

SAS/ETS® 14.1 User's Guide



The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2015. *SAS/ETS® 14.1 User's Guide*. Cary, NC: SAS Institute Inc.

SAS/ETS® 14.1 User's Guide

Copyright © 2015, SAS Institute Inc., Cary, NC, USA

All Rights Reserved. Produced in the United States of America.

For a hard-copy book: No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

For a web download or e-book: Your use of this publication shall be governed by the terms established by the vendor at the time you acquire this publication.

The scanning, uploading, and distribution of this book via the Internet or any other means without the permission of the publisher is illegal and punishable by law. Please purchase only authorized electronic editions and do not participate in or encourage electronic piracy of copyrighted materials. Your support of others' rights is appreciated.

U.S. Government License Rights; Restricted Rights: The Software and its documentation is commercial computer software developed at private expense and is provided with RESTRICTED RIGHTS to the United States Government. Use, duplication, or disclosure of the Software by the United States Government is subject to the license terms of this Agreement pursuant to, as applicable, FAR 12.212, DFAR 227.7202-1(a), DFAR 227.7202-3(a), and DFAR 227.7202-4, and, to the extent required under U.S. federal law, the minimum restricted rights as set out in FAR 52.227-19 (DEC 2007). If FAR 52.227-19 is applicable, this provision serves as notice under clause (c) thereof and no other notice is required to be affixed to the Software or documentation. The Government's rights in Software and documentation shall be only those set forth in this Agreement.

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

July 2015

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

Contents

I General Information **1**

Chapter 1. What's New in SAS/ETS 14.1 . . . 3
Chapter 2. Introduction . . . 13
Chapter 3. Shared Concepts and Topics . . . 61
Chapter 4. Working with Time Series Data . . . 91
Chapter 5. Date Intervals, Formats, and Functions . . . 151
Chapter 6. SAS Macros and Functions . . . 179
Chapter 7. Nonlinear Optimization Methods . . . 195

II Procedure Reference **215**

Chapter 8. The ARIMA Procedure . . . 217
Chapter 9. The AUTOREG Procedure . . . 331
Chapter 10. The COMPUTAB Procedure . . . 481
Chapter 11. The COPULA Procedure . . . 533
Chapter 12. The COUNTREG Procedure . . . 581
Chapter 13. The DATASOURCE Procedure . . . 697
Chapter 14. The ENTROPY Procedure (Experimental) . . . 785
Chapter 15. The ESM Procedure . . . 847
Chapter 16. The EXPAND Procedure . . . 883
Chapter 17. The FORECAST Procedure . . . 935
Chapter 18. The HPCDM Procedure (Experimental) . . . 989
Chapter 19. The HPCOPULA Procedure . . . 1057
Chapter 20. The HPCOUNTREG Procedure . . . 1071
Chapter 21. The HPPANEL Procedure . . . 1115
Chapter 22. The HPQLIM Procedure . . . 1143
Chapter 23. The HPSEVERITY Procedure . . . 1193
Chapter 24. The LOAN Procedure . . . 1355
Chapter 25. The MDC Procedure . . . 1393
Chapter 26. The MODEL Procedure . . . 1469
Chapter 27. The PANEL Procedure . . . 1793
Chapter 28. The PDLREG Procedure . . . 1925
Chapter 29. The QLIM Procedure . . . 1947
Chapter 30. The SEVERITY Procedure . . . 2067
Chapter 31. The SIMILARITY Procedure . . . 2235
Chapter 32. The SIMLIN Procedure . . . 2301
Chapter 33. The SPECTRA Procedure . . . 2329
Chapter 34. The SSM Procedure . . . 2355
Chapter 35. The STATESPACE Procedure . . . 2503
Chapter 36. The SYSLIN Procedure . . . 2547
Chapter 37. The TIMEDATA Procedure . . . 2605
Chapter 38. The TIMEID Procedure . . . 2635
Chapter 39. The TIMESERIES Procedure . . . 2657
Chapter 40. The TSCSREG Procedure . . . 2729

Chapter 41. The UCM Procedure . . . 2743
Chapter 42. The VARMAX Procedure . . . 2855
Chapter 43. The X11 Procedure . . . 3085
Chapter 44. The X12 Procedure . . . 3149
Chapter 45. The X13 Procedure . . . 3151

III Data Access Engines **3277**
Chapter 46. The SASECRSP Interface Engine . . . 3279
Chapter 47. The SASEFAME Interface Engine . . . 3325
Chapter 48. The SASEFRED Interface Engine . . . 3381
Chapter 49. The SASEHAVR Interface Engine . . . 3431
Chapter 50. The SASEQUAN Interface Engine . . . 3479
Chapter 51. The SASEXCCM Interface Engine . . . 3519
Chapter 52. The SASEXFSD Interface Engine . . . 3557

IV Time Series Forecasting System **3605**
Chapter 53. Overview of the Time Series Forecasting System . . . 3607
Chapter 54. Getting Started with Time Series Forecasting . . . 3611
Chapter 55. Creating Time ID Variables . . . 3667
Chapter 56. Specifying Forecasting Models . . . 3681
Chapter 57. Choosing the Best Forecasting Model . . . 3719
Chapter 58. Using Predictor Variables . . . 3739
Chapter 59. Command Reference . . . 3771
Chapter 60. Window Reference . . . 3779
Chapter 61. Forecasting Process Details . . . 3883

V Investment Analysis **3915**
Chapter 62. Overview . . . 3917
Chapter 63. Portfolios . . . 3921
Chapter 64. Investments . . . 3929
Chapter 65. Computations . . . 3971
Chapter 66. Analyses . . . 3983
Chapter 67. Details . . . 3997

Subject Index **4009**

Syntax Index **4058**

Credits and Acknowledgments

Credits

Documentation

Editing	Anne Baxter, Ed Huddleston
Technical Review	Evan L. Anderson, Wen Bardsley, Jennifer Beeman, Ming-Chun Chang, Owen Chen, Xilong Chen, Jan Chvosta, Allison Crutchfield, Gül Ege, Bruce Elsheimer, Donald J. Erdman, Kelly Fellingham, Bobby Gutierrez, Sanggohn Han, Laura Jackson, Wilma S. Jackson, Mahesh V. Joshi, Marc Kessler, Kathleen Kiernan, Jerry Leonard, Michael J. Leonard, Li C. Li, Mark R. Little, Christian Macaro, Kevin Meyer, Gina Marie Mondello, Youngjin Park, Richard Potter, Kenneth Sanford, Jim Seabolt, David Schlotzhauer, Rajesh Selukar, Mark Traccarella, Oleksiy Tokovenko, Michele A. Trovero, Charles Sun, Gunce Walton, Donna Woodward, Guohui Wu, Wan Xu
Documentation Production	Tim Arnold

Software

The procedures in SAS/ETS software were implemented by members of the Advanced Analytics Division. Program development includes design, programming, debugging, support, documentation, and technical review. In the following list, the name of the developer who currently has principal support responsibility for the procedure is given first.

ARIMA	Rajesh Selukar, Michael J. Leonard, Terry Woodfield
AUTOREG	Xilong Chen, Jan Chvosta, Richard Potter, John P. Sall
COMPUTAB	Richard Potter, Michael J. Leonard, Alan R. Eaton

COPULA	Jan Chvosta, Hao Chen
COUNTREG	Richard Potter, Guohui Wu, Christian Macaro, Jan Chvosta, Laura Jackson
DATASOURCE	Kelly Fellingham, Meltem Narter
ENTROPY	Xilong Chen, Donald J. Erdman
ESM	Michael J. Leonard
EXPAND	Michael J. Leonard, Marc Kessler, Mark R. Little
FORECAST	Rajesh Selukar, Michael J. Leonard, Mark R. Little, John P. Sall
HPCDM	Mahesh V. Joshi, Richard Potter, Jan Chvosta
HPCOPULA	Richard Potter, Jan Chvosta, Hao Chen
HPCOUNTREG	Richard Potter, Jan Chvosta
HPPANEL	Bobby Gutierrez, Linxia Ren, Richard Potter
HPQLIM	Christian Macaro, Richard Potter, Jan Chvosta
HPSEVERITY	Mahesh V. Joshi
LOAN	Richard Potter, Gül Ege
MDC	Owen Chen, Jan Chvosta
MODEL	Marc Kessler, Donald J. Erdman, Mark R. Little, John P. Sall
PANEL	Bobby Gutierrez, Linxia Ren, Jan Chvosta
PDLREG	Xilong Chen, Richard Potter, Jan Chvosta, Leigh A. Ihnen
QLIM	Christian Macaro, Gunce Walton, Jan Chvosta
SASECRSP	Kelly Fellingham, Richard D. Langston
SASEFAME	Kelly Fellingham
SASEFRED	Kelly Fellingham, Fatemeh Sayyady
SASEHAVR	Kelly Fellingham

SASEXCCM	Kelly Fellingham, Peng Zang
SASEXFSD	Kelly Fellingham, Fatemeh Sayyady, William McNeill, Richard D. Langston
SEVERITY	Mahesh V. Joshi
SIMILARITY	Michael J. Leonard
SIMLIN	Richard Potter, Mark R. Little, John P. Sall
SPECTRA	Michael J. Leonard, Marc Kessler, Rajesh Selukar, Donald J. Erdman, John P. Sall
SSM	Rajesh Selukar
STATESPACE	Rajesh Selukar, Donald J. Erdman, Michael J. Leonard
SYSLIN	Richard Potter, Laura Jackson, Donald J. Erdman, Leigh A. Ihnen, John P. Sall
TIMEDATA	Michael J. Leonard
TIMEID	Michael J. Leonard, Marc Kessler
TIMESERIES	Michael J. Leonard, Marc Kessler
TSCSREG	Bobby Gutierrez, Jan Chvosta
UCM	Rajesh Selukar
VARMAX	Xilong Chen, Youngjin Park
X11	Wilma S. Jackson, R. Bart Killam, Leigh A. Ihnen, Richard D. Langston
X12	Wilma S. Jackson
X13	Wilma S. Jackson
Time Series Forecasting System	Evan L. Anderson, Michael J. Leonard, Meltem Narter, Gül Ege
Investment Analysis System	Gül Ege, Scott Gray, Michael J. Leonard
Compiler and Symbolic Differentiation	Andrew Henrick, Stacey Christian, Mark R. Little, John P. Sall

Testing	Shu An, Jennifer Beeman, Ming-Chun Chang, Allison Crutchfield, Bruce Elsheimer, Kelly Fellingham, Wanxi Gu, Laura Gold, Sanggohn Han, Karen Hoffman, Li C. Li, Charles Sun, Oleksiy Tokovenko, Wan Xu, Kit-Woon Zhao

Technical Support

Members	Rob Agnelli, Wen Bardsley, Kathleen Kiernan, Bari Lawhorn, Jerry Leonard, Gina Marie Mondello, Charley Mullin, Paul Savarese, David Schlotzhauer, Jill Tao, Donna Woodward

Acknowledgments

Hundreds of people have helped the SAS System in many ways since its inception. The following individuals have been especially helpful in the development of the procedures in SAS/ETS software. Acknowledgments for the SAS System generally appear in Base SAS software documentation and SAS/ETS software documentation.

David Amick	Idaho Office of Highway Safety
Lai Cheng	Haver Analytics
David M. DeLong	Duke University
David Dickey	North Carolina State University
Douglas J. Drummond	Center for Survey Statistics
Janet Eder	Chicago Booth Center for Research in Security Prices
George Essig	Federal Reserve Bank of St. Louis
Michel Ferland	Statistics Canada
Susie Fortier	Statistics Canada
William Fortney	Boeing Computer Services
Wayne Fuller	Iowa State University
A. Ronald Gallant	Fuqua School, Duke University
Phil Hanser	Sacramento Municipal Utilities District
Maurine Haver	Haver Analytics
Marvin Jochimsen	Mississippi R&O Center
Tammer Kamel	Quandl Search Engine
Jeff Kaplan	SunGard Data Management Solutions
Ken Kraus	Chicago Booth Center for Research in Security Prices
Dominique Ladiray	INSEE
Derek Manning	Factset Research Systems, Inc.
George McCollister	San Diego Gas & Electric
Douglas Miller	Purdue University
Brian C. Monsell	U.S. Census Bureau
Robert Parks	Washington University
Benoit Quenneville	Statistics Canada
Gregory Sali	Idaho Office of Highway Safety
Artur Shepilko	Chicago Booth Center for Research in Security Prices
Bob Spatz	Chicago Booth Center for Research in Security Prices
Mary Young	Salt River Project

The final responsibility for the SAS System lies with SAS alone. We hope that you will always let us know your opinions about the SAS System and its documentation. It is through your participation that SAS software is continuously improved.

Part I

General Information

Chapter 1
What's New in SAS/ETS 14.1

Contents

Overview 3
Highlights of Changes and Enhancements 4
Highlights of Enhancements in SAS/ETS 13.2 4
COUNTREG Procedure 5
HPCOUNTREG Procedure 6
HPPANEL Procedure 6
MODEL Procedure 6
PANEL Procedure 7
QLIM Procedure 7
SASECRSP Engine 8
SASEFRED Engine 8
SASEXCCM Engine 9
SASEXFSD Engine 9
SASEQUAN Engine 9
SSM Procedure 10
VARMAX Procedure 10
X13 Procedure 11
References 12

Overview

This chapter summarizes the new features available in SAS/ETS 14.1 software.

If you have used SAS/ETS procedures in the past, you can review this chapter to learn about the new features that have been added. When you see a new feature that might be useful for your work, turn to the appropriate chapter to read about the feature in detail.

All high-performance procedures that are available in SAS High-Performance Econometrics software for distributed computing are also available in SAS/ETS software for use in single-machine mode. These procedures are now documented both in *SAS/ETS User's Guide* and in *SAS/ETS User's Guide: High-Performance Procedures*.

Highlights of Changes and Enhancements

The following procedure has been added to SAS/ETS software:

- X13 procedure

New features have been added to the following SAS/ETS components:

- COUNTREG procedure
- HPCOUNTREG procedure
- HPPANEL procedure
- MODEL procedure
- PANEL procedure
- QLIM procedure
- SASECRSP interface engine
- SASEFRED interface engine
- SASEXCCM interface engine
- SASEXFSD interface engine
- SASEQUAN interface engine
- SSM procedure
- VARMAX procedure

Highlights of Enhancements in SAS/ETS 13.2

Some users might be unfamiliar with updates made in the previous releases. SAS/ETS 13.2 introduced the SASEQUAN interface engine. Following are highlights of the other enhancements in SAS/ETS 13.2:

- The COUNTREG procedure was enhanced with the following Bayesian estimation features: the multithreaded Metropolis sampling algorithm, convergence diagnostic tools, the automated reiterative MCMC algorithm (which can search for the stationary region of the posterior distribution and determine the length of posterior sample), and a weighted scheme that can recombine the samples from multiple MCMC chains. The PERFORMANCE statement and item store support for BY groups were also added to the COUNTREG procedure.
- Support for panel data analysis was added to the HPCOUNTREG procedure.

- The HPSEVERITY procedure was enhanced to support the CLASS statement, classification and interaction effects in the SCALEMODEL statement, OUTSTORE= and INSTORE= options to store and read models that contain richer scale regression effects, PLOTS= and OUTCDF= options for single-machine mode, specification of probability of observability in the LOSS statement, and a richer set of options in the NLOPTIONS statement.
- The following enhancments were added to the PANEL procedure: first-differenced methods for one-way and two-way models, cross-sectional dependence tests, Lagrange multiplier (LM) tests for cross-sectional and time effects, and tests for serial correlation and cross-sectional effects.
- Enhanced Bayesian estimation features were added to the QLIM procedure. These features include an automated reiterative MCMC algorithm (which can search for the stationary region of the posterior distribution and determine the length of posterior samples) and a weighted scheme that can recombine the samples from multiple MCMC chains.
- The SASEFRED engine was enhanced by adding error checking to avoid situations in which the UNITS= option, the OUTPUT= option, and the VINTAGES= option are incompatible.
- The SEVERITY procedure was enhanced to support the CLASS statement, classification and interaction effects in the SCALEMODEL statement, OUTSTORE= and INSTORE= options to store and read models that contain richer scale regression effects, and an INITSAMPLE= option to limit the number of observations used for initialization that requires EDF computations.
- The VARMAX procedure was enhanced by adding *p*-values for the Johansen cointegration rank test and a multistep forecast for multivariate GARCH models.
- The X12 procedure was enhanced by adding the ALPHA= option to the FORECAST statement to specify the size of forecast confidence limits; adding the ALMOST=, ALPHA=, CVMETHOD=, LSRUN=, METHOD=, and TCRATE= options in the OUTLIER statement to enhance outlier detection; and by enabling you to specify seasonal filters in the SEASONALMA= option in the X11 statement for each seasonal period.

COUNTREG Procedure

The following features have been added to the COUNTREG procedure:

- The following Bayesian analysis features have been added:
  - harmonic mean evaluation of the marginal likelihood to compare competing models (this estimator does not require additional simulations after the posterior samples have been obtained)
  - evaluation of the marginal likelihood using an importance sampling algorithm based on the cross-entropy theory (this estimator requires additional importance sampling simulations after the posterior samples have been obtained)
- The TEST statement has been added.

- Spatial lag models have been added, enabling you to include spatial effects in a model. The SPATIALEFFECTS, SPATIALDISPEFFECTS, and SPATIALZEROEFFECTS statements have been added to allow for spatial effects in the MODEL, DISPMODEL, and ZEROMODEL statements, respectively. In addition, variable selection functionality has been added to the DISPMODEL, SPATIALEFFECTS, SPATIALDISPEFFECTS, and SPATIALZEROEFFECTS statements.

HPCOUNTREG Procedure

The following features have been added to the HPCOUNTREG procedure:

- TEST statement
- support for the Conway-Maxwell Poisson distribution

HPPANEL Procedure

The following features have been added to the HPPANEL procedure:

- the between-groups estimator, which is equivalent to regression on cross-sectional means (to obtain this estimator, specify the BTWNG option in the MODEL statement)
- the between-time-periods estimator, which is equivalent to regression on means at each time point (to obtain this estimator, specify the BTWNT option in the MODEL statement)
- pooled OLS regression, which is obtained by specifying the POOLED option in the MODEL statement

MODEL Procedure

The %EQAR and %EQMA macros have been added to the MODEL procedure. Model programs that are expressed using general form equations can now use the %EQAR macro to specify autoregressive error processes, or the %EQMA macro to specify moving average error processes. The %EQAR and %EQMA macros modify general form equations similarly to how the %AR and %MA macros modify normal form equations.

PANEL Procedure

The following features have been added to the PANEL procedure:

- The Hausman and Taylor (1981) and Amemiya and MaCurdy (1986) estimators, which are hybrids that combine the desirable properties of fixed-effects and random-effects models. Under the right circumstances, these estimators afford you the consistency of fixed effects and the efficiency and wider applicability of random effects. Both estimators are instrumental-variables regressions, where you stipulate a set of regressors as correlated with individual effects. The instrumental variables are then determined internally from the set of uncorrelated regressors, their individual-level means, and their deviations from individual-level means. To obtain these estimators, first specify the correlated regressors in the CORRELATED= option in the INSTRUMENTS statement, and then specify the HTAYLOR or AMACURDY option in the MODEL statement.
- Comparison tables for multiple models. The new COMPARE statement creates tables of side-by-side comparisons of parameter estimates and other model statistics. You can fit multiple models in the PANEL procedure by issuing multiple MODEL statements. Also specifying a COMPARE statement creates tables that compare the models. The COMPARE statement creates two tables: the first table compares model fit statistics such as R^2 and MSE; the second table compares regression coefficients, their standard errors, and (optionally) t tests.
- More general Hausman specification tests. In previous versions, Hausman tests for random effects required that the random-effects model contain no time-invariant regressors (regressors that would be dropped from the fixed-effects model). That requirement has been relaxed in SAS/ETS 14.1, and the Hausman test is now a comparison of regressors that are common to both the random- and fixed-effects models. A new column labeled “Coefficients” has been added to the output table for the Hausman test. The “Coefficients” column tells you how many coefficients are common to both models, and thus also tells you the nominal rank of the test.

QLIM Procedure

The following features have been added to the QLIM procedure:

- The following Bayesian analysis features have been added:
  - harmonic mean evaluation of the marginal likelihood to compare competing models (this estimator does not require additional simulations after the posterior samples have been obtained)
  - evaluation of the marginal likelihood using an importance sampling algorithm based on the cross-entropy theory (this estimator requires additional importance sampling simulations after the posterior samples have been obtained)
  - Bayesian analysis support for multivariate models in which the likelihood function is not available in closed form

- The RANDOM statement, which enables you to estimate the random-intercept models, has been added to PROC QLIM. Any single-equation model in PROC QLIM can be expanded to a random-intercept single-equation model. The RANDOM statement enables you to use panel data in your estimations, and it offers three methods to integrate out the random intercept. These methods are Gauss-Hermite quadrature, simulation, and quasi–Monte Carlo using a Halton sequence.

  The random-intercept models include the following:

  – random-intercept linear regression models
  – random-intercept discrete choice models, including binary probit, binary logit, ordinal probit, and ordinal logit models
  – random-intercept limited dependent variable models, including censored regression and truncated regression models
  – random-intercept stochastic frontier models

SASECRSP Engine

If you install SAS/ETS on a Windows system, you no longer need to install the CRSPAccess API, because it is now distributed automatically during the installation. Prior to running SASECRSP, your Windows setup requires that the CRSPDB_SASCAL environment variable be set to the path where your database calendar files reside.

SASEFRED Engine

The following features have been added to the SASEFRED engine:

- Linux X64 (LAX) host support has been added.
- The PROXY= option specifies a proxy server and port number to use if the connection times out without a proxy server.
- The RTSTART= and RTEND= options support the real-time periods for FRED data. Because the default is today, it is important to support the range of first to last available (the complete real-time period), in addition to other ranges for which data were available.
- You can use the URL= option to request the following useful information about categories, tags, groups, and releases:

  – a list of the vintage dates (release dates) for a particular series that is specified in the IDLIST= option
  – a list of the available series for a particular release or for a specified source, tag name, or category ID
  – a list of available sources for today's date

  - a list of the categories available for a specified series ID

- Blanks are now allowed in pathnames that are used in the options for SASEFRED.
- The DEBUG=ON option logs diagnostics in the SAS log.

SASEXCCM Engine

If you install SAS/ETS on a Windows system, you no longer need to install the CRSPAccess API, because it is now distributed automatically during the installation. Your Windows setup does not require any special environment variables.

SASEXFSD Engine

The following features have been added to the SASEXFSD engine:

- Linux X64 (LAX) host support has been added.
- The PROXY= option specifies a proxy server and port number to use if the connection times out without a proxy server.
- The CONNECT= option specifies whether to connect using the secure HTTP address in the PROXY= specification to obtain a secure connection.
- Blanks are now allowed in pathnames that are used in SASEXFSD options.
- The UNIVERSE= option is now supported on the ExtractFormulaHistory factlet.
- The DEBUG=ON option logs diagnostics in the SAS log.

SASEQUAN Engine

The following features have been added to the SASEQUAN engine:

- Linux X64 (LAX) host support has been added.
- The PROXY= option specifies a proxy server and port number to use if the connection times out without a proxy server.
- When a long variable name (more than 32 bytes) is truncated, the 32-byte name might not be unique, so SASEQUAN appends the variable number to the name in order to create a unique variable name.

- Blanks are now allowed in pathnames that are used in SASEQUAN options.
- Up to nine Quandl codes are allowed in the IDLIST= option. SASEQUAN returns an error message when more than nine are specified.
- The DEBUG=ON option logs diagnostics in the SAS log.

SSM Procedure

The DEPLAG statement has been added to the SSM procedure. It simplifies the specification of models that have lagged values of response variables in the observation equation. The DEPLAG statement enables you to define a linear combination of lagged response variables, which can be subsequently used as a right-hand-side term in the MODEL statement. Models that include lagged response variables are permitted only if the data form a time series (either univariate or multivariate).

VARMAX Procedure

The following features have been added to the VARMAX procedure:

- Vector error correction models in ARMA-GARCH form are supported. You can use the COINTEG statement together with the Q= option in the MODEL statement and the GARCH statement to model the cointegration relationship between multiple time series that have GARCH-type innovations.
- The linear equality and inequality constraints for any parameters to be estimated in vector error correction models are supported. You can use the BOUND and RESTRICT statement to study the restricted cointegrated systems.
- The covariance and standard errors of the parameter estimates of the adjustment coefficient matrix and the covariance matrix of innovations in vector error correction models are supported. The outputs of parameter estimates of the long-run parameters and the error correction trend parameters are also supported.
- You can apply the Wald tests, by using the TEST statement, on any parameters in vector correction models except the long-run parameters and the error correction trend parameters.
- You can specify initial values, by using the INITIAL statement, for any parameters to be estimated in vector error correction models. If you do so, you must specify the full-rank initial matrices for both the adjustment coefficient matrix and the long-run parameters.
- A new estimation method, the conditional maximum likelihood method (CML), is supported. This method is especially suitable for estimating VARMAX models on large samples.
- The log likelihoods for all types of models are output. These outputs are especially useful if you need to execute the likelihood ratio (LR) tests.

- Definitions have been revised for the information criteria: Akaike's information criterion (AIC), the corrected Akaike's information criterion (AICC), the Hannan-Quinn criterion (HQC), and the Schwarz Bayesian criterion (SBC, also referred to as BIC). You can compare more types of models, including all forms of multivariate GARCH models.
- The ECTREND option is supported in the COINTEG statement. All options in the ECM= option in the MODEL statement are now supported in the COINTEG statement and the ECM= option becomes obsolete. Starting with SAS/ETS 14.1, it is recommended that you use the COINTEG statement instead of the ECM= option to fit vector error correction models.
- A new option, the NLC option, is supported in the COINTEG statement. By using this option, you can explicitly require that the adjustment coefficient matrix and the long-run parameters both be full rank when you numerically maximize the likelihood of a vector error correction model.

X13 Procedure

Because the US Census Bureau has included the X-12-ARIMA methodology in the X-13ARIMA-SEATS program, the X12 procedure has been renamed the X13 procedure. You can continue to specify PROC X12 as before, but the X-12-ARIMA methodology will be based on the current method in the X-13ARIMA-SEATS method, including enhancements and improvements.

The following features have been added to the X13 procedure:

- The DIFFID= option has been added to the AUTOMDL statement. The DIFFID= option specifies the estimation method of identifying difference orders.

  The DIFFID= option is experimental in SAS/ETS 14.1.
- The DIFFIDITER= option has been added to the AUTOMDL statement. The DIFFIDITER= option specifies the maximum iterations for exact likelihood for DIFFID=EXACTFIRST.

  The DIFFIDITER= option is experimental in SAS/ETS 14.1.
- The EXACT= option has been added to the ESTIMATE statement. The EXACT= option specifies the likelihood function to be used for estimating AR and MA parameters.

  The EXACT= option is experimental in SAS/ETS 14.1.
- The EASTERMEANS= option has been added to the REGRESSION statement. The EASTERMEANS= option specifies the method to be used to calculate the means for the Easter regression variable.
- The following tables have been added to the displayed output for automatic modeling: "Check of the Residual Ljung-Box Q Statistic," "Initial Automatic Model Selection," and "Final Checks for Identified Model." In addition, if the orders have been altered during the final checks, the new Orders Altered column in the "Final Automatic Model Selection" table displays a value of Yes. These changes clarify the final automatic model selection process.
- The default for the MAXITER= option in the ESTIMATE statement has been changed to 1,500.

- The following tables have been added to the tables available in the X11 statement: B7, B13, B17, B20, C1, D8B, and E18.

References

Amemiya, T., and MaCurdy, T. E. (1986). "Instrumental-Variable Estimation of an Error-Components Model." *Econometrica* 54:869–880.

Hausman, J. A., and Taylor, W. E. (1981). "Panel Data and Unobservable Individual Effects." *Econometrica* 49:1377–1398.

Chapter 2
Introduction

Contents

Overview of SAS/ETS Software **14**
Uses of SAS/ETS Software 15
Contents of SAS/ETS Software 16
Experimental Software **18**
About This Book **18**
Chapter Organization 19
Typographical Conventions 20
Options Used in Examples 20
Where to Turn for More Information **21**
Accessing the SAS/ETS Sample Library 21
Online Help System 21
SAS Short Courses 21
SAS Technical Support Services 21
Major Features of SAS/ETS Software **22**
Discrete Choice and Qualitative and Limited Dependent Variable Analysis 22
Regression with Autocorrelated and Heteroscedastic Errors 24
Simultaneous Systems Linear Regression 25
Linear Systems Simulation 26
Polynomial Distributed Lag Regression 27
Nonlinear Systems Regression and Simulation 27
ARIMA (Box-Jenkins) and ARIMAX (Box-Tiao) Modeling and Forecasting 29
Vector Time Series Analysis 30
State Space Modeling and Forecasting 31
Spectral Analysis 32
Seasonal Adjustment 33
Structural Time Series Modeling and Forecasting 34
Time Series Cross-Sectional Regression Analysis 34
Automatic Time Series Forecasting 35
Time Series Interpolation and Frequency Conversion 37
Trend and Seasonal Analysis on Transaction Databases 39
Access to Financial and Economic Databases 39
Spreadsheet Calculations and Financial Report Generation 42
Loan Analysis, Comparison, and Amortization 42
Time Series Forecasting System 43
Investment Analysis System 44
ODS Graphics 45

Related SAS Software **46**
Base SAS Software 46
SAS Forecast Studio 49
SAS High-Performance Forecasting 49
SAS/GRAPH Software 50
SAS/STAT Software 50
SAS/IML Software 51
SAS/IML Stat Studio 52
SAS/OR Software 53
SAS/QC Software 53
MLE for User-Defined Likelihood Functions 54
JMP Software 54
SAS Enterprise Guide 55
SAS Add-In for Microsoft Office 56
Enterprise Miner—Time Series nodes 56
SAS Risk Products 57
References **58**

Overview of SAS/ETS Software

SAS/ETS software, a component of the SAS System, provides SAS procedures for:

- econometric analysis
- time series analysis
- time series forecasting
- systems modeling and simulation
- discrete choice analysis
- analysis of qualitative and limited dependent variable models
- seasonal adjustment of time series data
- financial analysis and reporting
- access to economic and financial databases
- time series data management

In addition to SAS procedures, SAS/ETS software also includes seamless access to economic and financial databases and interactive environments for time series forecasting and investment analysis.

Uses of SAS/ETS Software

SAS/ETS software provides tools for a wide variety of applications in business, government, and academia. Major uses of SAS/ETS procedures are economic analysis, forecasting, economic and financial modeling, time series analysis, financial reporting, and manipulation of time series data.

The common theme relating the many applications of the software is time series data: SAS/ETS software is useful whenever it is necessary to analyze or predict processes that take place over time or to analyze models that involve simultaneous relationships.

Although SAS/ETS software is most closely associated with business, finance and economics, time series data also arise in many other fields. SAS/ETS software is useful whenever time dependencies, simultaneous relationships, or dynamic processes complicate data analysis. For example, an environmental quality study might use SAS/ETS software's time series analysis tools to analyze pollution emissions data. A pharmacokinetic study might use SAS/ETS software's features for nonlinear systems to model the dynamics of drug metabolism in different tissues.

The diversity of problems for which econometrics and time series analysis tools are needed is reflected in the applications reported by SAS users. The following listed items are some applications of SAS/ETS software presented by SAS users at past annual conferences of the SAS Users Group International (SUGI).

- forecasting college enrollment (Calise and Earley 1997)
- fitting a pharmacokinetic model (Morelock et al. 1995)
- testing interaction effect in reducing sudden infant death syndrome (Fleming, Gibson, and Fleming 1996)
- forecasting operational indices to measure productivity changes (McCarty 1994)
- spectral decomposition and reconstruction of nuclear plant signals (Hoyer and Gross 1993)
- estimating parameters for the constant-elasticity-of-substitution translog model (Hisnanick 1993)
- applying econometric analysis for mass appraisal of real property (Amal and Weselowski 1993)
- forecasting telephone usage data (Fischetti, Heathcote, and Perry 1993)
- forecasting demand and utilization of inpatient hospital services (Hisnanick 1992)
- using conditional demand estimation to determine electricity demand (Keshani and Taylor 1992)
- estimating tree biomass for measurement of forestry yields (Parresol and Thomas 1991)
- evaluating the theory of input separability in the production function of U.S. manufacturing (Hisnanick 1991)
- forecasting dairy milk yields and composition (Benseman 1990)
- predicting the gloss of coated aluminum products subject to weathering (Khan 1990)
- learning curve analysis for predicting manufacturing costs of aircraft (LeBouton 1989)

- analyzing Dow Jones stock index trends (Earley, Sweeney, and Zekavat 1989)
- analyzing the usefulness of the composite index of leading economic indicators for forecasting the economy (Lin and Myers 1988)

Contents of SAS/ETS Software

Procedures

SAS/ETS software includes the following SAS procedures:

ARIMA	ARIMA (Box-Jenkins) and ARIMAX (Box-Tiao) modeling and forecasting
AUTOREG	regression analysis with autocorrelated or heteroscedastic errors and ARCH and GARCH modeling
COMPUTAB	spreadsheet calculations and financial report generation
COPULA	fitting and simulating multivariate distributions by using copula methods
COUNTREG	regression modeling for dependent variables that represent counts
DATASOURCE	access to financial and economic databases
ENTROPY	maximum entropy-based regression
ESM	forecasting by using exponential smoothing models with optimized smoothing weights
EXPAND	time series interpolation, frequency conversion, and transformation of time series
FORECAST	automatic forecasting (deprecated procedure)
LOAN	loan analysis and comparison
MDC	multinomial discrete choice analysis
MODEL	nonlinear simultaneous equations regression and nonlinear systems modeling and simulation
PANEL	panel data models
PDLREG	polynomial distributed lag regression
QLIM	qualitative and limited dependent variable analysis
SEVERITY	modeling the statistical distribution of the severity of losses and other events
SIMILARITY	similarity analysis of time series data for time series data mining
SIMLIN	linear systems simulation
SPECTRA	spectral and cross-spectral analysis
SSM	state space modeling of time series
STATESPACE	state space modeling and automated forecasting of multivariate time series
SYSLIN	linear simultaneous equations models

TIMEDATA	analyzes time-stamped transactional data with respect to time and accumulates the data into a time series format
TIMEID	identifying the time frequency for data sets that contain time series data
TIMESERIES	analysis of time-stamped transactional data
TSCSREG	time series cross-sectional regression analysis
UCM	unobserved components analysis of time series
VARMAX	vector autoregressive and moving average with modeling and forecasting
X11	seasonal adjustment (Census X-11 and X-11 ARIMA)
X12	seasonal adjustment (Census X-12 ARIMA)

Access Interfaces to Economic and Financial Databases

SAS/ETS software includes the following LIBNAME statement engines to provide access to financial and economic databases:

SASECRSP	LIBNAME engine for accessing time series and event data that reside in a CRSPAccess database
SASEFAME	LIBNAME engine for accessing time series or case series data that reside in a FAME database
SASEFRED	LIBNAME engine to retrieve economic data from the FRED website, which is hosted by the Economic Research Division of the Federal Reserve Bank of St. Louis
SASEHAVR	LIBNAME engine for accessing time series that reside in a Haver Analytics Data Link Express (DLX) database
SASEQUAN	LIBNAME engine to retrieve economic data from the Quandl website, which offers access to 8 million time series data sets from 400 sources in finance, economics, society, health, energy, demography, and more
SASEXCCM	LIBNAME engine for accessing data items that reside in the CRSP US Stock (STK) Database, the CRSP US Stock and Indices (IND) Database, the CRSP US Treasury (TRS) Database, or the CRSP/Compustat Merged (CCM) Database, which is created from data delivered via Standard & Poor's Compustat Xpressfeed product
SASEXFSD	LIBNAME engine for accessing both FactSet data and FactSet sourced data that are provided by the FactSet OnDemand service

Macros

SAS/ETS software includes the following SAS macros:

%AR	generates statements to define autoregressive error models for the MODEL procedure
%BOXCOXAR	investigates Box-Cox transformations useful for modeling and forecasting a time series
%DFPVALUE	computes probabilities for Dickey-Fuller test statistics
%DFTEST	performs Dickey-Fuller tests for unit roots in a time series process

%LOGTEST	tests to determine whether a log transformation is appropriate for modeling and forecasting a time series
%MA	generates statements to define moving-average error models for the MODEL procedure
%PDL	generates statements to define polynomial distributed lag models for the MODEL procedure

These macros are part of the SAS AUTOCALL facility and are automatically available for use in your SAS program. Refer to *SAS Macro Language: Reference* for information about the SAS macro facility.

The Time Series Forecasting System

SAS/ETS software includes an interactive forecasting system, described in *Part IV*. This graphical user interface to SAS/ETS forecasting features was developed with SAS/AF software and uses PROC ARIMA and other internal routines to perform time series forecasting. The Time Series Forecasting System makes it easy to forecast time series and provides many features for graphical data exploration and graphical comparisons of forecasting models and forecasts. (You must have SAS/GRAPH installed to use the graphical features of the system.)

The Investment Analysis System

The Investment Analysis System, described in *Part V*, is an interactive environment for analyzing the time-value of money in a variety of investments. Various analyses are provided to help analyze the value of investment alternatives: time value, periodic equivalent, internal rate of return, benefit-cost ratio, and break-even analysis.

Experimental Software

Experimental software is sometimes included as part of a production-release product. It is provided to customers in order to obtain feedback. All experimental features are marked Experimental in this document. Whenever an experimental procedure, statement, or option is used, a message is displayed in the SAS log to indicate that it is experimental. The design and syntax of experimental software might change before any production release. Experimental software has been tested prior to release, but it has not necessarily been tested to production-quality standards, so it should be used with care.

About This Book

This book is a user's guide to SAS/ETS software. Since SAS/ETS software is a part of the SAS System, this book assumes that you are familiar with Base SAS software and have the books *SAS Language Reference: Dictionary* and *Base SAS Procedures Guide* available for reference. It also assumes that you are familiar with SAS data sets, the SAS DATA step, and with basic SAS procedures such as PROC PRINT and PROC SORT. Chapter 4, "Working with Time Series Data," in this book summarizes the aspects of Base SAS software that are most relevant to the use of SAS/ETS software.

Chapter Organization

Following a brief What's New, this book is divided into five major parts. *Part I* contains general information to aid you in working with SAS/ETS Software. *Part II* explains the SAS procedures of SAS/ETS software. *Part III* describes the available data access interfaces for economic and financial databases. *Part IV* is the reference for the Time Series Forecasting System, an interactive forecasting menu system that uses PROC ARIMA and other routines to perform time series forecasting. Finally, *Part V* is the reference for the Investment Analysis System.

The new features added to SAS/ETS software since the publication of *SAS/ETS Software: Changes and Enhancements for Release 13.2* are summarized in Chapter 1, "What's New in SAS/ETS 14.1." If you have used SAS/ETS software in the past, you may want to skim this chapter to see what's new.

Part I contains the following chapters.

Chapter 2, the current chapter, provides an overview of SAS/ETS software and summarizes related SAS publications, products, and services.

Chapter 4, "Working with Time Series Data," discusses the use of SAS data management and programming features for time series data.

Chapter 5, "Date Intervals, Formats, and Functions," summarizes the time intervals, date and datetime informats, date and datetime formats, and date and datetime functions available in the SAS System.

Chapter 6, "SAS Macros and Functions," documents SAS macros and DATA step financial functions provided with SAS/ETS software. The macros use SAS/ETS procedures to perform Dickey-Fuller tests, test for the need for log transformations, or select optimal Box-Cox transformation parameters for time series data.

Chapter 7, "Nonlinear Optimization Methods," documents the NonLinear Optimization subsystem used by some ETS procedures to perform nonlinear optimization tasks.

Part II contains chapters that explain the SAS procedures that make up SAS/ETS software. These chapters appear in alphabetical order by procedure name.

Part III contains chapters that document the ETS access interfaces to economic and financial databases.

Each of the chapters that document the SAS/ETS procedures (*Part II*) and the SAS/ETS access interfaces (*Part III*) is organized as follows:

1. The "Overview" section gives a brief description of the procedure.
2. The "Getting Started" section provides a tutorial introduction on how to use the procedure.
3. The "Syntax" section is a reference to the SAS statements and options that control the procedure.
4. The "Details" section discusses various technical details.
5. The "Examples" section contains examples of the use of the procedure.
6. The "References" section contains technical references on methodology.

Part IV contains the chapters that document the features of the Time Series Forecasting System.

Part V contains chapters that document the features of the Investment Analysis System.

Typographical Conventions

This book uses several type styles for presenting information. The following list explains the meaning of the typographical conventions used in this book:

roman	is the standard type style used for most text.
UPPERCASE ROMAN	is used for SAS statements, options, and other SAS language elements when they appear in the text. However, you can enter these elements in your own SAS programs in lowercase, uppercase, or a mixture of the two.
UPPERCASE BOLD	is used in the "Syntax" sections' initial lists of SAS statements and options.
oblique	is used for user-supplied values for options in the syntax definitions. In the text, these values are written in *italic*.
helvetica	is used for the names of variables and data sets when they appear in the text.
bold	is used to refer to matrices and vectors and to refer to commands.
italic	is used for terms that are defined in the text, for emphasis, and for references to publications.
`bold monospace`	is used for example code. In most cases, this book uses lowercase type for SAS statements.

Options Used in Examples

The HTMLBLUE style is used to create the graphs and the HTML tables that appear in the online documentation. The PEARLJ style is used to create the PDF tables that appear in the documentation. A style template controls stylistic elements such as colors, fonts, and presentation attributes. You can specify a style template in an ODS destination statement as follows:

```
ods html style=HTMLBlue;
. . .
ods html close;

ods pdf style=PearlJ;
. . .
ods pdf close;
```

Most of the PDF tables are produced by using the following SAS System option:

```
options papersize=(6.5in 9in);
```

If you run the examples, you might get slightly different output. This is a function of the SAS System options that are used and the precision that your computer uses for floating-point calculations.

Where to Turn for More Information

This section describes other sources of information about SAS/ETS software.

Accessing the SAS/ETS Sample Library

The SAS/ETS Sample Library includes many examples that illustrate the use of SAS/ETS software, including the examples used in this documentation. To access these sample programs, select **Help** from the menu and then select **SAS Help and Documentation**. From the **Contents** list, select the section **Sample SAS Programs** under **Learning to Use SAS**.

Online Help System

You can access online help information about SAS/ETS software in two ways, depending on whether you are using the SAS windowing environment in the command line mode or the pull-down menu mode.

If you are using a command line, you can access the SAS/ETS help menus by typing **help** on the SAS windowing environment command line. Or you can issue the command **help ARIMA** (or another procedure name) to display the help for that particular procedure.

If you are using the SAS windowing environment pull-down menus, you can pull-down the **Help** menu and make the following selections:

- **SAS Help and Documentation**
- **Learning to Use SAS** in the Contents list
- **SAS Products**
- **SAS/ETS**

The content of the Online Help System follows closely that of this book.

SAS Short Courses

The SAS Education Division offers a number of training courses that might be of interest to SAS/ETS users. Please check the SAS web site for the current list of available training courses.

SAS Technical Support Services

As with all SAS products, the SAS Technical Support staff is available to respond to problems and answer technical questions regarding the use of SAS/ETS software.

Major Features of SAS/ETS Software

The following sections briefly summarize major features of SAS/ETS software. See the chapters on individual procedures for more detailed information.

Discrete Choice and Qualitative and Limited Dependent Variable Analysis

The MDC procedure provides maximum likelihood (ML) or simulated maximum likelihood estimates of multinomial discrete choice models in which the choice set consists of unordered multiple alternatives.

The MDC procedure supports the following models and features:

- conditional logit
- nested logit
- heteroscedastic extreme value
- multinomial probit
- mixed logit
- pseudo-random or quasi-random numbers for simulated maximum likelihood estimation
- bounds imposed on the parameter estimates
- linear restrictions imposed on the parameter estimates
- SAS data set containing predicted probabilities and linear predictor ($\mathbf{x}'\boldsymbol{\beta}$) values
- decision tree and nested logit
- model fit and goodness-of-fit measures including
  - likelihood ratio
  - Aldrich-Nelson
  - Cragg-Uhler 1
  - Cragg-Uhler 2
  - Estrella
  - Adjusted Estrella
  - McFadden's LRI
  - Veall-Zimmermann
  - Akaike Information Criterion (AIC)
  - Schwarz Criterion or Bayesian Information Criterion (BIC)

The QLIM procedure analyzes univariate and multivariate limited dependent variable models where dependent variables take discrete values or dependent variables are observed only in a limited range of values. This procedure includes logit, probit, Tobit, and general simultaneous equations models. The QLIM procedure supports the following models:

- linear regression model with heteroscedasticity
- probit with heteroscedasticity
- logit with heteroscedasticity
- Tobit (censored and truncated) with heteroscedasticity
- Box-Cox regression with heteroscedasticity
- bivariate probit
- bivariate Tobit
- sample selection models
- multivariate limited dependent models

The COUNTREG procedure provides regression models in which the dependent variable takes nonnegative integer count values. The COUNTREG procedure supports the following models:

- Poisson regression
- negative binomial regression with quadratic and linear variance functions
- zero inflated Poisson (ZIP) model
- zero inflated negative binomial (ZINB) model
- fixed and random effect Poisson panel data models
- fixed and random effect NB (negative binomial) panel data models

The PANEL procedure deals with panel data sets that consist of time series observations on each of several cross-sectional units.

The models and methods the PANEL procedure uses to analyze are as follows:

- one-way and two-way models
- fixed and random effects
- autoregressive models
  - the Parks method
  - dynamic panel estimator
  - the Da Silva method for moving-average disturbances

Regression with Autocorrelated and Heteroscedastic Errors

The AUTOREG procedure provides regression analysis and forecasting of linear models with autocorrelated or heteroscedastic errors. The AUTOREG procedure includes the following features:

- estimation and prediction of linear regression models with autoregressive errors
- any order autoregressive or subset autoregressive process
- optional stepwise selection of autoregressive parameters
- choice of the following estimation methods:
  - exact maximum likelihood
  - exact nonlinear least squares
  - Yule-Walker
  - iterated Yule-Walker
- tests for any linear hypothesis that involves the structural coefficients
- restrictions for any linear combination of the structural coefficients
- forecasts with confidence limits
- estimation and forecasting of ARCH (autoregressive conditional heteroscedasticity), GARCH (generalized autoregressive conditional heteroscedasticity), I-GARCH (integrated GARCH), E-GARCH (exponential GARCH), and GARCH-M (GARCH in mean) models
- combination of ARCH and GARCH models with autoregressive models, with or without regressors
- estimation and testing of general heteroscedasticity models
- variety of model diagnostic information including the following:
  - autocorrelation plots
  - partial autocorrelation plots
  - Durbin-Watson test statistic and generalized Durbin-Watson tests to any order
  - Durbin h and Durbin t statistics
  - Akaike information criterion
  - Schwarz information criterion
  - tests for ARCH errors
  - Ramsey's RESET test
  - Chow and PChow tests
  - Phillips-Perron stationarity test
  - CUSUM and CUMSUMSQ statistics
- exact significance levels (p-values) for the Durbin-Watson statistic
- embedded missing values

Simultaneous Systems Linear Regression

The SYSLIN and ENTROPY procedures provide regression analysis of a simultaneous system of linear equations.

The SYSLIN procedure includes the following features:

- estimation of parameters in simultaneous systems of linear equations
- full range of estimation methods including the following:
  - ordinary least squares (OLS)
  - two-stage least squares (2SLS)
  - three-stage least squares (3SLS)
  - iterated 3SLS (IT3SLS)
  - seemingly unrelated regression (SUR)
  - iterated SUR (ITSUR)
  - limited-information maximum likelihood (LIML)
  - full-information maximum likelihood (FIML)
  - minimum expected loss (MELO)
  - general K-class estimators
- weighted regression
- any number of restrictions for any linear combination of coefficients, within a single model or across equations
- tests for any linear hypothesis, for the parameters of a single model or across equations
- wide range of model diagnostics and statistics including the following:
  - usual ANOVA tables and R-square statistics
  - Durbin-Watson statistics
  - standardized coefficients
  - test for overidentifying restrictions
  - residual plots
  - standard errors and *t* tests
  - covariance and correlation matrices of parameter estimates and equation errors
- predicted values, residuals, parameter estimates, and variance-covariance matrices saved in output SAS data sets
- other features of the SYSLIN procedure that enable you to do the following:
  - impose linear restrictions on the parameter estimates
  - test linear hypotheses about the parameters

  - write predicted and residual values to an output SAS data set
  - write parameter estimates to an output SAS data set
  - write the crossproducts matrix (SSCP) to an output SAS data set
  - use raw data, correlations, covariances, or cross products as input

The ENTROPY procedure supports the following models and features:

- generalized maximum entropy (GME) estimation
- generalized cross entropy (GCE) estimation
- normed moment generalized maximum entropy
- maximum entropy-based seemingly unrelated regression (MESUR) estimation
- pure inverse estimation
- estimation of parameters in simultaneous systems of linear equations
- Markov models
- unordered multinomial choice problems
- weighted regression
- any number of restrictions for any linear combination of coefficients, within a single model or across equations
- tests for any linear hypothesis, for the parameters of a single model or across equations

Linear Systems Simulation

The SIMLIN procedure performs simulation and multiplier analysis for simultaneous systems of linear regression models. The SIMLIN procedure includes the following features:

- reduced form coefficients
- interim multipliers
- total multipliers
- dynamic multipliers
- multipliers for higher order lags
- dynamic forecasts and simulations
- goodness-of-fit statistics
- acceptance of the equation system coefficients estimated by the SYSLIN procedure as input

Polynomial Distributed Lag Regression

The PDLREG procedure provides regression analysis for linear models with polynomial distributed (Almon) lags. The PDLREG procedure includes the following features:

- entry of any number of regressors as a polynomial lag distribution and the use of any number of covariates
- use of any order lag length and degree polynomial for lag distribution
- optional upper and lower endpoint restrictions
- specification of any number of linear restrictions on covariates
- option to repeat analysis over a range of degrees for the lag distribution polynomials
- support for autoregressive errors to any lag
- forecasts with confidence limits

Nonlinear Systems Regression and Simulation

The MODEL procedure provides parameter estimation, simulation, and forecasting of dynamic nonlinear simultaneous equation models. The MODEL procedure includes the following features:

- nonlinear regression analysis for systems of simultaneous equations, including weighted nonlinear regression
- full range of parameter estimation methods including the following:
  - nonlinear ordinary least squares (OLS)
  - nonlinear seemingly unrelated regression (SUR)
  - nonlinear two-stage least squares (2SLS)
  - nonlinear three-stage least squares (3SLS)
  - iterated SUR
  - iterated 3SLS
  - generalized method of moments (GMM)
  - nonlinear full-information maximum likelihood (FIML)
  - simulated method of moments (SMM)
- supports dynamic multi-equation nonlinear models of any size or complexity
- uses the full power of the SAS programming language for model definition, including left-hand-side expressions

- hypothesis tests of nonlinear functions of the parameter estimates
- linear and nonlinear restrictions of the parameter estimates
- bounds imposed on the parameter estimates
- computation of estimates and standard errors of nonlinear functions of the parameter estimates
- estimation and simulation of ordinary differential equations (ODE's)
- vector autoregressive error processes and polynomial lag distributions easily specified for the nonlinear equations
- variance modeling (ARCH, GARCH, and others)
- computation of goal-seeking solutions of nonlinear systems to find input values needed to produce target outputs
- dynamic, static, or *n*-period-ahead-forecast simulation modes
- simultaneous solution or single equation solution modes
- Monte Carlo simulation using parameter estimate covariance and across-equation residuals covariance matrices or user-specified random functions
- a variety of diagnostic statistics including the following
  - model R-square statistics
  - general Durbin-Watson statistics and exact *p*-values
  - asymptotic standard errors and *t* tests
  - first-stage R-square statistics
  - covariance estimates
  - collinearity diagnostics
  - simulation goodness-of-fit statistics
  - Theil inequality coefficient decompositions
  - Theil relative change forecast error measures
  - heteroscedasticity tests
  - Godfrey test for serial correlation
  - Hausman specification test
  - Chow tests
- block structure and dependency structure analysis for the nonlinear system
- listing and cross-reference of fitted model
- automatic calculation of needed derivatives by using exact analytic formula
- efficient sparse matrix methods used for model solution; choice of other solution methods

Model definition, parameter estimation, simulation, and forecasting can be performed interactively in a single SAS session or models can also be stored in files and reused and combined in later runs.

ARIMA (Box-Jenkins) and ARIMAX (Box-Tiao) Modeling and Forecasting

The ARIMA procedure provides the identification, parameter estimation, and forecasting of autoregressive integrated moving-average (Box-Jenkins) models, seasonal ARIMA models, transfer function models, and intervention models. The ARIMA procedure includes the following features:

- complete ARIMA (Box-Jenkins) modeling with no limits on the order of autoregressive or moving-average processes
- model identification diagnostics including the following:
  - autocorrelation function
  - partial autocorrelation function
  - inverse autocorrelation function
  - cross-correlation function
  - extended sample autocorrelation function
  - minimum information criterion for model identification
  - squared canonical correlations
- stationarity tests
- outlier detection
- intervention analysis
- regression with ARMA errors
- transfer function modeling with fully general rational transfer functions
- seasonal ARIMA models
- ARIMA model-based interpolation of missing values
- several parameter estimation methods including the following:
  - exact maximum likelihood
  - conditional least squares
  - exact nonlinear unconditional least squares (ELS or ULS)
- prewhitening transformations
- forecasts and confidence limits for all models
- forecasting tied to parameter estimation methods: finite memory forecasts for models estimated by maximum likelihood or exact nonlinear least squares methods and infinite memory forecasts for models estimated by conditional least squares
- diagnostic statistics to help judge the adequacy of the model including the following:
  - Akaike's information criterion (AIC)

  - Schwarz's Bayesian criterion (SBC or BIC)
  - Box-Ljung chi-square test statistics for white-noise residuals
  - autocorrelation function of residuals
  - partial autocorrelation function of residuals
  - inverse autocorrelation function of residuals
  - automatic outlier detection

Vector Time Series Analysis

The VARMAX procedure enables you to model the dynamic relationship both between the dependent variables and between the dependent and independent variables. The VARMAX procedure includes the following features:

- several modeling features:
  - vector autoregressive model
  - vector autoregressive model with exogenous variables
  - vector autoregressive and moving-average model
  - Bayesian vector autoregressive model
  - vector error correction model
  - Bayesian vector error correction model
  - GARCH-type multivariate conditional heteroscedasticity models
- criteria for automatically determining AR and MA orders:
  - Akaike information criterion (AIC)
  - corrected AIC (AICC)
  - Hannan-Quinn (HQ) criterion
  - final prediction error (FPE)
  - Schwarz Bayesian criterion (SBC), also known as Bayesian information criterion (BIC)
- AR order identification aids:
  - partial cross-correlations
  - Yule-Walker estimates
  - partial autoregressive coefficients
  - partial canonical correlations
- testing the presence of unit roots and cointegration:
  - Dickey-Fuller tests
  - Johansen cointegration test for nonstationary vector processes of integrated order one

  - Stock-Watson common trends test for the possibility of cointegration among nonstationary vector processes of integrated order one
  - Johansen cointegration test for nonstationary vector processes of integrated order two
- model parameter estimation methods:
  - least squares (LS)
  - maximum likelihood (ML)
- model checks and residual analysis using the following tests:
  - Durbin-Watson (DW) statistics
  - *F* test for autoregressive conditional heteroscedastic (ARCH) disturbance
  - *F* test for AR disturbance
  - Jarque-Bera normality test
  - Portmanteau test
- seasonal deterministic terms
- subset models
- multiple regression with distributed lags
- dead-start model that does not have present values of the exogenous variables
- Granger-causal relationships between two distinct groups of variables
- infinite order AR representation
- impulse response function (or infinite order MA representation)
- decomposition of the predicted error covariances
- roots of the characteristic functions for both the AR and MA parts to evaluate the proximity of the roots to the unit circle
- contemporaneous relationships among the components of the vector time series
- forecasts
- conditional covariances for GARCH models

State Space Modeling and Forecasting

The STATESPACE procedure provides automatic model selection, parameter estimation, and forecasting of state space models. (*State space models* encompass an alternative general formulation of multivariate ARIMA models.) The STATESPACE procedure includes the following features:

- multivariate ARIMA modeling by using the general state space representation of the stochastic process
- automatic model selection using Akaike's information criterion (AIC)
- user-specified state space models including restrictions
- transfer function models with random inputs
- any combination of simple and seasonal differencing; input series can be differenced to any order for any lag lengths
- forecasts with confidence limits
- ability to save selected and fitted model in a data set and reuse for forecasting
- wide range of output options including the ability to print any statistics concerning the data and their covariance structure, the model selection process, and the final model fit

Spectral Analysis

The SPECTRA procedure provides spectral analysis and cross-spectral analysis of time series. The SPECTRA procedure includes the following features:

- efficient calculation of periodogram and smoothed periodogram using fast finite Fourier transform and Chirp-Z algorithms
- multiple spectral analysis, including raw and smoothed spectral and cross-spectral function estimates, with user-specified window weights
- choice of kernel for smoothing
- output of the following spectral estimates to a SAS data set:
  - Fourier sine and cosine coefficients
  - periodogram
  - smoothed periodogram
  - cospectrum
  - quadrature spectrum
  - amplitude
  - phase spectrum
  - squared coherency
- Fisher's Kappa and Bartlett's Kolmogorov-Smirnov test statistic for testing a null hypothesis of white noise

Seasonal Adjustment

The X11 procedure provides seasonal adjustment of time series by using the Census X-11 or X-11 ARIMA method. The X11 procedure is based on the U.S. Bureau of the Census X-11 seasonal adjustment program and also supports the X-11 ARIMA method developed by Statistics Canada. The X11 procedure includes the following features:

- decomposition of monthly or quarterly series into seasonal, trend, trading day, and irregular components
- both multiplicative and additive form of the decomposition
- all the features of the Census Bureau program
- support of the X-11 ARIMA method
- support of sliding spans analysis
- processing of any number of variables at once with no maximum length for a series
- computation of tests for stable, moving, and combined seasonality
- optional printing or storing in SAS data sets of the individual X11 tables that show the various components at different stages of the computation; full control over what is printed or output
- ability to project seasonal component one year ahead, which enables reintroduction of seasonal factors for an extrapolated series

The X12 procedure provides seasonal adjustment of time series using the X-12 ARIMA method. The X12 procedure is based on the U.S. Bureau of the Census X-12 ARIMA seasonal adjustment program (version 0.3). It also supports the X-11 ARIMA method developed by Statistics Canada and the previous X-11 method of the U.S. Census Bureau. The X12 procedure includes the following features:

- decomposition of monthly or quarterly series into seasonal, trend, trading day, and irregular components
- support of multiplicative, additive, pseudo-additive, and log additive forms of decomposition
- support of the X-12 ARIMA method
- support of regARIMA modeling
- automatic identification of outliers
- support of TRAMO-based automatic model selection
- use of regressors to process missing values within the span of the series
- processing of any number of variables at once with no maximum length for a series
- computation of tests for stable, moving, and combined seasonality
- spectral analysis of original, seasonally adjusted, and irregular series

- optional printing or storing in a SAS data set of the individual X11 tables that show the various components at different stages of the decomposition; full control over what is printed or output
- optional projection of seasonal component one year ahead, which enables reintroduction of seasonal factors for an extrapolated series

Structural Time Series Modeling and Forecasting

The UCM procedure provides a flexible environment for analyzing time series data using structural time series models, also called unobserved components models (UCM). These models represent the observed series as a sum of suitably chosen components such as trend, seasonal, cyclical, and regression effects. You can use the UCM procedure to formulate comprehensive models that bring out all the salient features of the series under consideration. Structural models are applicable in the same situations where Box-Jenkins ARIMA models are applicable; however, the structural models tend to be more informative about the underlying stochastic structure of the series. The UCM procedure includes the following features:

- general unobserved components modeling where the models can include trend, multiple seasons and cycles, and regression effects
- maximum-likelihood estimation of the model parameters
- model diagnostics that include a variety of goodness-of-fit statistics, and extensive graphical diagnosis of the model residuals
- forecasts and confidence limits for the series and all the model components
- Model-based seasonal decomposition
- extensive plotting capability that includes the following:
  - forecast and confidence interval plots for the series and model components such as trend, cycles, and seasons
  - diagnostic plots such as residual plot, residual autocorrelation plots, and so on
  - seasonal decomposition plots such as trend, trend plus cycles, trend plus cycles plus seasons, and so on
- model-based interpolation of series missing values
- full sample (also called smoothed) estimates of the model components

Time Series Cross-Sectional Regression Analysis

The TSCSREG procedure provides combined time series cross-sectional regression analysis. The TSCSREG procedure includes the following features:

- estimation of the regression parameters under several common error structures:

  - Fuller and Battese method (variance component model)
  - Wansbeek-Kapteyn method
  - Parks method (autoregressive model)
  - Da Silva method (mixed variance component moving-average model)
  - one-way fixed effects
  - two-way fixed effects
  - one-way random effects
  - two-way random effects
- any number of model specifications
- unbalanced panel data for the fixed or random-effects models
- variety of estimates and statistics including the following:
  - underlying error components estimates
  - regression parameter estimates
  - standard errors of estimates
  - *t*-tests
  - R-square statistic
  - correlation matrix of estimates
  - covariance matrix of estimates
  - autoregressive parameter estimate
  - cross-sectional components estimates
  - autocovariance estimates
  - *F* tests of linear hypotheses about the regression parameters
  - specification tests

Automatic Time Series Forecasting

The ESM procedure provides a quick way to generate forecasts for many time series or transactional data in one step by using exponential smoothing methods. All parameters associated with the forecasting model are optimized based on the data.

You can use the following smoothing models:

- simple
- double
- linear
- damped trend

- seasonal
- Winters method (additive and multiplicative)

Additionally, PROC ESM can transform the data before applying the smoothing methods using any of these transformations:

- log
- square root
- logistic
- Box-Cox

In addition to forecasting, the ESM procedure can also produce graphic output.

The ESM procedure can forecast both time series data, whose observations are equally spaced at a specific time interval (for example, monthly, weekly), or transactional data, whose observations are not spaced with respect to any particular time interval. (Internet, inventory, sales, and similar data are typical examples of transactional data. For transactional data, the data are accumulated based on a specified time interval to form a time series.)

The ESM procedure is a replacement for the older FORECAST procedure. ESM is often more convenient to use than PROC FORECAST but it supports only exponential smoothing models.

The FORECAST procedure provides forecasting of univariate time series using automatic trend extrapolation. PROC FORECAST is an easy-to-use procedure for automatic forecasting and uses simple popular methods that do not require statistical modeling of the time series, such as exponential smoothing, time trend with autoregressive errors, and the Holt-Winters method.

The FORECAST procedure supplements the powerful forecasting capabilities of the econometric and time series analysis procedures described previously. You can use PROC FORECAST when you have many series to forecast and you want to extrapolate trends without developing a model for each series.

The FORECAST procedure includes the following features:

- choice of the following forecasting methods:
  - EXPO method—exponential smoothing: single, double, triple, or Holt two-parameter smoothing
  - exponential smoothing as an ARIMA Model
  - WINTERS method—using updating equations similar to exponential smoothing to fit model parameters
  - ADDWINTERS method—like the WINTERS method except that the seasonal parameters are added to the trend instead of multiplied with the trend
  - STEPAR method—stepwise autoregressive models with constant, linear, or quadratic trend and autoregressive errors to any order
  - Holt-Winters forecasting method with constant, linear, or quadratic trend
  - additive variant of the Holt-Winters method

- support for up to three levels of seasonality for Holt-Winters method: time-of-year, day-of-week, or time-of-day
- ability to forecast any number of variables at once
- forecast confidence limits for all methods

Time Series Interpolation and Frequency Conversion

The EXPAND procedure provides time interval conversion and missing value interpolation for time series. The EXPAND procedure includes the following features:

- conversion of time series frequency; for example, constructing quarterly estimates from annual series or aggregating quarterly values to annual values
- conversion of irregular observations to periodic observations
- interpolation of missing values in time series
- conversion of observation types; for example, estimate stocks from flows and vice versa. All possible conversions are supported between any of the following:
  - beginning of period
  - end of period
  - period midpoint
  - period total
  - period average
- conversion of time series phase shift; for example, conversion between fiscal years and calendar years
- identifying observations including the following:
  - identification of the time interval of the input values
  - validation of the input data set observations
  - computation of the ID values for the observations in the output data set
- choice of four interpolation methods:
  - cubic splines
  - linear splines
  - step functions
  - simple aggregation
- ability to perform extrapolation by a linear projection of the trend of the cubic spline curve fit to the input data

- ability to transform series before and after interpolation (or without interpolation) by using any of the following:
  - constant shift or scale
  - sign change or absolute value
  - logarithm, exponential, square root, square, logistic, inverse logistic
  - lags, leads, differences
  - classical decomposition
  - bounds, trims, reverse series
  - centered moving, cumulative, or backward moving average
  - centered moving, cumulative, or backward moving range
  - centered moving, cumulative, or backward moving geometric mean
  - centered moving, cumulative, or backward moving maximum
  - centered moving, cumulative, or backward moving median
  - centered moving, cumulative, or backward moving minimum
  - centered moving, cumulative, or backward moving product
  - centered moving, cumulative, or backward moving corrected sum of squares
  - centered moving, cumulative, or backward moving uncorrected sum of squares
  - centered moving, cumulative, or backward moving rank
  - centered moving, cumulative, or backward moving standard deviation
  - centered moving, cumulative, or backward moving sum
  - centered moving, cumulative, or backward moving median
  - centered moving, cumulative, or backward moving *t*-value
  - centered moving, cumulative, or backward moving variance
- support for a wide range of time series frequencies:
  - YEAR
  - SEMIYEAR
  - QUARTER
  - MONTH
  - SEMIMONTH
  - TENDAY
  - WEEK
  - WEEKDAY
  - DAY
  - HOUR
  - MINUTE
  - SECOND

- support for repeating of shifting the basic interval types to define a great variety of different frequencies, such as fiscal years, biennial periods, work shifts, and so forth

Refer to Chapter 4, "Working with Time Series Data," and Chapter 5, "Date Intervals, Formats, and Functions," for more information about time series data transformations.

Trend and Seasonal Analysis on Transaction Databases

The TIMESERIES procedure can accumulate transactional data to time series and perform trend and seasonal analysis on the accumulated time series.

Time series analyses performed by the TIMESERIES procedure include the follows:

- descriptive statistics relevant for time series data
- seasonal decomposition and seasonal adjustment analysis
- correlation analysis
- cross-correlation analysis

The TIMESERIES procedure includes the following features:

- ability to process large amounts of time-stamped transactional data
- statistical methods useful for large-scale time series analysis or (temporal) data mining
- output data sets stored in either a time series format (default) or a coordinate format (transposed)

The TIMESERIES procedure is normally used to prepare data for subsequent analysis that uses other SAS/ETS procedures or other parts of the SAS system. The time series format is most useful when the data are to be analyzed with SAS/ETS procedures. The coordinate format is most useful when the data are to be analyzed with SAS/STAT® procedures or SAS Enterprise Miner™. (For example, clustering time-stamped transactional data can be achieved by using the results of TIMESERIES procedure with the clustering procedures of SAS/STAT and the nodes of SAS Enterprise Miner.)

Access to Financial and Economic Databases

The DATASOURCE procedure and the SAS/ETS data access interface LIBNAME Engines (SASECRSP, SASEFAME, SASEHAVR and SASEXCCM) provide seamless, efficient access to time series data from data files supplied by a variety of commercial and governmental data vendors.

The DATASOURCE procedure includes the following features:

- support for data files distributed by the following data vendors:
  - DRI/McGraw-Hill

  - FAME Information Services
  - HAVER ANALYTICS
  - Standard & Poors Compustat Service
  - Center for Research in Security Prices (CRSP)
  - International Monetary Fund
  - U.S. Bureau of Labor Statistics
  - U.S. Bureau of Economic Analysis
  - Organization for Economic Cooperation and Development (OECD)
- ability to select the series, frequency, time range, and cross sections of extracted data
- ability to create an output data set containing descriptive information on the series available in the data file
- ability to read EBCDIC data on ASCII systems and vice versa

The SASECRSP interface LIBNAME engine includes the following features:

- enables random access to time series data residing in CRSPAccess databases
- provides a seamless interface between CRSP and SAS data processing
- uses the LIBNAME statement to enable you to specify which time series you would like to read from the CRSPAccess database, and how you would like to perform selection
- enables you access to CRSP Stock, CRSP/COMPUSTAT Merged (CCM) or CRSP Indices Data.
- provides convenient formats, informats, and functions for CRSP and SAS datetime conversions

The SASEFAME interface LIBNAME engine includes the following features:

- provides SAS and FAME users flexibility in accessing and processing time series data, case series, and formulas that reside in either a FAME database or a SAS data set
- provides a seamless interface between FAME and SAS data processing
- uses the LIBNAME statement to enable you to specify which time series you would like to read from the FAME database
- enables you to convert the selected time series to the same time scale
- works with the SAS DATA step to perform further subsetting and to store the resulting time series into a SAS data set
- performs more analysis if desired either in the same SAS session or in another session at a later time
- supports the FAME CROSSLIST function for subsetting via BYGROUPS using the CROSSLIST= option

  - you can use a FAME namelist that contains your BY variables for selection in the CROSSLIST
  - you can use a SAS input dataset, INSET, that contains the BY selection variables along with the WHERE= option in your SASEFAME libref
- supports the use of FAME in a client/server environment that uses the FAME CHLI capability on your FAME server
- enables access to your FAME remote data when you specify the port number of the TCP/IP service that is defined for your FAME server and the node name of your FAME master server in your SASEFAME libref's physical path

The SASEHAVR interface LIBNAME engine includes the following features:

- enables Windows users random access to economic and financial data residing in a HAVER ANALYTICS Data Link Express (DLX) database
- the following types of HAVER data sets are available:
  - United States Economic Indicators
  - Specialized Databases
  - Financial Indicators
  - Industry
  - Industrial Countries
  - Emerging Markets
  - International Organizations
  - Forecasts and As Reported Data
  - United States Regional
- enables you to limit the range of data that is read from the time series
- enables you to specify a desired conversion frequency. Start dates are recommended on the LIBNAME statement to help you save resources when processing large databases or when processing a large number of observations.
- enables you to use the WHERE, KEEP, or DROP statements in your DATA step to further subset your data
- supports use of the SQL procedure to create a view of your resulting SAS data set

The SASEXCCM interface LIBNAME engine includes the following experimental features:

- enables random access to time series data residing in CRSPAccess databases
- provides a seamless interface between CRSP, Compustat XpressFeed and SAS data processing
- uses the LIBNAME statement to enable you to specify which data items, data groups and time series you would like to read from the CRSPAccess database, and how you would like to perform selection

- supports data-item-handling access methods to CRSP Stock (STK), CRSP/COMPUSTAT Merged (CCM), CRSP Indices (IND) or CRSP Treasury (TRS) Data.
- provides selection based on keys such as GVKEY, PERMNO, INDNO, TREASNO, and TCUSIP for efficient access to data items.

Spreadsheet Calculations and Financial Report Generation

The COMPUTAB procedure generates tabular reports using a programmable data table.

The COMPUTAB procedure is especially useful when you need both the power of a programmable spreadsheet and a report-generation system and you want to set up a program to run in batch mode and generate routine reports. The COMPUTAB procedure includes the following features:

- report generation facility for creating tabular reports such as income statements, balance sheets, and other row and column reports for analyzing business or time series data
- ability to tailor report format to almost any desired specification
- use of the SAS programming language to provide complete control of the calculation and format of each item of the report
- ability to report definition in terms of a data table on which programming statements operate
- ability for a single reference to a row or column to bring the entire row or column into a calculation
- ability to create new rows and columns (such as totals, subtotals, and ratios) with a single programming statement
- access to individual table values when needed
- built-in features to provide consolidation reports over summarization variables

Loan Analysis, Comparison, and Amortization

The LOAN procedure provides analysis and comparison of mortgages and other installment loans; it includes the following features:

- ability to specify contract terms for any number of different loans and ability to analyze and compare various financing alternatives
- analysis of four different types of loan contracts including the following:
  - fixed rate
  - adjustable rate
  - buy-down rate

  - balloon payment
- full control over adjustment terms for adjustable rate loans: life caps, adjustment frequency, and maximum and minimum rates
- support for a wide variety of payment and compounding intervals
- ability to incorporate initialization costs, discount points, down payments, and prepayments (uniform or lump-sum) in loan calculations
- analysis of different rate adjustment scenarios for variable rate loans including the following:
  - worst case
  - best case
  - fixed rate case
  - estimated case
- ability to make loan comparisons at different points in time
- ability to make loan comparisons at each analysis date on the basis of five different economic criteria:
  - present worth of cost (net present value of all payments to date)
  - true interest rate (internal rate of return to date)
  - current periodic payment
  - total interest paid to date
  - outstanding balance
- ability to base loan comparisons on either after-tax or before-tax analysis
- report of the best alternative when loans of equal amount are compared
- amortization schedules for each loan contract
- output that shows payment dates, rather than just payment sequence numbers, when starting date is specified
- optional printing or output of the amortization schedules, loan summaries, and loan comparison information to SAS data sets
- ability to specify rounding of payments to any number of decimal places

Time Series Forecasting System

SAS/ETS software includes the Time Series Forecasting System, a point-and-click application for exploring and analyzing univariate time series data. You can use the automatic model selection facility to select the best-fitting model for each time series, or you can use the system's diagnostic features and time series modeling tools interactively to develop forecasting models customized to best predict your time series. The system provides both graphical and statistical features to help you choose the best forecasting method for each series.

The system can be invoked by selecting **Analysis►Solutions**, by the FORECAST command, and by clicking the **Forecasting** icon in the Data Analysis folder of the SAS Desktop.

The following is a brief summary of the features of the Time Series Forecasting system. With the system you can:

- use a wide variety of forecasting methods, including several kinds of exponential smoothing models, Winters method, and ARIMA (Box-Jenkins) models. You can also produce forecasts by combining the forecasts from several models.
- use predictor variables in forecasting models. Forecasting models can include time trend curves, regressors, intervention effects (dummy variables), adjustments you specify, and dynamic regression (transfer function) models.
- view plots of the data, predicted versus actual values, prediction errors, and forecasts with confidence limits. You can plot changes or transformations of series, zoom in on parts of the graphs, or plot autocorrelations.
- use hold-out samples to select the best forecasting method
- compare goodness-of-fit measures for any two forecasting models side-by-side or list all models sorted by a particular fit statistic
- view the predictions and errors for each model in a spreadsheet or view and compare the forecasts from any two models in a spreadsheet
- examine the fitted parameters of each forecasting model and their statistical significance
- control the automatic model selection process: the set of forecasting models considered, the goodness-of-fit measure used to select the best model, and the time period used to fit and evaluate models
- customize the system by adding forecasting models for the automatic model selection process and for point-and-click manual selection
- save your work in a project catalog
- print an audit trail of the forecasting process
- save and print system output including spreadsheets and graphs

Investment Analysis System

The Investment Analysis System is an interactive environment for analyzing the time-value of money for a variety of investments:

- loans
- savings
- depreciations

- bonds
- generic cash flows

Various tools are provided to help analyze the value of investment alternatives: time value, periodic equivalent, internal rate of return, benefit-cost ratio, and breakeven analysis.

These analyses can help answer a number of questions you might have about your investments:

- Which option is more profitable or less costly?
- Is it better to buy or rent?
- Are the extra fees for refinancing at a lower interest rate justified?
- What is the balance of this account after saving this amount periodically for so many years?
- How much is legally tax-deductible?
- Is this a reasonable price?

Investment Analysis can be beneficial to users in many industries for a variety of decisions:

- manufacturing: cost justification of automation or any capital investment, replacement analysis of major equipment, or economic comparison of alternative designs
- government: setting funds for services
- finance: investment analysis and portfolio management for fixed-income securities

ODS Graphics

Many SAS/ETS procedures produce graphical output using the SAS Output Delivery System (ODS). The ODS Graphics system provides several advantages:

- Plots and graphs are output objects in the Output Delivery System (ODS) and can be manipulated with ODS commands.
- There is no need to write SAS/GRAPH statements or use special plotting macros.
- There are multiple formats to choose from: html, gif, and rtf.
- Templates control the appearance of plots.
- Styles control the color scheme.
- You can edit or create templates and styles for all graphs.

To enable graphical output from SAS/ETS procedures, you must use the following statement in your SAS program.

```
ods graphics on;
```

The graphical output produced by many SAS/ETS procedures can be controlled using the PLOTS= option on the PROC statement.

For more information about the features of the ODS Graphics system, including the many ways that you can control or customize the plots produced by SAS procedures, refer to Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*). For more information about the SAS Output Delivery system, refer to the *SAS Output Delivery System: User's Guide*.

Related SAS Software

Many features not found in SAS/ETS software are available in other parts of the SAS System, such as Base SAS®, SAS® Forecast Server, SAS/STAT® software, SAS/OR® software, SAS/QC® software, SAS® Stat Studio, and SAS/IML® software.

If you do not find something you need in SAS/ETS software, you might be able to find it in SAS/STAT software and in Base SAS software. If you still do not find it, look in other SAS software products or contact SAS Technical Support staff.

The following subsections summarize the features of other SAS products that might be of interest to users of SAS/ETS software.

Base SAS Software

The features provided by SAS/ETS software are extensions to the features provided by Base SAS software. Many data management and reporting capabilities you need are part of Base SAS software. Refer to *SAS Language Reference: Dictionary* and *Base SAS Procedures Guide* for documentation of Base SAS software. In particular, refer to *Base SAS Procedures Guide: Statistical Procedures* for information about statistical analysis features included with Base SAS.

The following sections summarize Base SAS software features of interest to users of SAS/ETS software. See Chapter 4, "Working with Time Series Data," for further discussion of some of these topics as they relate to time series data and SAS/ETS software.

SAS DATA Step

The DATA step is your primary tool for reading and processing data in the SAS System. The DATA step provides a powerful general purpose programming language that enables you to perform all kinds of data processing tasks. The DATA step is documented in *SAS Language Reference: Dictionary*.

Base SAS Procedures

Base SAS software includes many useful SAS procedures, which are documented in *Base SAS Procedures Guide* and *Base SAS Procedures Guide: Statistical Procedures*. The following is a list of Base SAS procedures you might find useful:

CATALOG	for managing SAS catalogs
CHART	for printing charts and histograms
COMPARE	for comparing SAS data sets
CONTENTS	for displaying the contents of SAS data sets
COPY	for copying SAS data sets
CORR	for computing correlations
CPORT	for moving SAS data libraries between computer systems
DATASETS	for deleting or renaming SAS data sets
FCMP	for compiling functions for use in SAS programs. The SAS Function Compiler Procedure (FCMP) enables you to create, test, and store SAS functions and subroutines before you use them in other SAS procedures. PROC FCMP accepts slight variations of DATA step statements, and most features of the SAS program ing language can be used in functions and subroutines that are processed by PROC FCMP.
FREQ	for computing frequency crosstabulations
MEANS	for computing descriptive statistics and summarizing or collapsing data over cross sections
PLOT	for printing scatter plots
PRINT	for printing SAS data sets
PROTO	for accessing external functions from the SAS system. The PROTO procedure enables you to register external functions that are written in the C or C++ programming languages. You can use these functions in SAS as well as in C-language structures and types. After the C-language functions are registered in PROC PROTO, they can be called from any SAS function or subroutine that is declared in the FCMP procedure, as well as from any SAS function, subroutine, or method block that is declared in the COMPILE procedure.
RANK	for computing rankings or order statistics
SORT	for sorting SAS data sets
SQL	for processing SAS data sets with Structured Query Language
STANDARD	for standardizing variables to a fixed mean and variance
TABULATE	for printing descriptive statistics in tabular format
TIMEPLOT	for plotting variables over time
TRANSPOSE	for transposing SAS data sets
UNIVARIATE	for computing descriptive statistics

Global Statements

Global statements can be specified anywhere in your SAS program, and they remain in effect until changed. Global statements are documented in *SAS Language Reference: Dictionary*. You may find the following SAS global statements useful:

FILENAME	for accessing data files
FOOTNOTE	for printing footnote lines at the bottom of each page
%INCLUDE	for including files of SAS statements
LIBNAME	for accessing SAS data libraries
OPTIONS	for setting various SAS system options
QUIT	for ending an interactive procedure step
RUN	for executing the preceding SAS statements
TITLE	for printing title lines at the top of each page
X	for issuing host operating system commands from within your SAS session

Some Base SAS statements can be used with any SAS procedure, including SAS/ETS procedures. These statements are not global, and they affect only the SAS procedure they are used with. These statements are documented in *SAS Language Reference: Dictionary*.

The following Base SAS statements are useful with SAS/ETS procedures:

BY	for computing separate analyses for groups of observations
FORMAT	for assigning formats to variables
LABEL	for assigning descriptive labels to variables
WHERE	for subsetting data to restrict the range of data processed or to select or exclude observations from the analysis

SAS Functions

SAS functions can be used in DATA step programs and in the COMPUTAB and MODEL procedures. The following kinds of functions are available:

- character functions for manipulating character strings
- date and time functions for performing date and calendar calculations
- financial functions for performing financial calculations such as depreciation, net present value, periodic savings, and internal rate of return
- lagging and differencing functions for computing lags and differences
- mathematical functions for computing data transformations and other mathematical calculations
- probability functions for computing quantiles of statistical distributions and the significance of test statistics
- random number functions for simulation experiments
- sample statistics functions for computing means, standard deviations, kurtosis, and so forth

SAS functions are documented in *SAS Language Reference: Dictionary*. Chapter 4, "Working with Time Series Data," discusses the use of date, time, lagging, and differencing functions. Chapter 5, "Date Intervals, Formats, and Functions," contains a reference list of date and time functions.

Formats, Informats, and Time Intervals

Base SAS software provides formats to control the printing of data values, informats to read data values, and time intervals to define the frequency of time series. See Chapter 5, "Date Intervals, Formats, and Functions," for more information.

SAS Forecast Studio

SAS Forecast Studio is part of the SAS Forecast Server product. It provides an interactive environment for modeling and forecasting very large collections of hierarchically organized time series, such as SKUs in product lines and sales regions of a retail business. Forecast Studio greatly extends the capabilities provided by the Time Series Forecasting System included with SAS/ETS and described in *Part IV*.

Forecast Studio is documented in *SAS Forecast Studio User's Guide*.

SAS High-Performance Forecasting

SAS High-Performance Forecasting (HPF) software provides a system of SAS procedures for large-scale automatic forecasting in business, government, and academic applications. Major uses of High-Performance Forecasting procedures include: forecasting, forecast scoring, market response modeling, and time series data mining.

The software includes the following automatic forecasting process:

- accumulates the time-stamped data to form a fixed-interval time series
- diagnoses the time series using time series analysis techniques
- creates a list of candidate model specifications based on the diagnostics
- fits each candidate model specification to the time series
- generates forecasts for each candidate fitted model
- selects the most appropriate model specification based on either in-sample or holdout-sample evaluation using a model selection criterion
- refits the selected model specification to the entire range of the time series
- creates a forecast score from the selected fitted model
- generate forecasts from the forecast score
- evaluates the forecast using in-sample analysis
- provides for out-of-sample forecast performance analysis
- performs top-down, middle-out, or bottom-up reconciliations of forecasts in the hierarchy

SAS/GRAPH Software

SAS/GRAPH software includes procedures that create two- and three-dimensional high resolution color graphics plots and charts. You can generate output that graphs the relationship of data values to one another, enhance existing graphs, or simply create graphics output that is not tied to data.

With the addition of ODS Graphics features to SAS/ETS procedures, there is now less need for the use of SAS/GRAPH procedures with SAS/ETS. However, SAS/GRAPH procedures allow you to create additional graphical displays of your results.

SAS/GRAPH software can produce the following types of output:

- charts
- plots
- maps
- text
- three-dimensional graphs

With SAS/GRAPH software you can produce high-resolution color graphics plots of time series data.

SAS/STAT Software

SAS/STAT software is of interest to users of SAS/ETS software because many econometric and other statistical methods not included in SAS/ETS software are provided in SAS/STAT software.

SAS/STAT software includes procedures for a wide range of statistical methodologies including the following:

- logistic regression
- censored regression
- principal component analysis
- structural equation models using covariance structure analysis
- factor analysis
- survival analysis
- discriminant analysis
- cluster analysis
- categorical data analysis; log-linear and conditional logistic models
- general linear models

- mixed linear and nonlinear models
- generalized linear models
- response surface analysis
- kernel density estimation
- LOESS regression
- spline regression
- two-dimensional kriging
- multiple imputation for missing values
- survey data analysis

SAS/IML Software

SAS/IML software gives you access to a powerful and flexible programming language (Interactive Matrix Language) in a dynamic, interactive environment. The fundamental object of the language is a data matrix. You can use SAS/IML software interactively (at the statement level) to see results immediately, or you can store statements in a module and execute them later. The programming is dynamic because necessary activities such as memory allocation and dimensioning of matrices are done automatically.

You can access built-in operators and call routines to perform complex tasks such as matrix inversion or eigenvector generation. You can define your own functions and subroutines using SAS/IML modules. You can perform operations on an entire data matrix. You have access to a wide choice of data management commands. You can read, create, and update SAS data sets from inside SAS/IML software without ever using the DATA step.

SAS/IML software is of interest to users of SAS/ETS software because it enables you to program your own econometric and time series methods in the SAS System. It contains subroutines for time series operators and for general function optimization. If you need to perform a statistical calculation not provided as an automated feature by SAS/ETS or other SAS software, you can use SAS/IML software to program the matrix equations for the calculation.

Kalman Filtering and Time Series Analysis in SAS/IML

SAS/IML software includes CALL routines and functions for Kalman filtering and time series analysis, which perform the following:

- generate univariate, multivariate, and fractional time series
- compute likelihood function of ARMA, VARMA, and ARFIMA models
- compute an autocovariance function of ARMA, VARMA, and ARFIMA models
- check the stationarity of ARMA and VARMA models

- filter and smooth time series models using Kalman method
- fit AR, periodic AR, time-varying coefficient AR, VAR, and ARFIMA models
- handle Bayesian seasonal adjustment models

SAS/IML Stat Studio

SAS/IML Studio is a highly interactive tool for data exploration and analysis. SAS/IML Studio runs on a PC in the Microsoft Windows operating environment. You can use SAS/IML Studio to do the following:

- explore data through graphs linked across multiple windows
- transform data
- subset data
- analyze univariate distributions
- discover structure and features in multivariate data
- fit and evaluate explanatory models
- create your own customized statistical graphics
- add legends, curves, maps, or other custom features to statistical graphics
- develop interactive programs that use dialog boxes
- extend the built-in analyses by calling SAS procedures
- create custom analyses
- repeat an analysis on different data
- extend the results of SAS procedures by using IML
- share analyses with colleagues who also use SAS/IML Studio
- call functions from libraries written in R, C/C++, FORTRAN, or Java

See *SAS/IML Studio User's Guide* for more information.

SAS/OR Software

SAS/OR software provides SAS procedures for operations research and project planning and includes a menu driven system for project management. SAS/OR software has features for the following:

- solving transportation problems
- linear, integer, and mixed-integer programming
- nonlinear programming and optimization
- scheduling projects
- plotting Gantt charts
- drawing network diagrams
- solving optimal assignment problems
- network flow programming

SAS/OR software might be of interest to users of SAS/ETS software for its mathematical programming features. In particular, the NLP and OPTMODEL procedures in SAS/OR software solve nonlinear programming problems and can be used for constrained and unconstrained maximization of user-defined likelihood functions.

See *SAS/OR User's Guide: Mathematical Programming* for more information.

SAS/QC Software

SAS/QC software provides a variety of procedures for statistical quality control and quality improvement. SAS/QC software includes procedures for the following:

- Shewhart control charts
- cumulative sum control charts
- moving average control charts
- process capability analysis
- Ishikawa diagrams
- Pareto charts
- experimental design

SAS/QC software also includes the SQC menu system for interactive application of statistical quality control methods and the ADX Interface for experimental design.

MLE for User-Defined Likelihood Functions

There are several SAS procedures that enable you to do maximum likelihood estimation of parameters in an arbitrary model with a likelihood function that you define: PROC MODEL, PROC NLP, PROC OPTMODEL and PROC IML.

The MODEL procedure in SAS/ETS software enables you to minimize general log-likelihood functions for the error term of a model.

The NLP and OPTMODEL procedures in SAS/OR software are general nonlinear programming procedures that can maximize a general function subject to linear equality or inequality constraints. You can use PROC NLP or OPTMODEL to maximize a user-defined nonlinear likelihood function.

You can use the IML procedure in SAS/IML software for maximum likelihood problems. The optimization routines used by PROC NLP are available through IML subroutines. You can write the likelihood function in the SAS/IML matrix language and call the constrained and unconstrained nonlinear programming subroutines to maximize the likelihood function with respect to the parameter vector.

JMP® Software

JMP software uses a flexible graphical interface to display and analyze data. JMP dynamically links statistics and graphics so you can easily explore data, make discoveries, and gain the knowledge you need to make better decisions. JMP provides a comprehensive set of statistical tools as well as design of experiments (DOE) and advanced quality control (QC and SPC) tools for Six Sigma in a single package. JMP is software for interactive statistical graphics and includes:

- a data table window for editing, entering, and manipulating data
- a broad range of graphical and statistical methods for data analysis
- a facility for grouping data and computing summary statistics
- JMP scripting language (JSL)—a scripting language for saving and creating frequently used routines
- JMP automation
- Formula Editor—a formula editor for each table column to compute values as needed
- linear models, correlations, and multivariate
- design of experiments module
- options to highlight and display subsets of data
- statistical quality control and variability charts—special plots, charts, and communication capability for quality-improvement techniques
- survival analysis
- time series analysis, which includes the following:

  - Box-Jenkins ARIMA forecasting
  - seasonal ARIMA forecasting
  - transfer function modeling
  - smoothing models: Winters method, single, double, linear, damped trend linear, and seasonal exponential smoothing
  - diagnostic charts (autocorrelation, partial autocorrelation, and variogram) and statistics of fit
  - a model comparison table to compare all forecasts generated
  - spectral density plots and white noise tests
- tools for printing and for moving analyses results between applications

SAS Enterprise Guide®

SAS Enterprise Guide has the following features:

- integration with the SAS9 platform:
  - open metadata repository (OMR) integration
  - SAS report integration
    - create report interface
    - ODS support
    - Web report studio integration
  - access to information maps
  - ETL studio impact analysis
  - ESRI integration within the OLAP analyzer
  - data mining scoring task
- the user interface and workflow
  - process flow
  - ability to create stored processes from process flows
  - SAS folders window
  - project parameters
  - query builder interface
  - code node
  - OLAP analyzer
    - ESRI integration
    - tree-diagram-based OLAP explorer
    - SAS report snapshots
    - SAS Web OLAP viewer for .NET ability to create EG projects
  - workspace maximization

With Enterprise Guide, you can perform time series analysis with the following EG procedures:

- prepare time series data—the Prepare Time Series Data task can be used to make data more suitable for analysis by other time series tasks.
- create time series data—the Create Time Series Data wizard helps you convert transactional data into fixed-interval time series. Transactional data are time-stamped data collected over time with irregular or varied frequency.
- ARIMA Modeling and Forecasting task
- Basic Forecasting task
- Regression Analysis with Autoregressive Errors
- Regression Analysis of Panel Data

SAS® Add-In for Microsoft Office

The main time series tasks in SAS Add-in for Microsoft Office (AMO) are as follows:

- Prepare Time Series Data
- Basic Forecasting
- ARIMA Modeling and Forecasting
- Regression Analysis with Autoregressive Errors
- Regression Analysis of Panel Data
- Create Time Series Data
- Forecast Studio Create Project
- Forecast Studio Open Project
- Forecast Studio Submit Overrides

SAS Enterprise Miner™—Time Series Node

SAS Enterprise Miner™ is the SAS solution for data mining, streamlining the data mining process to create highly accurate predictive and descriptive models. Enterprise Miner's process flow diagram eliminates the need for manual coding and reduces the model development time for both business analysts and statisticians. The system is customizable and extensible; users can integrate their code and build new nodes for redistribution.

The Time Series node is a method of investigating time series data. It belongs to the Modify category of the SAS SEMMA (sample, explore, modify, model, assess) data mining process. The Time Series node enables you to understand trends and seasonal variation in large amounts of time series and transactional data.

The Time Series node in SAS Enterprise Miner enables you to do the following:

- perform time series analysis
- perform forecasting
- work with transactional data

SAS Risk Products

The SAS Risk products include SAS Risk Dimensions®, SAS Credit Risk Management for Banking, SAS OpRisk VaR, and SAS OpRisk Monitor.

The analytical methods of SAS Risk Dimensions measure market risk and credit risk. SAS Risk Dimensions creates an environment where market and position data are staged for analysis using SAS data access and warehousing methodologies. SAS Risk Dimensions delivers a full range of modern credit, market and operational risk analysis techniques including:

- mark-to-market
- scenario analysis
- profit/loss curves and surfaces
- sensitivity analysis
- delta normal VaR
- historical simulation VaR
- Monte Carlo VaR
- current exposure
- potential exposure
- credit VaR
- optimization

SAS Credit Risk Management for Banking is a complete end-to-end application for measuring, exploring, managing, and reporting credit risk. SAS Credit Risk Management for Banking integrates data access, mapping, enrichment, and aggregation with advanced analytics and flexible reporting, all in an open, extensible, client-server framework.

SAS Credit Risk Management for Banking enables you to do the following:

- access and aggregate credit risk data across disparate operating systems and sources
- seamlessly integrate credit scoring/internal rating with credit portfolio risk assessment
- accurately measure, monitor, and report potential credit risk exposures within entities of an organization and aggregated across the entire organization, both on the counterparty level and the portfolio level
- evaluate alternative strategies for pricing, hedging, or transferring credit risk
- optimize the allocation of credit risk mitigants or assign the mitigants to lower the regulatory capital requirement
- optimize the allocation of regulatory capital and economic capital
- facilitate regulatory compliance and risk disclosure requirements for a wide variety of regulations such as Basel I, Basel II, and the Capital Requirements Directive (CAD III)

References

Amal, S., and Weselowski, R. (1993). "Practical Econometric Analysis for Assessment of Real Property Using the SAS System on Personal Computers." In *Proceedings of the Eighteenth Annual SAS Users Group International Conference*, 385–390. Cary, NC: SAS Institute Inc. http://www.sascommunity.org/sugi/SUGI93/Sugi-93-62%20Amal%20Weselowski.pdf.

Benseman, B. R. (1990). "Better Forecasting with SAS/ETS Software." In *Proceedings of the Fifteenth Annual SAS Users Group International Conference*, 494–497. Cary, NC: SAS Institute Inc. http://www.sascommunity.org/sugi/SUGI90/Sugi-90-74%20Benseman.pdf.

Calise, A., and Earley, J. (1997). "Forecasting College Enrollment Using the SAS System." In *Proceedings of the Twenty-Second Annual SAS Users Group International Conference*, 1326–1329. Cary, NC: SAS Institute Inc. http://www2.sas.com/proceedings/sugi22/STATS/PAPER282.PDF.

Earley, J., Sweeney, S. J., and Zekavat, S. M. (1989). "PROC ARIMA and the Dow Jones Stock Index." In *Proceedings of the Fourteenth Annual SAS Users Group International Conference*, 371–375. Cary, NC: SAS Institute Inc. http://www.sascommunity.org/sugi/SUGI89/Sugi-89-57%20Earley%20Sweeney%20Zekavat.pdf.

Fischetti, T., Heathcote, S., and Perry, D. (1993). "Using SAS to Create a Modular Forecasting System." In *Proceedings of the Eighteenth Annual SAS Users Group International Conference*, 580–585. Cary, NC: SAS Institute Inc. http://www.sascommunity.org/sugi/SUGI93/Sugi-93-93%20Fischetti%20Heathcote%20Perry.pdf.

Fleming, N. S., Gibson, E., and Fleming, D. G. (1996). "The Use of PROC ARIMA to Test an Intervention Effect." In *Proceedings of the Twenty-First Annual SAS Users Group International Conference*, 1317–1326. Cary, NC: SAS Institute Inc. http://www.sascommunity.org/sugi/SUGI96/Sugi-96-220%20Fleming%20Gibson%20Fleming.pdf.

Hisnanick, J. J. (1991). "SAS/ETS in Applied Econometrics: Evaluating Input Separability in a Model of the U.S. Manufacturing Sector." In *Proceedings of the Sixteenth Annual SAS Users Group International Conference*, 688–693. Cary, NC: SAS Institute Inc. http://www.sascommunity.org/sugi/SUGI91/Sugi-91-116%20Hisnanick.pdf.

Hisnanick, J. J. (1992). "Using PROC ARIMA in Forecasting the Demand and Utilization of Inpatient Hospital Services." In *Proceedings of the Seventeenth Annual SAS Users Group International Conference*, 383–391. Cary, NC: SAS Institute Inc. http://www.sascommunity.org/sugi/SUGI92/Sugi-92-64%20Hisnanick.pdf.

Hisnanick, J. J. (1993). "Using SAS/ETS in Applied Econometrics: Parameters Estimates for the CES-Translog Specification." In *Proceedings of the Eighteenth Annual SAS Users Group International Conference*, 275–279. Cary, NC: SAS Institute Inc. http://www.sascommunity.org/sugi/SUGI93/Sugi-93-46%20Hisnanick.pdf.

Hoyer, K. K., and Gross, K. C. (1993). "Spectral Decomposition and Reconstruction of Nuclear Plant Signals." In *Proceedings of the Eighteenth Annual SAS Users Group International Conference*, 1153–1158. Cary, NC: SAS Institute Inc. http://www.sascommunity.org/sugi/SUGI93/Sugi-93-193%20Hoyer%20Gross.pdf.

Keshani, D. A., and Taylor, T. N. (1992). "Weather Sensitive Appliance Load Curves; Conditional Demand Estimation." In *Proceedings of the Seventeenth Annual SAS Users Group International Conference*, 422–430. Cary, NC: SAS Institute Inc. http://www.sascommunity.org/sugi/SUGI92/Sugi-92-69%20Keshani%20Taylor.pdf.

Khan, M. H. (1990). "Transfer Function Model for Gloss Prediction of Coated Aluminum Using the ARIMA Procedure." In *Proceedings of the Fifteenth Annual SAS Users Group International Conference*, 517–522. Cary, NC: SAS Institute Inc. http://www.sascommunity.org/sugi/SUGI90/Sugi-90-77%20Khan.pdf.

LeBouton, K. J. (1989). "Performance Function for Aircraft Production Using PROC SYSLIN and L^2 Norm Estimation." In *Proceedings of the Fourteenth Annual SAS Users Group International Conference*, 424–426. Cary, NC: SAS Institute Inc. http://www.sascommunity.org/sugi/SUGI89/Sugi-89-66%20LeBouton.pdf.

Lin, L.-H., and Myers, S. C. (1988). "Forecasting the Economy Using the Composite Leading Index, Its Components, and a Rational Expectations Alternative." In *Proceedings of the Thirteenth Annual SAS Users Group International Conference*, 181–186. Cary, NC: SAS Institute Inc. http://www.sascommunity.org/sugi/SUGI88/Sugi-13-34%20Lin%20Myers.pdf.

McCarty, L. (1994). "Forecasting Operational Indices Using SAS/ETS Software." In *Proceedings of the Nineteenth Annual SAS Users Group International Conference*, 844–848. Cary, NC: SAS Institute Inc. http://www.sascommunity.org/sugi/SUGI94/Sugi-94-147%20McCarty.pdf.

Morelock, M. M., Pargellis, C. A., Graham, E. T., Lamarre, D., and Jung, G. (1995). "Time-Resolved Ligand Exchange Reactions: Kinetic Models for Competitive Inhibitors with Recombinant Human Renin." *Journal of Medical Chemistry* 38:1751–1761.

Parresol, B. R., and Thomas, C. E. (1991). "Econometric Modeling of Sweetgum Stem Biomass Using the IML and SYSLIN Procedures." In *Proceedings of the Sixteenth Annual SAS Users Group International Conference*, 694–699. Cary, NC: SAS Institute Inc. http://www.sascommunity.org/sugi/SUGI91/Sugi-91-117%20Parresol%20Thomas.pdf.

Chapter 3
Shared Concepts and Topics

Contents

Overview **61**
Processing Modes **62**
Single-Machine Mode 62
Distributed Mode 62
Controlling the Execution Mode with Environment Variables and Performance Statement Options 62
Determining Single-Machine Mode or Distributed Mode 63
Data Access Modes **67**
Single-Machine Data Access Mode 67
Distributed Data Access Mode 68
Determining the Data Access Mode 69
Alongside-the-Database Execution **70**
Alongside-LASR Distributed Execution **72**
Running High-Performance Analytical Procedures Alongside a SAS LASR Analytic Server in Distributed Mode **73**
Starting a SAS LASR Analytic Server Instance 73
Associating a SAS Libref with the SAS LASR Analytic Server Instance 74
Running a High-Performance Analytical Procedure Alongside the SAS LASR Analytic Server Instance 74
Terminating a SAS LASR Analytic Server Instance 75
Alongside-LASR Distributed Execution on a Subset of the Appliance Nodes **75**
Running High-Performance Analytical Procedures in Asymmetric Mode **76**
Running in Asymmetric Mode on Distinct Appliances 76
Alongside-HDFS Execution **79**
Alongside-HDFS Execution by Using the SASHDAT Engine 79
Alongside-HDFS Execution by Using the Hadoop Engine 81
Output Data Sets **85**
Working with Formats **85**
PERFORMANCE Statement **87**

Overview

This chapter describes the modes of execution in which SAS high-performance analytical procedures can execute. If you have SAS/ETS installed, you can run any procedure in this book on a single machine.

However, to run procedures in this book in distributed mode, you must also have SAS High-Performance Econometrics software installed. For more information about these modes, see the next section.

This chapter provides details of how you can control the modes of execution and includes the syntax for the PERFORMANCE statement, which is common to all high-performance analytical procedures.

Processing Modes

Single-Machine Mode

Single-machine mode is a computing model in which multiple processors or multiple cores are controlled by a single operating system and can access shared resources, such as disks and memory. In this book, single-machine mode refers to an application running multiple concurrent threads on a multicore machine in order to take advantage of parallel execution on multiple processing units. More simply, single-machine mode for high-performance analytical procedures means multithreading on the client machine.

All high-performance analytical procedures are capable of running in single-machine mode, and this is the default mode when a procedure runs on the client machine. The procedure uses the number of CPUs (cores) on the machine to determine the number of concurrent threads. High-performance analytical procedures use different methods to map core count to the number of concurrent threads, depending on the analytic task. Using one thread per core is not uncommon for the procedures that implement data-parallel algorithms.

Distributed Mode

Distributed mode is a computing model in which several nodes in a distributed computing environment participate in the calculations. In this book, the distributed mode of a high-performance analytical procedure refers to the procedure performing the analytics on an appliance that consists of a cluster of nodes. This appliance can be one of the following:

- a database management system (DBMS) appliance on which the SAS High-Performance Analytics infrastructure is also installed
- a cluster of nodes that have the SAS High-Performance Analytics infrastructure installed but no DBMS software installed

Controlling the Execution Mode with Environment Variables and Performance Statement Options

You control the execution mode by using environment variables or by specifying options in the PERFORMANCE statement in high-performance analytical procedures, or by a combination of these methods.

The important environment variables follow:

- *grid host* identifies the domain name system (DNS) or IP address of the appliance node to which the SAS High-Performance Econometrics software connects to run in distributed mode.
- *installation location* identifies the directory where the SAS High-Performance Econometrics software is installed on the appliance.

You can set an environment variable directly from the SAS program by using the OPTION SET= command. For example, the following statements define the grid host and the location where the SAS High-Performance software is installed on the appliance:

```
option set=GRIDHOST      ="hpa.sas.com";
option set=GRIDINSTALLLOC="/opt/TKGrid";
```

Alternatively, you can set the parameters in the PERFORMANCE statement in high-performance analytical procedures. For example:

```
performance host      ="hpa.sas.com"
            install   ="/opt/TKGrid";
```

A specification in the PERFORMANCE statement overrides a specification of an environment variable without resetting its value. An environment variable that you set in the SAS session by using an OPTION SET= command remains in effect until it is modified or until the SAS session terminates.

The key variable that determines whether a high-performance analytical procedure executes in single-machine or distributed mode is the *grid host*. The installation location is needed to ensure that a connection to the grid host can be made, given that a host is specified. This book assumes that the installation location has been set by your system administrator.

The following sets of SAS statements are functionally equivalent:

```
proc hpreduce;
   reduce unsupervised x:;
   performance host="hpa.sas.com";
run;

option set=GRIDHOST="hpa.sas.com";
proc hpreduce;
   reduce unsupervised x:;
run;
```

Determining Single-Machine Mode or Distributed Mode

High-performance analytical procedures use the following rules to determine whether they run in single-machine mode or distributed mode:

- If a grid host is not specified, the analysis is carried out in single-machine mode on the client machine that runs the SAS session.

- If a grid host is specified, the behavior depends on whether the execution is alongside the database or alongside HDFS. If the data are local to the client (that is, not stored in the distributed database or HDFS on the appliance), you need to use the NODES= option in the PERFORMANCE statement to specify the number of nodes on the appliance or cluster that you want to engage in the analysis. If the procedure executes alongside the database or alongside HDFS, you do not need to specify the NODES= option.

The following example shows single-machine and client-data distributed configurations for a data set of 100,000 observations that are simulated from a logistic regression model. The following DATA step generates the data:

```
data simData;
   array _a{8} _temporary_ (0,0,0,1,0,1,1,1);
   array _b{8} _temporary_ (0,0,1,0,1,0,1,1);
   array _c{8} _temporary_ (0,1,0,0,1,1,0,1);
   do obsno=1 to 100000;
      x  = rantbl(1,0.28,0.18,0.14,0.14,0.03,0.09,0.08,0.06);
      a  = _a{x};
      b  = _b{x};
      c  = _c{x};
      x1 = int(ranuni(1)*400);
      x2 = 52 + ranuni(1)*38;
      x3 = ranuni(1)*12;
      lp = 6. -0.015*(1-a) + 0.7*(1-b) + 0.6*(1-c) + 0.02*x1 -0.05*x2 - 0.1*x3;
      y  = ranbin(1,1,(1/(1+exp(lp))));
      output;
   end;
   drop x lp;
run;
```

The following statements run PROC HPLOGISTIC to fit a logistic regression model:

```
proc hplogistic data=simData;
   class a b c;
   model y = a b c x1 x2 x3;
run;
```

Figure 3.1 shows the results from the analysis.

Figure 3.1 Results from Logistic Regression in Single-Machine Mode

The HPLOGISTIC Procedure

Performance Information	
Execution Mode	Single-Machine
Number of Threads	4

Data Access Information			
Data	**Engine**	**Role**	**Path**
WORK.SIMDATA	V9	Input	On Client

Figure 3.1 *continued*

Model Information	
Data Source	WORK.SIMDATA
Response Variable	y
Class Parameterization	GLM
Distribution	Binary
Link Function	Logit
Optimization Technique	Newton-Raphson with Ridging

Parameter Estimates

Parameter	Estimate	Standard Error	DF	t Value	Pr > |t|
Intercept	5.7011	0.2539	Infty	22.45	<.0001
a 0	-0.01020	0.06627	Infty	-0.15	0.8777
a 1	0	.	.	.	.
b 0	0.7124	0.06558	Infty	10.86	<.0001
b 1	0	.	.	.	.
c 0	0.8036	0.06456	Infty	12.45	<.0001
c 1	0	.	.	.	.
x1	0.01975	0.000614	Infty	32.15	<.0001
x2	-0.04728	0.003098	Infty	-15.26	<.0001
x3	-0.1017	0.009470	Infty	-10.74	<.0001

The entries in the "Performance Information" table show that the HPLOGISTIC procedure runs in single-machine mode and uses four threads, which are chosen according to the number of CPUs on the client machine. You can force a certain number of threads on any machine that is involved in the computations by specifying the NTHREADS option in the PERFORMANCE statement. Another indication of execution on the client is the following message, which is issued in the SAS log by all high-performance analytical procedures:

```
NOTE: The HPLOGISTIC procedure is executing in single-machine mode.
```

The following statements use 10 nodes (in distributed mode) to analyze the data on the appliance; results appear in Figure 3.2:

```
proc hplogistic data=simData;
   class a b c;
   model y = a b c x1 x2 x3;
   performance host="hpa.sas.com" nodes=10;
run;
```

Figure 3.2 Results from Logistic Regression in Distributed Mode

The HPLOGISTIC Procedure

Performance Information	
Host Node	hpa.sas.com
Execution Mode	Distributed
Number of Compute Nodes	10
Number of Threads per Node	24

Data Access Information			
Data	**Engine**	**Role**	**Path**
WORK.SIMDATA	V9	Input	From Client

Model Information	
Data Source	WORK.SIMDATA
Response Variable	y
Class Parameterization	GLM
Distribution	Binary
Link Function	Logit
Optimization Technique	Newton-Raphson with Ridging

Parameter Estimates

Parameter	Estimate	Standard Error	DF	t Value	Pr > |t|
Intercept	5.7011	0.2539	Infty	22.45	<.0001
a 0	-0.01020	0.06627	Infty	-0.15	0.8777
a 1	0	.	.	.	.
b 0	0.7124	0.06558	Infty	10.86	<.0001
b 1	0	.	.	.	.
c 0	0.8036	0.06456	Infty	12.45	<.0001
c 1	0	.	.	.	.
x1	0.01975	0.000614	Infty	32.15	<.0001
x2	-0.04728	0.003098	Infty	-15.26	<.0001
x3	-0.1017	0.009470	Infty	-10.74	<.0001

The specification of a host causes the "Performance Information" table to display the name of the host node of the appliance. The "Performance Information" table also indicates that the calculations were performed in a distributed environment on the appliance. Twenty-four threads on each of 10 nodes were used to perform the calculations—for a total of 240 threads.

Another indication of distributed execution on the appliance is the following message, which is issued in the SAS log by all high-performance analytical procedures:

```
NOTE: The HPLOGISTIC procedure is executing in the distributed
      computing environment with 10 worker nodes.
```

You can override the presence of a grid host and force the computations into single-machine mode by specifying the NODES=0 option in the PERFORMANCE statement:

```
proc hplogistic data=simData;
   class a b c;
   model y = a b c x1 x2 x3;
   performance host="hpa.sas.com" nodes=0;
run;
```

Figure 3.3 shows the "Performance Information" table. The numeric results are not reproduced here, but they agree with the previous analyses, which are shown in Figure 3.1 and Figure 3.2.

Figure 3.3 Single-Machine Mode Despite Host Specification

The HPLOGISTIC Procedure

Performance Information	
Execution Mode	Single-Machine
Number of Threads	4

Data Access Information			
Data	**Engine**	**Role**	**Path**
WORK.SIMDATA	V9	Input	On Client

The "Performance Information" table indicates that the HPLOGISTIC procedure executes in single-machine mode on the client. This information is also reported in the following message, which is issued in the SAS log:

```
NOTE: The HPLOGISTIC procedure is executing in single-machine mode.
```

In the analysis shown previously in Figure 3.2, the data set Work.simData is local to the client, and the HPLOGISTIC procedure distributed the data to 10 nodes on the appliance. The High-Performance Analytics infrastructure does not keep these data on the appliance. When the procedure terminates, the in-memory representation of the input data on the appliance is freed.

When the input data set is large, the time that is spent sending client-side data to the appliance might dominate the execution time. In practice, transfer speeds are usually lower than the theoretical limits of the network connection or disk I/O rates. At a transfer rate of 40 megabytes per second, sending a 10-gigabyte data set to the appliance requires more than four minutes. If analytic execution time is in the range of seconds, the "performance" of the process is dominated by data movement.

The alongside-the-database execution model, unique to high-performance analytical procedures, enables you to read and write data in distributed form from the database that is installed on the appliance.

Data Access Modes

Single-Machine Data Access Mode

When high-performance analytical procedures run in single-machine mode, they access data in the same way as traditional SAS procedures. They use Base SAS to access input and output SAS data sets on the

client machine, and they use the relevant SAS/ACCESS interface to bring data from other sources, such as third-party databases, Hadoop, and SAS LASR servers, to the client.

Distributed Data Access Mode

When high-performance analytical procedures run in distributed mode, input data must be brought to the computation that is performed on the nodes of the grid, and output data must be sent from the computational nodes. This can be accomplished in several ways:

- Client-data (local-data) mode: The input and output data for the analytic task are stored on the client machine where the high-performance procedure is invoked. When the procedure runs, the SAS High-Performance Analytics infrastructure sends input data from the client to the distributed computing environment and sends output data from the distributed computing environment to the client.
- Parallel symmetric mode: Input and output data are stored on the same nodes that are used for the distributed computation, and the data move in parallel from the data store to the computational nodes without crossing node boundaries. Parallel symmetric mode is available with the following distributed data sources:
  - Data in Greenplum databases that are collocated with the computational nodes. This access mode is also called alongside-the-database mode. For more information, see the section "Alongside-the-Database Execution" on page 70.
  - Data in SASHDAT format in the Hadoop Distributed File System (HDFS) that is collocated with the computational nodes. This access mode is also called alongside-HDFS mode. For more information, see the section "Alongside-HDFS Execution by Using the SASHDAT Engine" on page 79.
  - Data in a SAS LASR Analytic Server that is collocated with the computational nodes. This access mode is also called alongside-LASR mode. For more information, see the section "Running High-Performance Analytical Procedures Alongside a SAS LASR Analytic Server in Distributed Mode" on page 73.
- Parallel asymmetric mode: The primary reason for providing this mode is to enable you to manage and house data on appliances (the data appliances) and to run high-performance analytical procedures on a different appliance (the computing appliance). The high-performance analytical procedures run in a SAS process on the computing appliance. For each data source that is accessed in parallel asymmetric mode, a SAS Embedded Process must run on the associated data appliance. Data are requested by a SAS data feeder that runs on the computing appliance and communicates with the SAS Embedded Process on the data appliance. The SAS Embedded Process transfers the data in parallel to the SAS data feeder that runs on each of the nodes of the computing appliance. This mode is called asymmetric mode because the number of nodes on the data appliance does not need to match the number of nodes on the computing appliance. Parallel asymmetric mode is supported for data in Teradata, Greenplum, and Oracle databases and for data in HDFS and SAP HANA. In these cases, the parallel asymmetric access is somewhat loosely described as being asymmetric alongside access, even though the data storage and computation can occur on different appliances. For more information, see the section "Running High-Performance Analytical Procedures in Asymmetric Mode" on page 76.

- Through-the-client mode: When data can be accessed through a SAS/ACCESS interface but the data reside in a file system or in a distributed data source on which a SAS Embedded Process is not running, those data cannot be accessed in parallel in either symmetric or asymmetric mode. The SAS/ACCESS interface is used to transfer input data from the data source to the client machine on which the high-performance procedure is invoked, and the data are then sent to the distributed computing environment by the SAS High-Performance Analytics infrastructure. The data path is reversed for output data. This mode of data access is referred to as through-the-client access.

Determining the Data Access Mode

High-performance analytical procedures determine the data access mode individually for each data set that is used in the analysis. When high-performance analytical procedures run in distributed mode, parallel symmetric or parallel asymmetric mode is used whenever possible. There are two reasons why parallel access might not be possible. The first reason is that for a particular data set, the required SAS Embedded Process is not installed on the appliance that houses the data. In such cases, access to those data reverts to through-the-client access, and a note like the following is reported in the SAS log:

```
NOTE: The data MYLIB.MYDATA are being routed through the client because a
      SAS Embedded Process is not running on the associated data server.
```

The second reason why parallel data access might not be possible for a particular data set is that the required driver software might not be installed on the compute nodes. In this case, the required data feeder that moves the data from the compute nodes to the data source cannot be successfully loaded, and a note like the following is reported in the SAS log:

```
NOTE: The data MYLIB.MYDATA are being routed through the client because
      the ORACLE data feeder could not be loaded on the specified grid host.
```

For distributed data in SASHDAT format in HDFS or data in a SAS LASR Analytic Server, parallel symmetric access is used when the data nodes and compute nodes are collocated on the same appliance. For data in a LASR Analytic Server that cannot be accessed in parallel symmetric mode, through-the-client mode is used. Through-the-client access is not supported for data in SASHDAT format in HDFS.

For data in Greenplum databases, parallel symmetric access is used if the compute nodes and the data nodes are collocated on the same appliance and you do not specify the NODES=*n* option in a PERFORMANCE statement. In this case, the number of nodes that are used is determined by the number of nodes across which the data are distributed. If you specify NODES=*n*, then parallel asymmetric access is used.

High-performance analytical procedures produce a “Data Access Information” table that shows you how each data set that is used in the analysis is accessed. The following statements provide an example in which PROC HPDS2 is used to copy a distributed data set named Neuralgia (which is stored in SASHDAT format in HDFS) to a SAS data set on the client machine:

```
libname hdatlib sashdat
                host='hpa.sas.com';
                hdfs_path="/user/hps";

proc hpds2 data=hdatlib.neuralgia out=neuralgia;
```

```
   performance host='hpa.sas.com';
   data DS2GTF.out;
      method run();
     set DS2GTF.in;
   end;
  enddata;
run;
```

Figure 3.4 shows the output that PROC HPDS2 produces. The "Performance Information" table shows that PROC HPDS2 ran in distributed mode on a 13-node grid. The "Data Access Information" table shows that the input data were accessed in parallel symmetric mode and the output data set was sent to the client, where the V9 (base) engine stored it as a SAS data set in the Work directory.

Figure 3.4 Performance Information and Data Access Information Tables

The HPDS2 Procedure

Performance Information	
Host Node	hpa.sas.com
Execution Mode	Distributed
Number of Compute Nodes	12
Number of Threads per Node	24

Data Access Information

Data	Engine	Role	Path
HDATLIB.NEURALGIA	SASHDAT	Input	Parallel, Symmetric
WORK.NEURALGIA	V9	Output	To Client

Alongside-the-Database Execution

High-performance analytical procedures interface with the distributed database management system (DBMS) on the appliance in a unique way. If the input data are stored in the DBMS and the grid host is the appliance that houses the data, high-performance analytical procedures create a distributed computing environment in which an analytic process is collocated with the nodes of the DBMS. Data then pass from the DBMS to the analytic process on each node. Instead of moving across the network and possibly back to the client machine, the data pass locally between the processes on each node of the appliance.

Because the analytic processes on the appliance are separate from the database processes, the technique is referred to as alongside-the-database execution in contrast to in-database execution, where the analytic code executes in the database process.

In general, when you have a large amount of input data, you can achieve the best performance from high-performance analytical procedures if execution is alongside the database.

Before you can run alongside the database, you must distribute the data to the appliance. The following statements use the HPDS2 procedure to distribute the data set Work.simData into the mydb database on the hpa.sas.com appliance. In this example, the appliance houses a Greenplum database.

```
option set=GRIDHOST="green.sas.com";
libname applianc greenplm
        server  ="green.sas.com"
        user    =XXXXXX
        password=YYYYY
        database=mydb;


option set=GRIDHOST="compute_appliance.sas.com";

proc datasets lib=applianc nolist; delete simData;
proc hpds2 data=simData
           out =applianc.simData(distributed_by='distributed randomly');
  performance commit=10000 nodes=all;
  data DS2GTF.out;
     method run();
        set DS2GTF.in;
     end;
  enddata;
run;
```

If the output table applianc.simData exists, the DATASETS procedure removes the table from the Greenplum database because a DBMS does not usually support replacement operations on tables.

Note that the libref for the output table points to the appliance. The data set option informs the HPDS2 procedure to distribute the records randomly among the data segments of the appliance. The statements that follow the PERFORMANCE statement are the DS2 program that copies the input data to the output data without further transformations.

Because you loaded the data into a database on the appliance, you can use the following HPLOGISTIC statements to perform the analysis on the appliance in the alongside-the-database mode. These statements are almost identical to the first PROC HPLOGISTIC example in a previous section, which executed in single-machine mode.

```
proc hplogistic data=applianc.simData;
   class a b c;
   model y = a b c x1 x2 x3;
run;
```

The subtle differences are as follows:

- The grid host environment variable that you specified in an OPTION SET= command is still in effect.
- The DATA= option in the high-performance analytical procedure uses a libref that identifies the data source as being housed on the appliance. This libref was specified in a prior LIBNAME statement.

Figure 3.5 shows the results from this analysis. The "Performance Information" table shows that the execution was in distributed mode, and the "Data Access Information" table shows that the data were accessed asymmetrically in parallel from the Greenplum database. The numeric results agree with the previous analyses, which are shown in Figure 3.1 and Figure 3.2.

Figure 3.5 Alongside-the-Database Execution on Greenplum

The HPLOGISTIC Procedure

Performance Information	
Host Node	compute_appliance.sas.com
Execution Mode	Distributed
Number of Compute Nodes	141
Number of Threads per Node	32

Data Access Information			
Data	**Engine**	**Role**	**Path**
APPLIANC.SIMDATA	GREENPLM	Input	Parallel, Asymmetric

Model Information	
Data Source	APPLIANC.SIMDATA
Response Variable	y
Class Parameterization	GLM
Distribution	Binary
Link Function	Logit
Optimization Technique	Newton-Raphson with Ridging

Parameter Estimates

Parameter	Estimate	Standard Error	DF	t Value	Pr > |t|
Intercept	5.7011	0.2539	Infty	22.45	<.0001
a 0	-0.01020	0.06627	Infty	-0.15	0.8777
a 1	0	.	.	.	.
b 0	0.7124	0.06558	Infty	10.86	<.0001
b 1	0	.	.	.	.
c 0	0.8036	0.06456	Infty	12.45	<.0001
c 1	0	.	.	.	.
x1	0.01975	0.000614	Infty	32.15	<.0001
x2	-0.04728	0.003098	Infty	-15.26	<.0001
x3	-0.1017	0.009470	Infty	-10.74	<.0001

Alongside-LASR Distributed Execution

You can execute high-performance analytical procedures in distributed mode alongside a SAS LASR Analytic Server. When high-performance analytical procedures run in this mode, the data are preloaded in distributed form in memory that is managed by a LASR Analytic Server. The data on the nodes of the appliance are accessed in parallel in the process that runs the LASR Analytic Server, and they are transferred to the process where the high-performance analytical procedure runs. In general, each high-performance analytical procedure copies the data to memory that persists only while that procedure executes. Hence, when a high-performance analytical procedure runs alongside a LASR Analytic Server, both the high-performance analytical procedure and the LASR Analytic Server have a copy of the subset of the data that is used by the high-performance analytical procedure. The advantage of running high-performance analytical procedures alongside a LASR Analytic Server (as opposed to running alongside a DBMS table or alongside HDFS) is

that the initial transfer of data from the LASR Analytic Server to the high-performance analytical procedure is a memory-to-memory operation that is faster than the disk-to-memory operation when the procedure runs alongside a DBMS or HDFS. When the cost of preloading a table into a LASR Analytic Server is amortized by multiple uses of these data in separate runs of high-performance analytical procedures, using the LASR Analytic Server can result in improved performance.

Running High-Performance Analytical Procedures Alongside a SAS LASR Analytic Server in Distributed Mode

This section provides an example of steps that you can use to start and load data into a SAS LASR Analytic Server instance and then run high-performance analytical procedures alongside this LASR Analytic Server instance.

Starting a SAS LASR Analytic Server Instance

The following statements create a SAS LASR Analytic Server instance and load it with the simData data set that is used in the preceding examples. The data that are loaded into the LASR Analytic Server persist in memory across procedure boundaries until these data are explicitly deleted or until the server instance is terminated.

```
proc lasr port=54545
          data=simData
          path="/tmp/";
   performance host="hpa.sas.com" nodes=ALL;
run;
```

The PORT= option specifies a network port number to use. The PATH= option specifies the directory in which the server and table signature files are to be stored. The specified directory must exist on each machine in the cluster. The DATA= option specifies the name of a data set that is loaded into this LASR Analytic Server instance. (You do not need to specify the DATA= option at this time because you can add tables to the LASR Analytic Server instance at any stage of its life.) For more information about starting and using a LASR Analytic Server, see the *SAS LASR Analytic Server: Reference Guide*.

The NODES=ALL option in the PERFORMANCE statement specifies that the LASR Analytic Server run on all the nodes on the appliance. You can start a LASR Analytic Server on a subset of the nodes on an appliance, but this might affect whether high-performance analytical procedures can run alongside the LASR Analytic Server. For more information, see the section "Alongside-LASR Distributed Execution on a Subset of the Appliance Nodes" on page 75.

Figure 3.6 shows the "Performance Information" and "Data Access Information" tables, which show that the LASR procedure ran in distributed mode on 13 nodes and that the data were sent from the client to the appliance.

Figure 3.6 Performance and Data Access Information

The LASR Procedure

Performance Information	
Host Node	hpa.sas.com
Execution Mode	Distributed
Number of Compute Nodes	12

Data Access Information			
Data	**Engine**	**Role**	**Path**
WORK.SIMDATA	V9	Input	From Client

Associating a SAS Libref with the SAS LASR Analytic Server Instance

The following statements use a LIBNAME statement that associates a SAS libref (named MyLasr) with tables on the server instance as follows:

```
libname MyLasr sasiola port=54545 host="hpa.sas.com";
```

The SASIOLA option requests that the MyLasr libref use the SASIOLA engine, and the PORT= value associates this libref with the appropriate server instance. For more information about creating a libref that uses the SASIOLA engine, see the *SAS LASR Analytic Server: Reference Guide*.

Running a High-Performance Analytical Procedure Alongside the SAS LASR Analytic Server Instance

You can use the MyLasr libref to specify the input data for high-performance analytical procedures. You can also create output data sets in the SAS LASR Analytic Server instance by using this libref to request that the output data set be held in memory by the server instance as follows:

```
proc hplogistic data=MyLasr.simData;
   class a b c;
   model y = a b c x1 x2 x3;
   output out=MyLasr.simulateScores pred=PredictedProbabliity;
run;
```

Because you previously specified the GRIDHOST= environment variable and the input data are held in distributed form in the associated server instance, this PROC HPLOGISTIC step runs in distributed mode alongside the LASR Analytic Server, as indicated in the "Performance Information" table shown in Figure 3.7.

Figure 3.7 Performance and Data Access Information

The HPLOGISTIC Procedure

Performance Information	
Host Node	hpa.sas.com
Execution Mode	Distributed
Number of Compute Nodes	12
Number of Threads per Node	24

Data Access Information			
Data	**Engine**	**Role**	**Path**
MYLASR.SIMDATA	SASIOLA	Input	Parallel, Symmetric
MYLASR.SIMULATESCORES	SASIOLA	Output	Parallel, Symmetric

The "Data Access Information" table shows that both the input and output data were read and written, respectively, in parallel symmetric mode.

The preceding OUTPUT statement creates an output table that is added to the LASR Analytic Server instance. Output data sets do not have to be created in the same server instance that holds the input data. You can use a different LASR Analytic Server instance to hold the output data set. However, in order for the output data to be created in parallel symmetric mode, all the nodes that are used by the server instance that holds the input data must also be used by the server instance that holds the output data.

Terminating a SAS LASR Analytic Server Instance

You can continue to run high-performance analytical procedures and add and delete tables from the SAS LASR Analytic Server instance until you terminate the server instance as follows:

```
proc lasr term port=54545;
run;
```

Alongside-LASR Distributed Execution on a Subset of the Appliance Nodes

When you run PROC LASR to start a SAS LASR Analytic Server, you can specify the NODES= option in a PERFORMANCE statement to control how many nodes the LASR Analytic Server executes on. Similarly, a high-performance analytical procedure can execute on a subset of the nodes either because you specify the NODES= option in a PERFORMANCE statement or because you run alongside a DBMS or HDFS with an input data set that is distributed on a subset of the nodes on an appliance. In such situations, if a high-performance analytical procedure uses nodes on which the LASR Analytic Server is not running, then running alongside LASR is not supported. You can avoid this issue by specifying the NODES=ALL in the PERFORMANCE statement when you use PROC LASR to start the LASR Analytic Server.

Running High-Performance Analytical Procedures in Asymmetric Mode

This section provides examples of how you can run high-performance analytical procedures in asymmetric mode.

Asymmetric mode is commonly used when the data appliance and the computing appliance are distinct appliances. In order to be able to use an appliance as a data provider for high-performance analytical procedures that run in asymmetric mode on another appliance, it is not necessary that SAS High-Performance Econometrics be installed on the data appliance. However, it is essential that a SAS Embedded Process be installed on the data appliance and that SAS High-Performance Econometrics be installed on the computing appliance.

The following examples use a 24-node data appliance named "data_appliance.sas.com," which houses a Teradata DBMS and has a SAS Embedded Process installed.

The following statements load the simData data set of the preceding sections onto the data appliance:

```
libname dataLib teradata
        server  ="tera2650"
        user    =XXXXXX
        password=YYYYY
        database=mydb;

data dataLib.simData;
   set simData;
run;
```

NOTE: You can provision the appliance with data even if SAS High-Performance Econometrics software is not installed on the appliance.

The following subsections show how you can run the HPLOGISTIC procedure asymmetrically on distinct data and computing appliances.

Running in Asymmetric Mode on Distinct Appliances

Usually, there is no advantage to executing high-performance analytical procedures in asymmetric mode on one appliance, because data might have to be unnecessarily moved between nodes. The following example demonstrates the more typical use of asymmetric mode. In this example, the specified grid host "compute_appliance.sas.com" is a 142-node computing appliance that is different from the 24-node data appliance "data_appliance.sas.com," which houses the Teradata DBMS where the data reside.

The advantage of using different computing and data appliances is that the data appliance is not affected by the execution of high-performance analytical procedures except during the initial parallel data transfer. A potential disadvantage of this asymmetric mode of execution is that the performance can be limited by the bandwidth with which data can be moved between the appliances. However, because this data movement takes place in parallel from the nodes of the data appliance to the nodes of the computing appliance, this

potential performance bottleneck can be overcome with appropriately provisioned hardware. The following statements show how this is done:

```
proc hplogistic data=dataLib.simData;
   class a b c;
   model y = a b c x1 x2 x3;
   performance host = "compute_appliance.sas.com" nodes=30;
run;
```

Figure 3.8 shows the "Performance Information" and "Data Access Information" tables.

Figure 3.8 Asymmetric Mode with Distinct Data and Computing Appliances

The HPLOGISTIC Procedure

Performance Information	
Host Node	compute_appliance.sas.com
Execution Mode	Distributed
Number of Compute Nodes	30
Number of Threads per Node	32

Data Access Information			
Data	**Engine**	**Role**	**Path**
DATALIB.simData	TERADATA	Input	Parallel, Asymmetric

PROC HPLOGISTIC ran on 30 nodes of the computing appliance, even though the data were partitioned across the 24 nodes of the data appliance. The numeric results are not reproduced here, but they agree with the previous analyses shown in Figure 3.1 and Figure 3.2.

Every time you run a high-performance analytical procedure in asymmetric mode that uses different computing and data appliances, data are transferred between these appliances. If you plan to make repeated use of the same data, then it might be advantageous to temporarily persist the data that you need on the computing appliance. One way to persist the data is to store them as a table in a SAS LASR Analytic Server that runs on the computing appliance. By running PROC LASR in asymmetric mode, you can load the data in parallel from the data appliance nodes to the nodes on which the LASR Analytic Server runs on the computing appliance. You can then use a LIBNAME statement that associates a SAS libref with tables on the LASR Analytic Server. The following statements show how you do this:

```
proc lasr port=54345
          data=dataLib.simData
          path="/tmp/";
   performance host ="compute_appliance.sas.com" nodes=30;
run;

libname MyLasr sasiola tag="dataLib" port=54345 host="compute_appliance.sas.com" ;
```

Figure 3.9 show the "Performance Information" and "Data Access Information" tables.

Figure 3.9 PROC LASR Running in Asymmetric Mode

The LASR Procedure

Performance Information	
Host Node	compute_appliance.sas.com
Execution Mode	Distributed
Number of Compute Nodes	30

Data Access Information			
Data	**Engine**	**Role**	**Path**
DATALIB.simData	TERADATA	Input	Parallel, Asymmetric

By default, all the nodes on the computing appliance would be used. However, because NODES=30 was specified in the PERFORMANCE statement, PROC LASR ran on only 30 nodes of the computing appliance. The data were loaded asymmetrically in parallel from the 24 data appliance nodes to the 30 compute nodes on which PROC LASR ran.

After the data are loaded into a LASR Analytic Server that runs on the computing appliance, you can run high-performance analytical procedures alongside this LASR Analytic Server as shown by the following statements:

```
proc hplogistic data=MyLasr.simData;
   class a b c;
   model y = a b c x1 x2 x3;
   output out=MyLasr.myOutputData pred=myPred;
   performance host = "compute_appliance.sas.com";
run;
```

The following note, which appears in the SAS log, confirms that the output data set is created successfully:

```
NOTE: The table DATALIB.MYOUTPUTDATA has been added to the LASR Analytic Server
      with port 54345. The Libname is MYLASR.
```

You can use the dataLib libref that you used to load the data onto the data appliance to create an output data set on the data appliance.

```
proc hplogistic data=MyLasr.simData;
   class a b c;
   model y = a b c x1 x2 x3;
   output out=dataLib.myOutputData pred=myPred;
   performance host  = "compute_appliance.sas.com";
run;
```

The following note, which appears in the SAS log, confirms that the output data set is created successfully on the data appliance:

```
NOTE: The data set DATALIB.myOutputData has 100000 observations and 1 variables.
```

When you run a high-performance analytical procedure on a computing appliance and either read data from or write data to a different data appliance on which a SAS Embedded Process is running, the Read and Write operations take place in parallel without any movement of data to and from the SAS client.

When you no longer need the data in the SAS LASR Analytic Server, you should terminate the server instance as follows:

```
proc lasr term port=54345;
    performance host="compute_appliance.sas.com";
run;
```

If you configured Hadoop on the computing appliance, then you can create output data tables that are stored in the HDFS on the computing appliance. You can do this by using the SASHDAT engine as described in the section "Alongside-HDFS Execution" on page 79.

Alongside-HDFS Execution

Running high-performance analytical procedures alongside HDFS shares many features with running alongside the database. You can execute high-performance analytical procedures alongside HDFS by using either the SASHDAT engine or the Hadoop engine.

You use the SASHDAT engine to read and write data that are stored in HDFS in a proprietary SASHDAT format. In SASHDAT format, metadata that describe the data in the Hadoop files are included with the data. This enables you to access files in SASHDAT format without supplying any additional metadata. Additionally, you can also use the SASHDAT engine to read data in CSV (comma-separated value) format, but you need supply metadata that describe the contents of the CSV data. The SASHDAT engine provides highly optimized access to data in HDFS that are stored in SASHDAT format.

The Hadoop engine reads data that are stored in various formats from HDFS and writes data to HDFS in CSV format. This engine can use metadata that are stored in Hive, which is a data warehouse that supplies metadata about data that are stored in Hadoop files. In addition, this engine can use metadata that you create by using the HDMD procedure.

The following subsections provide details about using the SASHDAT and Hadoop engines to execute high-performance analytical procedures alongside HDFS.

Alongside-HDFS Execution by Using the SASHDAT Engine

If the grid host is a cluster that houses data that have been distributed by using the SASHDAT engine, then high-performance analytical procedures can analyze those data in the alongside-HDFS mode. The procedures use the distributed computing environment in which an analytic process is collocated with the nodes of the cluster. Data then pass from HDFS to the analytic process on each node of the cluster.

Before you can run a procedure alongside HDFS, you must distribute the data to the cluster. The following statements use the SASHDAT engine to distribute to HDFS the simData data set that was used in the previous two sections:

```
option set=GRIDHOST="hpa.sas.com";

libname hdatLib sashdat
        path="/hps";

 data hdatLib.simData (replace = yes) ;
     set simData;
 run;
```

In this example, the GRIDHOST is a cluster where the SAS Data in HDFS Engine is installed. If a data set that is named simData already exists in the hps directory in HDFS, it is overwritten because the REPLACE=YES data set option is specified. For more information about using this LIBNAME statement, see the section "LIBNAME Statement for the SAS Data in HDFS Engine" in the *SAS LASR Analytic Server: Reference Guide*.

The following HPLOGISTIC procedure statements perform the analysis in alongside-HDFS mode. These statements are almost identical to the PROC HPLOGISTIC example in the previous two sections, which executed in single-machine mode and alongside-the-database distributed mode, respectively.

Figure 3.10 shows the "Performance Information" and "Data Access Information" tables. You see that the procedure ran in distributed mode and that the input data were read in parallel symmetric mode. The numeric results shown in Figure 3.11 agree with the previous analyses shown in Figure 3.1, Figure 3.2, and Figure 3.5.

Figure 3.10 Alongside-HDFS Execution Performance Information

The HPLOGISTIC Procedure

Performance Information	
Host Node	hpa.sas.com
Execution Mode	Distributed
Number of Compute Nodes	12
Number of Threads per Node	24

Data Access Information			
Data	**Engine**	**Role**	**Path**
HDATLIB.SIMDATA	SASHDAT	Input	Parallel, Symmetric

Figure 3.11 Alongside-HDFS Execution Model Information

Model Information	
Data Source	HDATLIB.SIMDATA
Response Variable	y
Class Parameterization	GLM
Distribution	Binary
Link Function	Logit
Optimization Technique	Newton-Raphson with Ridging

Figure 3.11 *continued*

Parameter Estimates

Parameter	Estimate	Standard Error	DF	t Value	Pr > |t|
Intercept	5.7011	0.2539	Infty	22.45	<.0001
a 0	-0.01020	0.06627	Infty	-0.15	0.8777
a 1	0	.	.	.	.
b 0	0.7124	0.06558	Infty	10.86	<.0001
b 1	0	.	.	.	.
c 0	0.8036	0.06456	Infty	12.45	<.0001
c 1	0	.	.	.	.
x1	0.01975	0.000614	Infty	32.15	<.0001
x2	-0.04728	0.003098	Infty	-15.26	<.0001
x3	-0.1017	0.009470	Infty	-10.74	<.0001

Alongside-HDFS Execution by Using the Hadoop Engine

The following LIBNAME statement sets up a libref that you can use to access data that are stored in HDFS and have metadata in Hive:

```
libname hdoopLib hadoop
        server       = "hpa.sas.com"
        user         = XXXXX
        password     = YYYYY
        database     = myDB
        config       = "demo.xml" ;
```

For more information about LIBNAME options available for the Hadoop engine, see the LIBNAME topic in the Hadoop section of *SAS/ACCESS for Relational Databases: Reference*. The configuration file that you specify in the CONFIG= option contains information that is needed to access the Hive server. It also contains information that enables this configuration file to be used to access data in HDFS without using the Hive server. This information can also be used to specify replication factors and block sizes that are used when the engine writes data to HDFS.

The following DATA step uses the Hadoop engine to distribute to HDFS the simData data set that was used in the previous sections. The engine creates metadata for the data set in Hive.

```
data hdoopLib.simData;
    set simData;
run;
```

After you have loaded data or if you are accessing preexisting data in HDFS that have metadata in Hive, you can access this data alongside HDFS by using high-performance analytical procedures. The following HPLOGISTIC procedure statements perform the analysis in alongside-HDFS mode. These statements are similar to the PROC HPLOGISTIC example in the previous sections.

```
proc hplogistic data=hdoopLib.simData;
   class a b c;
   model y = a b c x1 x2 x3;
   performance host = "compute_appliance.sas.com";
run;
```

Figure 3.12 shows the "Performance Information" and "Data Access Information" tables. You see that the procedure ran in distributed mode and that the input data were read in parallel asymmetric mode. The numeric results shown in Figure 3.13 agree with the previous analyses.

Figure 3.12 Alongside-HDFS Execution by Using the Hadoop Engine

The HPLOGISTIC Procedure

Performance Information	
Host Node	compute_appliance.sas.com
Execution Mode	Distributed
Number of Compute Nodes	141
Number of Threads per Node	32

Data Access Information			
Data	**Engine**	**Role**	**Path**
GRIDLIB.SIMDATA	HADOOP	Input	Parallel, Asymmetric

Figure 3.13 Alongside-HDFS Execution by Using the Hadoop Engine

Model Information	
Data Source	GRIDLIB.SIMDATA
Response Variable	y
Class Parameterization	GLM
Distribution	Binary
Link Function	Logit
Optimization Technique	Newton-Raphson with Ridging

Parameter Estimates

Parameter	Estimate	Standard Error	DF	t Value	Pr > |t|
Intercept	5.7011	0.2539	Infty	22.45	<.0001
a 0	-0.01020	0.06627	Infty	-0.15	0.8777
a 1	0	.	.	.	.
b 0	0.7124	0.06558	Infty	10.86	<.0001
b 1	0	.	.	.	.
c 0	0.8036	0.06456	Infty	12.45	<.0001
c 1	0	.	.	.	.
x1	0.01975	0.000614	Infty	32.15	<.0001
x2	-0.04728	0.003098	Infty	-15.26	<.0001
x3	-0.1017	0.009470	Infty	-10.74	<.0001

The Hadoop engine also enables you to access tables in HDFS that are stored in various formats and that are not registered in Hive. You can use the HDMD procedure to generate metadata for tables that are stored in the following file formats:

- delimited text
- fixed-record length binary
- sequence files
- XML text

To read any other kind of file in Hadoop, you can write a custom file reader plug-in in Java for use with PROC HDMD. For more information about LIBNAME options available for the Hadoop engine, see the LIBNAME topic in the Hadoop section of *SAS/ACCESS for Relational Databases: Reference*.

The following example shows how you can use PROC HDMD to register metadata for CSV data independently from Hive and then analyze these data by using high-performance analytical procedures. The CSV data in the table csvExample.csv is stored in HDFS in the directory /user/demo/data. Each record in this table consists of the following fields, in the order shown and separated by commas.

1. a string of at most six characters
2. a numeric field with values of 0 or 1
3. a numeric field with real numbers

Suppose you want to fit a logistic regression model to these data, where the second field represents a target variable named Success, the third field represents a regressor named Dose, and the first field represents a classification variable named Group.

The first step is to use PROC HDMD to create metadata that are needed to interpret the table, as in the following statements:

```
libname hdoopLib hadoop
                 server       = "hpa.sas.com"
                 user         = XXXXX
                 password     = YYYYY
                 HDFS_PERMDIR = "/user/demo/data"
                 HDFS_METADIR = "/user/demo/meta"
                 config       = "demo.xml"
                 DBCREATE_TABLE_EXTERNAL=YES;

proc hdmd name=hdoopLib.csvExample data_file='csvExample.csv'
          format=delimited encoding=utf8 sep = ',';

     column Group    char(6);
     column Success  double;
     column Dose     double;
run;
```

The metadata that are created by PROC HDMD for this table are stored in the directory /user/demo/meta that you specified in the HDFS_METADIR = option in the preceding LIBNAME statement. After you create the metadata, you can execute high-performance analytical procedures with these data by using the hdoopLib libref. For example, the following statements fit a logistic regression model to the CSV data that are stored in csvExample.csv table.

```
proc hplogistic data=hdoopLib.csvExample;
    class Group;
    model Success = Dose;
    performance host     = "compute_appliance.sas.com"
                gridmode = asym;
run;
```

Figure 3.14 shows the results of this analysis. You see that the procedure ran in distributed mode and that the input data were read in parallel asymmetric mode. The metadata that you created by using the HDMD procedure have been used successfully in executing this analysis.

Figure 3.14 Alongside-HDFS Execution with CSV Data

The HPLOGISTIC Procedure

Performance Information	
Host Node	compute_appliance.sas.com
Execution Mode	Distributed
Number of Compute Nodes	141
Number of Threads per Node	32

Data Access Information			
Data	**Engine**	**Role**	**Path**
GRIDLIB.CSVEXAMPLE	HADOOP	Input	Parallel, Asymmetric

Model Information	
Data Source	GRIDLIB.CSVEXAMPLE
Response Variable	Success
Class Parameterization	GLM
Distribution	Binary
Link Function	Logit
Optimization Technique	Newton-Raphson with Ridging

Class Level Information		
Class	**Levels**	**Values**
Group	3	group1 group2 group3

Number of Observations Read	1000
Number of Observations Used	1000

Parameter Estimates

Parameter	Estimate	Standard Error	DF	t Value	Pr > |t|
Intercept	0.1243	0.1295	Infty	0.96	0.3371
Dose	-0.2674	0.2216	Infty	-1.21	0.2277

Output Data Sets

In the alongside-the-database mode, the data are read in distributed form, minimizing data movement for best performance. Similarly, when you write output data sets and a high-performance analytical procedure executes in distributed mode, the data can be written in parallel into the database.

For example, in the following statements, the HPLOGISTIC procedure executes in distributed mode by using eight nodes on the appliance to perform the logistic regression on work.simData:

```
proc hplogistic data=simData;
   class a b c;
   model y = a b c x1 x2 x3;
   id a;
   output out=applianc.simData_out pred=p;
   performance host="hpa.sas.com" nodes=8;
run;
```

The output data set applianc.simData_out is written in parallel into the database. Although the data are fed on eight nodes, the database might distribute the data on more nodes.

When a high-performance analytical procedure executes in single-machine mode, all output objects are created on the client. If the libref of the output data sets points to the appliance, the data are transferred to the database on the appliance. This can lead to considerable performance degradation compared to execution in distributed mode.

Many procedures in SAS software add the variables from the input data set when an observationwise output data set is created. The assumption of high-performance analytical procedures is that the input data sets can be large and contain many variables. For performance reasons, the output data set contains the following:

- variables that are explicitly created by the statement
- variables that are listed in the ID statement, as described in Chapter 4, "Shared Statistical Concepts" (*SAS/STAT User's Guide: High-Performance Procedures*)
- distribution keys or hash keys that are transferred from the input data set

Including this information enables you to add to the output data set information necessary for subsequent SQL joins without copying the entire input data set to the output data set.

Working with Formats

You can use SAS formats and user-defined formats with high-performance analytical procedures as you can with other procedures in the SAS System. However, because the analytic work is carried out in a distributed environment and might depend on the formatted values of variables, some special handling can improve the efficiency of work with formats.

High-performance analytical procedures examine the variables that are used in an analysis for association with user-defined formats. Any user-defined formats that are found by a procedure are transmitted automatically to the appliance. If you are running multiple high-performance analytical procedures in a SAS session and the analysis variables depend on user-defined formats, you can preprocess the formats. This step involves generating an XML stream (a file) of the formats and passing the stream to the high-performance analytical procedures.

Suppose that the following formats are defined in your SAS program:

```
proc format;
     value YesNo        1='Yes'        0='No';
     value checkThis    1='ThisisOne'  2='ThisisTwo';
     value $cityChar    1='Portage'    2='Kinston';
run;
```

The next group of SAS statements create the XML stream for the formats in the file *Myfmt.xml*, associate that file with the file reference myxml, and pass the file reference with the FMTLIBXML= option in the PROC HPLOGISTIC statement:

```
filename myxml 'Myfmt.xml';
libname  myxml XML92 xmltype=sasfmt tagset=tagsets.XMLsuv;
proc format cntlout=myxml.allfmts;
run;


proc hplogistic data=six fmtlibxml=myxml;
   class wheeze cit age;
   format wheeze best4. cit $cityChar.;
   model wheeze = cit age;
run;
```

Generation and destruction of the stream can be wrapped in convenience macros:

```
%macro Make_XMLStream(name=tempxml);
    filename &name 'fmt.xml';
    libname  &name XML92 xmltype=sasfmt tagset=tagsets.XMLsuv;
    proc format cntlout=&name..allfmts;
    run;
%mend;

%macro Delete_XMLStream(fref);
    %let rc=%sysfunc(fdelete(&fref));
%mend;
```

If you do not pass an XML stream to a high-performance analytical procedure that supports the FMTLIBXML= option, the procedure generates an XML stream as needed when it is invoked.

PERFORMANCE Statement

PERFORMANCE < *performance-options* > **;**

The PERFORMANCE statement defines performance parameters for multithreaded and distributed computing, passes variables that describe the distributed computing environment, and requests detailed results about the performance characteristics of a high-performance analytical procedure.

You can also use the PERFORMANCE statement to control whether a high-performance analytical procedure executes in single-machine or distributed mode.

You can specify the following *performance-options* in the PERFORMANCE statement:

COMMIT=*n*

requests that the high-performance analytical procedure write periodic updates to the SAS log when observations are sent from the client to the appliance for distributed processing.

High-performance analytical procedures do not have to use input data that are stored on the appliance. You can perform distributed computations regardless of the origin or format of the input data, provided that the data are in a format that can be read by the SAS System (for example, because a SAS/ACCESS engine is available).

In the following example, the HPREG procedure performs LASSO variable selection where the input data set is stored on the client:

```
proc hpreg data=work.one;
   model y = x1-x500;
   selection method=lasso;
   performance nodes=10 host='mydca' commit=10000;
run;
```

In order to perform the work as requested using 10 nodes on the appliance, the data set Work.One needs to be distributed to the appliance.

High-performance analytical procedures send the data in blocks to the appliance. Whenever the number of observations sent exceeds an integer multiple of the COMMIT= size, a SAS log message is produced. The message indicates the actual number of observations distributed, and not an integer multiple of the COMMIT= size.

DETAILS

requests a table that shows a timing breakdown of the procedure steps.

GRIDHOST=*"name"*

HOST=*"name"*

specifies the name of the appliance host in single or double quotation marks. If this option is specified, it overrides the value of the GRIDHOST environment variable.

GRIDMODE=SYM | ASYM

MODE=SYM | ASYM

is a deprecated option that specifies whether to run the high-performance analytical procedure in symmetric (SYM) mode or asymmetric (ASYM) mode. This option overrides the GRIDMODE environment variable.

GRIDTIMEOUT=*s*

TIMEOUT=*s*

specifies the time-out in seconds for a high-performance analytical procedure to wait for a connection to the appliance and establish a connection back to the client. The default is 120 seconds. If jobs are submitted to the appliance through workload management tools that might suspend access to the appliance for a longer period, you might want to increase the time-out value.

INSTALL=*"name"*

INSTALLLOC=*"name"*

specifies the directory in which the shared libraries for the high-performance analytical procedure are installed on the appliance. Specifying the INSTALL= option overrides the GRIDINSTALLLOC environment variable.

LASRSERVER=*"path"*

LASR=*"path"*

specifies the fully qualified path to the description file of a SAS LASR Analytic Server instance. If the input data set is held in memory by this LASR Analytic Server instance, then the procedure runs alongside LASR. This option is not needed to run alongside LASR if the DATA= specification of the input data uses a libref that is associated with a LASR Analytic Server instance. For more information, see the section "Alongside-LASR Distributed Execution" on page 72 and the *SAS LASR Analytic Server: Reference Guide*.

NODES=ALL | *n*

NNODES=ALL | *n*

specifies the number of nodes in the distributed computing environment, provided that the data are not processed alongside the database.

Specifying NODES=0 indicates that you want to process the data in single-machine mode on the client machine. If the input data are not alongside the database, this is the default. The high-performance analytical procedures then perform the analysis on the client. For example, the following sets of statements are equivalent:

```
proc hplogistic data=one;
   model y = x;
run;
```

```
proc hplogistic data=one;
   model y = x;
   performance nodes=0;
run;
```

If the data are not read alongside the database, the NODES= option specifies the number of nodes on the appliance that are involved in the analysis. For example, the following statements perform the analysis in distributed mode by using 10 units of work on the appliance that is identified in the HOST= option:

```
proc hplogistic data=one;
   model y = x;
   performance nodes=10 host="hpa.sas.com";
run;
```

If the number of nodes can be modified by the application, you can specify a NODES=*n* option, where *n* exceeds the number of physical nodes on the appliance. The SAS High-Performance Econometrics software then *oversubscribes* the nodes and associates nodes with multiple units of work. For example, on a system that has 16 appliance nodes, the following statements oversubscribe the system by a factor of 3:

```
proc hplogistic data=one;
   model y = x;
   performance nodes=48 host="hpa.sas.com";
run;
```

Usually, it is not advisable to oversubscribe the system because the analytic code is optimized for a certain level of multithreading on the nodes that depends on the CPU count. You can specify NODES=ALL if you want to use all available nodes on the appliance without oversubscribing the system.

If the data are read alongside the distributed database on the appliance, specifying a nonzero value for the NODES= option has no effect. The number of units of work in the distributed computing environment is then determined by the distribution of the data and cannot be altered. For example, if you are running alongside an appliance with 24 nodes, the NODES= option in the following statements is ignored:

```
libname GPLib greenplm server=gpdca user=XXX password=YYY
              database=ZZZ;
proc hplogistic data=gplib.one;
   model y = x;
   performance nodes=10 host="hpa.sas.com";
run;
```

NTHREADS=*n*

THREADS=*n*

specifies the number of threads for analytic computations and overrides the SAS system option THREADS | NOTHREADS. If you do not specify the NTHREADS= option, the number of threads is determined based on the number of CPUs on the host on which the analytic computations execute. The algorithm by which a CPU count is converted to a thread count is specific to the high-performance analytical procedure. Most procedures create one thread per CPU for the analytic computations.

By default, high-performance analytical procedures run in multiple concurrent threads unless multithreading has been turned off by the NOTHREADS system option or you force single-threaded execution by specifying NTHREADS=1. The largest number that can be specified for *n* is 256. Individual high-performance analytical procedures can impose more stringent limits if called for by algorithmic considerations.

NOTE: The SAS system options THREADS | NOTHREADS apply to the client machine on which the SAS high-performance analytical procedures execute. They do not apply to the compute nodes in a distributed environment.

Chapter 4
Working with Time Series Data

Contents

Overview **92**
Time Series and SAS Data Sets **93**
Introduction 93
Reading a Simple Time Series 94
Dating Observations **95**
SAS Date, Datetime, and Time Values 95
Reading Date and Datetime Values with Informats 97
Formatting Date and Datetime Values 97
The Variables DATE and DATETIME 99
Sorting by Time 99
Subsetting Data and Selecting Observations **100**
Subsetting SAS Data Sets 100
Using the WHERE Statement with SAS Procedures 101
Using SAS Data Set Options 102
Storing Time Series in a SAS Data Set **102**
Standard Form of a Time Series Data Set 103
Several Series with Different Ranges 104
Missing Values and Omitted Observations 104
Cross-Sectional Dimensions and BY Groups 105
Interleaved Time Series 107
Output Data Sets of SAS/ETS Procedures 109
Time Series Periodicity and Time Intervals **110**
Specifying Time Intervals 111
Using Intervals with SAS/ETS Procedures 111
Time Intervals, the Time Series Forecasting System, and the Time Series Viewer . . . 112
Plotting Time Series **112**
Using the Time Series Viewer 112
Using PROC SGPLOT 113
Using PROC GPLOT 118
Calendar and Time Functions **119**
Computing Dates from Calendar Variables 119
Computing Calendar Variables from Dates 120
Converting between Date, Datetime, and Time Values 121
Computing Datetime Values 121
Computing Calendar and Time Variables 121

Interval Functions INTNX and INTCK **122**
Incrementing Dates by Intervals 123
Alignment of SAS Dates 124
Computing the Width of a Time Interval 125
Computing the Ceiling of an Interval 125
Counting Time Intervals 126
Checking Data Periodicity 127
Filling In Omitted Observations in a Time Series Data Set 127
Using Interval Functions for Calendar Calculations 128
Lags, Leads, Differences, and Summations **128**
The LAG and DIF Functions 129
Multiperiod Lags and Higher-Order Differencing 132
Percent Change Calculations 133
Leading Series 135
Summing Series 136
Transforming Time Series **137**
Log Transformation 138
Other Transformations 139
The EXPAND Procedure and Data Transformations 140
Manipulating Time Series Data Sets **140**
Splitting and Merging Data Sets 140
Transposing Data Sets 141
Time Series Interpolation **145**
Interpolating Missing Values 146
Interpolating to a Higher or Lower Frequency 146
Interpolating between Stocks and Flows, Levels and Rates 146
Reading Time Series Data **147**
Reading a Simple List of Values 147
Reading Fully Described Time Series in Transposed Form 148

Overview

This chapter discusses working with time series data in the SAS System. The following topics are included:

- dating time series and working with SAS date and datetime values
- subsetting data and selecting observations
- storing time series data in SAS data sets
- specifying time series periodicity and time intervals
- plotting time series

- using calendar and time interval functions
- computing lags and other functions across time
- transforming time series
- transposing time series data sets
- interpolating time series
- reading time series data recorded in different ways

In general, this chapter focuses on using features of the SAS programming language and not on features of SAS/ETS software. However, since SAS/ETS procedures are used to analyze time series, understanding how to use the SAS programming language to work with time series data is important for the effective use of SAS/ETS software.

You do not need to read this chapter to use SAS/ETS procedures. If you are already familiar with SAS programming you might want to skip this chapter, or you can refer to sections of this chapter for help on specific time series data processing questions.

Time Series and SAS Data Sets

Introduction

To analyze data with the SAS System, data values must be stored in a SAS data set. A SAS data set is a matrix (or table) of data values organized into variables and observations.

The *variables* in a SAS data set label the columns of the data matrix, and the *observations* in a SAS data set are the rows of the data matrix. You can also think of a SAS data set as a kind of file, with the observations representing records in the file and the variables representing fields in the records. (See *SAS Language Reference: Concepts* for more information about SAS data sets.)

Usually, each observation represents the measurement of one or more variables for the individual subject or item observed. Often, the values of some of the variables in the data set are used to identify the individual subjects or items that the observations measure. These identifying variables are referred to as *ID variables*.

For many kinds of statistical analysis, only relationships among the variables are of interest, and the identity of the observations does not matter. ID variables might not be relevant in such a case.

However, for time series data the identity and order of the observations are crucial. A time series is a set of observations made at a succession of equally spaced points in time.

For example, if the data are monthly sales of a company's product, the variable measured is sales of the product and the unit observed is the operation of the company during each month. These observations can be identified by year and month. If the data are quarterly gross national product, the variable measured is final goods production and the unit observed is the economy during each quarter. These observations can be identified by year and quarter.

For time series data, the observations are identified and related to each other by their position in time. Since SAS does not assume any particular structure to the observations in a SAS data set, there are some special considerations needed when storing time series in a SAS data set.

The main considerations are how to associate dates with the observations and how to structure the data set so that SAS/ETS procedures and other SAS procedures recognize the observations of the data set as constituting time series. These issues are discussed in following sections.

Reading a Simple Time Series

Time series data can be recorded in many different ways. The section "Reading Time Series Data" on page 147 discusses some of the possibilities. The example below shows a simple case.

The following SAS statements read monthly values of the U.S. Consumer Price Index for June 1990 through July 1991. The data set USCPI is shown in Figure 4.1.

```
data uscpi;
   input year month cpi;
datalines;
1990  6 129.9
1990  7 130.4
1990  8 131.6

   ... more lines ...


proc print data=uscpi;
run;
```

Figure 4.1 Time Series Data

Obs	year	month	cpi
1	1990	6	129.9
2	1990	7	130.4
3	1990	8	131.6
4	1990	9	132.7
5	1990	10	133.5
6	1990	11	133.8
7	1990	12	133.8
8	1991	1	134.6
9	1991	2	134.8
10	1991	3	135.0
11	1991	4	135.2
12	1991	5	135.6
13	1991	6	136.0
14	1991	7	136.2

When a time series is stored in the manner shown by this example, the terms *series* and *variable* can be used interchangeably. There is one observation per row and one series/variable per column.

Dating Observations

The SAS System supports special date, datetime, and time values, which make it easy to represent dates, perform calendar calculations, and identify the time period of observations in a data set.

The preceding example uses the ID variables YEAR and MONTH to identify the time periods of the observations. For a quarterly data set, you might use YEAR and QTR as ID variables. A daily data set might have the ID variables YEAR, MONTH, and DAY. Clearly, it would be more convenient to have a single ID variable that could be used to identify the time period of observations, regardless of their frequency.

The following section, "SAS Date, Datetime, and Time Values" on page 95, discusses how the SAS System represents dates and times internally and how to specify date, datetime, and time values in a SAS program. The section "Reading Date and Datetime Values with Informats" on page 97 discusses how to read in date and time values from data records and how to control the display of date and datetime values in SAS output. Later sections discuss other issues concerning date and datetime values, specifying time intervals, data periodicity, and calendar calculations.

SAS date and datetime values and the other features discussed in the following sections are also described in *SAS Language Reference: Dictionary*. Reference documentation on these features is also provided in Chapter 5, "Date Intervals, Formats, and Functions."

SAS Date, Datetime, and Time Values

SAS Date Values

SAS software represents dates as the number of days since a reference date. The reference date, or date zero, used for SAS date values is 1 January 1960. For example, 3 February 1960 is represented by SAS as 33. The SAS date for 17 October 1991 is 11612.

SAS software correctly represents dates from the year 1582 to the year 20,000.

Dates represented in this way are called SAS *date values*. Any numeric variable in a SAS data set whose values represent dates in this way is called a SAS *date variable*.

Representing dates as the number of days from a reference date makes it easy for the computer to store them and perform calendar calculations, but these numbers are not meaningful to users. However, you never have to use SAS date values directly, since SAS automatically converts between this internal representation and ordinary ways of expressing dates, provided that you indicate the format with which you want the date values to be displayed. (Formatting of date values is explained in the section "Formatting Date and Datetime Values" on page 97.)

Century of Dates Represented with Two-Digit Year Values

SAS software informats, functions, and formats can process dates that are represented with two-digit year values. The century assumed for a two-digit year value can be controlled with the YEARCUTOFF= option in the OPTIONS statement. The YEARCUTOFF= system option controls how dates with two-digit year values are interpreted by specifying the first year of a 100-year span. The default value for the YEARCUTOFF= option is 1920. Thus by default the year '17' is interpreted as 2017, while the year '25' is interpreted as 1925. (See *SAS Language Reference: Dictionary* for more information about YEARCUTOFF=.)

SAS Date Constants

SAS date values are written in a SAS program by placing the dates in single quotes followed by a D. The date is represented by the day of the month, the three letter abbreviation of the month name, and the year.

For example, SAS reads the value '17OCT1991'D the same as 11612, the SAS date value for 17 October 1991. Thus, the following SAS statements print DATE=11612:

```
data _null_;
  date = '17oct1991'd;
  put date=;
run;
```

The year value can be given with two or four digits, so '17OCT91'D is the same as '17OCT1991'D.

SAS Datetime Values and Datetime Constants

To represent both the time of day and the date, SAS uses *datetime values*. SAS datetime values represent the date and time as the number of seconds the time is from a reference time. The reference time, or time zero, used for SAS datetime values is midnight, 1 January 1960. Thus, for example, the SAS datetime value for 17 October 1991 at 2:45 in the afternoon is 1003329900.

To specify datetime constants in a SAS program, write the date and time in single quotes followed by DT. To write the date and time in a SAS datetime constant, write the date part using the same syntax as for date constants, and follow the date part with the hours, the minutes, and the seconds, separating the parts with colons. The seconds are optional.

For example, in a SAS program you would write 17 October 1991 at 2:45 in the afternoon as '17OCT91:14:45'DT. SAS reads this as 1003329900. Table 4.1 shows some other examples of datetime constants.

Table 4.1 Examples of Datetime Constants

Datetime Constant	Time
'17OCT1991:14:45:32'DT	32 seconds past 2:45 p.m., 17 October 1991
'17OCT1991:12:5'DT	12:05 p.m., 17 October 1991
'17OCT1991:2:0'DT	2:00 a.m., 17 October 1991
'17OCT1991:0:0'DT	midnight, 17 October 1991

SAS Time Values

The SAS System also supports *time values*. SAS time values are just like datetime values, except that the date part is not given. To write a time value in a SAS program, write the time the same as for a datetime constant, but use T instead of DT. For example, 2:45:32 p.m. is written '14:45:32'T. Time values are represented by a number of seconds since midnight, so SAS reads '14:45:32'T as 53132.

SAS time values are not very useful for identifying time series, since usually both the date and the time of day are needed. Time values are not discussed further in this book.

Reading Date and Datetime Values with Informats

SAS provides a selection of *informats* for reading SAS date and datetime values from date and time values recorded in ordinary notations.

A SAS informat is an instruction that converts the values from a character-string representation into the internal numerical value of a SAS variable. Date informats convert dates from ordinary notations used to enter them to SAS date values; datetime informats convert date and time from ordinary notation to SAS datetime values.

For example, the following SAS statements read monthly values of the U.S. Consumer Price Index. Since the data are monthly, you could identify the date with the variables YEAR and MONTH, as in the previous example. Instead, in this example the time periods are coded as a three-letter month abbreviation followed by the year. The informat MONYY. is used to read month-year dates coded this way and to express them as SAS date values for the first day of the month, as follows:

```
data uscpi;
   input date : monyy7. cpi;
   format date monyy7.;
   label cpi = "US Consumer Price Index";
datalines;
jun1990 129.9
jul1990 130.4
aug1990 131.6

   ... more lines ...
```

The SAS System provides informats for most common notations for dates and times. See Chapter 5 for more information about the date and datetime informats available.

Formatting Date and Datetime Values

SAS provides *formats* to convert the internal representation of date and datetime values used by SAS to ordinary notations for dates and times. Several different formats are available for displaying dates and datetime values in most of the commonly used notations.

A SAS format is an instruction that converts the internal numerical value of a SAS variable to a character string that can be printed or displayed. Date formats convert SAS date values to a readable form; datetime formats convert SAS datetime values to a readable form.

In the preceding example, the variable DATE was set to the SAS date value for the first day of the month for each observation. If the data set USCPI were printed or otherwise displayed, the values shown for DATE would be the number of days since 1 January 1960. (See the "DATE with no format" column in Figure 4.2.) To display date values appropriately, use the FORMAT statement.

The following example processes the data set USCPI to make several copies of the variable DATE and uses a FORMAT statement to give different formats to these copies. The format cases shown are the MONYY7. format (for the DATE variable), the DATE9. format (for the DATE1 variable), and no format (for the DATE0 variable). The PROC PRINT output in Figure 4.2 shows the effect of the different formats on how the date values are printed.

```
data fmttest;
   set uscpi;
   date0 = date;
   date1 = date;
   label date  = "DATE with MONYY7. format"
         date1 = "DATE with DATE9. format"
         date0 = "DATE with no format";
   format date monyy7. date1 date9.;
run;

proc print data=fmttest label;
run;
```

Figure 4.2 SAS Date Values Printed with Different Formats

Obs	DATE with MONYY7. format	US Consumer Price Index	DATE with no format	DATE with DATE9. format
1	JUN1990	129.9	11109	01JUN1990
2	JUL1990	130.4	11139	01JUL1990
3	AUG1990	131.6	11170	01AUG1990
4	SEP1990	132.7	11201	01SEP1990
5	OCT1990	133.5	11231	01OCT1990
6	NOV1990	133.8	11262	01NOV1990
7	DEC1990	133.8	11292	01DEC1990
8	JAN1991	134.6	11323	01JAN1991
9	FEB1991	134.8	11354	01FEB1991
10	MAR1991	135.0	11382	01MAR1991
11	APR1991	135.2	11413	01APR1991
12	MAY1991	135.6	11443	01MAY1991
13	JUN1991	136.0	11474	01JUN1991
14	JUL1991	136.2	11504	01JUL1991

The appropriate format to use for SAS date or datetime valued ID variables depends on the sampling frequency or periodicity of the time series. Table 4.2 shows recommended formats for common data sampling frequencies and shows how the date '17OCT1991'D or the datetime value '17OCT1991:14:45:32'DT is displayed by these formats.

Table 4.2 Formats for Different Sampling Frequencies

ID values	Periodicity	FORMAT	Example
SAS date	annual	YEAR4.	1991
	quarterly	YYQC6.	1991:4
	monthly	MONYY7.	OCT1991
	weekly	WEEKDATX23.	Thursday, 17 Oct 1991
	daily	DATE9.	17OCT1991
SAS datetime	hourly	DATETIME10.	17OCT91:14
	minutes	DATETIME13.	17OCT91:14:45
	seconds	DATETIME16.	17OCT91:14:45:32

See Chapter 5, "Date Intervals, Formats, and Functions," for more information about the date and datetime formats available.

The Variables DATE and DATETIME

SAS/ETS procedures enable you to identify time series observations in many different ways to suit your needs. As discussed in preceding sections, you can use a combination of several ID variables, such as YEAR and MONTH for monthly data.

However, using a single SAS date or datetime ID variable is more convenient and enables you to take advantage of some features SAS/ETS procedures provide for processing ID variables. One such feature is automatic extrapolation of the ID variable to identify forecast observations. These features are discussed in following sections.

Thus, it is a good practice to include a SAS date or datetime ID variable in all the time series SAS data sets you create. It is also a good practice to always give the date or datetime ID variable a format appropriate for the data periodicity. (For information about creating SAS date and datetime values from multiple ID variables, see the section "Computing Dates from Calendar Variables" on page 119.)

You can assign a SAS date- or datetime-valued ID variable any name that conforms to SAS variable name requirements. However, you might find working with time series data in SAS easier and less confusing if you adopt the practice of always using the same name for the SAS date or datetime ID variable.

This book always names the date- or datetime-values ID variable DATE if it contains SAS date values or DATETIME if it contains SAS datetime values. This makes it easy to recognize the ID variable and also makes it easy to recognize whether this ID variable uses SAS date or datetime values.

Sorting by Time

Many SAS/ETS procedures assume the data are in chronological order. If the data are not in time order, you can use the SORT procedure to sort the data set. For example,

```
proc sort data=a;
   by date;
run;
```

There are many ways of coding the time ID variable or variables, and some ways do not sort correctly. If you use SAS date or datetime ID values as suggested in the preceding section, you do not need to be concerned with this issue. But if you encode date values in nonstandard ways, you need to consider whether your ID variables will sort.

SAS date and datetime values always sort correctly, as do combinations of numeric variables such as YEAR, MONTH, and DAY used together. Julian dates also sort correctly. (Julian dates are numbers of the form *yyddd*, where *yy* is the year and *ddd* is the day of the year. For example, 17 October 1991 has the Julian date value 91290.)

Calendar dates such as numeric values coded as *mmddyy* or *ddmmyy* do not sort correctly. Character variables that contain display values of dates, such as dates in the notation produced by SAS date formats, generally do not sort correctly.

Subsetting Data and Selecting Observations

It is often necessary to subset data for analysis. You might need to subset data to do the following:

- restrict the time range. For example, you want to perform a time series analysis using only recent data and ignoring observations from the distant past.
- select cross sections of the data. (See the section "Cross-Sectional Dimensions and BY Groups" on page 105.) For example, you have a data set with observations over time for each of several states, and you want to analyze the data for a single state.
- select particular kinds of time series from an interleaved-form data set. (See the section "Interleaved Time Series" on page 107.) For example, you have an output data set produced by the FORECAST procedure that contains both forecast and confidence limits observations, and you want to extract only the forecast observations.
- exclude particular observations. For example, you have an outlier in your time series, and you want to exclude this observation from the analysis.

You can subset data either by using the DATA step to create a subset data set or by using a WHERE statement with the SAS procedure that analyzes the data.

A typical WHERE statement used in a procedure has the following form:

```
proc arima data=full;
   where '31dec1993'd < date < '26mar1994'd;
   identify var=close;
run;
```

For complete reference documentation on the WHERE statement, see *SAS Language Reference: Dictionary*.

Subsetting SAS Data Sets

To create a subset data set, specify the name of the subset data set in the DATA statement, bring in the full data set with a SET statement, and specify the subsetting criteria with either subsetting IF statements or WHERE statements.

For example, suppose you have a data set that contains time series observations for each of several states. The following DATA step uses a WHERE statement to exclude observations with dates before 1970 and uses a subsetting IF statement to select observations for the state NC:

```
data subset;
   set full;
   where date >= '1jan1970'd;
   if state = 'NC';
run;
```

In this case, it makes no difference logically whether the WHERE statement or the IF statement is used, and you can combine several conditions in one subsetting statement. The following statements produce the same results as the previous example:

```
data subset;
   set full;
   if date >= '1jan1970'd & state = 'NC';
run;
```

The WHERE statement acts on the input data sets specified in the SET statement before observations are processed by the DATA step program, whereas the IF statement is executed as part of the DATA step program. If the input data set is indexed, using the WHERE statement can be more efficient than using the IF statement. However, the WHERE statement can refer only to variables in the input data set, not to variables computed by the DATA step program.

To subset the variables of a data set, use KEEP or DROP statements or use KEEP= or DROP= data set options. See *SAS Language Reference: Dictionary* for information about KEEP and DROP statements and SAS data set options.

For example, suppose you want to subset the data set as in the preceding example, but you want to include in the subset data set only the variables DATE, X, and Y. You could use the following statements:

```
data subset;
   set full;
   if date >= '1jan1970'd & state = 'NC';
   keep date x y;
run;
```

Using the WHERE Statement with SAS Procedures

Use the WHERE statement with SAS procedures to process only a subset of the input data set. For example, suppose you have a data set that contains monthly observations for each of several states, and you want to use the AUTOREG procedure to analyze data since 1970 for the state NC. You could use the following statements:

```
proc autoreg data=full;
   where date >= '1jan1970'd & state = 'NC';
  ... additional statements ...
run;
```

You can specify any number of conditions in the WHERE statement. For example, suppose that a strike created an outlier in May 1975, and you want to exclude that observation. You could use the following statements:

```
proc autoreg data=full;
   where date >= '1jan1970'd & state = 'NC'
       & date ^= '1may1975'd;
  ... additional statements ...
run;
```

Using SAS Data Set Options

You can use the OBS= and FIRSTOBS= data set options to subset the input data set.

For example, the following statements print observations 20 through 25 of the data set FULL:

```
proc print data=full(firstobs=20 obs=25);
run;
```

Figure 4.3 Partial Listing of Data Set FULL

Obs	date	state	i	x	y	close
20	21OCT1993	NC	20	0.44803	0.35302	0.44803
21	22OCT1993	NC	21	0.03186	1.67414	0.03186
22	23OCT1993	NC	22	-0.25232	-1.61289	-0.25232
23	24OCT1993	NC	23	0.42524	0.73112	0.42524
24	25OCT1993	NC	24	0.05494	-0.88664	0.05494
25	26OCT1993	NC	25	-0.29096	-1.17275	-0.29096

You can use KEEP= and DROP= data set options to exclude variables from the input data set. See *SAS Language Reference: Dictionary* for information about SAS data set options.

Storing Time Series in a SAS Data Set

This section discusses aspects of storing time series in SAS data sets. The topics discussed are the standard form of a time series data set, storing several series with different time ranges in the same data set, omitted observations, cross-sectional dimensions and BY groups, and interleaved time series.

Any number of time series can be stored in a SAS data set. Normally, each time series is stored in a separate variable. For example, the following statements augment the USCPI data set read in the previous example with values for the producer price index:

```
data usprice;
   input date : monyy7. cpi ppi;
   format date monyy7.;
   label cpi = "Consumer Price Index"
         ppi = "Producer Price Index";
datalines;
jun1990 129.9 114.3
jul1990 130.4 114.5
aug1990 131.6 116.5

   ... more lines ...


proc print data=usprice;
run;
```

Figure 4.4 Time Series Data Set Containing Two Series

Obs	date	cpi	ppi
1	JUN1990	129.9	114.3
2	JUL1990	130.4	114.5
3	AUG1990	131.6	116.5
4	SEP1990	132.7	118.4
5	OCT1990	133.5	120.8
6	NOV1990	133.8	120.1
7	DEC1990	133.8	118.7
8	JAN1991	134.6	119.0
9	FEB1991	134.8	117.2
10	MAR1991	135.0	116.2
11	APR1991	135.2	116.0
12	MAY1991	135.6	116.5
13	JUN1991	136.0	116.3
14	JUL1991	136.2	116.0

Standard Form of a Time Series Data Set

The simple way the CPI and PPI time series are stored in the USPRICE data set in the preceding example is termed the *standard form* of a time series data set. A time series data set in standard form has the following characteristics:

- The data set contains one variable for each time series.
- The data set contains exactly one observation for each time period.
- The data set contains an ID variable or variables that identify the time period of each observation.
- The data set is sorted by the ID variables associated with date time values, so the observations are in time sequence.
- The data are equally spaced in time. That is, successive observations are a fixed time interval apart, so the data set can be described by a single sampling interval such as hourly, daily, monthly, quarterly, yearly, and so forth. This means that time series with different sampling frequencies are not mixed in the same SAS data set.

Most SAS/ETS procedures that process time series expect the input data set to contain time series in this standard form, and this is the simplest way to store time series in SAS data sets. (The EXPAND and TIMESERIES procedures can be helpful in converting your data to this standard form.) There are more complex ways to represent time series in SAS data sets.

You can incorporate cross-sectional dimensions with BY groups, so that each BY group is like a standard form time series data set. This method is discussed in the section "Cross-Sectional Dimensions and BY Groups" on page 105.

You can interleave time series, with several observations for each time period identified by another ID variable. Interleaved time series data sets are used to store several series in the same SAS variable. Interleaved time

series data sets are often used to store series of actual values, predicted values, and residuals, or series of forecast values and confidence limits for the forecasts. This is discussed in the section "Interleaved Time Series" on page 107.

Several Series with Different Ranges

Different time series can have values recorded over different time ranges. Since a SAS data set must have the same observations for all variables, when time series with different ranges are stored in the same data set, missing values must be used for the periods in which a series is not available.

Suppose that in the previous example you did not record values for CPI before August 1990 and did not record values for PPI after June 1991. The USPRICE data set could be read with the following statements:

```
data usprice;
   input date : monyy7. cpi ppi;
   format date monyy7.;
datalines;
jun1990     . 114.3
jul1990     . 114.5
aug1990 131.6 116.5
sep1990 132.7 118.4
oct1990 133.5 120.8
nov1990 133.8 120.1
dec1990 133.8 118.7
jan1991 134.6 119.0
feb1991 134.8 117.2
mar1991 135.0 116.2
apr1991 135.2 116.0
may1991 135.6 116.5
jun1991 136.0 116.3
jul1991 136.2     .
;
```

The decimal points with no digits in the data records represent missing data and are read by SAS as missing value codes.

In this example, the time range of the USPRICE data set is June 1990 through July 1991, but the time range of the CPI variable is August 1990 through July 1991, and the time range of the PPI variable is June 1990 through June 1991.

SAS/ETS procedures ignore missing values at the beginning or end of a series. That is, the series is considered to begin with the first nonmissing value and end with the last nonmissing value.

Missing Values and Omitted Observations

Missing data can also occur within a series. Missing values that appear after the beginning of a time series and before the end of the time series are called *embedded missing values*.

Suppose that in the preceding example you did not record values for CPI for November 1990 and did not record values for PPI for both November 1990 and March 1991. The USPRICE data set could be read with the following statements:

```
data usprice;
   input date : monyy. cpi ppi;
   format date monyy.;
datalines;
jun1990     . 114.3
jul1990     . 114.5
aug1990 131.6 116.5
sep1990 132.7 118.4
oct1990 133.5 120.8
nov1990     .     .
dec1990 133.8 118.7
jan1991 134.6 119.0
feb1991 134.8 117.2
mar1991 135.0     .
apr1991 135.2 116.0
may1991 135.6 116.5
jun1991 136.0 116.3
jul1991 136.2     .
;
```

In this example, the series CPI has one embedded missing value, and the series PPI has two embedded missing values. The ranges of the two series are the same as before.

Note that the observation for November 1990 has missing values for both CPI and PPI; there is no data for this period. This is an example of a *missing observation*.

You might ask why the data record for this period is included in the example at all, since the data record contains no data. However, deleting the data record for November 1990 from the example would cause an *omitted observation* in the USPRICE data set. SAS/ETS procedures expect input data sets to contain observations for a contiguous time sequence. If you omit observations from a time series data set and then try to analyze the data set with SAS/ETS procedures, the omitted observations will cause errors. When all data are missing for a period, a missing observation should be included in the data set to preserve the time sequence of the series.

If observations are omitted from the data set, the EXPAND procedure can be used to fill in the gaps with missing values (or to interpolate nonmissing values) for the time series variables and with the appropriate date or datetime values for the ID variable.

Cross-Sectional Dimensions and BY Groups

Often, time series in a collection are related by a cross sectional dimension. For example, the national average U.S. consumer price index data shown in the previous example can be disaggregated to show price indexes for major cities. In this case, there are several related time series: CPI for New York, CPI for Chicago, CPI for Los Angeles, and so forth. When these time series are considered as one data set, the city whose price level is measured is a cross sectional dimension of the data.

There are two basic ways to store such related time series in a SAS data set. The first way is to use a standard form time series data set with a different variable for each series.

For example, the following statements read CPI series for three major U.S. cities:

```
data citycpi;
   input date : monyy7. cpiny cpichi cpila;
   format date monyy7.;
datalines;
nov1989  133.200  126.700  130.000
dec1989  133.300  126.500  130.600
jan1990  135.100  128.100  132.100

   ... more lines ...
```

The second way is to store the data in a time series cross-sectional form. In this form, the series for all cross sections are stored in one variable and a cross section ID variable is used to identify observations for the different series. The observations are sorted by the cross section ID variable and by time within each cross section.

The following statements indicate how to read the CPI series for U.S. cities in time series cross-sectional form:

```
data cpicity;
   length city $11;
   input city $11. date : monyy. cpi;
   format date monyy.;
datalines;
New York       JAN1990   135.100
New York       FEB1990   135.300
New York       MAR1990   136.600

   ... more lines ...


proc sort data=cpicity;
   by city date;
run;
```

When processing a time series cross sectional form data set with most SAS/ETS procedures, use the cross section ID variable in a BY statement to process the time series separately. The data set must be sorted by the cross section ID variable and sorted by date within each cross section. The PROC SORT step in the preceding example ensures that the CPICITY data set is correctly sorted.

When the cross section ID variable is used in a BY statement, each BY group in the data set is like a standard form time series data set. Thus, SAS/ETS procedures that expect a standard form time series data set can process time series cross sectional data sets when a BY statement is used, producing an independent analysis for each cross section.

It is also possible to analyze time series cross-sectional data jointly. The PANEL procedure (and the older TSCSREG procedure) expects the input data to be in the time series cross-sectional form described here. See Chapter 27, "The PANEL Procedure," for more information.

Interleaved Time Series

Normally, a time series data set has only one observation for each time period, or one observation for each time period within a cross section for a time series cross-sectional-form data set. However, it is sometimes useful to store several related time series in the same variable when the different series do not correspond to levels of a cross-sectional dimension of the data.

In this case, the different time series can be interleaved. An interleaved time series data set is similar to a time series cross-sectional data set, except that the observations are sorted differently and the ID variable that distinguishes the different time series does not represent a cross-sectional dimension.

Some SAS/ETS procedures produce interleaved output data sets. The interleaved time series form is a convenient way to store procedure output when the results consist of several different kinds of series for each of several input series. (Interleaved time series are also easy to process with plotting procedures. See the section "Plotting Time Series" on page 112.)

For example, the FORECAST procedure fits a model to each input time series and computes predicted values and residuals from the model. The FORECAST procedure then uses the model to compute forecast values beyond the range of the input data and also to compute upper and lower confidence limits for the forecast values.

Thus, the output from PROC FORECAST consists of up to five related time series for each variable forecast. The five resulting time series for each input series are stored in a single output variable with the same name as the series that is being forecast. The observations for the five resulting series are identified by values of the variable _TYPE_. These observations are interleaved in the output data set with observations for the same date grouped together.

The following statements show how to use PROC FORECAST to forecast the variable CPI in the USCPI data set. Figure 4.5 shows part of the output data set produced by PROC FORECAST and illustrates the interleaved structure of this data set.

```
proc forecast data=uscpi interval=month lead=12
              out=foreout outfull outresid;
   var cpi;
   id date;
run;

proc print data=foreout(obs=6);
run;
```

Figure 4.5 Partial Listing of Output Data Set Produced by PROC FORECAST

Obs	date	_TYPE_	_LEAD_	cpi
1	JUN1990	ACTUAL	0	129.900
2	JUN1990	FORECAST	0	130.817
3	JUN1990	RESIDUAL	0	-0.917
4	JUL1990	ACTUAL	0	130.400
5	JUL1990	FORECAST	0	130.678
6	JUL1990	RESIDUAL	0	-0.278

Observations with _TYPE_=ACTUAL contain the values of CPI read from the input data set. Observations with _TYPE_=FORECAST contain one-step-ahead predicted values for observations with dates in the range of the input series and contain forecast values for observations for dates beyond the range of the input series. Observations with _TYPE_=RESIDUAL contain the difference between the actual and one-step-ahead predicted values. Observations with _TYPE_=U95 and _TYPE_=L95 contain the upper and lower bounds, respectively, of the 95% confidence interval for the forecasts.

Using Interleaved Data Sets as Input to SAS/ETS Procedures

Interleaved time series data sets are not directly accepted as input by SAS/ETS procedures. However, it is easy to use a WHERE statement with any procedure to subset the input data and select one of the interleaved time series as the input.

For example, to analyze the residual series contained in the PROC FORECAST output data set with another SAS/ETS procedure, include a WHERE _TYPE_='RESIDUAL' statement. The following statements perform a spectral analysis of the residuals produced by PROC FORECAST in the preceding example:

```
proc spectra data=foreout out=spectout;
   var cpi;
   where _type_='RESIDUAL';
run;
```

Combined Cross Sections and Interleaved Time Series Data Sets

Interleaved time series output data sets produced from BY-group processing of time series cross-sectional input data sets have a complex structure that combines a cross-sectional dimension, a time dimension, and the values of the _TYPE_ variable. For example, consider the PROC FORECAST output data set produced by the following statements:

```
title "FORECAST Output Data Set with BY Groups";

proc forecast data=cpicity interval=month
              method=expo lead=2
              out=foreout outfull outresid;
   var cpi;
   id date;
   by city;
run;

proc print data=foreout(obs=6);
run;
```

The output data set FOREOUT contains many different time series in the single variable CPI. (The first few observations of FOREOUT are shown in Figure 4.6.) BY groups that are identified by the variable CITY contain the result series for the different cities. Within each value of CITY, the actual, forecast, residual, and confidence limits series are stored in interleaved form, with the observations for the different series identified by the values of _TYPE_.

Figure 4.6 Combined Cross Sections and Interleaved Time Series Data

FORECAST Output Data Set with BY Groups

Obs	city	date	_TYPE_	_LEAD_	cpi
1	Chicago	JAN90	ACTUAL	0	128.100
2	Chicago	JAN90	FORECAST	0	128.252
3	Chicago	JAN90	RESIDUAL	0	-0.152
4	Chicago	FEB90	ACTUAL	0	129.200
5	Chicago	FEB90	FORECAST	0	128.896
6	Chicago	FEB90	RESIDUAL	0	0.304

Output Data Sets of SAS/ETS Procedures

Some SAS/ETS procedures (such as PROC FORECAST) produce interleaved output data sets, and other SAS/ETS procedures produce standard form time series data sets. The form a procedure uses depends on whether the procedure is normally used to produce multiple result series for each of many input series in one step (as PROC FORECAST does).

For example, the ARIMA procedure can output actual series, forecast series, residual series, and confidence limit series just as the FORECAST procedure does. The PROC ARIMA output data set uses the standard form because PROC ARIMA is designed for the detailed analysis of one series at a time and so forecasts only one series at a time.

The following statements show the use of the ARIMA procedure to produce a forecast of the USCPI data set. Figure 4.7 shows part of the output data set that is produced by the ARIMA procedure's FORECAST statement. (The printed output from PROC ARIMA is not shown.) Compare the PROC ARIMA output data set shown in Figure 4.7 with the PROC FORECAST output data set shown in Figure 4.6.

```
title "PROC ARIMA Output Data Set";

proc arima data=uscpi;
   identify var=cpi(1);
   estimate q=1;
   forecast id=date interval=month
                    lead=12 out=arimaout;
run;

proc print data=arimaout(obs=6);
run;
```

Figure 4.7 Partial Listing of Output Data Set Produced by PROC ARIMA

PROC ARIMA Output Data Set

Obs	date	cpi	FORECAST	STD	L95	U95	RESIDUAL
1	JUN1990	129.9	.	.	.	.	.
2	JUL1990	130.4	130.368	0.36160	129.660	131.077	0.03168
3	AUG1990	131.6	130.881	0.36160	130.172	131.590	0.71909
4	SEP1990	132.7	132.354	0.36160	131.645	133.063	0.34584
5	OCT1990	133.5	133.306	0.36160	132.597	134.015	0.19421
6	NOV1990	133.8	134.046	0.36160	133.337	134.754	-0.24552

The output data set produced by the ARIMA procedure's FORECAST statement stores the actual values in a variable with the same name as the response series, stores the forecast series in a variable named FORECAST, stores the residuals in a variable named RESIDUAL, stores the 95% confidence limits in variables named L95 and U95, and stores the standard error of the forecast in the variable STD.

This method of storing several different result series as a standard form time series data set is simple and convenient. However, it works well only for a single input series. The forecast of a single series can be stored in the variable FORECAST. But if two series are forecast, two different FORECAST variables are needed.

The STATESPACE procedure handles this problem by generating forecast variable names FOR1, FOR2, and so forth. The SPECTRA procedure uses a similar method. Names such as FOR1, FOR2, RES1, RES2, and so forth require you to remember the order in which the input series are listed. This is why PROC FORECAST, which is designed to forecast a whole list of input series at once, stores its results in interleaved form.

Other SAS/ETS procedures are often used for a single input series but can also be used to process several series in a single step. Thus, they are not clearly like PROC FORECAST nor clearly like PROC ARIMA in the number of input series they are designed to work with. These procedures use a third method for storing multiple result series in an output data set. These procedures store output time series in standard form (as PROC ARIMA does) but require an OUTPUT statement to give names to the result series.

Time Series Periodicity and Time Intervals

A fundamental characteristic of time series data is how frequently the observations are spaced in time. How often the observations of a time series occur is called the *sampling frequency* or the *periodicity* of the series. For example, a time series with one observation each month has a monthly sampling frequency or monthly periodicity and so is called a monthly time series.

In SAS, data periodicity is described by specifying periodic *time intervals* into which the dates of the observations fall. For example, the SAS time interval MONTH divides time into calendar months.

Many SAS/ETS procedures enable you to specify the periodicity of the input data set with the INTERVAL= option. For example, specifying INTERVAL=MONTH indicates that the procedure should expect the ID variable to contain SAS date values, and that the date value for each observation should fall in a separate calendar month. The EXPAND procedure uses interval name values with the FROM= and TO= options to control the interpolation of time series from one periodicity to another.

SAS also uses time intervals in several other ways. In addition to indicating the periodicity of time series data sets, time intervals are used with the interval functions INTNX and INTCK and for controlling the plot axis and reference lines for plots of data over time.

Specifying Time Intervals

Intervals are specified in SAS by using *interval names* such as YEAR, QTR, MONTH, DAY, and so forth. Table 4.3 summarizes the basic types of intervals.

Table 4.3 Basic Interval Types

Name	Periodicity
YEAR	yearly
SEMIYEAR	semiannual
QTR	quarterly
MONTH	monthly
SEMIMONTH	1st and 16th of each month
TENDAY	1st, 11th, and 21st of each month
WEEK	weekly
WEEKDAY	daily ignoring weekend days
DAY	daily
HOUR	hourly
MINUTE	every minute
SECOND	every second

Interval names can be abbreviated in various ways. For example, you could specify monthly intervals as MONTH, MONTHS, MONTHLY, or just MON. SAS accepts all these forms as equivalent.

Interval names can also be qualified with a multiplier to indicate multi-period intervals. For example, biennial intervals are specified as YEAR2.

Interval names can also be qualified with a shift index to indicate intervals with different starting points. For example, fiscal years starting in July are specified as YEAR.7.

Intervals are classified as either date or datetime intervals. Date intervals are used with SAS date values, while datetime intervals are used with SAS datetime values. The interval types YEAR, SEMIYEAR, QTR, MONTH, SEMIMONTH, TENDAY, WEEK, WEEKDAY, and DAY are date intervals. HOUR, MINUTE, and SECOND are datetime intervals. Date intervals can be turned into datetime intervals for use with datetime values by prefixing the interval name with 'DT'. Thus DTMONTH intervals are like MONTH intervals but are used with datetime ID values instead of date ID values.

See Chapter 5, "Date Intervals, Formats, and Functions," for more information about specifying time intervals and for a detailed reference to the different kinds of intervals available.

Using Intervals with SAS/ETS Procedures

SAS/ETS procedures use the date or datetime interval and the ID variable in the following ways:

- to validate the data periodicity. The ID variable is used to check the data and verify that successive observations have valid ID values that correspond to successive time intervals.
- to check for gaps in the input observations. For example, if INTERVAL=MONTH and an input observation for January 1990 is followed by an observation for April 1990, there is a gap in the input data with two omitted observations.
- to label forecast observations in the output data set. The values of the ID variable for the forecast observations after the end of the input data set are extrapolated according to the frequency specifications of the INTERVAL= option.

Time Intervals, the Time Series Forecasting System, and the Time Series Viewer

Time intervals are used in the Time Series Forecasting System and Time Series Viewer to identify the number of seasonal cycles or seasonality associated with a DATE, DATETIME, or TIME ID variable. For example, monthly time series have a seasonality of 12 because there are 12 months in a year; quarterly time series have a seasonality of 4 because there are four quarters in a year. The seasonality is used to analyze seasonal properties of time series data and to estimate seasonal forecasting methods.

Plotting Time Series

This section discusses SAS procedures that are available for plotting time series data, but it covers only certain aspects of the use of these procedures with time series data.

The Time Series Viewer displays and analyzes time series plots for time series data sets that do not contain cross sections. See Chapter 54, "Getting Started with Time Series Forecasting."

The SGPLOT procedure produces high resolution color graphics plots. See the *SAS ODS Graphics: Procedures Guide* and *SAS/GRAPH: Reference* for more information.

Using the Time Series Viewer

The following command starts the Time Series Viewer to display the plot of CPI in the USCPI data set against DATE. (The USCPI data set was shown in the previous example; the time series used in the following example contains more observations than previously shown.)

```
tsview data=uscpi var=cpi timeid=date
```

The TSVIEW DATA= option specifies the data set to be viewed; the VAR= option specifies the variable that contains the time series observations; the TIMEID= option specifies the time series ID variable.

The Time Series Viewer can also be invoked by selecting **Solutions►Analyze►Time Series Viewer** from the menu in the SAS Display Manager.

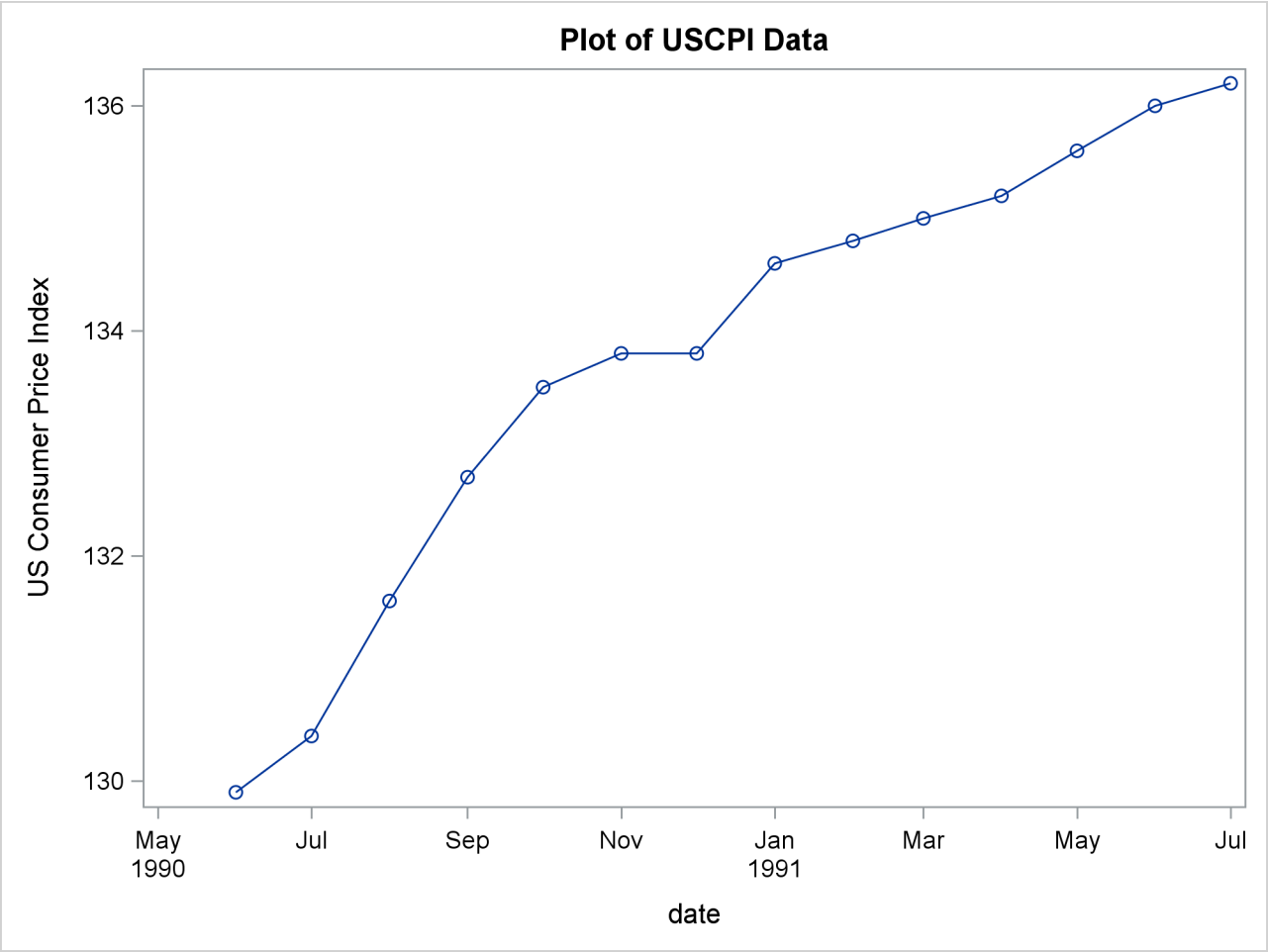
Using PROC SGPLOT

The following statements use the SGPLOT procedure to plot CPI in the USCPI data set against DATE. (The USCPI data set was shown in a previous example; the data set plotted in the following example contains more observations than shown previously.)

```
title "Plot of USCPI Data";
proc sgplot data=uscpi;
   series x=date y=cpi / markers;
run;
```

The plot is shown in Figure 4.8.

Figure 4.8 Plot of Monthly CPI Over Time



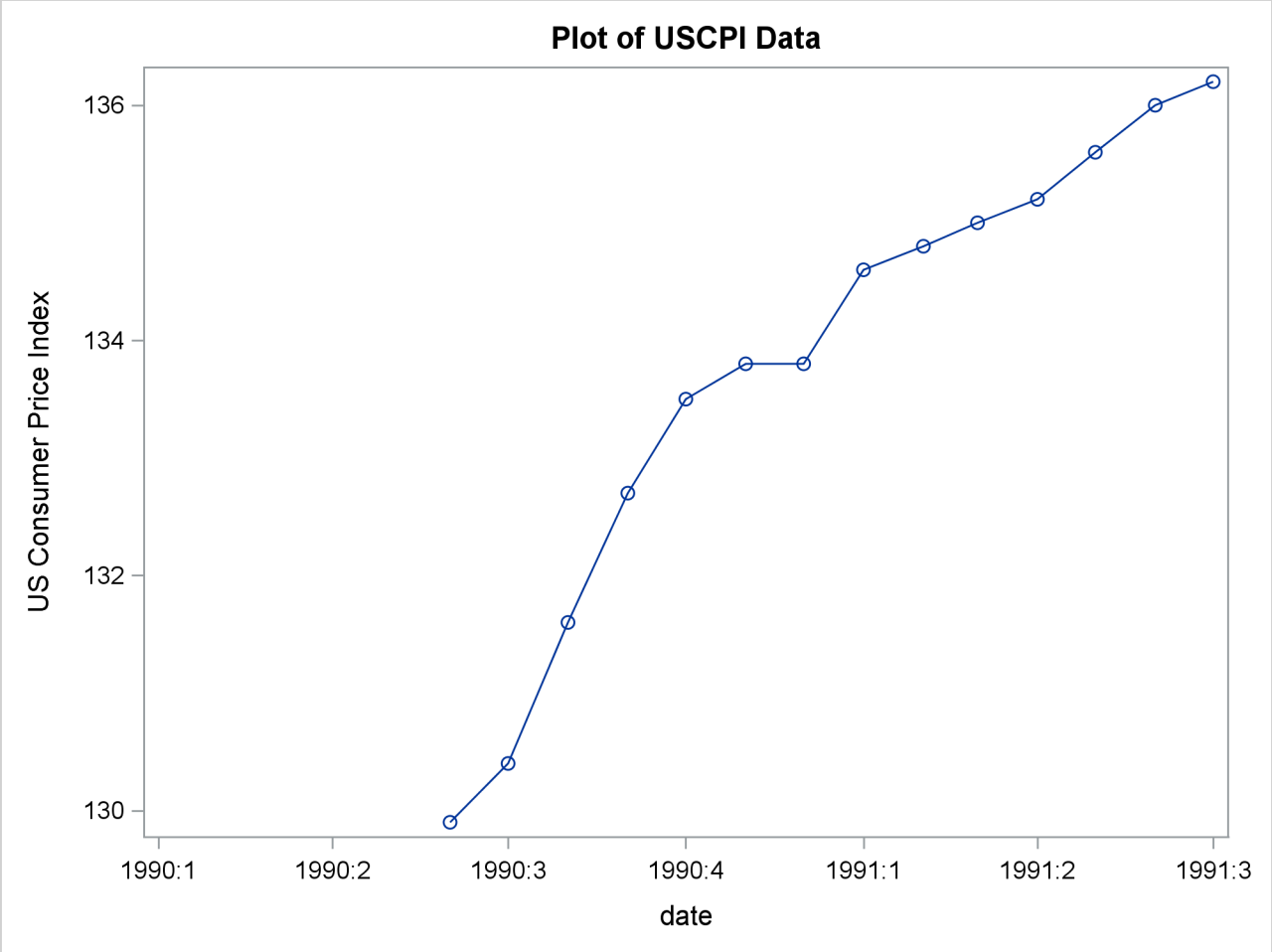
Controlling the Time Axis: Tick Marks and Reference Lines

It is possible to control the spacing of the tick marks on the time axis. The following statements use the XAXIS statement to tell PROC SGPLOT to mark the axis at the start of each quarter:

```
proc sgplot data=uscpi;
   series x=date y=cpi / markers;
   format date yyqc.;
   xaxis values=('1jan90'd to '1jul91'd by qtr);
run;
```

The plot is shown in Figure 4.9.

Figure 4.9 Plot of Monthly CPI Over Time



Overlay Plots of Different Variables

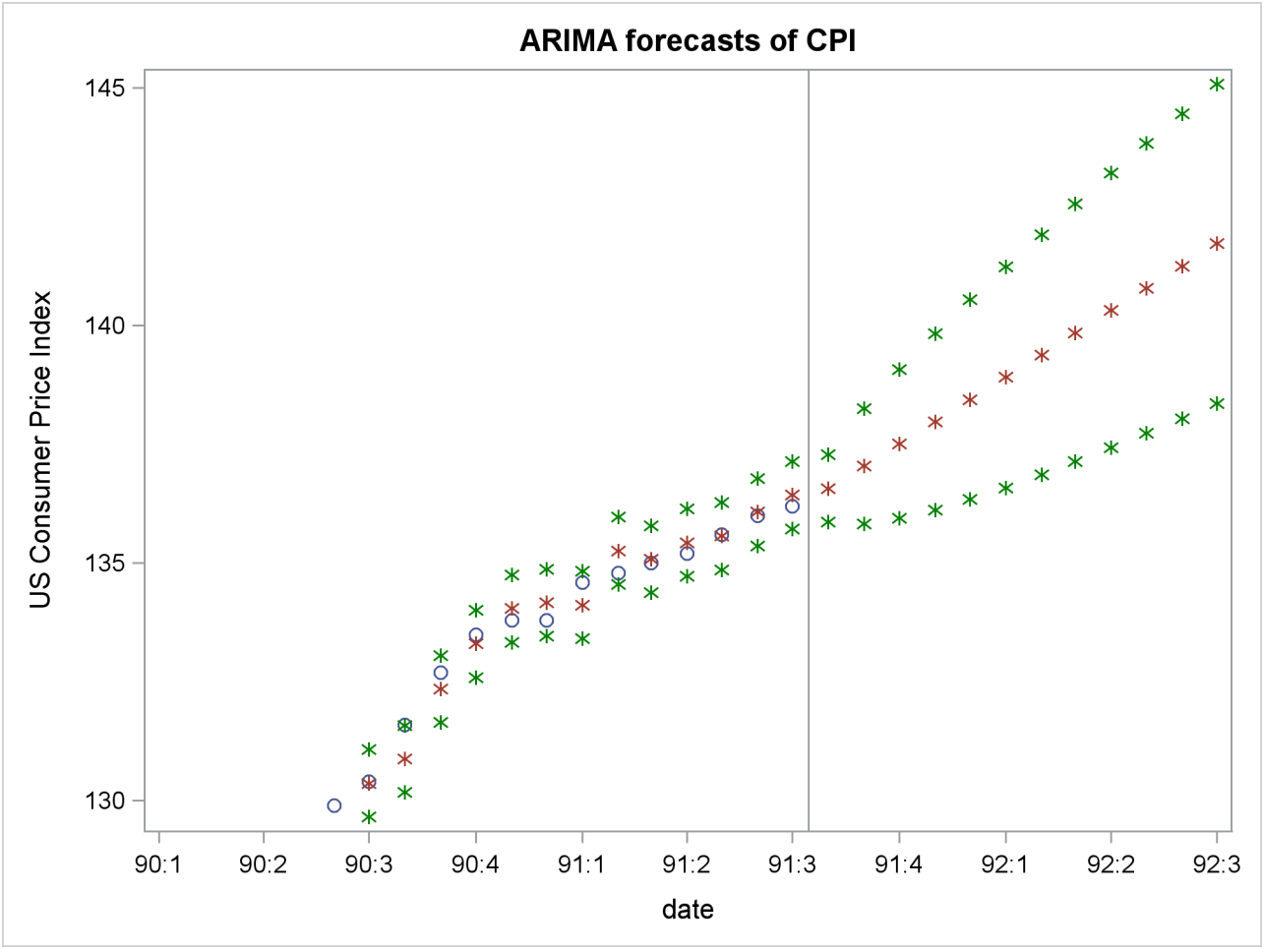
You can plot two or more series stored in different variables on the same graph by specifying multiple plot requests in one SGPLOT statement.

For example, the following statements plot the CPI, FORECAST, L95, and U95 variables produced by PROC ARIMA in a previous example. A reference line is drawn to mark the start of the forecast period. Quarterly tick marks with YYQC format date values are used.

```
title "ARIMA Forecasts of CPI";

proc arima data=uscpi;
   identify var=cpi(1);
   estimate q=1;
   forecast id=date interval=month lead=12 out=arimaout;
run;

title "ARIMA forecasts of CPI";
proc sgplot data=arimaout noautolegend;
   scatter x=date y=cpi;
   scatter x=date y=forecast / markerattrs=(symbol=asterisk);
   scatter x=date y=l95 / markerattrs=(symbol=asterisk color=green);
   scatter x=date y=u95 / markerattrs=(symbol=asterisk color=green);
   format date yyqc4.;
   xaxis values=('1jan90'd to '1jul92'd by qtr);
   refline '15jul91'd / axis=x;
run;
```

The plot is shown in Figure 4.10.

Figure 4.10 Plot of ARIMA Forecast



Overlay Plots of Interleaved Series

You can also plot several series on the same graph when the different series are stored in the same variable in interleaved form. Plot interleaved time series by using the values of the ID variable in GROUP= option to distinguish the different series.

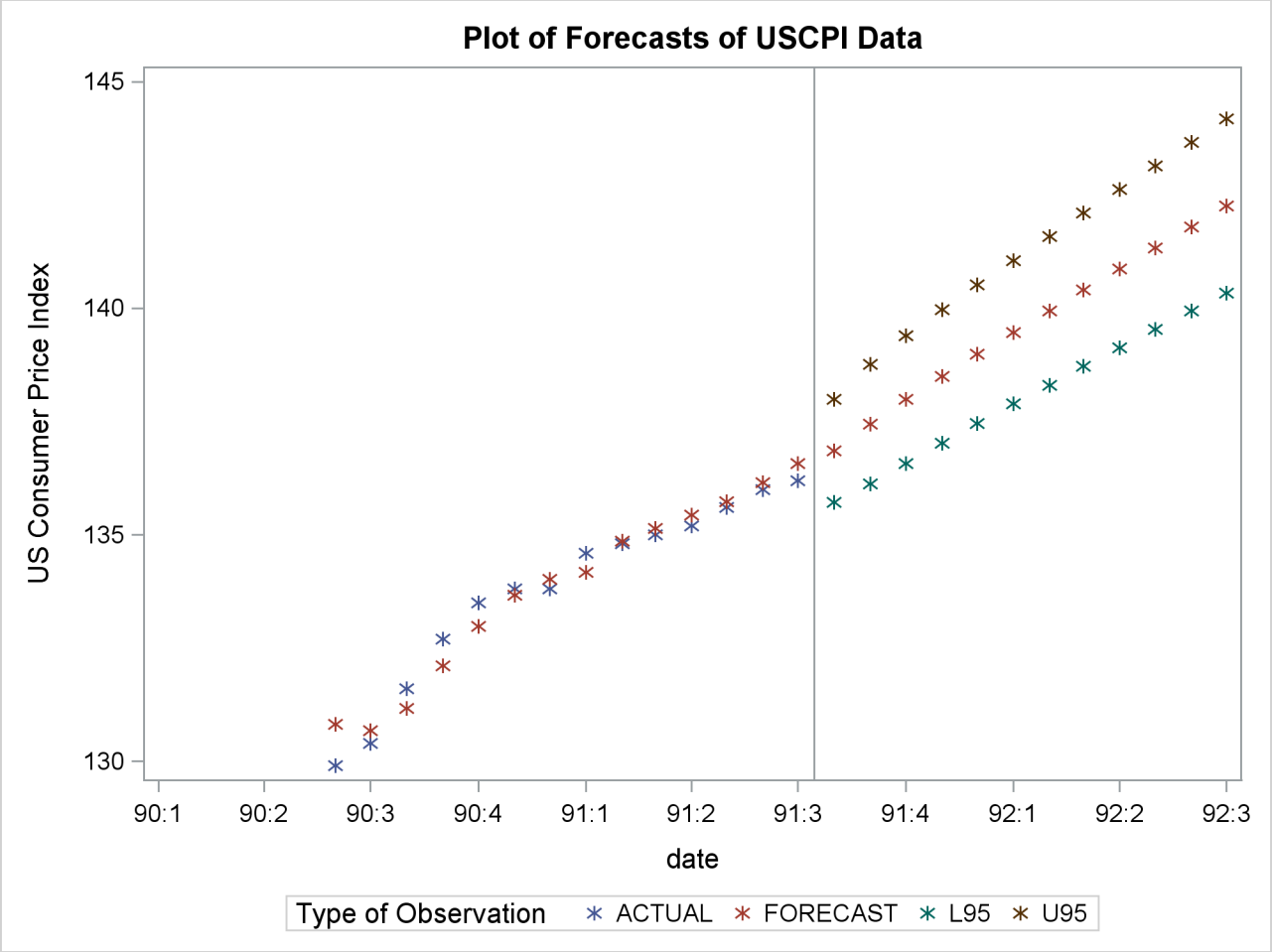
The following example plots the output data set produced by PROC FORECAST in a previous example. Since the residual series has a different scale than the other series, it is excluded from the plot with a WHERE statement.

The _TYPE_ variable is used in the PLOT statement to identify the different series and to select the SCATTER statements to use for each plot.

```
title "Plot of Forecasts of USCPI Data";

proc forecast data=uscpi interval=month lead=12
              out=foreout outfull outresid;
   var cpi;
   id date;
run;

proc sgplot data=foreout;
   where _type_ ^= 'RESIDUAL';
   scatter x=date y=cpi / group=_type_ markerattrs=(symbol=asterisk);
   format date yyqc4.;
   xaxis values=('1jan90'd to '1jul92'd by qtr);
   refline '15jul91'd / axis=x;
run;
```

The plot is shown in Figure 4.11.

Figure 4.11 Plot of Forecast



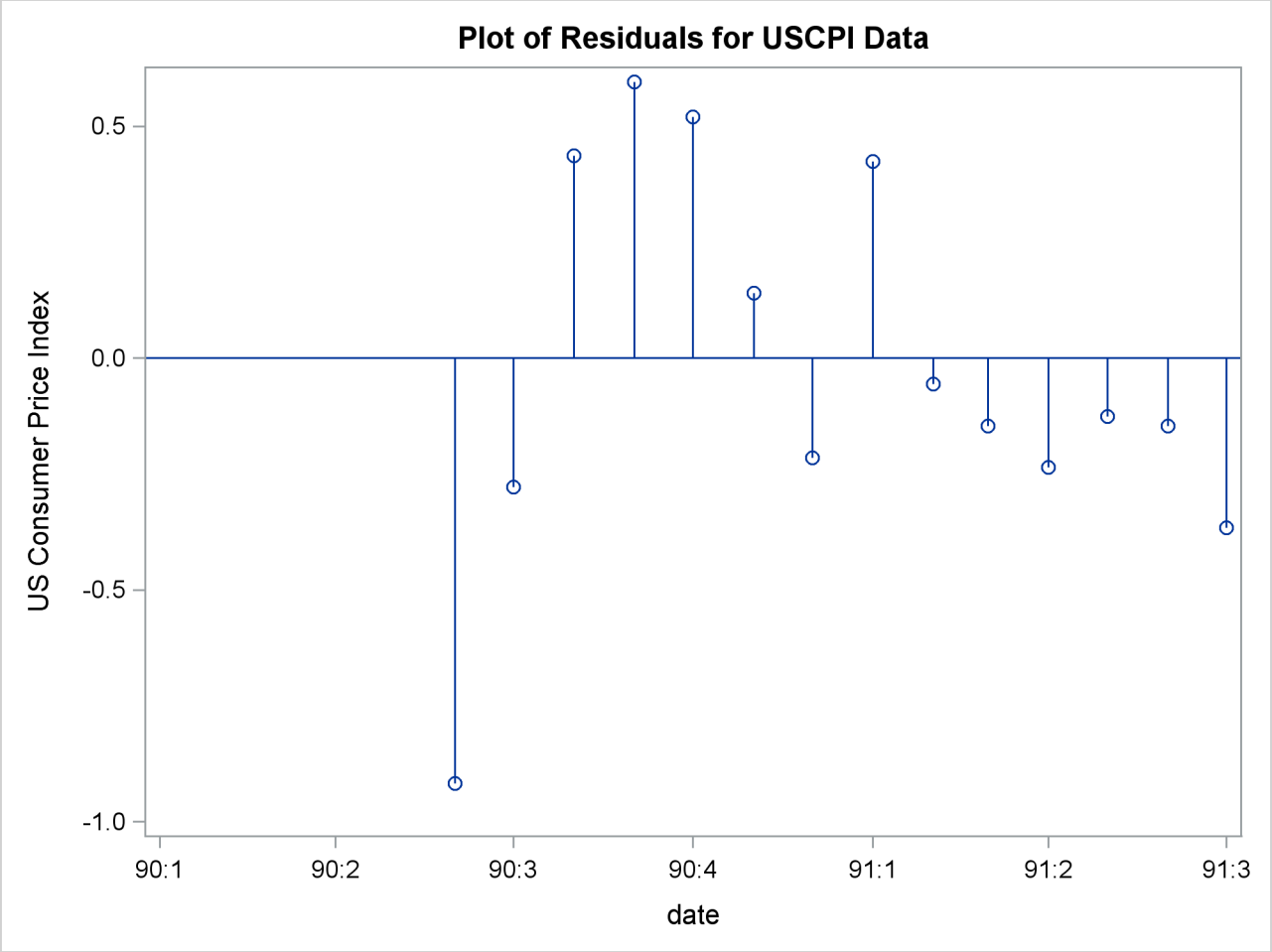
Residual Plots

The following example plots the residuals series that was excluded from the plot in the previous example. The NEEDLE statement specifies a needle plot, so that each residual point is plotted as a vertical line showing deviation from zero.

```
title "Plot of Residuals for USCPI Data";
proc sgplot data=foreout;
   where _type_ = 'RESIDUAL';
   needle x=date y=cpi / markers;
   format date yyqc4.;
   xaxis values=('1jan90'd to '1jul91'd by qtr);
run;
```

The plot is shown in Figure 4.12.

Figure 4.12 Plot of Residuals



Using PROC GPLOT

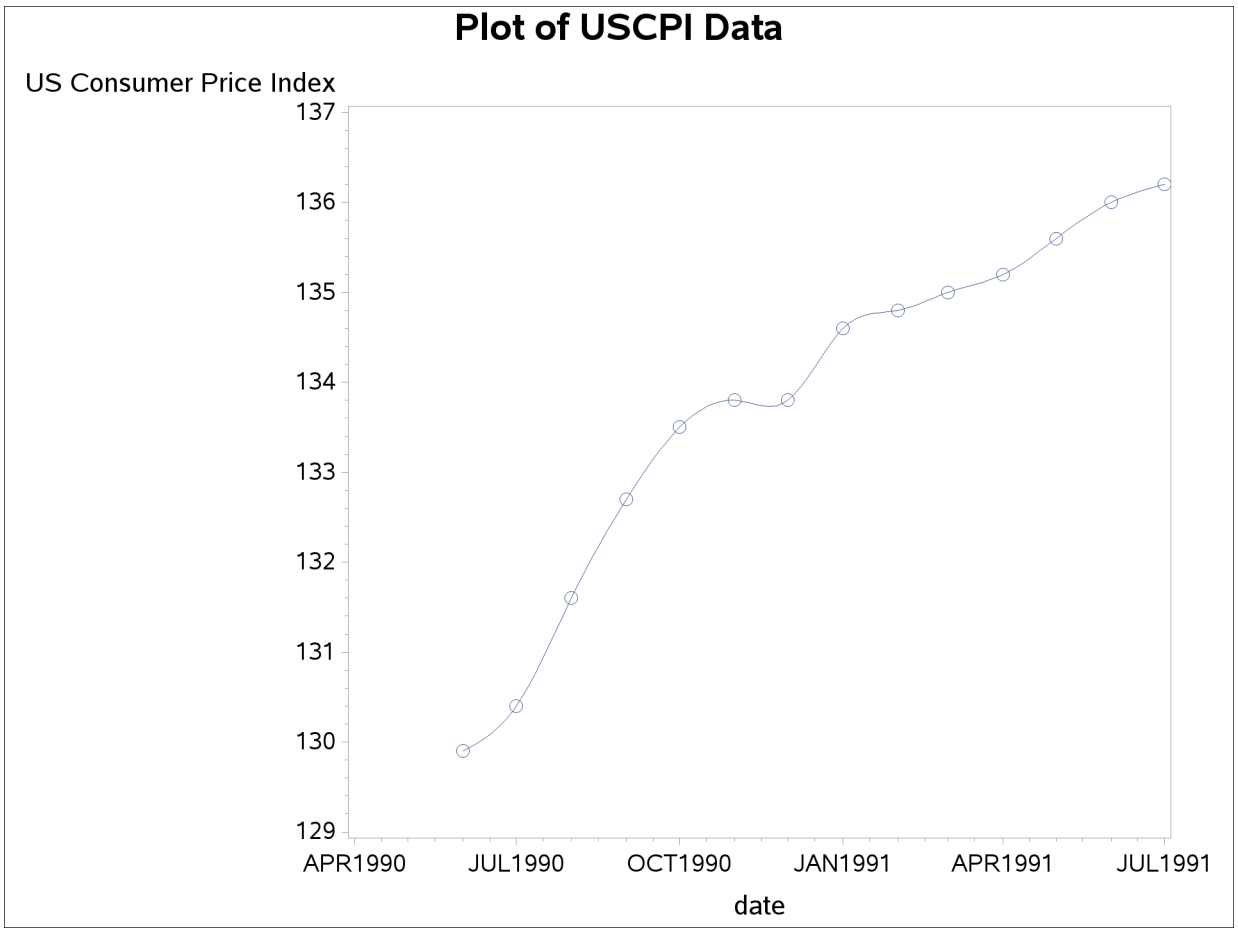
The GPLOT procedure in SAS/GRAPH software can also be used to plot time series data, although the newer SGPLOT procedure is easier to use.

The following is an example of how GPLOT can be used to produce a plot similar to the graph produced by PROC SGPLOT in the preceding section.

```
title "Plot of USCPI Data";
proc gplot data=uscpi;
   symbol i=spline v=circle h=2;
   plot cpi * date;
run;
```

The plot is shown in Figure 4.13.

Figure 4.13 Plot of Monthly CPI Over Time



For more information about the GPLOT procedure, see *SAS/GRAPH: Reference*.

Calendar and Time Functions

Calendar and time functions convert calendar and time variables such as YEAR, MONTH, DAY, and HOUR, MINUTE, SECOND into SAS date or datetime values, and vice versa.

The SAS calendar and time functions are DATEJUL, DATEPART, DAY, DHMS, HMS, HOUR, JULDATE, MDY, MINUTE, MONTH, QTR, SECOND, TIMEPART, WEEKDAY, YEAR, and YYQ. See *SAS Language Reference: Dictionary* for more details about these functions.

Computing Dates from Calendar Variables

The MDY function converts MONTH, DAY, and YEAR values to a SAS date value. For example, MDY(2010,17,91) returns the SAS date value '17OCT2010'D.

The YYQ function computes the SAS date for the first day of a quarter. For example, YYQ(2010,4) returns the SAS date value '1OCT2010'D.

The DATEJUL function computes the SAS date for a Julian date. For example, DATEJUL(91290) returns the SAS date '17OCT2010'D.

The YYQ and MDY functions are useful for creating SAS date variables when the ID values recorded in the data are year and quarter; year and month; or year, month, and day.

For example, the following statements read quarterly data from records in which dates are coded as separate year and quarter values. The YYQ function is used to compute the variable DATE.

```
data usecon;
   input year qtr gnp;
   date = yyq( year, qtr );
   format date yyqc.;
datalines;
1990 1 5375.4
1990 2 5443.3
1990 3 5514.6

   ... more lines ...
```

The monthly USCPI data shown in a previous example contained time ID values represented in the MONYY format. If the data records instead contain separate year and month values, the data can be read in and the DATE variable computed with the following statements:

```
data uscpi;
   input month year cpi;
   date = mdy( month, 1, year );
   format date monyy.;
datalines;
 6 90 129.9
 7 90 130.4
 8 90 131.6

   ... more lines ...
```

Computing Calendar Variables from Dates

The functions YEAR, MONTH, DAY, WEEKDAY, and JULDATE compute calendar variables from SAS date values.

Returning to the example of reading the USCPI data from records that contain date values represented in the MONYY format, you can find the month and year of each observation from the SAS dates of the observations by using the following statements.

```
data uscpi;
   input date monyy7. cpi;
   format date monyy7.;
   year  = year( date );
```

```
   month = month( date );
datalines;
jun1990 129.9
jul1990 130.4
aug1990 131.6

   ... more lines ...
```

Converting between Date, Datetime, and Time Values

The DATEPART function computes the SAS date value for the date part of a SAS datetime value. The TIMEPART function computes the SAS time value for the time part of a SAS datetime value.

The HMS function computes SAS time values from HOUR, MINUTE, and SECOND time variables. The DHMS function computes a SAS datetime value from a SAS date value and HOUR, MINUTE, and SECOND time variables.

See the section "SAS Date, Time, and Datetime Functions" on page 171 for more information about these functions.

Computing Datetime Values

To compute datetime ID values from calendar and time variables, first compute the date and then compute the datetime with DHMS.

For example, suppose you read tri-hourly temperature data with time recorded as YEAR, MONTH, DAY, and HOUR. The following statements show how to compute the ID variable DATETIME:

```
data weather;
   input year month day hour temp;
   datetime = dhms( mdy( month, day, year ), hour, 0, 0 );
   format datetime datetime10.;
datalines;
91 10 16 21  61
91 10 17  0  56
91 10 17  3  53
91 10 17  6  54

   ... more lines ...
```

Computing Calendar and Time Variables

The functions HOUR, MINUTE, and SECOND compute time variables from SAS datetime values. The DATEPART function and the date-to-calendar variables functions can be combined to compute calendar variables from datetime values.

For example, suppose the date and time of the tri-hourly temperature data in the preceding example were recorded as datetime values in the datetime format. The following statements show how to compute the YEAR, MONTH, DAY, and HOUR of each observation and include these variables in the SAS data set:

```
data weather;
   input datetime : datetime13. temp;
   format datetime datetime10.;
   hour = hour( datetime );
   date = datepart( datetime );
   year = year( date );
   month = month( date );
   day = day( date );
datalines;
16oct91:21:00  61
17oct91:00:00  56
17oct91:03:00  53
17oct91:06:00  54

   ... more lines ...
```

Interval Functions INTNX and INTCK

The SAS interval functions INTNX and INTCK perform calculations with date values, datetime values, and time intervals. They can be used for calendar calculations with SAS date values to increment date values or datetime values by intervals and to count time intervals between dates.

The INTNX function increments dates by intervals. INTNX computes the date or datetime of the start of the interval a specified number of intervals from the interval that contains a given date or datetime value.

The form of the INTNX function is

INTNX *(interval, from, n < , alignment >)* **;**

The arguments to the INTNX function are as follows:

interval
is a character constant or variable that contains an interval name

from
is a SAS date value (for date intervals) or datetime value (for datetime intervals)

n
is the number of intervals to increment from the interval that contains the *from* value

alignment
controls the alignment of SAS dates, within the interval, used to identify output observations. Allowed values are BEGINNING, MIDDLE, END, and SAMEDAY.

The number of intervals to increment, *n*, can be positive, negative, or zero.

For example, the statement NEXTMON=INTNX('MONTH',DATE,1) assigns to the variable NEXTMON the date of the first day of the month following the month that contains the value of DATE. Thus INTNX('MONTH','21OCT2007'D,1) returns the date 1 November 2007.

The INTCK function counts the number of interval boundaries between two date values or between two datetime values.

The form of the INTCK function is

INTCK *(interval, from, to)* **;**

The arguments of the INTCK function are as follows:

interval
is a character constant or variable that contains an interval name

from
is the starting date value (for date intervals) or datetime value (for datetime intervals)

to
is the ending date value (for date intervals) or datetime value (for datetime intervals)

For example, the statement NEWYEARS=INTCK('YEAR',DATE1,DATE2) assigns to the variable NEWYEARS the number of New Year's Days between the two dates.

Incrementing Dates by Intervals

Use the INTNX function to increment dates by intervals. For example, suppose you want to know the date of the start of the week that is six weeks from the week of 17 October 1991. The function INTNX('WEEK','17OCT91'D,6) returns the SAS date value '24NOV1991'D.

One practical use of the INTNX function is to generate periodic date values. For example, suppose the monthly U.S. Consumer Price Index data in a previous example were recorded without any time identifier on the data records. Given that you know the first observation is for June 1990, the following statements use the INTNX function to compute the ID variable DATE for each observation:

```
data uscpi;
   input cpi;
   date = intnx( 'month', '1jun1990'd, _n_-1 );
   format date monyy7.;
datalines;
129.9
130.4
131.6

   ... more lines ...
```

The automatic variable _N_ counts the number of times the DATA step program has executed; in this case _N_ contains the observation number. Thus _N_–1 is the increment needed from the first observation date. Alternatively, you could increment from the month before the first observation, in which case the INTNX function in this example would be written INTNX('MONTH','1MAY1990'D,_N_).

Alignment of SAS Dates

Any date within the time interval that corresponds to an observation of a periodic time series can serve as an ID value for the observation. For example, the USCPI data in a previous example might have been recorded with dates at the 15th of each month. The person recording the data might reason that since the CPI values are monthly averages, midpoints of the months might be the appropriate ID values.

However, as far as SAS/ETS procedures are concerned, what is important about monthly data is the month of each observation, not the exact date of the ID value. If you indicate that the data are monthly (with an INTERVAL=MONTH) option, SAS/ETS procedures ignore the day of the month in processing the ID variable. The MONYY format also ignores the day of the month.

Thus, you could read in the monthly USCPI data with mid-month DATE values by using the following statements:

```
data uscpi;
   input date : date9. cpi;
   format date monyy7.;
datalines;
15jun1990 129.9
15jul1990 130.4
15aug1990 131.6

   ... more lines ...
```

The results of using this version of the USCPI data set for analysis with SAS/ETS procedures would be the same as with first-of-month values for DATE. Although you can use any date within the interval as an ID value for the interval, you might find working with time series in SAS less confusing if you always use date ID values normalized to the start of the interval.

For some applications it might be preferable to use end of period dates, such as 31Jan1994, 28Feb1994, 31Mar1994, . . . , 31Dec1994. For other applications, such as plotting time series, it might be more convenient to use interval midpoint dates to identify the observations.

(Some SAS/ETS procedures provide an ALIGN= option to control the alignment of dates for output time series observations. In addition, the INTNX library function supports an optional argument to specify the alignment of the returned date value.)

To normalize date values to the start of intervals, use the INTNX function with a 0 increment. The INTNX function with an increment of 0 computes the date of the first day of the interval (or the first second of the interval for datetime values).

For example, INTNX('MONTH','17OCT1991'D,0,'BEG') returns the date '1OCT1991'D.

The following statements show how the preceding example can be changed to normalize the mid-month DATE values to first-of-month and end-of-month values. For exposition, the first-of-month value is transformed back into a middle-of-month value.

```
data uscpi;
   input date : date9. cpi;
   format date monyy7.;
   monthbeg = intnx( 'month', date, 0, 'beg' );
   midmonth = intnx( 'month', monthbeg, 0, 'mid' );
   monthend = intnx( 'month', date, 0, 'end' );
```

```
datalines;
15jun1990 129.9
15jul1990 130.4
15aug1990 131.6

   ... more lines ...
```

If you want to compute the date of a particular day within an interval, you can use calendar functions, or you can increment the starting date of the interval by a number of days. The following example shows three ways to compute the seventh day of the month:

```
data test;
   set uscpi;
   mon07_1 = mdy( month(date), 7, year(date) );
   mon07_2 = intnx( 'month', date, 0, 'beg' ) + 6;
   mon07_3 = intnx( 'day', date, 6 );
run;
```

Computing the Width of a Time Interval

To compute the width of a time interval, subtract the ID value of the start of the next interval from the ID value of the start of the current interval. If the ID values are SAS dates, the width is in days. If the ID values are SAS datetime values, the width is in seconds.

For example, the following statements show how to add a variable WIDTH to the USCPI data set that contains the number of days in the month for each observation:

```
data uscpi;
   input date : date9. cpi;
   format date monyy7.;
   width = intnx( 'month', date, 1 ) - intnx( 'month', date, 0 );
datalines;
15jun1990 129.9
15jul1990 130.4
15aug1990 131.6
15sep1990 132.7

   ... more lines ...
```

Computing the Ceiling of an Interval

To shift a date to the start of the next interval if it is not already at the start of an interval, subtract 1 from the date and use INTNX to increment the date by 1 interval.

For example, the following statements add the variable NEWYEAR to the monthly USCPI data set. The variable NEWYEAR contains the date of the next New Year's Day. NEWYEAR contains the same value as DATE when the DATE value is the start of year and otherwise contains the date of the start of the next year.

```
data test;
   set uscpi;
   newyear = intnx( 'year', date - 1, 1 );
   format newyear date.;
run;
```

Counting Time Intervals

Use the INTCK function to count the number of interval boundaries between two dates.

Note that the INTCK function counts the number of times the beginning of an interval is reached in moving from the first date to the second. It does not count the number of complete intervals between two dates. Following are two examples:

- The function INTCK('MONTH','1JAN1991'D,'31JAN1991'D) returns 0, since the two dates are within the same month.
- The function INTCK('MONTH','31JAN1991'D,'1FEB1991'D) returns 1, since the two dates lie in different months that are one month apart.

When the first date is later than the second date, INTCK returns a negative count. For example, the function INTCK('MONTH','1FEB1991'D,'31JAN1991'D) returns –1.

The following example shows how to use the INTCK function with shifted interval specifications to count the number of Sundays, Mondays, Tuesdays, and so forth, in each month. The variables NSUNDAY, NMONDAY, NTUESDAY, and so forth, are added to the USCPI data set.

```
data uscpi;
   set uscpi;
   d0 = intnx( 'month', date, 0 ) - 1;
   d1 = intnx( 'month', date, 1 ) - 1;
   nSunday  = intck( 'week.1', d0, d1 );
   nMonday  = intck( 'week.2', d0, d1 );
   nTuesday = intck( 'week.3', d0, d1 );
   nWedday  = intck( 'week.4', d0, d1 );
   nThurday = intck( 'week.5', d0, d1 );
   nFriday  = intck( 'week.6', d0, d1 );
   nSatday  = intck( 'week.7', d0, d1 );
   drop d0 d1;
run;
```

Since the INTCK function counts the number of interval beginning dates between two dates, the number of Sundays is computed by counting the number of week boundaries between the last day of the previous month and the last day of the current month. To count Mondays, Tuesdays, and so forth, shifted week intervals are used. The interval type WEEK.2 specifies weekly intervals starting on Mondays, WEEK.3 specifies weeks starting on Tuesdays, and so forth.

Checking Data Periodicity

Suppose you have a time series data set and you want to verify that the data periodicity is correct, the observations are dated correctly, and the data set is sorted by date. You can use the INTCK function to compare the date of the current observation with the date of the previous observation and verify that the dates fall into consecutive time intervals.

For example, the following statements verify that the data set USCPI is a correctly dated monthly data set. The RETAIN statement is used to hold the date of the previous observation, and the automatic variable _N_ is used to start the verification process with the second observation.

```
data _null_;
   set uscpi;
   retain prevdate;
   if _n_ > 1 then
      if intck( 'month', prevdate, date ) ^= 1 then
         put "Bad date sequence at observation number " _n_;
   prevdate = date;
run;
```

Filling In Omitted Observations in a Time Series Data Set

Most SAS/ETS procedures expect input data to be in the standard form, with no omitted observations in the sequence of time periods. When data are missing for a time period, the data set should contain a missing observation, in which all variables except the ID variables have missing values.

You can replace omitted observations in a time series data set with missing observations with the EXPAND procedure.

The following statements create a monthly data set, OMITTED, from data lines that contain records for an intermittent sample of months. (Data values are not shown.) The OMITTED data set is sorted to make sure it is in time order.

```
data omitted;
   input date : monyy7. x y z;
   format date monyy7.;
datalines;
jan1991  ...
mar1991  ...
apr1991  ...
jun1991  ...
 ... etc. ...
;

proc sort data=omitted;
   by date;
run;
```

This data set is converted to a standard form time series data set by the following PROC EXPAND step. The TO= option specifies that monthly data is to be output, while the METHOD=NONE option specifies that no

interpolation is to be performed, so that the variables X, Y, and Z in the output data set STANDARD will have missing values for the omitted time periods that are filled in by the EXPAND procedure.

```
proc expand data=omitted
            out=standard
            to=month
            method=none;
   id date;
run;
```

Using Interval Functions for Calendar Calculations

With a little thought, you can come up with a formula that involves INTNX and INTCK functions and different interval types to perform almost any calendar calculation.

For example, suppose you want to know the date of the third Wednesday in the month of October 1991. The answer can be computed as

```
intnx( 'week.4', '1oct91'd - 1, 3 )
```

which returns the SAS date value ’16OCT91’D.

Consider this more complex example: how many weekdays are there between 17 October 1991 and the second Friday in November 1991, inclusive? The following formula computes the number of weekdays between the date value contained in the variable DATE and the second Friday of the following month (including the ending dates of this period):

```
n = intck( 'weekday', date - 1,
     intnx( 'week.6', intnx( 'month', date, 1 ) - 1, 2 ) + 1 );
```

Setting DATE to ’17OCT91’D and applying this formula produces the answer, N=17.

Lags, Leads, Differences, and Summations

When working with time series data, you sometimes need to refer to the values of a series in previous or future periods. For example, the usual interest in the consumer price index series shown in previous examples is how fast the index is changing, rather than the actual level of the index. To compute a percent change, you need both the current and the previous values of the series. When you model a time series, you might want to use the previous values of other series as explanatory variables.

This section discusses how to use the DATA step to perform operations over time: lags, differences, leads, summations over time, and percent changes.

The EXPAND procedure can also be used to perform many of these operations; see Chapter 16, “The EXPAND Procedure,” for more information. See also the section “Transforming Time Series” on page 137.

The LAG and DIF Functions

The DATA step provides two functions, LAG and DIF, for accessing previous values of a variable or expression. These functions are useful for computing lags and differences of series.

For example, the following statements add the variables CPILAG and CPIDIF to the USCPI data set. The variable CPILAG contains lagged values of the CPI series. The variable CPIDIF contains the changes of the CPI series from the previous period; that is, CPIDIF is CPI minus CPILAG. The new data set is shown in part in Figure 4.14.

```
data uscpi;
   set uscpi;
   cpilag = lag( cpi );
   cpidif = dif( cpi );
run;

proc print data=uscpi;
run;
```

Figure 4.14 USCPI Data Set with Lagged and Differenced Series

Plot of USCPI Data

Obs	date	cpi	cpilag	cpidif
1	JUN1990	129.9	.	.
2	JUL1990	130.4	129.9	0.5
3	AUG1990	131.6	130.4	1.2
4	SEP1990	132.7	131.6	1.1
5	OCT1990	133.5	132.7	0.8
6	NOV1990	133.8	133.5	0.3
7	DEC1990	133.8	133.8	0.0
8	JAN1991	134.6	133.8	0.8
9	FEB1991	134.8	134.6	0.2
10	MAR1991	135.0	134.8	0.2
11	APR1991	135.2	135.0	0.2
12	MAY1991	135.6	135.2	0.4
13	JUN1991	136.0	135.6	0.4
14	JUL1991	136.2	136.0	0.2

Understanding the DATA Step LAG and DIF Functions

When used in this simple way, LAG and DIF act as lag and difference functions. However, it is important to keep in mind that, despite their names, the LAG and DIF functions available in the DATA step are not true lag and difference functions.

Rather, LAG and DIF are queuing functions that remember and return argument values from previous calls. The LAG function remembers the value you pass to it and returns as its result the value you passed to it on the previous call. The DIF function works the same way but returns the difference between the current argument and the remembered value. (LAG and DIF return a missing value the first time the function is called.)

A true lag function does not return the value of the argument for the "previous call," as do the DATA step LAG and DIF functions. Instead, a true lag function returns the value of its argument for the "previous observation," regardless of the sequence of previous calls to the function. Thus, for a true lag function to be possible, it must be clear what the "previous observation" is.

If the data are sorted chronologically, then LAG and DIF act as true lag and difference functions. If in doubt, use PROC SORT to sort your data before using the LAG and DIF functions. Beware of missing observations, which can cause LAG and DIF to return values that are not the actual lag and difference values.

The DATA step is a powerful tool that can read any number of observations from any number of input files or data sets, can create any number of output data sets, and can write any number of output observations to any of the output data sets, all in the same program. Thus, in general, it is not clear what "previous observation" means in a DATA step program. In a DATA step program, the "previous observation" exists only if you write the program in a simple way that makes this concept meaningful.

Since, in general, the previous observation is not clearly defined, it is not possible to make true lag or difference functions for the DATA step. Instead, the DATA step provides queuing functions that make it easy to compute lags and differences.

Pitfalls of DATA Step LAG and DIF Functions

The LAG and DIF functions compute lags and differences provided that the sequence of calls to the function corresponds to the sequence of observations in the output data set. However, any complexity in the DATA step that breaks this correspondence causes the LAG and DIF functions to produce unexpected results.

For example, suppose you want to add the variable CPILAG to the USCPI data set, as in the previous example, and you also want to subset the series to 1991 and later years. You might use the following statements:

```
data subset;
   set uscpi;
   if date >= '1jan1991'd;
   cpilag = lag( cpi );  /* WRONG PLACEMENT! */
run;
```

If the subsetting IF statement comes before the LAG function call, the value of CPILAG will be missing for January 1991, even though a value for December 1990 is available in the USCPI data set. To avoid losing this value, you must rearrange the statements to ensure that the LAG function is actually executed for the December 1990 observation.

```
data subset;
   set uscpi;
   cpilag = lag( cpi );
   if date >= '1jan1991'd;
run;
```

In other cases, the subsetting statement should come before the LAG and DIF functions. For example, the following statements subset the FOREOUT data set shown in a previous example to select only _TYPE_=RESIDUAL observations and also to compute the variable LAGRESID:

```
data residual;
   set foreout;
   if _type_ = "RESIDUAL";
   lagresid = lag( cpi );
run;
```

Another pitfall of LAG and DIF functions arises when they are used to process time series cross-sectional data sets. For example, suppose you want to add the variable CPILAG to the CPICITY data set shown in a previous example. You might use the following statements:

```
data cpicity;
   set cpicity;
   cpilag = lag( cpi );
run;
```

However, these statements do not yield the desired result. In the data set produced by these statements, the value of CPILAG for the first observation for the first city is missing (as it should be), but in the first observation for all later cities, CPILAG contains the last value for the previous city. To correct this, set the lagged variable to missing at the start of each cross section, as follows:

```
data cpicity;
   set cpicity;
   by city date;
   cpilag = lag( cpi );
   if first.city then cpilag = .;
run;
```

Alternatives to LAG and DIF Functions

You can also use the EXPAND procedure to compute lags and differences. For example, the following statements compute lag and difference variables for CPI:

```
proc expand data=uscpi out=uscpi method=none;
   id date;
   convert cpi=cpilag / transform=( lag 1 );
   convert cpi=cpidif / transform=( dif 1 );
run;
```

You can also calculate lags and differences in the DATA step without using LAG and DIF functions. For example, the following statements add the variables CPILAG and CPIDIF to the USCPI data set:

```
data uscpi;
   set uscpi;
   retain cpilag;
   cpidif = cpi - cpilag;
   output;
   cpilag = cpi;
run;
```

The RETAIN statement prevents the DATA step from reinitializing CPILAG to a missing value at the start of each iteration and thus allows CPILAG to retain the value of CPI assigned to it in the last statement. The OUTPUT statement causes the output observation to contain values of the variables before CPILAG is reassigned the current value of CPI in the last statement. This is the approach that must be used if you want to build a variable that is a function of its previous lags.

LAG and DIF Functions in PROC MODEL

The preceding discussion of LAG and DIF functions applies to LAG and DIF functions available in the DATA step. However, LAG and DIF functions are also used in the MODEL procedure.

The MODEL procedure LAG and DIF functions do not work like the DATA step LAG and DIF functions. The LAG and DIF functions supported by PROC MODEL are true lag and difference functions, not queuing functions.

Unlike the DATA step, the MODEL procedure processes observations from a single input data set, so the "previous observation" is always clearly defined in a PROC MODEL program. Therefore, PROC MODEL is able to define LAG and DIF as true lagging functions that operate on values from the previous observation. See Chapter 26, "The MODEL Procedure," for more information about LAG and DIF functions in the MODEL procedure.

Multiperiod Lags and Higher-Order Differencing

To compute lags at a lagging period greater than 1, add the lag length to the end of the LAG keyword to specify the lagging function needed. For example, the LAG2 function returns the value of its argument two calls ago, the LAG3 function returns the value of its argument three calls ago, and so forth.

To compute differences at a lagging period greater than 1, add the lag length to the end of the DIF keyword. For example, the DIF2 function computes the differences between the value of its argument and the value of its argument two calls ago. (The maximum lagging period is 100.)

The following statements add the variables CPILAG12 and CPIDIF12 to the USCPI data set. CPILAG12 contains the value of CPI from the same month one year ago. CPIDIF12 contains the change in CPI from the same month one year ago. (In this case, the first 12 values of CPILAG12 and CPIDIF12 are missing.)

```
data uscpi;
   set uscpi;
   cpilag12 = lag12( cpi );
   cpidif12 = dif12( cpi );
run;
```

To compute second differences, take the difference of the difference. To compute higher-order differences, nest DIF functions to the order needed. For example, the following statements compute the second difference of CPI:

```
data uscpi;
   set uscpi;
   cpi2dif = dif( dif( cpi ) );
run;
```

Multiperiod lags and higher-order differencing can be combined. For example, the following statements compute monthly changes in the inflation rate, with inflation rate computed as percent change in CPI from the same month one year ago:

```
data uscpi;
   set uscpi;
   infchng = dif( 100 * dif12( cpi ) / lag12( cpi ) );
run;
```

Percent Change Calculations

There are several common ways to compute the percent change in a time series. This section illustrates the use of LAG and DIF functions by showing SAS statements for various kinds of percent change calculations.

Computing Period-to-Period Change

To compute percent change from the previous period, divide the difference of the series by the lagged value of the series and multiply by 100.

```
data uscpi;
   set uscpi;
   pctchng = dif( cpi ) / lag( cpi ) * 100;
   label pctchng = "Monthly Percent Change, At Monthly Rates";
run;
```

Often, changes from the previous period are expressed at annual rates. This is done by exponentiation of the current-to-previous period ratio to the number of periods in a year and expressing the result as a percent change. For example, the following statements compute the month-over-month change in CPI as a percent change at annual rates:

```
data uscpi;
   set uscpi;
   pctchng = ( ( cpi / lag( cpi ) ) ** 12 - 1 ) * 100;
   label pctchng = "Monthly Percent Change, At Annual Rates";
run;
```

Computing Year-over-Year Change

To compute percent change from the same period in the previous year, use LAG and DIF functions with a lagging period equal to the number of periods in a year. (For quarterly data, use LAG4 and DIF4. For monthly data, use LAG12 and DIF12.)

For example, the following statements compute monthly percent change in CPI from the same month one year ago:

```
data uscpi;
   set uscpi;
   pctchng = dif12( cpi ) / lag12( cpi ) * 100;
   label pctchng = "Percent Change from One Year Ago";
run;
```

To compute year-over-year percent change measured at a given period within the year, subset the series of percent changes from the same period in the previous year to form a yearly data set. Use an IF or WHERE statement to select observations for the period within each year on which the year-over-year changes are based.

For example, the following statements compute year-over-year percent change in CPI from December of the previous year to December of the current year:

```
data annual;
   set uscpi;
   pctchng = dif12( cpi ) / lag12( cpi ) * 100;
   label pctchng = "Percent Change: December to December";
   if month( date ) = 12;
   format date year4.;
run;
```

Computing Percent Change in Yearly Averages

To compute changes in yearly averages, first aggregate the series to an annual series by using the EXPAND procedure, and then compute the percent change of the annual series. (See Chapter 16, "The EXPAND Procedure," for more information about PROC EXPAND.)

For example, the following statements compute percent changes in the annual averages of CPI:

```
proc expand data=uscpi out=annual from=month to=year;
   convert cpi / observed=average method=aggregate;
run;

data annual;
   set annual;
   pctchng = dif( cpi ) / lag( cpi ) * 100;
   label pctchng = "Percent Change in Yearly Averages";
run;
```

It is also possible to compute percent change in the average over the most recent yearly span. For example, the following statements compute monthly percent change in the average of CPI over the most recent 12 months from the average over the previous 12 months:

```
data uscpi;
   retain sum12 0;
   drop sum12 ave12 cpilag12;
   set uscpi;
   sum12 = sum12 + cpi;
   cpilag12 = lag12( cpi );
   if cpilag12 ^= . then sum12 = sum12 - cpilag12;
   if lag11( cpi ) ^= . then ave12 = sum12 / 12;
   pctchng = dif12( ave12 ) / lag12( ave12 ) * 100;
   label pctchng = "Percent Change in 12 Month Moving Ave.";
run;
```

This example is a complex use of LAG and DIF functions that requires care in handling the initialization of the moving-window averaging process. The LAG12 of CPI is checked for missing values to determine when more than 12 values have been accumulated, and older values must be removed from the moving sum. The LAG11 of CPI is checked for missing values to determine when at least 12 values have been accumulated; AVE12 will be missing when LAG11 of CPI is missing. The DROP statement prevents temporary variables from being added to the data set.

Note that the DIF and LAG functions must execute for every observation, or the queues of remembered values will not operate correctly. The CPILAG12 calculation must be separate from the IF statement. The PCTCHNG calculation must not be conditional on the IF statement.

The EXPAND procedure provides an alternative way to compute moving averages.

Leading Series

Although the SAS System does not provide a function to look ahead at the "next" value of a series, there are a couple of ways to perform this task.

The most direct way to compute leads is to use the EXPAND procedure. For example:

```
proc expand data=uscpi out=uscpi method=none;
   id date;
   convert cpi=cpilead1 / transform=( lead 1 );
   convert cpi=cpilead2 / transform=( lead 2 );
run;
```

Another way to compute lead series in SAS software is by lagging the time ID variable, renaming the series, and merging the result data set back with the original data set.

For example, the following statements add the variable CPILEAD to the USCPI data set. The variable CPILEAD contains the value of CPI in the following month. (The value of CPILEAD is missing for the last observation, of course.)

```
data temp;
   set uscpi;
   keep date cpi;
   rename cpi = cpilead;
   date = lag( date );
   if date ^= .;
run;

data uscpi;
   merge uscpi temp;
   by date;
run;
```

To compute leads at different lead lengths, you must create one temporary data set for each lead length. For example, the following statements compute CPILEAD1 and CPILEAD2, which contain leads of CPI for 1 and 2 periods, respectively:

```
data temp1(rename=(cpi=cpilead1))
     temp2(rename=(cpi=cpilead2));
   set uscpi;
   keep date cpi;
   date = lag( date );
   if date ^= . then output temp1;
   date = lag( date );
   if date ^= . then output temp2;
run;

data uscpi;
   merge uscpi temp1 temp2;
   by date;
run;
```

Summing Series

Simple cumulative sums are easy to compute using SAS sum statements. The following statements show how to compute the running sum of variable X in data set A, adding XSUM to the data set.

```
data a;
   set a;
   xsum + x;
run;
```

The SAS sum statement automatically retains the variable XSUM and initializes it to 0, and the sum statement treats missing values as 0. The sum statement is equivalent to using a RETAIN statement and the SUM function. The previous example could also be written as follows:

```
data a;
   set a;
   retain xsum;
   xsum = sum( xsum, x );
run;
```

You can also use the EXPAND procedure to compute summations. For example:

```
proc expand data=a out=a method=none;
   convert x=xsum / transform=( sum );
run;
```

Like differencing, summation can be done at different lags and can be repeated to produce higher-order sums. To compute sums over observations separated by lags greater than 1, use the LAG and SUM functions together, and use a RETAIN statement that initializes the summation variable to zero.

For example, the following statements add the variable XSUM2 to data set A. XSUM2 contains the sum of every other observation, with even-numbered observations containing a cumulative sum of values of X from even observations, and odd-numbered observations containing a cumulative sum of values of X from odd observations.

```
data a;
   set a;
   retain xsum2 0;
   xsum2 = sum( lag( xsum2 ), x );
run;
```

Assuming that A is a quarterly data set, the following statements compute running sums of X for each quarter. XSUM4 contains the cumulative sum of X for all observations for the same quarter as the current quarter. Thus, for a first-quarter observation, XSUM4 contains a cumulative sum of current and past first-quarter values.

```
data a;
   set a;
   retain xsum4 0;
   xsum4 = sum( lag3( xsum4 ), x );
run;
```

To compute higher-order sums, repeat the preceding process and sum the summation variable. For example, the following statements compute the first and second summations of X:

```
data a;
   set a;
   xsum + x;
   x2sum + xsum;
run;
```

The following statements compute the second order four-period sum of X:

```
data a;
   set a;
   retain xsum4 x2sum4 0;
   xsum4 = sum( lag3( xsum4 ), x );
   x2sum4 = sum( lag3( x2sum4 ), xsum4 );
run;
```

You can also use PROC EXPAND to compute cumulative statistics and moving window statistics. See Chapter 16, "The EXPAND Procedure," for details.

Transforming Time Series

It is often useful to transform time series for analysis or forecasting. Many time series analysis and forecasting methods are most appropriate for time series with an unrestricted range, a linear trend, and a constant variance. Series that do not conform to these assumptions can often be transformed to series for which the methods are appropriate.

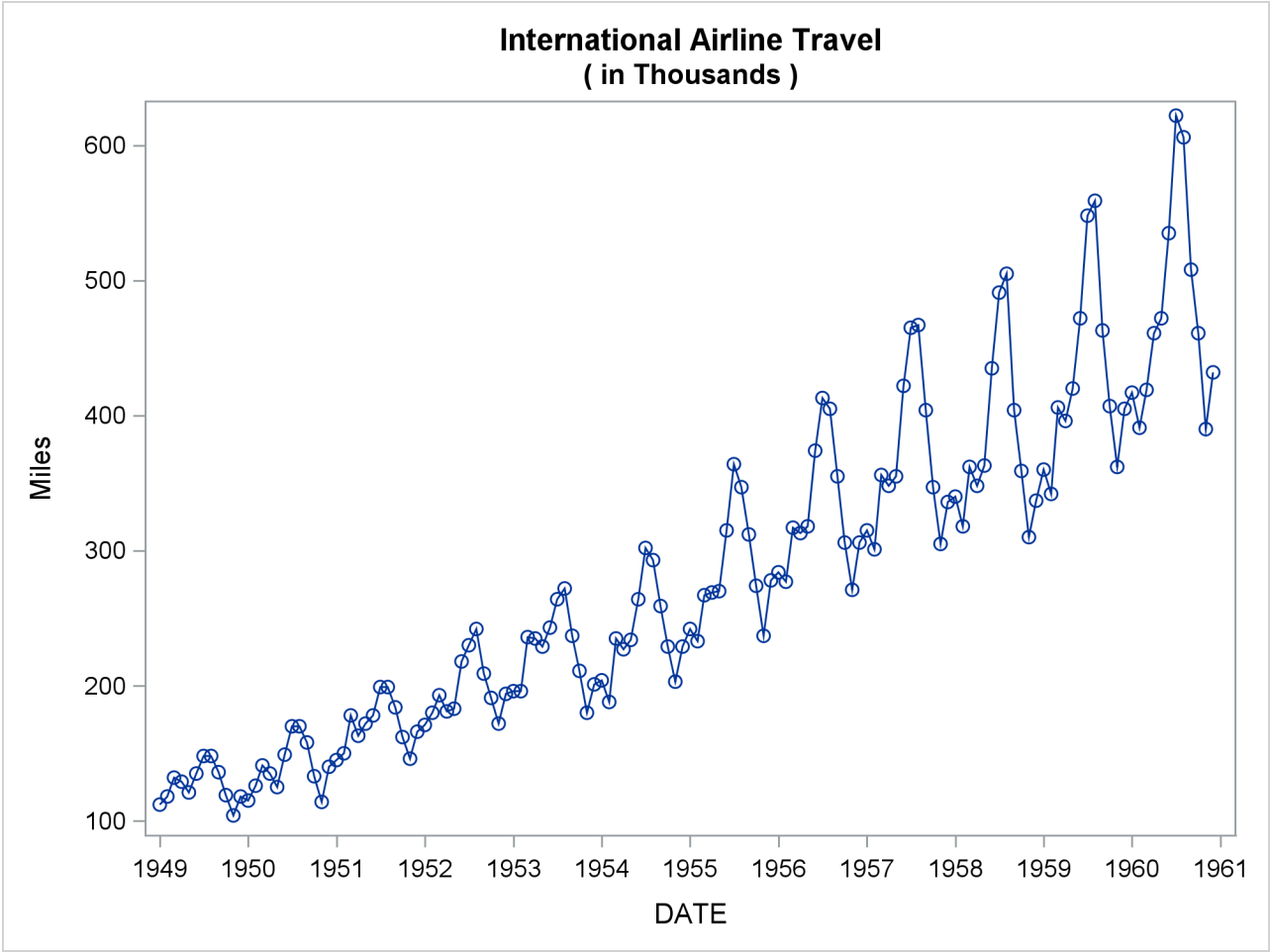
Transformations can be useful for the following:

- range restrictions. Many time series cannot have negative values or can be limited to a maximum possible value. You can often create a transformed series with an unbounded range.
- nonlinear trends. Many economic time series grow exponentially. Exponential growth corresponds to linear growth in the logarithms of the series.
- series variability that changes over time. Various transformations can be used to stabilize the variance.
- nonstationarity. The %DFTEST macro can be used to test a series for nonstationarity which can then be removed by differencing.

Log Transformation

The logarithmic transformation is often useful for series that must be greater than zero and that grow exponentially. For example, Figure 4.15 shows a plot of an airline passenger miles series. Notice that the series has exponential growth and the variability of the series increases over time. Airline passenger miles must also be zero or greater.

Figure 4.15 Airline Series

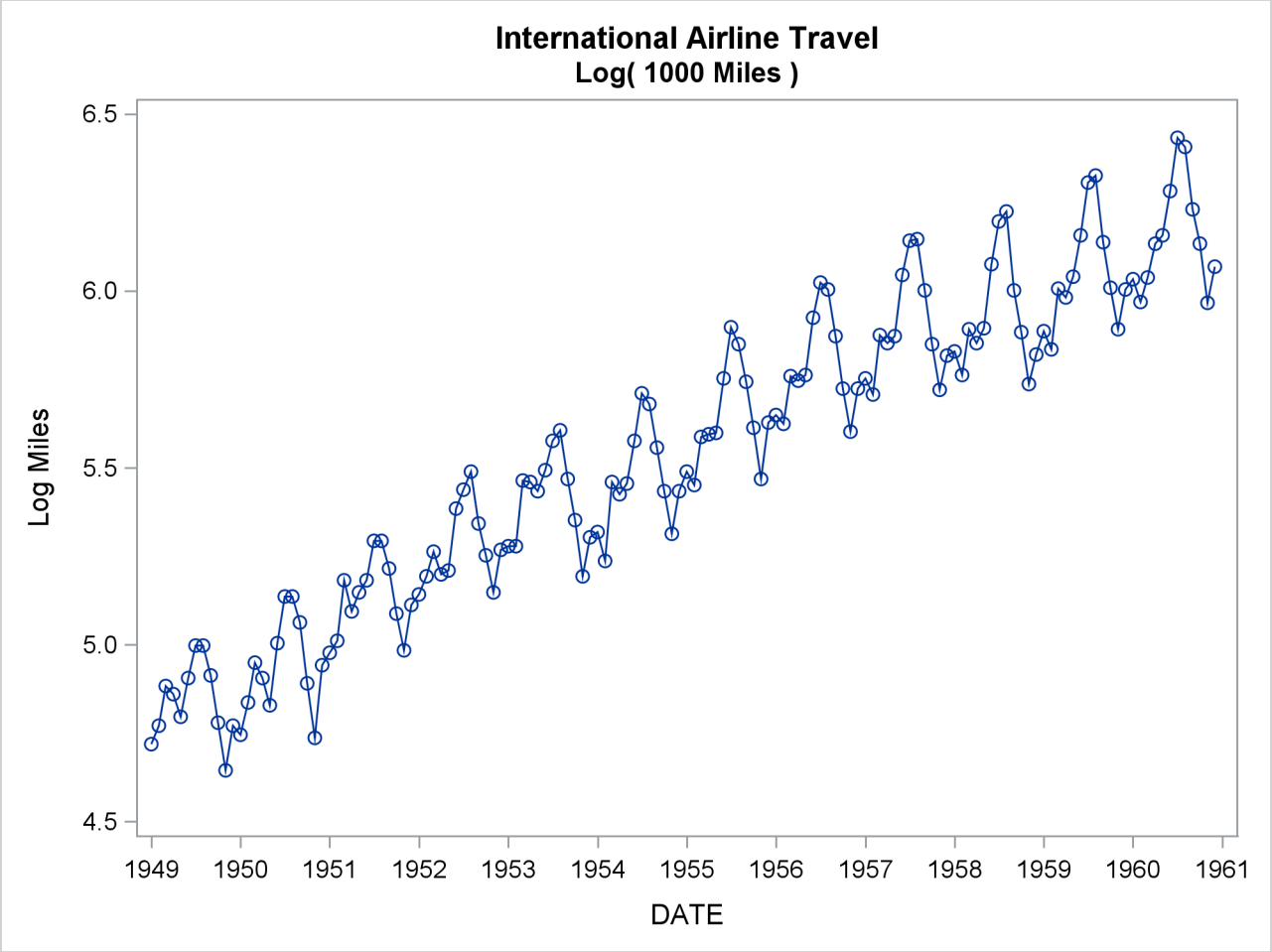


The following statements compute the logarithms of the airline series:

```
data lair;
   set sashelp.air;
   logair = log( air );
run;
```

Figure 4.16 shows a plot of the log-transformed airline series. Notice that the log series has a linear trend and constant variance.

Figure 4.16 Log Airline Series



The %LOGTEST macro can help you decide if a log transformation is appropriate for a series. See Chapter 6, "SAS Macros and Functions," for more information about the %LOGTEST macro.

Other Transformations

The Box-Cox transformation is a general class of transformations that includes the logarithm as a special case. The %BOXCOXAR macro can be used to find an optimal Box-Cox transformation for a time series. See Chapter 6 for more information about the %BOXCOXAR macro.

The logistic transformation is useful for variables with both an upper and a lower bound, such as market shares. The logistic transformation is useful for proportions, percent values, relative frequencies, or probabilities. The logistic function transforms values between 0 and 1 to values that can range from $-\infty$ to $+\infty$.

For example, the following statements transform the variable SHARE from percent values to an unbounded range:

```
data a;
   set a;
   lshare = log( share / ( 100 - share ) );
run;
```

Many other data transformation can be used. You can create virtually any desired data transformation using DATA step statements.

The EXPAND Procedure and Data Transformations

The EXPAND procedure provides a convenient way to transform series. For example, the following statements add variables for the logarithm of AIR and the logistic of SHARE to data set A:

```
proc expand data=a out=a method=none;
   convert air=logair   / transform=( log );
   convert share=lshare / transform=( / 100 logit );
run;
```

See Table 16.2 in Chapter 16, "The EXPAND Procedure," for a complete list of transformations supported by PROC EXPAND.

Manipulating Time Series Data Sets

This section discusses merging, splitting, and transposing time series data sets and interpolating time series data to a higher or lower sampling frequency.

Splitting and Merging Data Sets

In some cases, you might want to separate several time series that are contained in one data set into different data sets. In other cases, you might want to combine time series from different data sets into one data set.

To split a time series data set into two or more data sets that contain subsets of the series, use a DATA step to create the new data sets and use the KEEP= data set option to control which series are included in each new data set. The following statements split the USPRICE data set shown in a previous example into two data sets, USCPI and USPPI:

```
data uscpi(keep=date cpi)
     usppi(keep=date ppi);
   set usprice;
run;
```

If the series have different time ranges, you can subset the time ranges of the output data sets accordingly. For example, if you know that CPI in USPRICE has the range August 1990 through the end of the data set, while PPI has the range from the beginning of the data set through June 1991, you could write the previous example as follows:

```
data uscpi(keep=date cpi)
     usppi(keep=date ppi);
   set usprice;
   if date >= '1aug1990'd then output uscpi;
   if date <= '1jun1991'd then output usppi;
run;
```

To combine time series from different data sets into one data set, list the data sets to be combined in a MERGE statement and specify the dating variable in a BY statement. The following statements show how to combine the USCPI and USPPI data sets to produce the USPRICE data set. It is important to use the BY DATE statement so that observations are matched by time before merging.

```
data usprice;
   merge uscpi usppi;
   by date;
run;
```

Transposing Data Sets

The TRANSPOSE procedure is used to transpose data sets from one form to another. The TRANSPOSE procedure can transpose variables and observations, or transpose variables and observations within BY groups. This section discusses some applications of the TRANSPOSE procedure relevant to time series data sets. See the *Base SAS Procedures Guide* for more information about PROC TRANSPOSE.

Transposing from Interleaved to Standard Time Series Form

The following statements transpose part of the interleaved-form output data set FOREOUT, produced by PROC FORECAST in a previous example, to a standard form time series data set. To reduce the volume of output produced by the example, a WHERE statement is used to subset the input data set.

Observations with _TYPE_=ACTUAL are stored in the new variable ACTUAL; observations with _TYPE_=FORECAST are stored in the new variable FORECAST; and so forth. Note that the method used in this example works only for a single variable.

```
title "Original Data Set";
proc print data=foreout(obs=10);
   where date > '1may1991'd & date < '1oct1991'd;
run;

proc transpose data=foreout out=trans(drop=_name_);
   var cpi;
   id _type_;
   by date;
   where date > '1may1991'd & date < '1oct1991'd;
run;

title "Transposed Data Set";
proc print data=trans(obs=10);
run;
```

The TRANSPOSE procedure adds the variables _NAME_ and _LABEL_ to the output data set. These variables contain the names and labels of the variables that were transposed. In this example, there is only one transposed variable, so _NAME_ has the value CPI for all observations. Thus, _NAME_ and _LABEL_ are of no interest and are dropped from the output data set by using the DROP= data set option. (If none of the variables transposed have a label, PROC TRANSPOSE does not output the _LABEL_ variable and the DROP=_LABEL_ option produces a warning message. You can ignore this message, or you can prevent the message by omitting _LABEL_ from the DROP= list.)

The original and transposed data sets are shown in Figure 4.17 and Figure 4.18. (The observation numbers shown for the original data set reflect the operation of the WHERE statement.)

Figure 4.17 Original Data Sets

Original Data Set

Obs	date	_TYPE_	_LEAD_	cpi
37	JUN1991	ACTUAL	0	136.000
38	JUN1991	FORECAST	0	136.146
39	JUN1991	RESIDUAL	0	-0.146
40	JUL1991	ACTUAL	0	136.200
41	JUL1991	FORECAST	0	136.566
42	JUL1991	RESIDUAL	0	-0.366
43	AUG1991	FORECAST	1	136.856
44	AUG1991	L95	1	135.723
45	AUG1991	U95	1	137.990
46	SEP1991	FORECAST	2	137.443

Figure 4.18 Transposed Data Sets

Transposed Data Set

Obs	date	_LABEL_	ACTUAL	FORECAST	RESIDUAL	L95	U95
1	JUN1991	US Consumer Price Index	136.0	136.146	-0.14616	.	.
2	JUL1991	US Consumer Price Index	136.2	136.566	-0.36635	.	.
3	AUG1991	US Consumer Price Index	.	136.856	.	135.723	137.990
4	SEP1991	US Consumer Price Index	.	137.443	.	136.126	138.761

Transposing Cross-Sectional Dimensions

The following statements transpose the variable CPI in the CPICITY data set shown in a previous example from time series cross-sectional form to a standard form time series data set. (Only a subset of the data shown in the previous example is used here.) Note that the method shown in this example works only for a single variable.

```
title "Original Data Set";
proc print data=cpicity;
run;

proc sort data=cpicity out=temp;
   by date city;
run;

proc transpose data=temp out=citycpi(drop=_name_);
   var cpi;
   id city;
   by date;
run;

title "Transposed Data Set";
proc print data=citycpi;
run;
```

The names of the variables in the transposed data sets are taken from the city names in the ID variable CITY. The original and the transposed data sets are shown in Figure 4.19 and Figure 4.20.

Figure 4.19 Original Data Sets

Original Data Set

Obs	city	date	cpi	cpilag
1	Chicago	JAN90	128.1	.
2	Chicago	FEB90	129.2	128.1
3	Chicago	MAR90	129.5	129.2
4	Chicago	APR90	130.4	129.5
5	Chicago	MAY90	130.4	130.4
6	Chicago	JUN90	131.7	130.4
7	Chicago	JUL90	132.0	131.7
8	Los Angeles	JAN90	132.1	.
9	Los Angeles	FEB90	133.6	132.1
10	Los Angeles	MAR90	134.5	133.6
11	Los Angeles	APR90	134.2	134.5
12	Los Angeles	MAY90	134.6	134.2
13	Los Angeles	JUN90	135.0	134.6
14	Los Angeles	JUL90	135.6	135.0
15	New York	JAN90	135.1	.
16	New York	FEB90	135.3	135.1
17	New York	MAR90	136.6	135.3
18	New York	APR90	137.3	136.6
19	New York	MAY90	137.2	137.3
20	New York	JUN90	137.1	137.2
21	New York	JUL90	138.4	137.1

Figure 4.20 Transposed Data Sets

Transposed Data Set

Obs	date	Chicago	Los_Angeles	New_York
1	JAN90	128.1	132.1	135.1
2	FEB90	129.2	133.6	135.3
3	MAR90	129.5	134.5	136.6
4	APR90	130.4	134.2	137.3
5	MAY90	130.4	134.6	137.2
6	JUN90	131.7	135.0	137.1
7	JUL90	132.0	135.6	138.4

The following statements transpose the CITYCPI data set back to the original form of the CPICITY data set. The variable _NAME_ is added to the data set to tell PROC TRANSPOSE the name of the variable in which to store the observations in the transposed data set. (If the (DROP=_NAME_ _LABEL_) option were omitted from the first PROC TRANSPOSE step, this would not be necessary. PROC TRANSPOSE assumes ID _NAME_ by default.)

The NAME=CITY option in the PROC TRANSPOSE statement causes PROC TRANSPOSE to store the names of the transposed variables in the variable CITY. Because PROC TRANSPOSE recodes the values of the CITY variable to create valid SAS variable names in the transposed data set, the values of the variable CITY in the retransposed data set are not the same as in the original. The retransposed data set is shown in Figure 4.21.

```
data temp;
   set citycpi;
   _name_ = 'CPI';
run;

proc transpose data=temp out=retrans name=city;
   by date;
run;

proc sort data=retrans;
   by city date;
run;

title "Retransposed Data Set";
proc print data=retrans;
run;
```

Figure 4.21 Data Set Transposed Back to Original Form

Retransposed Data Set

Obs	date	city	CPI
1	JAN90	Chicago	128.1
2	FEB90	Chicago	129.2
3	MAR90	Chicago	129.5
4	APR90	Chicago	130.4
5	MAY90	Chicago	130.4
6	JUN90	Chicago	131.7
7	JUL90	Chicago	132.0
8	JAN90	Los_Angeles	132.1
9	FEB90	Los_Angeles	133.6
10	MAR90	Los_Angeles	134.5
11	APR90	Los_Angeles	134.2
12	MAY90	Los_Angeles	134.6
13	JUN90	Los_Angeles	135.0
14	JUL90	Los_Angeles	135.6
15	JAN90	New_York	135.1
16	FEB90	New_York	135.3
17	MAR90	New_York	136.6
18	APR90	New_York	137.3
19	MAY90	New_York	137.2
20	JUN90	New_York	137.1
21	JUL90	New_York	138.4

Time Series Interpolation

The EXPAND procedure interpolates time series. This section provides a brief summary of the use of PROC EXPAND for different kinds of time series interpolation problems. Most of the issues discussed in this section are explained in greater detail in Chapter 16.

By default, the EXPAND procedure performs interpolation by first fitting cubic spline curves to the available data and then computing needed interpolating values from the fitted spline curves. Other interpolation methods can be requested.

Note that interpolating values of a time series does not add any real information to the data because the interpolation process is not the same process that generated the other (nonmissing) values in the series. While time series interpolation can sometimes be useful, great care is needed in analyzing time series that contain interpolated values.

Interpolating Missing Values

To use the EXPAND procedure to interpolate missing values in a time series, specify the input and output data sets in the PROC EXPAND statement, and specify the time ID variable in an ID statement. For example, the following statements cause PROC EXPAND to interpolate values for missing values of all numeric variables in the data set USPRICE:

```
proc expand data=usprice out=interpl;
   id date;
run;
```

Interpolated values are computed only for embedded missing values in the input time series. Missing values before or after the range of a series are ignored by the EXPAND procedure.

In the preceding example, PROC EXPAND assumes that all series are measured at points in time given by the value of the ID variable. In fact, the series in the USPRICE data set are monthly averages. PROC EXPAND can produce a better interpolation if this is taken into account. The following example uses the FROM=MONTH option to tell PROC EXPAND that the series is monthly and uses the CONVERT statement with the OBSERVED=AVERAGE to specify that the series values are averages over each month:

```
proc expand data=usprice out=interpl
            from=month;
   id date;
   convert cpi ppi / observed=average;
run;
```

Interpolating to a Higher or Lower Frequency

You can use PROC EXPAND to interpolate values of time series at a higher or lower sampling frequency than the input time series. To change the periodicity of time series, specify the time interval of the input data set with the FROM= option, and specify the time interval for the desired output frequency with the TO= option. For example, the following statements compute interpolated weekly values of the monthly CPI and PPI series:

```
proc expand data=usprice out=interpl
            from=month to=week;
   id date;
   convert cpi ppi / observed=average;
run;
```

Interpolating between Stocks and Flows, Levels and Rates

A distinction is made between variables that are measured at points in time and variables that represent totals or averages over an interval. Point-in-time values are often called *stocks* or *levels*. Variables that represent totals or averages over an interval are often called *flows* or *rates*.

For example, the annual series Gross National Product represents the final goods production of over the year and also the yearly average rate of that production. However, the monthly variable Inventory represents the cost of a stock of goods at the end of the month.

The EXPAND procedure can convert between point-in-time values and period average or total values. To convert observation characteristics, specify the input and output characteristics with the OBSERVED= option in the CONVERT statement. For example, the following statements use the monthly average price index values in USPRICE to compute interpolated estimates of the price index levels at the midpoint of each month.

```
proc expand data=usprice out=midpoint
            from=month;
   id date;
   convert cpi ppi / observed=(average,middle);
run;
```

Reading Time Series Data

Time series data can be coded in many different ways. The SAS System can read time series data recorded in almost any form. Earlier sections of this chapter show how to read time series data coded in several commonly used ways. This section shows how to read time series data from data records coded in two other commonly used ways not previously introduced.

Several time series databases distributed by major data vendors can be read into SAS data sets by the DATASOURCE procedure. See Chapter 13, "The DATASOURCE Procedure," for more information.

The SASECRSP, SASEFAME, and SASEHAVR interface engines enable SAS users to access and process time series data in CRSPAccess data files, FAME databases, and Haver Analytics Data Link Express (DLX) data bases, respectively. See Chapter 46, "The SASECRSP Interface Engine," Chapter 47, "The SASEFAME Interface Engine," and Chapter 49, "The SASEHAVR Interface Engine," for more details.

Reading a Simple List of Values

Time series data can be coded as a simple list of values without dating information and with an arbitrary number of observations on each data record. In this case, the INPUT statement must use the trailing "@@" option to retain the current data record after reading the values for each observation, and the time ID variable must be generated with programming statements.

For example, the following statements read the USPRICE data set from data records that contain pairs of values for CPI and PPI. This example assumes you know that the first pair of values is for June 1990.

```
data usprice;
   input cpi ppi @@;
   date = intnx( 'month', '1jun1990'd, _n_-1 );
   format date monyy7.;
datalines;
129.9 114.3  130.4 114.5  131.6 116.5
132.7 118.4  133.5 120.8  133.8 120.1 133.8 118.7
134.6 119.0  134.8 117.2  135.0 116.2 135.2 116.0
135.6 116.5  136.0 116.3  136.2 116.0
;
```

Reading Fully Described Time Series in Transposed Form

Data for several time series can be coded with separate groups of records for each time series. Data files coded this way are transposed from the form required by SAS procedures. Time series data can also be coded with descriptive information about the series included with the data records.

The following example reads time series data for the USPRICE data set coded with separate groups of records for each series. The data records for each series consist of a series description record and one or more value records. The series description record gives the series name, starting month and year of the series, number of values in the series, and a series label. The value records contain the observations of the time series.

The data are first read into a temporary data set that contains one observation for each value of each series.

```
data temp;
   length _name_ $8 _label_ $40;
   keep _name_ _label_ date value;
   format date monyy.;
   input _name_ month year nval _label_ &;
   date = mdy( month, 1, year );
   do i = 1 to nval;
      input value @;
      output;
      date = intnx( 'month', date, 1 );
   end;
datalines;
cpi      8 90  12  Consumer Price Index
131.6 132.7 133.5 133.8 133.8 134.6 134.8 135.0
135.2 135.6 136.0 136.2
ppi      6 90  13  Producer Price Index
114.3 114.5 116.5 118.4 120.8 120.1 118.7 119.0
117.2 116.2 116.0 116.5 116.3
;
```

The following statements sort the data set by date and series name, and the TRANSPOSE procedure is used to transpose the data into a standard form time series data set.

```
proc sort data=temp;
   by date _name_;
run;

proc transpose data=temp out=usprice(drop=_name_);
   by date;
   var value;
run;

proc contents data=usprice;
run;

proc print data=usprice;
run;
```

The final data set is shown in Figure 4.23.

Figure 4.22 Contents of USPRICE Data Set

Retransposed Data Set

The CONTENTS Procedure

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Label
3	cpi	Num	8		Consumer Price Index
1	date	Num	8	MONYY.	
2	ppi	Num	8		Producer Price Index

Figure 4.23 Listing of USPRICE Data Set

Retransposed Data Set

Obs	date	ppi	cpi
1	JUN90	114.3	.
2	JUL90	114.5	.
3	AUG90	116.5	131.6
4	SEP90	118.4	132.7
5	OCT90	120.8	133.5
6	NOV90	120.1	133.8
7	DEC90	118.7	133.8
8	JAN91	119.0	134.6
9	FEB91	117.2	134.8
10	MAR91	116.2	135.0
11	APR91	116.0	135.2
12	MAY91	116.5	135.6
13	JUN91	116.3	136.0
14	JUL91	.	136.2

Chapter 5
Date Intervals, Formats, and Functions

Contents

Overview **151**
Time Intervals **152**
Constructing Interval Names 152
Shifted Intervals 153
Beginning Dates and Datetimes of Intervals 154
Summary of Interval Types 155
Examples of Interval Specifications 158
Custom Time Intervals **159**
Date and Datetime Informats **164**
Date, Time, and Datetime Formats **165**
Date Formats 167
Datetime and Time Formats 170
Alignment of SAS Dates **171**
SAS Date, Time, and Datetime Functions **171**
References **177**

Overview

This chapter summarizes the time intervals, date and datetime informats, date and datetime formats, and date, time, and datetime functions available in SAS software. The use of these features is explained in Chapter 4, "Working with Time Series Data." The material in this chapter is also contained in *SAS Language Reference: Concepts* and *SAS Language Reference: Dictionary*. Because these features are useful for work with time series data, documentation of these features is consolidated and repeated here for easy reference.

Time Intervals

This section provides a reference for the different kinds of time intervals supported by SAS software, but it does not cover how they are used. For an introduction to the use of time intervals, see Chapter 4, "Working with Time Series Data."

Some interval names are used with SAS date values, while other interval names are used with SAS datetime values. The interval names used with SAS date values are YEAR, SEMIYEAR, QTR, MONTH, SEMIMONTH, TENDAY, WEEK, WEEKDAY, DAY, YEARV, R445YR, R454YR, R544YR, R445QTR, R454QTR, R544QTR, R445MON, R454MON, R544MON, and WEEKV. The interval names used with SAS datetime or time values are HOUR, MINUTE, and SECOND. Various abbreviations of these names are also allowed, as described in the section "Summary of Interval Types" on page 155.

Interval names for use with SAS date values can be prefixed with 'DT' to construct interval names for use with SAS datetime values. The interval names DTYEAR, DTSEMIYEAR, DTQTR, DTMONTH, DTSEMIMONTH, DTTENDAY, DTWEEK, DTWEEKDAY, DTDAY, DTYEARV, DTR445YR, DTR454YR, DTR544YR, DTR445QTR, DTR454QTR, DTR544QTR, DTR445MON, DTR454MON, DTR544MON, and DTWEEKV are used with SAS datetime values.

Constructing Interval Names

Multipliers and shift indexes can be used with the basic interval names to construct more complex interval specifications. The general form of an interval name is as follows:

NAMEn.s

The three parts of the interval name are shown below:

NAME	the name of the basic interval type. For example, YEAR specifies yearly intervals.
n	an optional multiplier that specifies that the interval is a multiple of the period of the basic interval type. For example, the interval YEAR2 consists of two-year (biennial) periods.
s	an optional starting subperiod index that specifies that the intervals are shifted to later starting points. For example, YEAR.3 specifies yearly periods shifted to start on the first of March of each calendar year and to end in February of the following year.

Both the multiplier *n* and the shift index *s* are optional and default to 1. For example, YEAR, YEAR1, YEAR.1, and YEAR1.1 are all equivalent ways of specifying ordinary calendar years.

To test for a valid interval specification, use the INTTEST function:

```
interval = 'MONTH3.2';
valid = INTTEST( interval );
valid = INTTEST( 'YEAR4');
```

INTTEST returns a value of 0 if the argument is not a valid interval specification and 1 if the argument is a valid interval specification. The INTTEST function can also be used in a DATA step to test an interval before calling an interval function:

```
valid = INTTEST( interval );
if ( valid = 1 ) then do;
    end_date = INTNX( interval, date, 0, 'E' );
    Status = 'Success';
end;
if ( valid = 0 ) then Status = 'Failure';
```

For more information about the INTTEST function, see the *SAS Language Reference: Dictionary*.

Shifted Intervals

Different kinds of intervals are shifted by different subperiods:

- YEAR, SEMIYEAR, QTR, and MONTH intervals are shifted by calendar months.
- WEEK and DAY intervals are shifted by days.
- SEMIMONTH intervals are shifted by semimonthly periods.
- TENDAY intervals are shifted by 10-day periods.
- YEARV intervals are shifted by WEEKV intervals.
- R445YR, R445QTR, and R445MON intervals are shifted by R445MON intervals.
- R454YR, R454QTR, and R454MON intervals are shifted by R454MON intervals.
- R544YR, R544QTR, and R544MON intervals are shifted by R544MON intervals.
- WEEKV intervals are shifted by days.
- WEEKDAY intervals are shifted by weekdays.
- HOUR intervals are shifted by hours.
- MINUTE intervals are shifted by minutes.
- SECOND intervals are shifted by seconds.

The INTSHIFT function returns the shift interval:

```
interval = 'MONTH3.2';
shift_interval = INTSHIFT( interval );
```

In this example, the value of shift_interval is 'MONTH'. For more information about the INTSHIFT function, see the *SAS Language Reference: Dictionary*.

If a subperiod is specified, the shift index cannot be greater than the number of subperiods in the whole interval. For example, you can use YEAR2.24, but YEAR2.25 is an error because there is no 25th month in a two-year interval.

For interval types that shift by subperiods that are the same as the basic interval type, only multiperiod intervals can be shifted. For example, MONTH type intervals shift by MONTH subintervals; thus, monthly intervals cannot be shifted because there is only one month in MONTH. However, bimonthly intervals can be shifted because there are two MONTH intervals in each MONTH2 interval. The interval name MONTH2.2 specifies bimonthly periods that start on the first day of even-numbered months.

Beginning Dates and Datetimes of Intervals

Intervals that represent divisions of a year begin with the start of the year (1 January). YEARV, R445YR, R454YR, and R544YR intervals begin with the first week of the International Organization for Standardization (ISO) year, the Monday on or immediately preceding January 4th. R445QTR, R454QTR, and R544QTR intervals begin with the 1st, 14th, 27th, and 40th weeks of the ISO year. MONTH2 periods begin with odd-numbered months (January, March, May, and so on).

Likewise, intervals that represent divisions of a day begin with the start of the day (midnight). Thus, HOUR8.7 intervals divide the day into the periods 06:00 to 14:00, 14:00 to 22:00, and 22:00 to 06:00.

Intervals that do not nest within years or days begin relative to the SAS date or datetime value 0. The arbitrary reference time of midnight on January 1, 1960, is used as the origin for nonshifted intervals, and shifted intervals are defined relative to that reference point. For example, MONTH13 defines the intervals January 1, 1960, February 1, 1961, March 1, 1962, and so forth, and the intervals December 1, 1959, November 1, 1958, and so on before the base date January 1, 1960.

Similarly, the WEEK2 interval begins relative to the Sunday of the week of January 1, 1960. The interval specification WEEK6.13 defines six-week periods that start on second Fridays, and the convention of counting relative to the period that contains January 1, 1960, indicates the starting date or datetime of the interval closest to January 1, 1960, that corresponds to the second Fridays of six-week intervals.

Intervals always begin on the date or datetime defined by the base interval name, the multiplier, and the shift value. The end of the interval immediately precedes the beginning of the next interval. However, an interval can be identified by any date or datetime value between its starting and ending values, inclusive. See the section "Alignment of SAS Dates" on page 171 for more information about generating identifying dates for intervals.

Summary of Interval Types

The interval types are summarized as follows:

YEAR
specifies yearly intervals. Abbreviations are YEAR, YEARS, YEARLY, YR, ANNUAL, ANNUALLY, and ANNUALS. The starting subperiod *s* is in months (MONTH).

YEARV
specifies ISO 8601 yearly intervals. The ISO 8601 year starts on the Monday on or immediately preceding January 4th. Note that it is possible for the ISO 8601 year to start in December of the preceding year. Also, some ISO 8601 years contain a leap week. For further discussion of ISO weeks, see Technical Committee ISO/TC 154 (Processes, Data Elements, and Documents in Commerce, Industry, and Administration) (2004). The starting subperiod *s* is in ISO 8601 weeks (WEEKV).

R445YR
is the same as YEARV except that the starting subperiod *s* is in retail 4-4-5 months (R445MON).

R454YR
is the same as YEARV except that the starting subperiod *s* is in retail 4-5-4 months (R454MON). For a discussion of the retail 4-5-4 calendar, see National Retail Federation (2007).

R544YR
is the same as YEARV except that the starting subperiod *s* is in retail 5-4-4 months (R544MON).

SEMIYEAR
specifies semiannual intervals (every six months). Abbreviations are SEMIYEAR, SEMIYEARS, SEMIYEARLY, SEMIYR, SEMIANNUAL, and SEMIANN.

The starting subperiod *s* is in months (MONTH). For example, SEMIYEAR.3 intervals are March–August and September–February.

QTR
specifies quarterly intervals (every three months). Abbreviations are QTR, QUARTER, QUARTERS, QUARTERLY, QTRLY, and QTRS. The starting subperiod *s* is in months (MONTH).

R445QTR
specifies retail 4-4-5 quarterly intervals (every 13 ISO 8601 weeks). Some fourth quarters contain a leap week. The starting subperiod *s* is in retail 4-4-5 months (R445MON).

R454QTR
specifies retail 4-5-4 quarterly intervals (every 13 ISO 8601 weeks). Some fourth quarters contain a leap week. For a discussion of the retail 4-5-4 calendar, see National Retail Federation (2007). The starting subperiod *s* is in retail 4-5-4 months (R454MON).

R544QTR
specifies retail 5-4-4 quarterly intervals (every 13 ISO 8601 weeks). Some fourth quarters contain a leap week. The starting subperiod *s* is in retail 5-4-4 months (R544MON).

MONTH
specifies monthly intervals. Abbreviations are MONTH, MONTHS, MONTHLY, and MON. The starting subperiod *s* is in months (MONTH). For example, MONTH2.2 intervals are February–March, April–May, June–July, August–September, October–November, and December–January of the following year.

R445MON
specifies retail 4-4-5 monthly intervals. The 3rd, 6th, 9th, and 12th months are five ISO 8601 weeks long with the exception that some 12th months contain leap weeks. All other months are four ISO 8601 weeks long. R445MON intervals begin with the 1st, 5th, 9th, 14th, 18th, 22nd, 27th, 31st, 35th, 40th, 44th, and 48th weeks of the ISO year. The starting subperiod *s* is in retail 4-4-5 months (R445MON).

R454MON
specifies retail 4-5-4 monthly intervals. The 2nd, 5th, 8th, and 11th months are five ISO 8601 weeks long. All other months are four ISO 8601 weeks long with the exception that some 12th months contain leap weeks. R454MON intervals begin with the 1st, 5th, 10th, 14th, 18th, 23rd, 27th, 31st, 36th, 40th, 44th, and 49th weeks of the ISO year. For a discussion of the retail 4-5-4 calendar, see National Retail Federation (2007). The starting subperiod *s* is in retail 4-5-4 months (R454MON).

R544MON
specifies retail 5-4-4 monthly intervals. The 1st, 4th, 7th, and 10th months are five ISO 8601 weeks long. All other months are four ISO 8601 weeks long with the exception that some 12th months contain leap weeks. R544MON intervals begin with the 1st, 6th, 10th, 14th, 19th, 23rd, 27th, 32nd, 36th, 40th, 45th, and 49th weeks of the ISO year. The starting subperiod *s* is in retail 5-4-4 months (R544MON).

SEMIMONTH
specifies semimonthly intervals. SEMIMONTH breaks each month into two periods, starting on the 1st and 16th days. Abbreviations are SEMIMONTH, SEMIMONTHS, SEMIMONTHLY, and SEMIMON. The starting subperiod *s* is in SEMIMONTH periods. For example, SEMIMONTH2.2 specifies intervals from the 16th of one month through the 15th of the next month.

TENDAY
specifies 10-day intervals. TENDAY breaks the month into three periods, the 1st through the 10th day of the month, the 11th through the 20th day of the month, and the remainder of the month. (TENDAY is a special interval typically used for reporting automobile sales data.) The starting subperiod *s* is in TENDAY periods. For example, TENDAY4.2 defines 40-day periods that start at the second TENDAY period.

WEEK
specifies weekly intervals of seven days. Abbreviations are WEEK, WEEKS, and WEEKLY. The starting subperiod *s* is in days (DAY), with the days of the week numbered as 1=Sunday, 2=Monday, 3=Tuesday, 4=Wednesday, 5=Thursday, 6=Friday, and 7=Saturday. For example, WEEK.7 means weekly with Saturday as the first day of the week.

WEEKV

specifies ISO 8601 weekly intervals of seven days. Each week starts on Monday. The starting subperiod *s* is in days (DAY). Note that WEEKV differs from WEEK in that WEEKV.1 starts on Monday, WEEKV.2 starts on Tuesday, and so forth.

WEEKDAY

WEEKDAY*d*W

WEEKDAY*dd*W

WEEKDAY*ddd*W

specifies daily intervals with weekend days included in the preceding weekday. Note that for a five-day work week that starts on Monday, the appropriate interval is WEEKDAY5.2. Abbreviations are WEEKDAY and WEEKDAYS. The starting subperiod *s* is in weekdays (WEEKDAY).

The WEEKDAY interval is the same as DAY except that weekend days are absorbed into the preceding weekday. Thus, there are five WEEKDAY intervals in a calendar week: Monday, Tuesday, Wednesday, Thursday, and the three-day period Friday-Saturday-Sunday.

The default weekend days are Saturday and Sunday, but any one to six weekend days can be listed after the WEEKDAY string and followed by a W. Weekend days are specified as '1' for Sunday, '2' for Monday, and so forth. For example, WEEKDAY67W specifies a Friday-Saturday weekend. WEEKDAY1W specifies a six-day work week with a Sunday weekend. WEEKDAY17W is the same as WEEKDAY.

DAY

specifies daily intervals. Abbreviations are DAY, DAYS, and DAILY. The starting subperiod *s* is in days (DAY).

HOUR

specifies hourly intervals. Aliases are HOUR, DTHOUR, HOURS, DTHOURS, HOURLY, DTHOURLY, HR, and DTHR. The starting subperiod *s* is in hours (HOUR).

MINUTE

specifies minute intervals. Aliases are MINUTE, DTMINUTE, MINUTES, DTMINUTES, MIN, and DTMIN. The starting subperiod *s* is in minutes (MINUTE).

SECOND

specifies second intervals. Aliases are SECOND, DTSECOND, SECONDS, DTSECONDS, SEC and DTSEC. The starting subperiod *s* is in seconds (SECOND).

Examples of Interval Specifications

Table 5.1 shows examples of different kinds of interval specifications.

Table 5.1 Examples of Intervals

Name	Description of Interval
YEAR	Years that start in January
YEAR.10	Years that start in October
YEAR2.7	Biennial intervals that start in July of even years
YEAR2.19	Biennial intervals that start in July of odd years
YEAR4.11	Four-year intervals that start in November of leap years (frequency of U.S. presidential elections)
YEAR4.35	Four-year intervals that start in November of even years between leap years (frequency of U.S. midterm elections)
YEARV	Years that start on the Monday on or immediately preceding January 4th
YEARV.2	Years that start on the Monday immediately following January 4th
R445MON	Months that start on the 1st, 5th, 9th, 14th, 18th, 22nd, 27th, 31st, 35th, 40th, 44th, and 48th Monday of the year. The 1st Monday is the Monday on or immediately preceding January 4th
R445MON3	Three-month intervals that start on the 1st, 14th, 27th, and 40th Monday of the year. This is equivalent to R445QTR
R445MON3.2	Three-month intervals that start on the 5th, 18th, 31th, and 44th Monday of the year. This is equivalent to R445QTR.2
WEEK	Weekly intervals that start on Sundays
WEEK2	Biweekly intervals that start on first Sundays
WEEK1.1	Same as WEEK
WEEK.2	Weekly intervals that start on Mondays
WEEK6.3	Six-week intervals that start on first Tuesdays
WEEK6.11	Six-week intervals that start on second Wednesdays
WEEKDAY	Daily with Friday-Saturday-Sunday counted as the same day (five-day work week with a Saturday-Sunday weekend)
WEEKDAY17W	Same as WEEKDAY
WEEKDAY5.2	Five weekdays that start on Monday. If WEEKDAY data are accumulated into weekly data, the interval of the accumulated data is WEEKDAY5.2
WEEKDAY67W	Daily with Thursday-Friday-Saturday counted as the same day (five-day work week with a Friday-Saturday weekend)
WEEKDAY1W	Daily with Saturday-Sunday counted as the same day (six-day work week with a Sunday weekend)
WEEKDAY3.2	Three-weekday intervals (with Friday-Saturday-Sunday counted as one weekday) with the cycle three-weekday periods aligned to Monday, January 4, 1960
HOUR8.7	Eight-hour intervals that start at 6 a.m., 2 p.m., and 10 p.m. (might be used for work shifts)

Custom Time Intervals

The standard time intervals described in the previous sections do not always fit the data. For example, you might want to use fiscal months that begin on the 10th of each month, but the MONTH interval begins on the 1st of each month. Or you might collect data hourly for a business that is closed at night, but using the DTHOUR interval results in gaps in the data that can cause problems in standard time series analysis. In another case, you might wish to calculate the number of business days between dates, excluding holidays and weekends, but holidays are counted when you use the INTCK function with the WEEKDAY interval. For more information about the INTCK function, see "Interval Functions INTNX and INTCK" on page 122.

Time series can be analyzed using observation numbers as the identifying reference. However, it is often desirable to maintain the time stamp for other types of modeling such as regression variables based on time or reconciliation.

To address these issues, you can define custom intervals within a given SAS program. The use of custom intervals requires the following two steps for each interval:

1 Associate a data set name with a custom interval name by using the INTERVALDS= system option. For more information about the INTERVALDS= option, see the *SAS Language Reference: Dictionary*. The following example associates the data set StoreHoursDS with the custom interval StoreHours.

```
options intervalds=(StoreHours=StoreHoursDS);
```

2 Create a data set that describes the custom interval. The data set must contain a BEGIN variable. It can also contain an END and a SEASON variable. It should contain a FORMAT statement for the BEGIN variable that specifies a SAS date, SAS datetime, or numeric format that matches the BEGIN variable data. If the END variable is present, it should also be included in the FORMAT statement. A numeric format that is not a SAS date or SAS datetime format indicates that the values are observation numbers. If the END variable is not present, then the implied value of END at each observation is one less than the value of BEGIN at the next observation.

   The span of the custom interval data set should include any dates or times that are necessary for performing calculations on the time series, including backcasting, forecasting, and other operations that might extend beyond the series (such as filters).

After the two preceding steps have been completed, the custom interval can be specified in SAS procedures and functions where a standard time interval can be specified.

The following DATA step creates the StoreHoursDS data set, which is appropriate for a business that is open 9AM to 6PM Monday through Friday and Saturday 9AM to 1PM:

```
options intervalds=(StoreHours=StoreHoursDS);
data StoreHoursDS(keep=BEGIN END);
   start = '01JAN2009'D;
   stop  = '31DEC2009'D;
   do date = start to stop;
      dow = WEEKDAY(date);
      datetime=dhms(date,0,0,0);
      if dow not in (1,7) then
         do hour = 9 to 17;
            begin=intnx('hour',datetime,hour,'b');
            end=intnx('hour',datetime,hour,'e');
            output;
         end;
      else if dow = 7 then
         do hour = 9 to 12;
            begin=intnx('hour',datetime,hour,'b');
            end=intnx('hour',datetime,hour,'e');
            output;
         end;
   end;
   format BEGIN END DATETIME.;
run;

title 'Store Hours Custom Interval';
proc print data=StoreHoursDS(obs=18);
run;
```

The first 18 observations of the custom interval data set are shown in Figure 5.1.

Figure 5.1 Store Hours Custom Interval

Store Hours Custom Interval

Obs	begin	end
1	01JAN09:09:00:00	01JAN09:09:59:59
2	01JAN09:10:00:00	01JAN09:10:59:59
3	01JAN09:11:00:00	01JAN09:11:59:59
4	01JAN09:12:00:00	01JAN09:12:59:59
5	01JAN09:13:00:00	01JAN09:13:59:59
6	01JAN09:14:00:00	01JAN09:14:59:59
7	01JAN09:15:00:00	01JAN09:15:59:59
8	01JAN09:16:00:00	01JAN09:16:59:59
9	01JAN09:17:00:00	01JAN09:17:59:59
10	02JAN09:09:00:00	02JAN09:09:59:59
11	02JAN09:10:00:00	02JAN09:10:59:59
12	02JAN09:11:00:00	02JAN09:11:59:59
13	02JAN09:12:00:00	02JAN09:12:59:59
14	02JAN09:13:00:00	02JAN09:13:59:59
15	02JAN09:14:00:00	02JAN09:14:59:59
16	02JAN09:15:00:00	02JAN09:15:59:59
17	02JAN09:16:00:00	02JAN09:16:59:59
18	02JAN09:17:00:00	02JAN09:17:59:59

The following DATA step creates the FMDS data set to define a custom interval FiscalMonth, which is appropriate for a business that uses fiscal months that start on the 10th of each month. The SAME alignment option of the INTNX function specifies that the dates generated by the INTNX function are the same day of the month as the date in the start variable. For more information about the INTNX function, see "SAS Date, Time, and Datetime Functions" on page 171. The MONTH function assigns the month of the BEGIN variable to the SEASON variable. This specifies monthly seasonality.

```
options intervalds=(FiscalMonth=FMDS);
data FMDS(keep=BEGIN SEASON);
   start = '10JAN1999'D;
   stop  = '10JAN2001'D;
   nmonths = INTCK('MONTH',start,stop);
   do i=0 to nmonths;
      BEGIN = INTNX('MONTH',start,i,'S');
      SEASON = MONTH(BEGIN);
      output;
   end;
   format BEGIN DATE.;
run;
```

The difference between the custom FiscalMonth interval and a standard interval can be seen in the following example. The output shown in Figure 5.2 compares how the data are accumulated. For the FiscalMonth interval, values in the first nine days of the month are accumulated with the interval that begins in the previous month. For the standard MONTH interval, values in the first nine days of the month are accumulated with the calendar month.

```
data sales(keep=DATE sales);
   do date = '01JAN2000'D to '31DEC2000'D;
      month = MONTH(date);
      dayofmonth = DAY(date);
      sales = 0;
      if ( dayofmonth lt 10 ) then sales = month/9;
      output;
   end;
   format date monyy.;
run;

proc timeseries data=sales out=dataInFiscalMonths;
   id DATE interval=FiscalMonth accumulate=total;
   var sales;
run;

proc timeseries data=sales out=dataInStdMonths;
   id DATE interval=Month accumulate=total;
   var sales;
run;

data compare;
     merge dataInFiscalMonths(rename=(sales=FM_sales))
           dataInStdMonths(rename=(sales=SM_sales));
     by DATE;
run;

title 'Standard Monthly Data vs. Fiscal Month Data';
proc print data=compare;
run;
```

Figure 5.2 Fiscal Months Custom Interval

Standard Monthly Data vs. Fiscal Month Data

Obs	date	FM_sales	SM_sales
1	10-DEC-1999	1	.
2	01-JAN-2000	.	1
3	10-JAN-2000	2	.
4	01-FEB-2000	.	2
5	10-FEB-2000	3	.
6	01-MAR-2000	.	3
7	10-MAR-2000	4	.
8	01-APR-2000	.	4
9	10-APR-2000	5	.
10	01-MAY-2000	.	5
11	10-MAY-2000	6	.
12	01-JUN-2000	.	6
13	10-JUN-2000	7	.
14	01-JUL-2000	.	7
15	10-JUL-2000	8	.
16	01-AUG-2000	.	8
17	10-AUG-2000	9	.
18	01-SEP-2000	.	9
19	10-SEP-2000	10	.
20	01-OCT-2000	.	10
21	10-OCT-2000	11	.
22	01-NOV-2000	.	11
23	10-NOV-2000	12	.
24	01-DEC-2000	.	12
25	10-DEC-2000	0	.

The next example uses custom intervals in the time function INTCK to omit holidays when counting business days. The result is shown in Figure 5.3.

```
options intervalds=(BankingDays=BankDayDS);
data BankDayDS(keep=BEGIN);
   start = '15DEC1998'D;
   stop  = '15JAN2002'D;
   nwkdays = INTCK('WEEKDAY',start,stop);
   do i = 0 to nwkdays;
      BEGIN = INTNX('WEEKDAY',start,i);
      year = YEAR(BEGIN);
      if BEGIN ne HOLIDAY("NEWYEAR",year) and
         BEGIN ne HOLIDAY("MLK",year) and
         BEGIN ne HOLIDAY("USPRESIDENTS",year) and
         BEGIN ne HOLIDAY("MEMORIAL",year) and
         BEGIN ne HOLIDAY("USINDEPENDENCE",year) and
         BEGIN ne HOLIDAY("LABOR",year) and
         BEGIN ne HOLIDAY("COLUMBUS",year) and
         BEGIN ne HOLIDAY("VETERANS",year) and
         BEGIN ne HOLIDAY("THANKSGIVING",year) and
         BEGIN ne HOLIDAY("CHRISTMAS",year) then
      output;
   end;
   format BEGIN DATE.;
run;

data CountDays;
   start = '01JAN1999'D;
   stop  = '31DEC2001'D;
   ActualDays = INTCK('DAYS',start,stop);
   Weekdays   = INTCK('WEEKDAYS',start,stop);
   BankDays   = INTCK('BankingDays',start,stop);
   format start stop DATE.;
run;

title 'Methods of Counting Days';
proc print data=CountDays;
run;
```

Figure 5.3 Bank Days Custom Interval

Methods of Counting Days

Obs	start	stop	ActualDays	Weekdays	BankDays
1	01JAN99	31DEC01	1095	781	757

Date and Datetime Informats

Table 5.2 lists some of the SAS date and datetime informats available to read date, time, and datetime values. See Chapter 4, "Working with Time Series Data," for a discussion of the use of date and datetime informats. See *SAS Language Reference: Concepts* for a complete description of these informats.

For each informat, Table 5.2 shows an example of a date or datetime value written in the style that the informat is designed to read. You can specify the width of each informat by adding *w*. For informats that include second values, you can specify the number of decimal digits for seconds by adding *d*. Table 5.2 shows the width range allowed by the informat and the default width. The date 17 October 1991 and the time 2:25:32 p.m. are used for the example in all cases.

Table 5.2 Frequently Used SAS Date and Datetime Informats

Informat and Example	Description	Width Range	Default Width
ANYDTDTE*w.*	Reads and extracts the date value from any of the following: DATE, DATETIME, DDMMYY, JULIAN, MDYAMPM, MMDDYY, MMxYY*, MONYY, TIME, YMDDTTM, YYMMDD, YYQ, YYxMM*, month-day-year	5–32	9
ANYDTDTM*w.*	Reads and extracts the datetime value from any of the following: DATE, DATETIME, DDMMYY, JULIAN, MMDDYY, MMxYY*, MONYY, TIME, YYMMDD, YYQ, YYxMM*, month-day-year	1–32	19
ANYDTTME*w.*	Reads and extracts the time value from any of the following: DATE, DATETIME, DDMMYY, JULIAN, MMDDYY, MONYY, TIME, YYMMDD, YYQ, month-day-year	1–32	8
DATE*w.* 17oct91	Day, month abbreviation, and year: *ddmonyy*	7–32	7
DATETIME*w.d* 17oct91:14:45:32	Date and time: *ddmonyy*:*hh*:*mm*:*ss*	13–40	18
DDMMYY*w.* 17/10/91	Day, month, year: *ddmmyy*, *dd*/*mm*/*yy*, *dd*-*mm*-*yy*, or *dd mm yy*	6–32	6

Table 5.2 *continued*

Informat and Example	Description	Width Range	Default Width
JULIAN*w.* 91290	Year and day of year (Julian dates): *yyddd*	5–32	5
MMDDYY*w.* 10/17/91	Month, day, year: *mmddyy*, *mm/dd/yy*, *mm-dd-yy*, or *mm dd yy*	6–32	6
MONYY*w.* Oct91	Month abbreviation and year: *monyy*	5–32	5
NENGO*w.* H.03/10/17	Japanese Nengo notation	7–32	10
TIME*w.d* 14:45:32	Hours, minutes, seconds: *hh*:*mm*:*ss* or hours, minutes: *hh*:*mm*	5–32	8
WEEKV*w.* 1991-W42-04	ISO 8601 year, week, day of week: *yyyy*-W*ww*-*dd*	3–200	11
YYMMDD*w.* 91/10/17	Year, month, day: *yymmdd*, *yy/mm/dd*, *yy-mm-dd*, or *yy mm dd*	6–32	6
YYQ*w.* 91Q4	Year and quarter of year: *yy*Q*q*	4–32	4

Date, Time, and Datetime Formats

Some of the commonly used SAS date and datetime formats are listed in Table 5.3 and Table 5.4. You can specify the width value for each format by adding *w*. The tables list the range of width values allowed and the default width value for each format.

The notation used by a format is abbreviated in different ways depending on the width option used. For example, the format MMDDYY8. writes the date 17 October 1991 as 10/17/91, while the format MMDDYY6. writes this date as 101791. In particular, formats that display the year show two-digit or four-digit year values depending on the width option. The examples shown in the tables use the default width.

The interval function INTFMT returns a recommended format for time ID values based on the interval that describes the frequency of the values. The following example uses INTFMT to select a format to display the quarterly time ID variable qtrDate. In this example, INTFMT returns the format YYQC6., which displays the year in four digits and the quarter in a single digit. This selected format is stored in a macro variable that is created by the CALL SYMPUT statement. The second argument to INTFMT controls the width of the year for date formats; it can take the value ‘long’ or ‘l’ to indicate 4 for the year width or the value ‘short’ or

's' to indicate 2 for the year width. For more information about the INTFMT function, see the *SAS Language Reference: Dictionary*. For more information about the CALL SYMPUT statement, see the *SAS Language Reference: Dictionary*.

The macro variable &FMT is then used in the FORMAT statement in the PROC PRINT step as follows:

```
data b(keep=qtrDate);
   interval = 'QTR';
   form = INTFMT( interval, 'long' );
   call symput('fmt',form);
   do i=1 to 4;
      qtrDate = INTNX( interval, '01jan00'd, i-1 );
      output;
   end;
run;

proc print;
   format qtrDate &fmt;
run;
```

It is also possible to display date and datetime values as strings by using the format that is identified by INTFMT. In the following example, INTFIT is used to identify the intervals of the sashelp.citiwk and sashelp.air data sets. Then INTFMT is used to identify formats that are based on the intervals. The formats are then used to convert the first SAS date value of each data set to a string. The START variable displays the date of the first observation of each data set. This method assumes that the interval of the data set can be identified by examining the first two observations. This is often the case for output data sets and data sets that have been properly prepared for input by using a procedure such as the TIMESERIES procedure. More than two observations might be required to identify the difference between a DAY interval and a WEEKDAY interval. This example would need to be modified if the DATE variable contained SAS datetime values.

```
data a(keep=DATE0 DATE INTERVAL FMT START);
   length START INTERVAL FMT $32;
   format date0 date DATE.;
   set sashelp.citiwk(obs=2) sashelp.air(obs=2);
   DATE0 = lag(date);
   if ( mod(_n_,2) eq 1 ) then delete;
   if ( mod(_n_,2) eq 0 ) then INTERVAL = intfit( DATE0, date, 'D' );
   if ( mod(_n_,2) eq 0 ) then FMT = INTFMT( interval, 'l' );
   START = putn( DATE0, FMT );
run;

proc print;
run;
```

See *SAS Language Reference: Concepts* for a complete description of these formats, including the variations of the formats produced by different width options. See Chapter 4, "Working with Time Series Data," for a discussion of the use of date and datetime formats.

Date Formats

Table 5.3 lists some of the available SAS date formats. For each format, an example is shown of a date value in the notation produced by the format. The date '17OCT91'D is used as the example.

Table 5.3 Frequently Used SAS Date Formats

Format and Example	Description	Width Range	Default Width
DATE*w.* 17OCT91	Day, month abbreviation, year: *ddmonyy*	5–9	7
DAY*w.* 17	Day of month	2–32	2
DDMMYY*w.* 17/10/91	Day, month, year: *dd*/*mm*/*yy*	2–8	8
DOWNAME*w.* Thursday	Name of day of the week	1–32	9
JULDAY*w.* 290	Day of year	3–32	3
JULIAN*w.* 91290	Year and day of year: *yyddd*	5–7	5
MMDDYY*w.* 10/17/91	Month, day, year: *mm*/*dd*/*yy*	2–8	8
MMYY*w.* 10M1991	Month and year: *mm*M*yyyy*	5–32	7
MMYYC*w.* 10:1991	Month and year: *mm*:*yyyy*	5–32	7
MMYYD*w.* 10-1991	Month and year: *mm*-*yyyy*	5–32	7
MMYYP*w.* 10.1991	Month and year: *mm*.*yyyy*	5–32	7
MMYYS*w.* 10/1991	Month and year: *mm*/*yyyy*	5–32	7
MMYYN*w.* 101991	Month and year: *mmyyyy*	5–32	6

Table 5.3 *continued*

Format and Example	Description	Width Range	Default Width
MONNAME*w.* October	Name of month	1–32	9
MONTH*w.* 10	Month of year	1–32	2
MONYY*w.* OCT91	Month abbreviation and year: *monyy*	5–7	5
QTR*w.* 4	Quarter of year	1–32	1
QTRR*w.* IV	Quarter in roman numerals	3–32	3
NENGO*w.* H.03/10/17	Japanese Nengo notation	2–10	10
WEEKDATE*w.* Thursday, October 17, 1991	*day-of-week*, *month-name dd*, *yyyy*	3–37	29
WEEKDATX*w.* Thursday, 17 October 1991	*day-of-week*, *dd month-name yyyy*	3–37	29
WEEKDAY*w.* 5	Day of week	1–32	1
WEEKV*w.* 1991-W42-04	ISO 8601 year, week, day of week: *yyyy*-W*ww*-*dd*	3–200	11
WORDDATE*w.* October 17, 1991	*month-name dd*, *yyyy*	3–32	18
WORDDATX*w.* 17 October 1991	*dd month-name yyyy*	3–32	18
YEAR*w.* 1991	Year: *yyyy*	2–32	4
YYMM*w.* 1991M10	Year and month: *yyyy*M*mm*	5–32	7
YYMMC*w.* 1991:10	Year and month: *yyyy*:*mm*	5–32	7

Table 5.3 *continued*

Format and Example	Description	Width Range	Default Width
YYMMD*w.* 1991-10	Year and month: *yyyy-mm*	5–32	7
YYMMP*w.* 1991.10	Year and month: *yyyy.mm*	5–32	7
YYMMS*w.* 1991/10	Year and month: *yyyy/mm*	5–32	7
YYMMN*w.* 199110	Year and month: *yyyymm*	5–32	7
YYMON*w.* 1991OCT	Year and month abbreviation: *yyyymon*	5–32	7
YYMMDD*w.* 91/10/17	Year, month, day: *yy/mm/dd*	2–8	8
YYQ*w.* 1991Q4	Year and quarter: *yyyy*Q*q*	4–6	6
YYQC*w.* 1991:4	Year and quarter: *yyyy*:*q*	4–32	6
YYQD*w.* 1991-4	Year and quarter: *yyyy-q*	4–32	6
YYQP*w.* 1991.4	Year and quarter: *yyyy.q*	4–32	6
YYQS*w.* 1991/4	Year and quarter: *yyyy/q*	4–32	6
YYQN*w.* 19914	Year and quarter: *yyyyq*	3–32	5
YYQR*w.* 1991QIV	Year and quarter in roman numerals: *yyyy*Q*rr*	6–32	8
YYQRC*w.* 1991:IV	Year and quarter in roman numerals: *yyyy*:*rr*	6–32	8

Table 5.3 *continued*

Format and Example	Description	Width Range	Default Width
YYQRD*w.* 1991-IV	Year and quarter in roman numerals: *yyyy-rr*	6–32	8
YYQRP*w.* 1991.IV	Year and quarter in roman numerals: *yyyy.rr*	6–32	8
YYQRS*w.* 1991/IV	Year and quarter in roman numerals: *yyyy/rr*	6–32	8
YYQRN*w.* 1991IV	Year and quarter in roman numerals: *yyyyrr*	6–32	8

Datetime and Time Formats

Table 5.4 lists some of the available SAS datetime and time formats. For each format, the example shows the formatted value. The value of the variable dt is '17OCT91:14:25:32'DT. You can specify the width of each format by adding *w*. For formats that allow a decimal value, you can specify the number of decimal digits by adding *d*.

Table 5.4 Frequently Used SAS Datetime and Time Formats

Format	Value and Example	Description	Width Range	Default Width
DATETIME*w.d*	dt 17OCT91:14:25:32	*ddmonyy*:*hh*:*mm*:*ss.ss*	7–40	16
DTWKDATX*w.*	dt Thursday, 17 October 1991	*day-of-week*, *dd month yyyy*	3–37	29
HHMM*w.d*	TIMEPART(dt) 14:26	Hour and minute: *hh*:*mm.mm*	2–20	5
HOUR*w.d*	TIMEPART(dt) 14	Hour: *hh.hh*	2–20	2
MMSS*w.d*	HMS(0,MINUTE(dt),SECOND(dt)) 25:32	Minutes and seconds: *mm*:*ss.ss*	2–20	5
TIME*w.d*	TIMEPART(dt) 14:25:32	Time of day: *hh*:*mm*:*ss.ss*	2–20	8

Table 5.4 *continued*

Format	Value and Example	Description	Width Range	Default Width
TOD*w.d*	dt 14:25:32	Time of day: *hh*:*mm*:*ss.ss*	2–20	8

Alignment of SAS Dates

SAS date values that are used to identify time series observations produced by SAS/ETS and SAS High-Performance Forecasting procedures are normally aligned with the beginning of the time intervals that correspond to the observations. For example, for monthly data for 1994, the date values that identify the observations are 1Jan94, 1Feb94, 1Mar94, . . . , 1Dec94.

However, for some applications it might be preferable to use end-of-period dates, such as 31Jan94, 28Feb94, 31Mar94, . . . , 31Dec94. For other applications, such as plotting time series, it might be more convenient to use interval midpoint dates to identify the observations.

Many SAS/ETS and SAS High-Performance Forecasting procedures provide an ALIGN= option to control the alignment of dates for outputting time series observations. SAS/ETS procedures that support the ALIGN= option are ARIMA, DATASOURCE, ESM, EXPAND, FORECAST, SIMILARITY, TIMESERIES, UCM, and VARMAX. SAS High-Performance Forecasting procedures that support the ALIGN= option are HPFRECONCILE, HPF, HPFDIAGNOSE, HPFENGINE, and HPFEVENTS.

ALIGN=

The ALIGN= option can have the following values:

BEGINNING specifies that dates be aligned to the start of the interval. This is the default. BEGINNING can be abbreviated as BEGIN, BEG, or B.

MIDDLE specifies that dates be aligned to the interval midpoint, the average of the beginning and ending values. MIDDLE can be abbreviated as MID or M.

ENDING specifies that dates be aligned to the end of the interval. ENDING can be abbreviated as END or E.

For information about the calculation of the beginning and ending values of intervals, see the section "Beginning Dates and Datetimes of Intervals" on page 154.

SAS Date, Time, and Datetime Functions

SAS date, time, and datetime functions are used to perform the following tasks:

- compute date, time, and datetime values from calendar and time-of-day values

- compute calendar and time-of-day values from date and datetime values
- convert between date, time, and datetime values
- perform calculations that involve time intervals
- provide information about time intervals
- provide information about seasonality

For all interval functions, you can supply the intervals and other character arguments either directly as a quoted string or as a SAS character variable. When you use a character variable, you should set the length of the character variable to at least the length of the longest string for that variable that is used in the DATA step.

Also, to ensure correct results when using interval functions, use date intervals with date values and datetime intervals with datetime values.

See *SAS Language Reference: Dictionary* for a complete description of these functions.

The following list shows SAS date, time, and datetime functions in alphabetical order.

DATE()
returns today's date as a SAS date value.

DATEJUL(*yyddd* **)**
returns the SAS date value when given the Julian date in *yyddd* or *yyyyddd* format. For example, `DATE = DATEJUL(99001);` assigns the SAS date value '01JAN99'D to DATE, and `DATE = DATEJUL(1999365);` assigns the SAS date value '31DEC1999'D to DATE.

DATEPART(*datetime* **)**
returns the date part of a SAS datetime value as a date value.

DATETIME()
returns the current date and time of day as a SAS datetime value.

DAY(*date* **)**
returns the day of the month from a SAS date value.

DHMS(*date, hour, minute, second* **)**
returns a SAS datetime value for date, hour, minute, and second values.

HMS(*hour, minute, second* **)**
returns a SAS time value for hour, minute, and second values.

HOLIDAY('*holiday*'**,** *year* **)**
returns a SAS date value for the holiday and year specified. Valid values for holiday are 'BOXING', 'CANADA', 'CANADAOBSERVED', 'CHRISTMAS', 'COLUMBUS', 'EASTER', 'FATHERS', 'HALLOWEEN', 'LABOR', 'MLK', 'MEMORIAL', 'MOTHERS', 'NEWYEAR','THANKSGIVING', 'THANKSGIVINGCANADA', 'USINDEPENDENCE', 'USPRESIDENTS', 'VALENTINES', 'VETERANS', 'VETERANSUSG', 'VETERANSUSPS', and 'VICTORIA'. For example: `EASTER2000 = HOLIDAY('EASTER', 2000);`

HOUR(*datetime* **)**
returns the hour from a SAS datetime or time value.

INTCINDEX('*date-interval*'**,** *date* **)**

INTCINDEX('*datetime-interval*'**,** *datetime* **)**
returns the index of the seasonal cycle when given an interval and an appropriate SAS date, datetime, or time value. For example, the seasonal cycle for INTERVAL='DAY' is 'WEEK', so `INTCINDEX('DAY','01SEP78'D);` returns 35 because September 1, 1978, is the sixth day of the 35th week of the year. For correct results, date intervals should be used with date values, and datetime intervals should be used with datetime values.

INTCK('*date-interval*'**,** *date1, date2* **<** *, 'method'* **>** **)**

INTCK('*datetime-interval*'**,** *datetime1, datetime2* **<** *, 'method'* **>** **)**
returns the number of boundaries of intervals of the given kind that lie between the two date or datetime values. The optional method argument specifies that the intervals are counted using either a discrete or a continuous method. The default DISCRETE (or DISC or D) method uses discrete time intervals. For the DISCRETE method, the distance in MONTHS between January 31, 2000, and February 1, 2000, is one month. The CONTINUOUS (or CONT or C) method uses continuous time intervals. For the CONTINUOUS method, the distance in MONTHS between January 15, 2000, and February 14, 2000, is zero, but the distance in MONTHS between January 15, 2000, and February 15, 2000, is one month.

INTCYCLE('*interval*' **<,** *seasonality* **>** **)**
returns the interval of the seasonal cycle, given a date, time, or datetime interval. For example, INTCYCLE('MONTH') returns 'YEAR' because the months January, February, ..., December constitute a yearly cycle. INTCYCLE('DAY') returns 'WEEK' because Sunday, Monday, ..., Saturday constitute a weekly cycle.

You can specify the optional *seasonality* argument to construct a cycle other than the default seasonal cycle. For example, INTCYCLE('MONTH', 3) returns 'QTR'. The optional second argument is the seasonal frequency.

INTFIT(*date1, date2*, '*D*' **)**

INTFIT(*datetime1, datetime2*, '*DT*' **)**

INTFIT(*obs1, obs2*, '*OBS*' **)**
returns an interval that fits exactly between two SAS date, datetime, or observation values. That is, if the interval result of the INTFIT function is used with *date1*, 1, and SAMEDAY alignment in the INTNX function, then the result is *date2*. This concept is illustrated in the following example, where result1 is the same as date1 and result2 is the same as date2.

```
FitInterval = INTFIT( date1, date2, 'D' );
result1 = INTNX( FitInterval, date1, 0, 'SAMEDAY');
result2 = INTNX( FitInterval, date1, 1, 'SAMEDAY');
```

More than one interval can fit the preceding definition. For example, two SAS date values that are seven days apart could be fit with either 'DAY7' or 'WEEK'. The INTFIT function chooses the more common interval, so 'WEEK' is the result when the dates are seven days apart. The INTFIT function can be used to detect the possible frequency of the time series or to analyze frequencies of other events in a time series, such as outliers or missing values.

INTFMT('*interval*' **,**'*size*'**)**

returns a recommended format when given a date, time, or datetime interval for displaying the time ID values associated with a time series of the given interval. The second argument to INTFMT controls the width of the year for date formats; it can take the value 'long' or 'l' to specify that the returned format display a four-digit year or the value 'short' or 's' to specify that the returned format display a two-digit year.

INTGET(*date1, date2, date3* **)**

INTGET(*datetime1, datetime2, datetime3* **)**

returns an interval that fits three consecutive SAS date or datetime values. The INTGET function examines two intervals: the first interval between *date1* and *date2*, and the second interval between *date2* and *date3*. In order for an interval to be detected, either the two intervals must be the same or one interval must be an integer multiple of the other interval. That is, INTGET assumes that at least two of the dates are consecutive points in the time series, and that the other two dates are also consecutive or represent the points before and after missing observations. The INTGET function assumes that large values are SAS datetime values, which are measured in seconds, and that smaller values are SAS date values, which are measured in days. The INTGET function can be used to detect the possible frequency of the time series or to analyze frequencies of other events in a time series, such as outliers or missing values.

INTINDEX('*date-interval*'**,** *date* **<,** *seasonality* **>)**

INTINDEX('*datetime-interval*'**,** *datetime* **<,** *seasonality* **>)**

returns the seasonal index for the specified date or datetime interval and an appropriate date or datetime value. The seasonal index is a number that represents the position of the date or datetime value in the seasonal cycle of the specified interval. For example, `INTINDEX('MONTH','01DEC2000'D);` returns 12 because monthly data is yearly periodic and DECEMBER is the 12th month of the year. However, `INTINDEX('DAY','01DEC2000'D);` returns 6 because daily data is weekly periodic and December 01, 2000, is a Friday, the sixth day of the week. To correctly identify the seasonal index, the interval specification should agree with the date or datetime value. For example, `INTINDEX('DTMONTH','01DEC2000'D);` and `INTINDEX('MONTH','01DEC2000:00:00:00'DT);` do not return the expected value of 12. However, both `INTINDEX('MONTH','01DEC2000'D);` and `INTINDEX('DTMONTH','01DEC2000:00:00:00'DT);` return the expected value of 12.

You can specify the optional *seasonality* argument to use a seasonal cycle other than the default seasonal cycle. For example, `INTINDEX('MONTH','01APR2000'D);` returns the value 4, to indicate the fourth month of the year. However, `INTINDEX('MONTH','01APR2000'D,3);` and `INTINDEX('MONTH','01APR2000'D,'QTR');` return the value 1 to indicate the first month of the quarter. Specifying either 3 or 'QTR' for the third argument uses a quarterly seasonal cycle instead of the default yearly seasonal cycle.

INTNX('*date-interval*'**,** *date***,** *n* **<,** *'alignment'***>)**

INTNX('*datetime-interval*'**,** *datetime***,** *n* **<,** *'alignment'***>)**

returns the date or datetime value of the beginning of the interval that is *n* intervals from the interval that contains the given date or datetime value. The optional *alignment* argument specifies that the returned date is aligned to the beginning, middle, or end of the interval. Beginning is the default. In addition, you can specify SAME (S) alignment. The SAME alignment bases the alignment of the calculated date or datetime value on the alignment of the input date or datetime value. As illustrated in the following example, the SAME alignment can be used to calculate the meaning of "same day next year" or "same day two weeks from now."

```
nextYear = INTNX( 'YEAR', '15Apr2007'D, 1, 'S' );
TwoWeeks = INTNX( 'WEEK', '15Apr2007'D, 2, 'S' );
```

The preceding example returns '15Apr2008'D for nextYear and '29Apr2007'D for TwoWeeks.

For all values of alignment, the number of discrete intervals *n* between the input date and the resulting date agrees with the input value. In the following example, the result is always that n2 = n1:

```
date2 = INTNX( interval, date1, n1, align );
n2 = INTCK( interval, date1, date2 );
```

The preceding example uses the DISCRETE method of the INTCK function by default. The result n2 = n1 does not always apply when the CONTINUOUS method of the INTCK function is specified.

INTSEAS('*interval*' < , *seasonality* >)

returns the length of the seasonal cycle for the specified date or datetime interval. The length of a seasonal cycle is the number of intervals in a seasonal cycle. For example, when the interval for a time series is described as monthly, many procedures use the option INTERVAL=MONTH to indicate that each observation in the data corresponds to a particular month. Monthly data are considered to be periodic for a one-year seasonal cycle. There are 12 months in one year, so the number of intervals (months) in a seasonal cycle (year) is 12. For quarterly data, there are 4 quarters in one year, so the number of intervals in a seasonal cycle is 4. The periodicity is not always one year. For example, INTERVAL=DAY is considered to have a seasonal cycle of one week, and because there are 7 days in a week, the number of intervals in a seasonal cycle is 7.

You can specify the optional *seasonality* argument to use a seasonal cycle other than the default seasonal cycle. For example, INTSEAS('MONTH', 3) and INTSEAS('MONTH', 'QTR') both specify a quarterly seasonal cycle and return the value 3. If the optional *seasonality* argument is numeric, it is the seasonal frequency. If the optional *seasonality* argument is character, it is the seasonal cycle.

INTSHIFT('*interval*')

returns the shift interval that applies to the shift index if a subperiod is specified. For example, YEAR intervals are shifted by MONTH, so INTSHIFT('YEAR') returns 'MONTH'.

INTTEST('*interval*')

returns 1 if the interval name is valid, 0 otherwise. For example, `VALID = INTTEST('MONTH');` should set VALID to 1, while `VALID = INTTEST('NOTANINTERVAL');` should set VALID to 0. The INTTEST function can be useful in verifying which values of multiplier *n* and the shift index *s* are valid in constructing an interval name.

JULDATE(*date*)

returns the Julian date from a SAS date value. The format of the Julian date is either *yyddd* or *yyyyddd* depending on the value of the system option YEARCUTOFF=. For example, using the default system option values, `JULDATE( '31DEC1999'D );` returns 99365, while `JULDATE( '31DEC1899'D );` returns 1899365.

MDY(*month, day, year*)

returns a SAS date value for month, day, and year values.

MINUTE(*datetime* **)**
returns the minute from a SAS time or datetime value.

MONTH(*date* **)**
returns the numerical value for the month of the year from a SAS date value. For example, `MONTH=MONTH('01JAN2000'D);` returns 1, the numerical value for January.

NWKDOM(*n, weekday, month, year* **)**
returns a SAS date value for the *n*th weekday of the month and year specified. For example, Thanksgiving is always the fourth (*n*=4) Thursday (*weekday*=5) in November (*month*=11). Thus `THANKS2000 = NWKDOM( 4, 5, 11, 2000);` returns the SAS date value for Thanksgiving in the year 2000. The last weekday of a month can be specified by using *n*=5. Memorial Day in the United States is the last (*n*=5) Monday (*weekday*=2) in May (*month*=5), and so `MEMORIAL2002 = NWKDOM( 5, 2, 5, 2002);` returns the SAS date value for Memorial Day in 2002. Because $n = 5$ always specifies the last occurrence of the month and most months have only 4 instances of each day, the result for $n = 5$ is often the same as the result for $n = 4$. NWKDOM is useful for calculating the SAS date values of holidays that are defined in this manner.

QTR(*date* **)**
returns the quarter of the year from a SAS date value.

SECOND(*date* **)**
returns the second from a SAS time or datetime value.

TIME()
returns the current time of day.

TIMEPART(*datetime* **)**
returns the time part of a SAS datetime value.

TODAY()
returns the current date as a SAS date value. (TODAY is another name for the DATE function.)

WEEK(*date* <*, 'descriptor'*> **)**
returns the week of year from a SAS date value. The algorithm used to calculate the week depends on the *descriptor*, which can take the value 'U', 'V', or 'W'.

If the descriptor is 'U,' weeks start on Sunday and the range is 0 to 53. If weeks 0 and 53 exist, they are only partial weeks. Week 52 can be a partial week.

If the descriptor is 'V', the result is equivalent to the ISO 8601 week of year definition. The range is 1 to 53. Week 53 is a leap week. The first week of the year, Week 1, and the last week of the year, Week 52 or 53, can include days in another Gregorian calendar year.

If the descriptor is 'W', weeks start on Monday and the range is 0 to 53. If weeks 0 and 53 exist, they are only partial weeks. Week 52 can be a partial week.

WEEKDAY(*date* **)**
returns the day of the week from a SAS date value. For example `WEEKDAY=WEEKDAY('17OCT1991'D);` returns 5, the numerical value for Thursday.

YEAR(*date* **)**
returns the year from a SAS date value.

YYQ(*year, quarter* **)**
returns a SAS date value for year and quarter values.

References

National Retail Federation (2007). "National Retail Federation 4-5-4 Calendar." http://www.nrf.com/.

Technical Committee ISO/TC 154 (Processes, Data Elements, and Documents in Commerce, Industry, and Administration) (2004). *ISO 8601:2004 Data Elements and Interchange Formats—Information Interchange—Representation of Dates and Times.* 3rd ed. Technical report, International Organization for Standardization.

Chapter 6
SAS Macros and Functions

Contents

SAS Macros **179**
BOXCOXAR Macro 180
DFPVALUE Macro 182
DFTEST Macro 183
LOGTEST Macro 185
Functions **187**
PROBDF Function for Dickey-Fuller Tests 187
References **193**

SAS Macros

This chapter describes several SAS macros and the SAS function PROBDF that are provided with SAS/ETS software. A SAS macro is a program that generates SAS statements. Macros make it easy to produce and execute complex SAS programs that would be time-consuming to write yourself.

SAS/ETS software includes the following macros:

%AR	generates statements to define autoregressive error models for the MODEL procedure.
%BOXCOXAR	investigates Box-Cox transformations useful for modeling and forecasting a time series.
%DFPVALUE	computes probabilities for Dickey-Fuller test statistics.
%DFTEST	performs Dickey-Fuller tests for unit roots in a time series process.
%LOGTEST	tests to see if a log transformation is appropriate for modeling and forecasting a time series.
%MA	generates statements to define moving-average error models for the MODEL procedure.
%PDL	generates statements to define polynomial-distributed lag models for the MODEL procedure.

These macros are part of the SAS AUTOCALL facility and are automatically available for use in your SAS program. See *SAS Macro Language: Reference* for information about the SAS macro facility.

Since the %AR, %MA, and %PDL macros are used only with PROC MODEL, they are documented with the MODEL procedure. See the sections on the %AR, %MA, and %PDL macros in Chapter 26, "The MODEL Procedure," for more information about these macros. The %BOXCOXAR, %DFPVALUE, %DFTEST, and %LOGTEST macros are described in the following sections.

BOXCOXAR Macro

The %BOXCOXAR macro finds the optimal Box-Cox transformation for a time series.

Transformations of the dependent variable are a useful way of dealing with nonlinear relationships or heteroscedasticity. For example, the logarithmic transformation is often used for modeling and forecasting time series that show exponential growth or that show variability proportional to the level of the series.

The Box-Cox transformation is a general class of power transformations that include the log transformation and no transformation as special cases. The Box-Cox transformation is

$$Y_t = \begin{cases} \frac{(X_t+c)^{\lambda}-1}{\lambda} & \text{for } \lambda \neq 0 \\ \ln(X_t + c) & \text{for } \lambda = 0 \end{cases}$$

The parameter λ controls the shape of the transformation. For example, λ=0 produces a log transformation, while λ=0.5 results in a square root transformation. When λ=1, the transformed series differs from the original series by $c - 1$.

The constant c is optional. It can be used when some X_t values are negative or 0. You choose c so that the series X_t is always greater than $-c$.

The %BOXCOXAR macro tries a range of λ values and reports which of the values tried produces the optimal Box-Cox transformation. To evaluate different λ values, the %BOXCOXAR macro transforms the series with each λ value and fits an autoregressive model to the transformed series. It is assumed that this autoregressive model is a reasonably good approximation to the true time series model appropriate for the transformed series. The likelihood of the data under each autoregressive model is computed, and the λ value that produces the maximum likelihood over the values tried is reported as the optimal Box-Cox transformation for the series.

The %BOXCOXAR macro prints and optionally writes to a SAS data set all of the λ values tried, the corresponding log-likelihood value, and related statistics for the autoregressive model.

You can control the range and number of λ values tried. You can also control the order of the autoregressive models fit to the transformed series. You can difference the transformed series before the autoregressive model is fit.

Note that the Box-Cox transformation might be appropriate when the data have a common distribution (apart from heteroscedasticity) but not when groups of observations for the variable are quite different. Thus the %BOXCOXAR macro is more often appropriate for time series data than for cross-sectional data.

Syntax

The form of the %BOXCOXAR macro is

%BOXCOXAR *(SAS-data-set, variable < , options >)* **;**

The first argument, *SAS-data-set*, specifies the name of the SAS data set that contains the time series to be analyzed. The second argument, *variable*, specifies the time series variable name to be analyzed. The first two arguments are required.

The following options can be used with the %BOXCOXAR macro. Options must follow the required arguments and are separated by commas.

AR=*n*
specifies the order of the autoregressive model fit to the transformed series. The default is AR=5.

CONST=*value*
specifies a constant c to be added to the series before transformation. Use the CONST= option when some values of the series are 0 or negative. The default is CONST=0.

DIF=*(differencing-list)*
specifies the degrees of differencing to apply to the transformed series before the autoregressive model is fit. The *differencing-list* is a list of positive integers separated by commas and enclosed in parentheses. For example, DIF=(1,12) specifies that the transformed series be differenced once at lag 1 and once at lag 12. For more details, see the section "IDENTIFY Statement" on page 253 in Chapter 8, "The ARIMA Procedure."

LAMBDAHI=*value*
specifies the maximum value of lambda for the grid search. The default is LAMBDAHI=1. A large (in magnitude) LAMBDAHI= value can result in problems with floating point arithmetic.

LAMBDALO=*value*
specifies the minimum value of lambda for the grid search. The default is LAMBDALO=0. A large (in magnitude) LAMBDALO= value can result in problems with floating point arithmetic.

NLAMBDA=*value*
specifies the number of lambda values considered, including the LAMBDALO= and LAMBDAHI= option values. The default is NLAMBDA=2.

OUT=*SAS-data-set*
writes the results to an output data set. The output data set includes the lambda values tried (LAMBDA), and for each lambda value, the log likelihood (LOGLIK), residual mean squared error (RMSE), Akaike Information Criterion (AIC), and Schwarz's Bayesian Criterion (SBC).

PRINT=YES | NO
specifies whether results are printed. The default is PRINT=YES. The printed output contains the lambda values, log likelihoods, residual mean square errors, Akaike Information Criterion (AIC), and Schwarz's Bayesian Criterion (SBC).

Results

The value of λ that produces the maximum log likelihood is returned in the macro variable &BOXCOXAR. The value of the variable &BOXCOXAR is "ERROR" if the %BOXCOXAR macro is unable to compute the best transformation due to errors. This might be the result of large lambda values. The Box-Cox transformation parameter involves exponentiation of the data, so that large lambda values can cause floating-point overflow.

Results are printed unless the PRINT=NO option is specified. Results are also stored in SAS data sets when the OUT= option is specified.

Details

Assume that the transformed series Y_t is a stationary pth order autoregressive process generated by independent normally distributed innovations.

$$(1 - \Theta(B))(Y_t - \mu) = \epsilon_t$$

$$\epsilon_t \sim iid\,\mathrm{N}(0, \sigma^2)$$

Given these assumptions, the log-likelihood function of the transformed data Y_t is

$$\begin{aligned} l_Y(\cdot) = & -\frac{n}{2}\ln(2\pi) - \frac{1}{2}\ln(|\Sigma|) - \frac{n}{2}\ln(\sigma^2) \\ & -\frac{1}{2\sigma^2}(\mathbf{Y} - \mathbf{1}\mu)'\Sigma^{-1}(\mathbf{Y} - \mathbf{1}\mu) \end{aligned}$$

In this equation, n is the number of observations, μ is the mean of Y_t, $\mathbf{1}$ is the n-dimensional column vector of 1s, σ^2 is the innovation variance, $\mathbf{Y} = (Y_1, \cdots, Y_n)'$, and Σ is the covariance matrix of Y.

The log-likelihood function of the original data $X_1, \cdots, X_n$ is

$$l_X(\cdot) = l_Y(\cdot) + (\lambda - 1)\sum_{t=1}^{n}\ln(X_t + c)$$

where c is the value of the CONST= option.

For each value of λ, the maximum log-likelihood of the original data is obtained from the maximum log-likelihood of the transformed data given the maximum likelihood estimate of the autoregressive model.

The maximum log-likelihood values are used to compute the Akaike Information Criterion (AIC) and Schwarz's Bayesian Criterion (SBC) for each λ value. The residual mean squared error based on the maximum likelihood estimator is also produced. To compute the mean squared error, the predicted values from the model are transformed again to the original scale (Pankratz 1983, pp. 256–258; Taylor 1986).

After differencing as specified by the DIF= option, the process is assumed to be a stationary autoregressive process. You can check for stationarity of the series with the %DFTEST macro. If the process is not stationary, differencing with the DIF= option is recommended. For a process with moving-average terms, a large value for the AR= option might be appropriate.

DFPVALUE Macro

The %DFPVALUE macro computes the significance of the Dickey-Fuller test. The %DFPVALUE macro evaluates the p-value for the Dickey-Fuller test statistic τ for the test of $\mathrm{H_0}$: "The time series has a unit root" versus H_a: "The time series is stationary" using tables published by Dickey (1976); Dickey, Hasza, and Fuller (1984).

The %DFPVALUE macro can compute p-values for tests of a simple unit root with lag 1 or for seasonal unit roots at lags 2, 4, or 12. The %DFPVALUE macro takes into account whether an intercept or deterministic time trend is assumed for the series.

The %DFPVALUE macro is used by the %DFTEST macro described later in this chapter.

Note that the %DFPVALUE macro has been superseded by the PROBDF function described later in this chapter. It remains for compatibility with past releases of SAS/ETS.

Syntax

The %DFPVALUE macro has the following form:

%DFPVALUE *(tau, nobs < , options >)* **;**

The first argument, *tau,* specifies the value of the Dickey-Fuller test statistic.

The second argument, *nobs,* specifies the number of observations on which the test statistic is based.

The first two arguments are required. The following options can be used with the %DFPVALUE macro. Options must follow the required arguments and are separated by commas.

DLAG=1 | 2 | 4 | 12
specifies the lag period of the unit root to be tested. DLAG=1 specifies a one-period unit root test. DLAG=2 specifies a test for a seasonal unit root with lag 2. DLAG=4 specifies a test for a seasonal unit root with lag 4. DLAG=12 specifies a test for a seasonal unit root with lag 12. The default is DLAG=1.

TREND=0 | 1 | 2
specifies the degree of deterministic time trend included in the model. TREND=0 specifies no trend and assumes the series has a zero mean. TREND=1 includes an intercept term. TREND=2 specifies both an intercept and a deterministic linear time trend term. The default is TREND=1. TREND=2 is not allowed with DLAG=2, 4, or 12.

Results

The computed *p*-value is returned in the macro variable &DFPVALUE. If the *p*-value is less than 0.01 or larger than 0.99, the macro variable &DFPVALUE is set to 0.01 or 0.99, respectively.

Minimum Observations

The minimum number of observations required by the %DFPVALUE macro depends on the value of the DLAG= option. The minimum observations are as follows:

DLAG=	Minimum Observations
1	9
2	6
4	4
12	12

DFTEST Macro

The %DFTEST macro performs the Dickey-Fuller unit root test. You can use the %DFTEST macro to decide whether a time series is stationary and to determine the order of differencing required for the time series analysis of a nonstationary series.

Most time series analysis methods require that the series to be analyzed is stationary. However, many economic time series are nonstationary processes. The usual approach to this problem is to difference the series. A time series that can be made stationary by differencing is said to have a *unit root*. For more information, see the discussion of this issue in the section "Getting Started: ARIMA Procedure" on page 219 of Chapter 8, "The ARIMA Procedure."

The Dickey-Fuller test is a method for testing whether a time series has a unit root. The %DFTEST macro tests the hypothesis H_0: "The time series has a unit root" versus H_a: "The time series is stationary" based on tables provided in Dickey (1976); Dickey, Hasza, and Fuller (1984). The test can be applied for a simple unit root with lag 1, or for seasonal unit roots at lag 2, 4, or 12.

Note that the %DFTEST macro has been superseded by the PROC ARIMA stationarity tests. See Chapter 8, "The ARIMA Procedure," for details.

Syntax

The %DFTEST macro has the following form:

%DFTEST *(SAS-data-set, variable < , options >)* **;**

The first argument, *SAS-data-set,* specifies the name of the SAS data set that contains the time series variable to be analyzed.

The second argument, *variable,* specifies the time series variable name to be analyzed.

The first two arguments are required. The following options can be used with the %DFTEST macro. Options must follow the required arguments and are separated by commas.

AR=*n*
specifies the order of autoregressive model fit after any differencing specified by the DIF= and DLAG= options. The default is AR=3.

DIF=*(differencing-list)*
specifies the degrees of differencing to be applied to the series. The differencing list is a list of positive integers separated by commas and enclosed in parentheses. For example, DIF=(1,12) specifies that the series be differenced once at lag 1 and once at lag 12. For more details, see the section "IDENTIFY Statement" on page 253 in Chapter 8, "The ARIMA Procedure."

If the option DIF=($d_1, \cdots, d_k$) is specified, the series analyzed is $(1 - B^{d_1}) \cdots (1 - B^{d_k}) Y_t$, where Y_t is the variable specified, and B is the backshift operator defined by $BY_t = Y_{t-1}$.

DLAG=1 | 2 | 4 | 12
specifies the lag to be tested for a unit root. The default is DLAG=1.

OUT=*SAS-data-set*
writes residuals to an output data set.

OUTSTAT=*SAS-data-set*
writes the test statistic, parameter estimates, and other statistics to an output data set.

TREND=0 | 1 | 2
specifies the degree of deterministic time trend included in the model. TREND=0 includes no deterministic term and assumes the series has a zero mean. TREND=1 includes an intercept term. TREND=2 specifies an intercept and a linear time trend term. The default is TREND=1. TREND=2 is not allowed with DLAG=2, 4, or 12.

Results

The computed p-value is returned in the macro variable &DFTEST. If the p-value is less than 0.01 or larger than 0.99, the macro variable &DFTEST is set to 0.01 or 0.99, respectively. (The same value is given in the macro variable &DFPVALUE returned by the %DFPVALUE macro, which is used by the %DFTEST macro to compute the p-value.)

Results can be stored in SAS data sets with the OUT= and OUTSTAT= options.

Minimum Observations

The minimum number of observations required by the %DFTEST macro depends on the value of the DLAG= option. Let s be the sum of the differencing orders specified by the DIF= option, let t be the value of the TREND= option, and let p be the value of the AR= option. The minimum number of observations required is as follows:

DLAG=	Minimum Observations
1	$1 + p + s + \max(9, p + t + 2)$
2	$2 + p + s + \max(6, p + t + 2)$
4	$4 + p + s + \max(4, p + t + 2)$
12	$12 + p + s + \max(12, p + t + 2)$

Observations are not used if they have missing values for the series or for any lag or difference used in the autoregressive model.

LOGTEST Macro

The %LOGTEST macro tests whether a logarithmic transformation is appropriate for modeling and forecasting a time series. The logarithmic transformation is often used for time series that show exponential growth or variability proportional to the level of the series.

The %LOGTEST macro fits an autoregressive model to a series and fits the same model to the log of the series. Both models are estimated by the maximum-likelihood method, and the maximum log-likelihood values for both autoregressive models are computed. These log-likelihood values are then expressed in terms of the original data and compared.

You can control the order of the autoregressive models. You can also difference the series and the log-transformed series before the autoregressive model is fit.

You can print the log-likelihood values and related statistics (AIC, SBC, and MSE) for the autoregressive models for the series and the log-transformed series. You can also output these statistics to a SAS data set.

Syntax

The %LOGTEST macro has the following form:

%LOGTEST *(SAS-data-set, variable,* < *options* > *)* **;**

The first argument, *SAS-data-set,* specifies the name of the SAS data set that contains the time series variable to be analyzed. The second argument, *variable,* specifies the time series variable name to be analyzed.

The first two arguments are required. The following options can be used with the %LOGTEST macro. Options must follow the required arguments and are separated by commas.

AR=*n*

specifies the order of the autoregressive model fit to the series and the log-transformed series. The default is AR=5.

CONST=*value*

specifies a constant to be added to the series before transformation. Use the CONST= option when some values of the series are 0 or negative. The series analyzed must be greater than the negative of the CONST= value. The default is CONST=0.

DIF=(*differencing-list* **)**

specifies the degrees of differencing applied to the original and log-transformed series before fitting the autoregressive model. The *differencing-list* is a list of positive integers separated by commas and enclosed in parentheses. For example, DIF=(1,12) specifies that the transformed series be differenced once at lag 1 and once at lag 12. For more details, see the section "IDENTIFY Statement" on page 253 in Chapter 8, "The ARIMA Procedure."

OUT=*SAS-data-set*

writes the results to an output data set. The output data set includes a variable TRANS that identifies the transformation (LOG or NONE), the log-likelihood value (LOGLIK), residual mean squared error (RMSE), Akaike Information Criterion (AIC), and Schwarz's Bayesian Criterion (SBC) for the log-transformed and untransformed cases.

PRINT=YES | NO

specifies whether the results are printed. The default is PRINT=NO. The printed output shows the log-likelihood value, residual mean squared error, Akaike Information Criterion (AIC), and Schwarz's Bayesian Criterion (SBC) for the log-transformed and untransformed cases.

Results

The result of the test is returned in the macro variable &LOGTEST. The value of the &LOGTEST variable is 'LOG' if the model fit to the log-transformed data has a larger log likelihood than the model fit to the untransformed series. The value of the &LOGTEST variable is 'NONE' if the model fit to the untransformed data has a larger log likelihood. The variable &LOGTEST is set to 'ERROR' if the %LOGTEST macro is unable to compute the test due to errors.

Results are printed when the PRINT=YES option is specified. Results are stored in SAS data sets when the OUT= option is specified.

Details

Assume that a time series X_t is a stationary pth order autoregressive process with normally distributed white noise innovations. That is,

$$(1 - \Theta(B))(X_t - \mu_{\mathbf{x}}) = \epsilon_t$$

where $\mu_{\mathbf{x}}$ is the mean of X_t.

The log likelihood function of X_t is

$$l_1(\cdot) = -\frac{n}{2}\ln(2\pi) - \frac{1}{2}\ln(|\Sigma_{\mathbf{xx}}|) - \frac{n}{2}\ln(\sigma^2_{\mathbf{e}}) - \frac{1}{2\sigma^2_{\mathbf{e}}}(\mathbf{X} - \mathbf{1}\mu_{\mathbf{x}})'\Sigma^{-1}_{\mathbf{xx}}(\mathbf{X} - \mathbf{1}\mu_{\mathbf{x}})$$

where n is the number of observations, $\mathbf{1}$ is the n-dimensional column vector of 1s, $\sigma_{\mathbf{e}}^2$ is the variance of the white noise, $\mathbf{X} = (X_1, \cdots, X_n)'$, and $\Sigma_{\mathbf{xx}}$ is the covariance matrix of $\mathbf{X}$.

On the other hand, if the log-transformed time series $Y_t = \ln(X_t + c)$ is a stationary pth order autoregressive process, the log-likelihood function of X_t is

$$l_0(\cdot) = -\frac{n}{2}\ln(2\pi) - \frac{1}{2}\ln(|\Sigma_{\mathbf{yy}}|) - \frac{n}{2}\ln(\sigma_{\mathbf{e}}^2) - \frac{1}{2\sigma_{\mathbf{e}}^2}(\mathbf{Y} - \mathbf{1}\mu_{\mathbf{y}})'\Sigma_{\mathbf{yy}}^{-1}(\mathbf{Y} - \mathbf{1}\mu_{\mathbf{y}}) - \sum_{t=1}^{n}\ln(X_t + c)$$

where $\mu_{\mathbf{y}}$ is the mean of Y_t, $\mathbf{Y} = (Y_1, \cdots, Y_n)'$, and $\Sigma_{\mathbf{yy}}$ is the covariance matrix of $\mathbf{Y}$.

The %LOGTEST macro compares the maximum values of $l_1(\cdot)$ and $l_0(\cdot)$ and determines which is larger.

The %LOGTEST macro also computes the Akaike Information Criterion (AIC), Schwarz's Bayesian Criterion (SBC), and residual mean squared error based on the maximum likelihood estimator for the autoregressive model. For the mean squared error, retransformation of forecasts is based on Pankratz (1983, pp. 256–258).

After differencing as specified by the DIF= option, the process is assumed to be a stationary autoregressive process. You might want to check for stationarity of the series using the %DFTEST macro. If the process is not stationary, differencing with the DIF= option is recommended. For a process with moving average terms, a large value for the AR= option might be appropriate.

Functions

PROBDF Function for Dickey-Fuller Tests

The PROBDF function calculates significance probabilities for Dickey-Fuller tests for unit roots in time series. The PROBDF function can be used wherever SAS library functions can be used, including DATA step programs, SCL programs, and PROC MODEL programs.

Syntax

PROBDF(*x, n* < *, d* < *, type* > >)

x is the test statistic.

n is the sample size. The minimum value of *n* allowed depends on the value specified for the third argument *d*. For *d* in the set (1,2,4,6,12), *n* must be an integer greater than or equal to $\max(2d, 5)$; for other values of *d* the minimum value of *n* is 24.

d is an optional integer giving the degree of the unit root tested for. Specify $d = 1$ for tests of a simple unit root $(1 - \mathrm{B})$. Specify *d* equal to the seasonal cycle length for tests for a seasonal unit root $(1 - \mathrm{B}^d)$. The default value of *d* is 1; that is, a test for a simple unit root $(1 - \mathrm{B})$ is assumed if *d* is not specified. The maximum value of *d* allowed is 12.

type is an optional character argument that specifies the type of test statistic used. The values of *type* are the following:

SZM studentized test statistic for the zero mean (no intercept) case

RZM regression test statistic for the zero mean (no intercept) case

SSM studentized test statistic for the single mean (intercept) case

RSM regression test statistic for the single mean (intercept) case

STR studentized test statistic for the deterministic time trend case

RTR regression test statistic for the deterministic time trend case

The values STR and RTR are allowed only when $d = 1$. The default value of *type* is SZM.

Details

Theoretical Background

When a time series has a unit root, the series is nonstationary and the ordinary least squares (OLS) estimator is not normally distributed. The limiting distribution of the OLS estimator of autoregressive models for time series with a simple unit root was studied by Dickey (1976); Dickey and Fuller (1979). Dickey, Hasza, and Fuller (1984) obtained the limiting distribution for time series with seasonal unit roots. We will mainly introduce the nonseasonal tests in the following and list references for the nonseasonal tests.

Consider the Dickey-Fuller regression first. The null hypothesis is that there is an autoregressive unit root $H_0 : \alpha = 1$, and the alternative is $H_a : |\alpha| < 1$, where α is the autoregressive coefficient of the time series

$$y_t = \alpha y_{t-1} + \epsilon_t$$

This is referred to as the zero mean (ZM) model. The standard Dickey-Fuller (DF) test assumes that errors ϵ_t are white noise. There are two other types of regression models that include a constant or a time trend as follows:

$$y_t = \mu + \alpha y_{t-1} + \epsilon_t$$
$$y_t = \mu + \beta t + \alpha y_{t-1} + \epsilon_t$$

These two models are referred to as the constant mean model (SM) and the trend model (TR), respectively. The constant mean model includes a constant mean μ of the time series. However, the interpretation of μ depends on the stationarity in the following sense: the mean in the stationary case when $\alpha < 1$ is the trend in the integrated case when $\alpha = 1$. Therefore, the null hypothesis should be the joint hypothesis that $\alpha = 1$ and $\mu = 0$. However for the unit root tests, the test statistics are concerned with the null hypothesis of $\alpha = 1$. The joint null hypothesis is not commonly used. This issue is address in Bhargava (1986) with a different nesting model.

Under the null of I(1) of the Dickey-Fuller test, the differenced process is not serially correlated. There is a great need for the generalization of this specification. The augmented Dickey-Fuller (ADF) test, originally proposed in Dickey and Fuller (1979), adjusts for the serial correlation in the time series by adding lagged first differences to the autoregressive model as follows. Consider the $(p + 1)$th-order autoregressive time series

$$y_t = \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + \cdots + \alpha_{p+1} y_{t-p-1} + e_t$$

and its characteristic equation

$$m^{p+1} - \alpha_1 m^p - \alpha_2 m^{p-1} - \cdots - \alpha_{p+1} = 0$$

If all the characteristic roots are less than 1 in absolute value, y_t is stationary. y_t is nonstationary if there is a unit root. If there is a unit root, the sum of the autoregressive parameters is 1, and hence you can test for a unit root by testing whether the sum of the autoregressive parameters is 1 or not. The no-intercept model is parameterized as

$$\nabla y_t = \delta y_{t-1} + \theta_1 \nabla y_{t-1} + \cdots + \theta_p \nabla y_{t-p} + e_t$$

where $\nabla y_t = y_t - y_{t-1}$ and

$$\delta = \alpha_1 + \cdots + \alpha_{p+1} - 1$$

$$\theta_k = -\alpha_{k+1} - \cdots - \alpha_{p+1}$$

The estimators are obtained by regressing ∇y_t on $y_{t-1}, \nabla y_{t-1}, \cdots, \nabla y_{t-p}$. The t statistic of the ordinary least squares estimator of δ is the test statistic for the unit root test.

If the *type* argument value specifies a test for a nonzero mean (intercept case), the autoregressive model includes a mean term α_0. If the *type* argument value specifies a test for a time trend, the model also includes a time trend term and the model is as follows:

$$\nabla y_t = \alpha_0 + \gamma t + \delta y_{t-1} + \theta_1 \nabla y_{t-1} + \cdots + \theta_p \nabla y_{t-p} + e_t$$

For testing for a seasonal unit root, consider the multiplicative model

$$(1 - \alpha_d B^d)(1 - \theta_1 B - \cdots - \theta_p B^p) y_t = e_t$$

Let $\nabla^d y_t \equiv y_t - y_{t-d}$. The test statistic is calculated in the following steps:

1. Regress $\nabla^d y_t$ on $\nabla^d y_{t-1} \cdots \nabla^d y_{t-p}$ to obtain the initial estimators $\hat{\theta}_i$ and compute residuals $\hat{e}_t$. Under the null hypothesis that $\alpha_d = 1$, $\hat{\theta}_i$ are consistent estimators of θ_i.
2. Regress $\hat{e}_t$ on $(1 - \hat{\theta}_1 B - \cdots - \hat{\theta}_p B^p) y_{t-d}, \nabla^d y_{t-1}, \cdots, \nabla^d y_{t-p}$ to obtain estimates of $\delta = \alpha_d - 1$ and $\theta_i - \hat{\theta}_i$.

The t ratio for the estimate of δ produced by the second step is used as a test statistic for testing for a seasonal unit root. The estimates of θ_i are obtained by adding the estimates of $\theta_i - \hat{\theta}_i$ from the second step to $\hat{\theta}_i$ from the first step.

The series $(1 - B^d) y_t$ is assumed to be stationary, where d is the value of the third argument to the PROBDF function.

If the series is an ARMA process, a large value of p might be desirable in order to obtain a reliable test statistic. To determine an appropriate value for p, see Said and Dickey (1984).

Test Statistics

The Dickey-Fuller test is used to test the null hypothesis that the time series exhibits a lag d unit root against the alternative of stationarity. The PROBDF function computes the probability of observing a test statistic more extreme than x under the assumption that the null hypothesis is true. You should reject the unit root hypothesis when PROBDF returns a small (significant) probability value.

Consider the Dickey-Fuller regression first. There are several different versions of the Dickey-Fuller test. The PROBDF function supports six versions, as selected by the *type* argument. Specify the *type* value that corresponds to the way that you calculated the test statistic *x*.

The last two characters of the *type* value specify the kind of regression model used to compute the Dickey-Fuller test statistic. The meaning of the last two characters of the *type* value are as follows:

ZM zero mean or no-intercept case. The test statistic x is assumed to be computed from the regression model

$$y_t = \alpha y_{t-1} + e_t$$

SM single mean or intercept case. The test statistic x is assumed to be computed from the regression model

$$y_t = \mu + \alpha y_{t-1} + e_t$$

TR intercept and deterministic time trend case. The test statistic x is assumed to be computed from the regression model

$$y_t = \mu + \gamma t + \alpha y_{t-1} + e_t$$

The first character of the *type* value specifies whether the regression test statistic or the studentized test statistic is used. Let $\hat{\alpha}$ be the estimated regression coefficient for the lag of the series, and let $\mathrm{se}_{\hat{\alpha}}$ be the standard error of $\hat{\alpha}$. The meaning of the first character of the *type* value is as follows:

R the regression-coefficient-based test statistic. The test statistic is

$$\rho = n(\hat{\alpha} - 1)$$

S the studentized test statistic. The test statistic is

$$DF_\tau = \frac{(\hat{\alpha} - 1)}{\mathrm{se}_{\hat{\alpha}}}$$

The first one is also called ρ-test and the second is called τ-test. For the zero mean model, the asymptotic distributions of the Dickey-Fuller test statistics are

$$n(\hat{\alpha} - 1) \Rightarrow \left(\int_0^1 W(r)dW(r)\right)\left(\int_0^1 W(r)^2 dr\right)^{-1}$$
$$DF_\tau \Rightarrow \left(\int_0^1 W(r)dW(r)\right)\left(\int_0^1 W(r)^2 dr\right)^{-1/2}$$

For the constant mean model, the asymptotic distributions are

$$n(\hat{\alpha} - 1) \Rightarrow \left([W(1)^2 - 1]/2 - W(1)\int_0^1 W(r)dr\right)\left(\int_0^1 W(r)^2 dr - \left(\int_0^1 W(r)dr\right)^2\right)^{-1}$$
$$DF_\tau \Rightarrow \left([W(1)^2 - 1]/2 - W(1)\int_0^1 W(r)dr\right)\left(\int_0^1 W(r)^2 dr - \left(\int_0^1 W(r)dr\right)^2\right)^{-1/2}$$

For the trend model, the asymptotic distributions are

$$n(\hat{\alpha}-1) \Rightarrow \left[W(r)dW + 12\left(\int_0^1 rW(r)dr - \frac{1}{2}\int_0^1 W(r)dr\right)\left(\int_0^1 W(r)dr - \frac{1}{2}W(1)\right) - W(1)\int_0^1 W(r)dr\right] D^{-1}$$

$$DF_\tau \Rightarrow \left[W(r)dW + 12\left(\int_0^1 rW(r)dr - \frac{1}{2}\int_0^1 W(r)dr\right)\left(\int_0^1 W(r)dr - \frac{1}{2}W(1)\right) - W(1)\int_0^1 W(r)dr\right] D^{1/2}$$

where

$$D = \int_0^1 W(r)^2 dr - 12\left(\int_0^1 r(W(r)dr\right)^2 + 12\int_0^1 W(r)dr\int_0^1 rW(r)dr - 4\left(\int_0^1 W(r)dr\right)^2$$

For more information about the Dickey-Fuller test null distribution see: Dickey and Fuller (1979); Dickey, Hasza, and Fuller (1984); Hamilton (1994). The preceding formulas are for the basic Dickey-Fuller test. The PROBDF function can also be used for the augmented Dickey-Fuller test, in which the error term e_t is modeled as an autoregressive process; however, the test statistic is computed somewhat differently for the augmented Dickey-Fuller test. For the nonseasonal augmented Dickey-Fuller test, the test statistics can take one of the two forms similar to Dickey-Fuller test. One is the OLS t value

$$\frac{\hat{\alpha}-1}{sd(\hat{\alpha})}$$

and the other is given by

$$\frac{n(\hat{\alpha}-1)}{1-\hat{\alpha}_1-\ldots-\hat{\alpha}_p}$$

The asymptotic distributions of the test statistics are the same as those of the standard Dickey-Fuller test statistics. For information about seasonal and nonseasonal augmented Dickey-Fuller tests see: Dickey, Hasza, and Fuller (1984); Hamilton (1994).

The PROBDF function is calculated from approximating functions fit to empirical quantiles that are produced by a Monte Carlo simulation that employs 10^8 replications for each simulation. Separate simulations were performed for selected values of n and for $d = 1, 2, 4, 6, 12$ (where n and d are the second and third arguments to the PROBDF function).

The maximum error of the PROBDF function is approximately $\pm 10^{-3}$ for d in the set (1,2,4,6,12) and can be slightly larger for other d values. Because the number of simulation replications used to produce the PROBDF function is much greater than the 60,000 replications used by Dickey and colleagues (Dickey and Fuller 1979; Dickey, Hasza, and Fuller 1984), the PROBDF function can be expected to produce results that are substantially more accurate than the critical values reported in those papers.

Examples

Suppose the data set TEST contains 104 observations of the time series variable Y, and you want to test the null hypothesis that there exists a lag 4 seasonal unit root in the Y series. The following statements illustrate how to perform the single-mean Dickey-Fuller regression coefficient test using PROC REG and PROBDF.

```
data test1;
   set test;
   y4 = lag4(y);
run;

proc reg data=test1 outest=alpha;
   model y = y4 / noprint;
run;

data _null_;
   set alpha;
   x = 100 * ( y4 - 1 );
   p = probdf( x, 100, 4, "RSM" );
   put p= pvalue5.3;
run;
```

To perform the augmented Dickey-Fuller test, regress the differences of the series on lagged differences and on the lagged value of the series, and compute the test statistic from the regression coefficient for the lagged series. The following statements illustrate how to perform the single-mean augmented Dickey-Fuller studentized test for a simple unit root using PROC REG and PROBDF:

```
data test1;
   set test;
   yl  = lag(y);
   yd  = dif(y);
   yd1 = lag1(yd); yd2 = lag2(yd);
   yd3 = lag3(yd); yd4 = lag4(yd);
run;

proc reg data=test1 outest=alpha covout;
   model yd = yl yd1-yd4 / noprint;
run;

data _null_;
   set alpha;
   retain a;
   if _type_ = 'PARMS' then a = yl ;
   if _type_ = 'COV' & _NAME_ = 'Y1' then do;
      x = a / sqrt(yl);
      p = probdf( x, 99, 1, "SSM" );
      put p= pvalue5.3;
      end;
run;
```

The %DFTEST macro provides an easier way to perform Dickey-Fuller tests. The following statements perform the same tests as the preceding example:

```
%dftest( test, y, ar=4 );
%put p=&dftest;
```

References

Bhargava, A. (1986). "On the Theory of Testing for Unit Roots in Observed Time Series." *Review of Economic Studies* 53:369–384.

Dickey, D. A. (1976). *Estimation and Testing of Nonstationary Time Series*. Ph.D. diss., Iowa State University.

Dickey, D. A., and Fuller, W. A. (1979). "Distribution of the Estimators for Autoregressive Time Series with a Unit Root." *Journal of the American Statistical Association* 74:427–431.

Dickey, D. A., Hasza, D. P., and Fuller, W. A. (1984). "Testing for Unit Roots in Seasonal Time Series." *Journal of the American Statistical Association* 79:355–367.

Hamilton, J. D. (1994). *Time Series Analysis*. Princeton, NJ: Princeton University Press.

Microsoft Corporation (2000). *Microsoft Excel 2000 Online Help*. Microsoft, Redmond, WA.

Pankratz, A. (1983). *Forecasting with Univariate Box-Jenkins Models: Concepts and Cases*. New York: John Wiley & Sons.

Said, S. E., and Dickey, D. A. (1984). "Testing for Unit Roots in ARMA Models of Unknown Order." *Biometrika* 71:599–607.

Taylor, J. M. G. (1986). "The Retransformed Mean after a Fitted Power Transformation." *Journal of the American Statistical Association* 81:114–118.

Chapter 7
Nonlinear Optimization Methods

Contents

Overview **195**
Options **195**
Details of Optimization Algorithms **205**
Overview 205
Choosing an Optimization Algorithm 206
Algorithm Descriptions 207
Remote Monitoring **210**
ODS Table Names **213**
References **214**

Overview

Several SAS/ETS procedures (COUNTREG, ENTROPY, MDC, QLIM, UCM, and VARMAX) use the nonlinear optimization (NLO) subsystem to perform nonlinear optimization. This chapter describes the options of the NLO system and some technical details of the available optimization methods. Note that not all options have been implemented for all procedures that use the NLO subsystem. You should check each procedure chapter for more details about which options are available.

Options

The following table summarizes the options available in the NLO system.

Table 7.1 NLO options

Option	Description
Optimization Specifications	
TECHNIQUE=	minimization technique
UPDATE=	update technique
LINESEARCH=	line-search method
LSPRECISION=	line-search precision
HESCAL=	type of Hessian scaling
INHESSIAN=	start for approximated Hessian

Table 7.1 *continued*

Option	Description
RESTART=	iteration number for update restart
Termination Criteria Specifications	
MAXFUNC=	maximum number of function calls
MAXITER=	maximum number of iterations
MINITER=	minimum number of iterations
MAXTIME=	upper limit seconds of CPU time
ABSCONV=	absolute function convergence criterion
ABSFCONV=	absolute function convergence criterion
ABSGCONV=	absolute gradient convergence criterion
ABSXCONV=	absolute parameter convergence criterion
FCONV=	relative function convergence criterion
FCONV2=	relative function convergence criterion
GCONV=	relative gradient convergence criterion
XCONV=	relative parameter convergence criterion
FSIZE=	used in FCONV, GCONV criterion
XSIZE=	used in XCONV criterion
Step Length Options	
DAMPSTEP=	damped steps in line search
MAXSTEP=	maximum trust region radius
INSTEP=	initial trust region radius
Printed Output Options	
PALL	display (almost) all printed optimization-related output
PHISTORY	display optimization history
PHISTPARMS	display parameter estimates in each iteration
PSHORT	reduce some default optimization-related output
PSUMMARY	reduce most default optimization-related output
NOPRINT	suppress all printed optimization-related output
Remote Monitoring Options	
SOCKET=	specify the fileref for remote monitoring

These options are described in alphabetical order.

ABSCONV=*r*

ABSTOL=*r*

specifies an absolute function convergence criterion. For minimization, termination requires $f(\theta^{(k)}) \leq r$. The default value of r is the negative square root of the largest double-precision value, which serves only as a protection against overflows.

ABSFCONV=*r*[*n*]

ABSFTOL=*r*[*n*]

specifies an absolute function convergence criterion. For all techniques except NMSIMP, termination requires a small change of the function value in successive iterations:

$$|f(\theta^{(k-1)}) - f(\theta^{(k)})| \leq r$$

The same formula is used for the NMSIMP technique, but $\theta^{(k)}$ is defined as the vertex with the lowest function value, and $\theta^{(k-1)}$ is defined as the vertex with the highest function value in the simplex. The default value is $r = 0$. The optional integer value n specifies the number of successive iterations for which the criterion must be satisfied before the process can be terminated.

ABSGCONV=*r*[*n*]

ABSGTOL=*r*[*n*]

specifies an absolute gradient convergence criterion. Termination requires the maximum absolute gradient element to be small:

$$\max_j |g_j(\theta^{(k)})| \leq r$$

This criterion is not used by the NMSIMP technique. The default value is $r = 1\mathrm{E}-5$. The optional integer value n specifies the number of successive iterations for which the criterion must be satisfied before the process can be terminated.

ABSXCONV=*r*[*n*]

ABSXTOL=*r*[*n*]

specifies an absolute parameter convergence criterion. For all techniques except NMSIMP, termination requires a small Euclidean distance between successive parameter vectors,

$$\| \theta^{(k)} - \theta^{(k-1)} \|_2 \leq r$$

For the NMSIMP technique, termination requires either a small length $\alpha^{(k)}$ of the vertices of a restart simplex,

$$\alpha^{(k)} \leq r$$

or a small simplex size,

$$\delta^{(k)} \leq r$$

where the simplex size $\delta^{(k)}$ is defined as the L1 distance from the simplex vertex $\xi^{(k)}$ with the smallest function value to the other n simplex points $\theta_l^{(k)} \neq \xi^{(k)}$:

$$\delta^{(k)} = \sum_{\theta_l \neq y} \| \theta_l^{(k)} - \xi^{(k)} \|_1$$

The default is $r = 1\mathrm{E}-8$ for the NMSIMP technique and $r = 0$ otherwise. The optional integer value n specifies the number of successive iterations for which the criterion must be satisfied before the process can terminate.

DAMPSTEP*[=r]*

specifies that the initial step length value $\alpha^{(0)}$ for each line search (used by the QUANEW, HYQUAN, CONGRA, or NEWRAP technique) cannot be larger than r times the step length value used in the former iteration. If the DAMPSTEP option is specified but r is not specified, the default is $r = 2$. The DAMPSTEP=*r* option can prevent the line-search algorithm from repeatedly stepping into regions where some objective functions are difficult to compute or where they could lead to floating point overflows during the computation of objective functions and their derivatives. The DAMPSTEP=*r* option can save time-costly function calls during the line searches of objective functions that result in very small steps.

FCONV=*r*[*n*]

FTOL=*r*[*n*]

specifies a relative function convergence criterion. For all techniques except NMSIMP, termination requires a small relative change of the function value in successive iterations,

$$\frac{|f(\theta^{(k)}) - f(\theta^{(k-1)})|}{\max(|f(\theta^{(k-1)})|, \text{FSIZE})} \leq r$$

where FSIZE is defined by the FSIZE= option. The same formula is used for the NMSIMP technique, but $\theta^{(k)}$ is defined as the vertex with the lowest function value, and $\theta^{(k-1)}$ is defined as the vertex with the highest function value in the simplex. The default value may depend on the procedure. In most cases, you can use the PALL option to find it.

FCONV2=*r*[*n*]

FTOL2=*r*[*n*]

specifies another function convergence criterion.

For all techniques except NMSIMP, termination requires a small predicted reduction

$$df^{(k)} \approx f(\theta^{(k)}) - f(\theta^{(k)} + s^{(k)})$$

of the objective function. The predicted reduction

$$\begin{aligned} df^{(k)} &= -g^{(k)T} s^{(k)} - \frac{1}{2} s^{(k)T} H^{(k)} s^{(k)} \\ &= -\frac{1}{2} s^{(k)T} g^{(k)} \\ &\leq r \end{aligned}$$

is computed by approximating the objective function f by the first two terms of the Taylor series and substituting the Newton step

$$s^{(k)} = -[H^{(k)}]^{-1} g^{(k)}$$

For the NMSIMP technique, termination requires a small standard deviation of the function values of the $n+1$ simplex vertices $\theta_l^{(k)}$, $l = 0, \ldots, n$, $\sqrt{\frac{1}{n+1} \sum_l \left[f(\theta_l^{(k)}) - \overline{f}(\theta^{(k)}) \right]^2} \leq r$ where $\overline{f}(\theta^{(k)}) = \frac{1}{n+1} \sum_l f(\theta_l^{(k)})$. If there are n_{act} boundary constraints active at $\theta^{(k)}$, the mean and standard deviation are computed only for the $n + 1 - n_{act}$ unconstrained vertices.

The default value is $r = 1\text{E}-6$ for the NMSIMP technique and $r = 0$ otherwise. The optional integer value *n* specifies the number of successive iterations for which the criterion must be satisfied before the process can terminate.

FSIZE=*r*

specifies the FSIZE parameter of the relative function and relative gradient termination criteria. The default value is $r = 0$. For more details, see the FCONV= and GCONV= options.

GCONV=*r*[*n*]

GTOL=*r*[*n*]

specifies a relative gradient convergence criterion. For all techniques except CONGRA and NMSIMP, termination requires that the normalized predicted function reduction is small,

$$frac g(\theta^{(k)})^T [H^{(k)}]^{-1} g(\theta^{(k)}) \max(|f(\theta^{(k)})|, \text{FSIZE}) \leq r$$

where FSIZE is defined by the FSIZE= option. For the CONGRA technique (where a reliable Hessian estimate H is not available), the following criterion is used:

$$\frac{\| g(\theta^{(k)}) \|_2^2 \quad \| s(\theta^{(k)}) \|_2}{\| g(\theta^{(k)}) - g(\theta^{(k-1)}) \|_2 \max(|f(\theta^{(k)})|, \text{FSIZE})} \leq r$$

This criterion is not used by the NMSIMP technique. The default value is $r = 1E-8$. The optional integer value n specifies the number of successive iterations for which the criterion must be satisfied before the process can terminate.

HESCAL=0|1|2|3

HS=0|1|2|3

specifies the scaling version of the Hessian matrix used in NRRIDG, TRUREG, NEWRAP, or DBLDOG optimization.

If HS is not equal to 0, the first iteration and each restart iteration sets the diagonal scaling matrix $D^{(0)} = \text{diag}(d_i^{(0)})$:

$$d_i^{(0)} = \sqrt{\max(|H_{i,i}^{(0)}|, \epsilon)}$$

where $H_{i,i}^{(0)}$ are the diagonal elements of the Hessian. In every other iteration, the diagonal scaling matrix $D^{(0)} = \text{diag}(d_i^{(0)})$ is updated depending on the HS option:

HS=0 specifies that no scaling is done.

HS=1 specifies the Moré (1978) scaling update:

$$d_i^{(k+1)} = \max\left[d_i^{(k)}, \sqrt{\max(|H_{i,i}^{(k)}|, \epsilon)}\right]$$

HS=2 specifies the Dennis, Gay, and Welsch (1981) scaling update:

$$d_i^{(k+1)} = \max\left[0.6 * d_i^{(k)}, \sqrt{\max(|H_{i,i}^{(k)}|, \epsilon)}\right]$$

HS=3 specifies that d_i is reset in each iteration:

$$d_i^{(k+1)} = \sqrt{\max(|H_{i,i}^{(k)}|, \epsilon)}$$

In each scaling update, ϵ is the relative machine precision. The default value is HS=0. Scaling of the Hessian can be time consuming in the case where general linear constraints are active.

INHESSIAN[= *r*]

INHESS[= *r*]

specifies how the initial estimate of the approximate Hessian is defined for the quasi-Newton techniques QUANEW and DBLDOG. There are two alternatives:

- If you do not use the *r* specification, the initial estimate of the approximate Hessian is set to the Hessian at $\theta^{(0)}$.
- If you do use the *r* specification, the initial estimate of the approximate Hessian is set to the multiple of the identity matrix rI.

By default, if you do not specify the option INHESSIAN=*r*, the initial estimate of the approximate Hessian is set to the multiple of the identity matrix rI, where the scalar r is computed from the magnitude of the initial gradient.

INSTEP=*r*

reduces the length of the first trial step during the line search of the first iterations. For highly nonlinear objective functions, such as the EXP function, the default initial radius of the trust-region algorithm TRUREG or DBLDOG or the default step length of the line-search algorithms can result in arithmetic overflows. If this occurs, you should specify decreasing values of $0 < r < 1$ such as INSTEP=1E − 1, INSTEP=1E − 2, INSTEP=1E − 4, and so on, until the iteration starts successfully.

- For trust-region algorithms (TRUREG, DBLDOG), the INSTEP= option specifies a factor $r > 0$ for the initial radius $\Delta^{(0)}$ of the trust region. The default initial trust-region radius is the length of the scaled gradient. This step corresponds to the default radius factor of $r = 1$.
- For line-search algorithms (NEWRAP, CONGRA, QUANEW), the INSTEP= option specifies an upper bound for the initial step length for the line search during the first five iterations. The default initial step length is $r = 1$.
- For the Nelder-Mead simplex algorithm, using TECH=NMSIMP, the INSTEP=*r* option defines the size of the start simplex.

LINESEARCH=*i*

LIS=*i*

specifies the line-search method for the CONGRA, QUANEW, and NEWRAP optimization techniques. See Fletcher (1987) for an introduction to line-search techniques. The value of i can be $1, \ldots, 8$. For CONGRA, QUANEW and NEWRAP, the default value is $i = 2$.

LIS=1 specifies a line-search method that needs the same number of function and gradient calls for cubic interpolation and cubic extrapolation; this method is similar to one used by the Harwell subroutine library.

LIS=2 specifies a line-search method that needs more function than gradient calls for quadratic and cubic interpolation and cubic extrapolation; this method is implemented as shown in Fletcher (1987) and can be modified to an exact line search by using the LSPRECISION= option.

LIS=3 specifies a line-search method that needs the same number of function and gradient calls for cubic interpolation and cubic extrapolation; this method is implemented as shown in Fletcher (1987) and can be modified to an exact line search by using the LSPRECISION= option.

LIS=4 specifies a line-search method that needs the same number of function and gradient calls for stepwise extrapolation and cubic interpolation.

LIS=5 specifies a line-search method that is a modified version of LIS=4.

LIS=6 specifies golden section line search (Polak 1971), which uses only function values for linear approximation.

LIS=7 specifies bisection line search (Polak 1971), which uses only function values for linear approximation.

LIS=8 specifies the Armijo line-search technique (Polak 1971), which uses only function values for linear approximation.

LSPRECISION=*r*

LSP=*r*

specifies the degree of accuracy that should be obtained by the line-search algorithms LIS=2 and LIS=3. Usually an imprecise line search is inexpensive and successful. For more difficult optimization problems, a more precise and expensive line search may be necessary (Fletcher 1987). The second line-search method (which is the default for the NEWRAP, QUANEW, and CONGRA techniques) and the third line-search method approach exact line search for small LSPRECISION= values. If you have numerical problems, you should try to decrease the LSPRECISION= value to obtain a more precise line search. The default values are shown in the following table.

Table 7.2 Line Search Precision Defaults

TECH=	UPDATE=	LSP default
QUANEW	DBFGS, BFGS	$r = 0.4$
QUANEW	DDFP, DFP	$r = 0.06$
CONGRA	all	$r = 0.1$
NEWRAP	no update	$r = 0.9$

For more details, see Fletcher (1987).

MAXFUNC=*i*

MAXFU=*i*

specifies the maximum number *i* of function calls in the optimization process. The default values are

- TRUREG, NRRIDG, NEWRAP: 125
- QUANEW, DBLDOG: 500
- CONGRA: 1000
- NMSIMP: 3000

Note that the optimization can terminate only after completing a full iteration. Therefore, the number of function calls that is actually performed can exceed the number that is specified by the MAXFUNC= option.

MAXITER=*i*

MAXIT=*i*

specifies the maximum number *i* of iterations in the optimization process. The default values are

- TRUREG, NRRIDG, NEWRAP: 50
- QUANEW, DBLDOG: 200
- CONGRA: 400
- NMSIMP: 1000

These default values are also valid when *i* is specified as a missing value.

MAXSTEP=*r*[*n*]

specifies an upper bound for the step length of the line-search algorithms during the first *n* iterations. By default, *r* is the largest double-precision value and *n* is the largest integer available. Setting this option can improve the speed of convergence for the CONGRA, QUANEW, and NEWRAP techniques.

MAXTIME=*r*

specifies an upper limit of *r* seconds of CPU time for the optimization process. The default value is the largest floating-point double representation of your computer. Note that the time specified by the MAXTIME= option is checked only once at the end of each iteration. Therefore, the actual running time can be much longer than that specified by the MAXTIME= option. The actual running time includes the rest of the time needed to finish the iteration and the time needed to generate the output of the results.

MINITER=*i*

MINIT=*i*

specifies the minimum number of iterations. The default value is 0. If you request more iterations than are actually needed for convergence to a stationary point, the optimization algorithms can behave strangely. For example, the effect of rounding errors can prevent the algorithm from continuing for the required number of iterations.

NOPRINT

suppresses the output. (See procedure documentation for availability of this option.)

PALL

displays all optional output for optimization. (See procedure documentation for availability of this option.)

PHISTORY

displays the optimization history. (See procedure documentation for availability of this option.)

PHISTPARMS

display parameter estimates in each iteration. (See procedure documentation for availability of this option.)

PINIT

displays the initial values and derivatives (if available). (See procedure documentation for availability of this option.)

PSHORT
restricts the amount of default output. (See procedure documentation for availability of this option.)

PSUMMARY
restricts the amount of default displayed output to a short form of iteration history and notes, warnings, and errors. (See procedure documentation for availability of this option.)

RESTART=*i* > 0

REST=*i* > 0
specifies that the QUANEW or CONGRA algorithm is restarted with a steepest descent/ascent search direction after, at most, *i* iterations. Default values are as follows:

- CONGRA
  UPDATE=PB: restart is performed automatically, *i* is not used.
- CONGRA
  UPDATE≠PB: $i = \min(10n, 80)$, where n is the number of parameters.
- QUANEW
  i is the largest integer available.

SOCKET=*fileref*
Specifies the fileref that contains the information needed for remote monitoring. See the section "Remote Monitoring" on page 210 for more details.

TECHNIQUE=*value*

TECH=*value*
specifies the optimization technique. Valid values are as follows:

- CONGRA
  performs a conjugate-gradient optimization, which can be more precisely specified with the UPDATE= option and modified with the LINESEARCH= option. When you specify this option, UPDATE=PB by default.
- DBLDOG
  performs a version of double-dogleg optimization, which can be more precisely specified with the UPDATE= option. When you specify this option, UPDATE=DBFGS by default.
- NMSIMP
  performs a Nelder-Mead simplex optimization.
- NONE
  does not perform any optimization. This option can be used as follows:
  - to perform a grid search without optimization
  - to compute estimates and predictions that cannot be obtained efficiently with any of the optimization techniques
- NEWRAP
  performs a Newton-Raphson optimization that combines a line-search algorithm with ridging. The line-search algorithm LIS=2 is the default method.
- NRRIDG
  performs a Newton-Raphson optimization with ridging.

- QUANEW
  performs a quasi-Newton optimization, which can be defined more precisely with the UPDATE= option and modified with the LINESEARCH= option. This is the default estimation method.
- TRUREG
  performs a trust region optimization.

UPDATE=*method*
UPD=*method*
specifies the update method for the QUANEW, DBLDOG, or CONGRA optimization technique. Not every update method can be used with each optimizer.

Valid methods are as follows:

- BFGS
  performs the original Broyden, Fletcher, Goldfarb, and Shanno (BFGS) update of the inverse Hessian matrix.
- DBFGS
  performs the dual BFGS update of the Cholesky factor of the Hessian matrix. This is the default update method.
- DDFP
  performs the dual Davidon, Fletcher, and Powell (DFP) update of the Cholesky factor of the Hessian matrix.
- DFP
  performs the original DFP update of the inverse Hessian matrix.
- PB
  performs the automatic restart update method of Powell (1977); Beale (1972).
- FR
  performs the Fletcher-Reeves update (Fletcher 1987).
- PR
  performs the Polak-Ribiere update (Fletcher 1987).
- CD
  performs a conjugate-descent update of Fletcher (1987).

XCONV=*r*[*n*]
XTOL=*r*[*n*]
specifies the relative parameter convergence criterion. For all techniques except NMSIMP, termination requires a small relative parameter change in subsequent iterations.

$$\frac{\max_j |\theta_j^{(k)} - \theta_j^{(k-1)}|}{\max(|\theta_j^{(k)}|, |\theta_j^{(k-1)}|, \text{XSIZE})} \leq r$$

For the NMSIMP technique, the same formula is used, but $\theta_j^{(k)}$ is defined as the vertex with the lowest function value and $\theta_j^{(k-1)}$ is defined as the vertex with the highest function value in the simplex. The default value is $r = 1\text{E} - 8$ for the NMSIMP technique and $r = 0$ otherwise. The optional integer value n specifies the number of successive iterations for which the criterion must be satisfied before the process can be terminated.

XSIZE=$r > 0$

specifies the XSIZE parameter of the relative parameter termination criterion. The default value is $r = 0$. For more detail, see the XCONV= option.

Details of Optimization Algorithms

Overview

There are several optimization techniques available. You can choose a particular optimizer with the TECH=$name$ option in the PROC statement or NLOPTIONS statement.

Table 7.3 Optimization Techniques

Algorithm	TECH=
trust region Method	TRUREG
Newton-Raphson method with line search	NEWRAP
Newton-Raphson method with ridging	NRRIDG
quasi-Newton methods (DBFGS, DDFP, BFGS, DFP)	QUANEW
double-dogleg method (DBFGS, DDFP)	DBLDOG
conjugate gradient methods (PB, FR, PR, CD)	CONGRA
Nelder-Mead simplex method	NMSIMP

No algorithm for optimizing general nonlinear functions exists that always finds the global optimum for a general nonlinear minimization problem in a reasonable amount of time. Since no single optimization technique is invariably superior to others, NLO provides a variety of optimization techniques that work well in various circumstances. However, you can devise problems for which none of the techniques in NLO can find the correct solution. Moreover, nonlinear optimization can be computationally expensive in terms of time and memory, so you must be careful when matching an algorithm to a problem.

All optimization techniques in NLO use $O(n^2)$ memory except the conjugate gradient methods, which use only $O(n)$ of memory and are designed to optimize problems with many parameters. These iterative techniques require repeated computation of the following:

- the function value (optimization criterion)
- the gradient vector (first-order partial derivatives)
- for some techniques, the (approximate) Hessian matrix (second-order partial derivatives)

However, since each of the optimizers requires different derivatives, some computational efficiencies can be gained. Table 7.4 shows, for each optimization technique, which derivatives are required. (*FOD* means that first-order derivatives or the gradient is computed; *SOD* means that second-order derivatives or the Hessian is computed.)

Table 7.4 Optimization Computations

Algorithm	FOD	SOD
TRUREG	x	x
NEWRAP	x	x
NRRIDG	x	x
QUANEW	x	-
DBLDOG	x	-
CONGRA	x	-
NMSIMP	-	-

Each optimization method employs one or more convergence criteria that determine when it has converged. The various termination criteria are listed and described in the previous section. An algorithm is considered to have converged when any one of the convergence criterion is satisfied. For example, under the default settings, the QUANEW algorithm will converge if $ABSGCONV < 1\text{E}-5$, $FCONV < 10^{-FDIGITS}$, or $GCONV < 1\text{E}-8$.

Choosing an Optimization Algorithm

The factors that go into choosing a particular optimization technique for a particular problem are complex and might involve trial and error.

For many optimization problems, computing the gradient takes more computer time than computing the function value, and computing the Hessian sometimes takes *much* more computer time and memory than computing the gradient, especially when there are many decision variables. Unfortunately, optimization techniques that do not use some kind of Hessian approximation usually require many more iterations than techniques that do use a Hessian matrix, and as a result the total run time of these techniques is often longer. Techniques that do not use the Hessian also tend to be less reliable. For example, they can more easily terminate at stationary points rather than at global optima.

A few general remarks about the various optimization techniques follow.

- The second-derivative methods TRUREG, NEWRAP, and NRRIDG are best for small problems where the Hessian matrix is not expensive to compute. Sometimes the NRRIDG algorithm can be faster than the TRUREG algorithm, but TRUREG can be more stable. The NRRIDG algorithm requires only one matrix with $n(n+1)/2$ double words; TRUREG and NEWRAP require two such matrices.
- The first-derivative methods QUANEW and DBLDOG are best for medium-sized problems where the objective function and the gradient are much faster to evaluate than the Hessian. The QUANEW and DBLDOG algorithms, in general, require more iterations than TRUREG, NRRIDG, and NEWRAP, but each iteration can be much faster. The QUANEW and DBLDOG algorithms require only the gradient to update an approximate Hessian, and they require slightly less memory than TRUREG or NEWRAP (essentially one matrix with $n(n+1)/2$ double words). QUANEW is the default optimization method.
- The first-derivative method CONGRA is best for large problems where the objective function and the gradient can be computed much faster than the Hessian and where too much memory is required to store the (approximate) Hessian. The CONGRA algorithm, in general, requires more iterations than

QUANEW or DBLDOG, but each iteration can be much faster. Since CONGRA requires only a factor of n double-word memory, many large applications can be solved only by CONGRA.

- The no-derivative method NMSIMP is best for small problems where derivatives are not continuous or are very difficult to compute.

Algorithm Descriptions

Some details about the optimization techniques are as follows.

Trust Region Optimization (TRUREG)

The trust region method uses the gradient $g(\theta_{(k)})$ and the Hessian matrix $H(\theta_{(k)})$; thus, it requires that the objective function $f(\theta)$ have continuous first- and second-order derivatives inside the feasible region.

The trust region method iteratively optimizes a quadratic approximation to the nonlinear objective function within a hyperelliptic trust region with radius Δ that constrains the step size that corresponds to the quality of the quadratic approximation. The trust region method is implemented using Dennis, Gay, and Welsch (1981); Gay (1983); Moré and Sorensen (1983).

The trust region method performs well for small- to medium-sized problems, and it does not need many function, gradient, and Hessian calls. However, if the computation of the Hessian matrix is computationally expensive, one of the (dual) quasi-Newton or conjugate gradient algorithms may be more efficient.

Newton-Raphson Optimization with Line Search (NEWRAP)

The NEWRAP technique uses the gradient $g(\theta_{(k)})$ and the Hessian matrix $H(\theta_{(k)})$; thus, it requires that the objective function have continuous first- and second-order derivatives inside the feasible region. If second-order derivatives are computed efficiently and precisely, the NEWRAP method can perform well for medium-sized to large problems, and it does not need many function, gradient, and Hessian calls.

This algorithm uses a pure Newton step when the Hessian is positive definite and when the Newton step reduces the value of the objective function successfully. Otherwise, a combination of ridging and line search is performed to compute successful steps. If the Hessian is not positive definite, a multiple of the identity matrix is added to the Hessian matrix to make it positive definite.

In each iteration, a line search is performed along the search direction to find an approximate optimum of the objective function. The default line-search method uses quadratic interpolation and cubic extrapolation (LIS=2).

Newton-Raphson Ridge Optimization (NRRIDG)

The NRRIDG technique uses the gradient $g(\theta_{(k)})$ and the Hessian matrix $H(\theta_{(k)})$; thus, it requires that the objective function have continuous first- and second-order derivatives inside the feasible region.

This algorithm uses a pure Newton step when the Hessian is positive definite and when the Newton step reduces the value of the objective function successfully. If at least one of these two conditions is not satisfied, a multiple of the identity matrix is added to the Hessian matrix.

The NRRIDG method performs well for small- to medium-sized problems, and it does not require many function, gradient, and Hessian calls. However, if the computation of the Hessian matrix is computationally expensive, one of the (dual) quasi-Newton or conjugate gradient algorithms might be more efficient.

Since the NRRIDG technique uses an orthogonal decomposition of the approximate Hessian, each iteration of NRRIDG can be slower than that of the NEWRAP technique, which works with Cholesky decomposition. Usually, however, NRRIDG requires fewer iterations than NEWRAP.

Quasi-Newton Optimization (QUANEW)

The (dual) quasi-Newton method uses the gradient $g(\theta_{(k)})$, and it does not need to compute second-order derivatives since they are approximated. It works well for medium to moderately large optimization problems where the objective function and the gradient are much faster to compute than the Hessian; but, in general, it requires more iterations than the TRUREG, NEWRAP, and NRRIDG techniques, which compute second-order derivatives. QUANEW is the default optimization algorithm because it provides an appropriate balance between the speed and stability required for most nonlinear mixed model applications.

The QUANEW technique is one of the following, depending upon the value of the UPDATE= option.

- the original quasi-Newton algorithm, which updates an approximation of the inverse Hessian
- the dual quasi-Newton algorithm, which updates the Cholesky factor of an approximate Hessian (default)

You can specify four update formulas with the UPDATE= option:

- DBFGS performs the dual Broyden, Fletcher, Goldfarb, and Shanno (BFGS) update of the Cholesky factor of the Hessian matrix. This is the default.
- DDFP performs the dual Davidon, Fletcher, and Powell (DFP) update of the Cholesky factor of the Hessian matrix.
- BFGS performs the original BFGS update of the inverse Hessian matrix.
- DFP performs the original DFP update of the inverse Hessian matrix.

In each iteration, a line search is performed along the search direction to find an approximate optimum. The default line-search method uses quadratic interpolation and cubic extrapolation to obtain a step size α satisfying the Goldstein conditions. One of the Goldstein conditions can be violated if the feasible region defines an upper limit of the step size. Violating the left-side Goldstein condition can affect the positive definiteness of the quasi-Newton update. In that case, either the update is skipped or the iterations are restarted with an identity matrix, resulting in the steepest descent or ascent search direction. You can specify line-search algorithms other than the default with the LIS= option.

The QUANEW algorithm performs its own line-search technique. All options and parameters (except the INSTEP= option) that control the line search in the other algorithms do not apply here. In several applications, large steps in the first iterations are troublesome. You can use the INSTEP= option to impose an upper bound for the step size α during the first five iterations. You can also use the INHESSIAN[=r] option to specify a different starting approximation for the Hessian. If you specify only the INHESSIAN option, the Cholesky factor of a (possibly ridged) finite difference approximation of the Hessian is used to initialize the quasi-Newton update process. The values of the LCSINGULAR=, LCEPSILON=, and LCDEACT= options, which control the processing of linear and boundary constraints, are valid only for the quadratic programming subroutine used in each iteration of the QUANEW algorithm.

Double-Dogleg Optimization (DBLDOG)

The double-dogleg optimization method combines the ideas of the quasi-Newton and trust region methods. In each iteration, the double-dogleg algorithm computes the step $s^{(k)}$ as the linear combination of the steepest descent or ascent search direction $s_1^{(k)}$ and a quasi-Newton search direction $s_2^{(k)}$.

$$s^{(k)} = \alpha_1 s_1^{(k)} + \alpha_2 s_2^{(k)}$$

The step is requested to remain within a prespecified trust region radius; see Fletcher (1987, p. 107). Thus, the DBLDOG subroutine uses the dual quasi-Newton update but does not perform a line search. You can specify two update formulas with the UPDATE= option:

- DBFGS performs the dual Broyden, Fletcher, Goldfarb, and Shanno update of the Cholesky factor of the Hessian matrix. This is the default.
- DDFP performs the dual Davidon, Fletcher, and Powell update of the Cholesky factor of the Hessian matrix.

The double-dogleg optimization technique works well for medium to moderately large optimization problems where the objective function and the gradient are much faster to compute than the Hessian. The implementation is based on Dennis and Mei (1979); Gay (1983). However, this implementation is extended to deal with boundary and linear constraints. The DBLDOG technique generally requires more iterations than the TRUREG, NEWRAP, or NRRIDG technique, which requires second-order derivatives; however, each of the DBLDOG iterations is computationally cheap. Furthermore, the DBLDOG technique requires only gradient calls for the update of the Cholesky factor of an approximate Hessian.

Conjugate Gradient Optimization (CONGRA)

Second-order derivatives are not required by the CONGRA algorithm and are not even approximated. The CONGRA algorithm can be expensive in function and gradient calls, but it requires only $O(n)$ memory for unconstrained optimization. In general, many iterations are required to obtain a precise solution, but each of the CONGRA iterations is computationally cheap. You can specify four different update formulas for generating the conjugate directions by using the UPDATE= option:

- PB performs the automatic restart update method of Powell (1977); Beale (1972). This is the default.
- FR performs the Fletcher-Reeves update (Fletcher 1987).
- PR performs the Polak-Ribiere update (Fletcher 1987).
- CD performs a conjugate-descent update of Fletcher (1987).

The default, UPDATE=PB, behaved best in most test examples. You are advised to avoid the option UPDATE=CD, which behaved worst in most test examples.

The CONGRA subroutine should be used for optimization problems with large n. For the unconstrained or boundary constrained case, CONGRA requires only $O(n)$ bytes of working memory, whereas all other optimization methods require order $O(n^2)$ bytes of working memory. During n successive iterations, uninterrupted by restarts or changes in the working set, the conjugate gradient algorithm computes a cycle of n conjugate search directions. In each iteration, a line search is performed along the search direction to find an approximate optimum of the objective function. The default line-search method uses quadratic

interpolation and cubic extrapolation to obtain a step size α satisfying the Goldstein conditions. One of the Goldstein conditions can be violated if the feasible region defines an upper limit for the step size. Other line-search algorithms can be specified with the LIS= option.

Nelder-Mead Simplex Optimization (NMSIMP)

The Nelder-Mead simplex method does not use any derivatives and does not assume that the objective function has continuous derivatives. The objective function itself needs to be continuous. This technique is quite expensive in the number of function calls, and it might be unable to generate precise results for n much greater than 40.

The original Nelder-Mead simplex algorithm is implemented and extended to boundary constraints. This algorithm does not compute the objective for infeasible points, but it changes the shape of the simplex by adapting to the nonlinearities of the objective function, which contributes to an increased speed of convergence. It uses a special termination criteria.

Remote Monitoring

The SAS/EmMonitor is an application for Windows that enables you to monitor and stop from your PC a CPU-intensive application performed by the NLO subsystem that runs on a remote server.

On the server side, a FILENAME statement assigns a fileref to a SOCKET-type device that defines the IP address of the client and the port number for listening. The fileref is then specified in the SOCKET= option in the PROC statement to control the EmMonitor. The following statements show an example of server-side statements for PROC ENTROPY.

```
data one;
   do t = 1 to 10;
      x1 = 5 * ranuni(456);
      x2 = 10 * ranuni( 456);
      x3 = 2 * rannor(1456);
      e1 = rannor(1456);
      e2 = rannor(4560);
      tmp1 = 0.5 * e1 - 0.1 * e2;
      tmp2 = -0.1 * e1 - 0.3 * e2;
      y1 = 7 + 8.5*x1 + 2*x2 + tmp1;
      y2 = -3 + -2*x1 + x2 + 3*x3 + tmp2;
      output;
   end;
run;

filename sock socket 'your.pc.address.com:6943';

proc entropy data=one tech=tr gmenm gconv=2.e-5 socket=sock;
   model y1 = x1 x2 x3;
run;
```

On the client side, the EmMonitor application is started with the following syntax:

EmMonitor *options*

The options are:

`-p port_number` defines the port number

`-t title` defines the title of the EmMonitor window

`-k` keeps the monitor alive when the iteration is completed

The default port number is 6943.

The server does not need to be running when you start the EmMonitor, and you can start or dismiss the server at any time during the iteration process. You only need to remember the port number.

Starting the PC client, or closing it prematurely, does not have any effect on the server side. In other words, the iteration process continues until one of the criteria for termination is met.

Figure 7.1 through Figure 7.4 show screenshots of the application on the client side.

Figure 7.1 Graph Tab Group 0

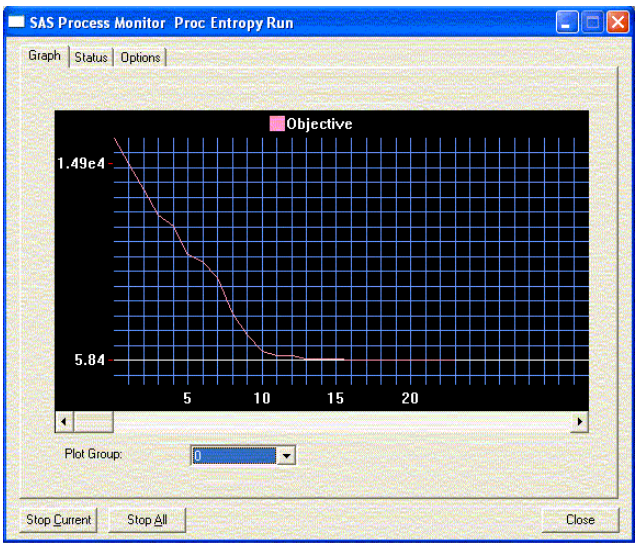


Figure 7.2 Graph Tab Group 1

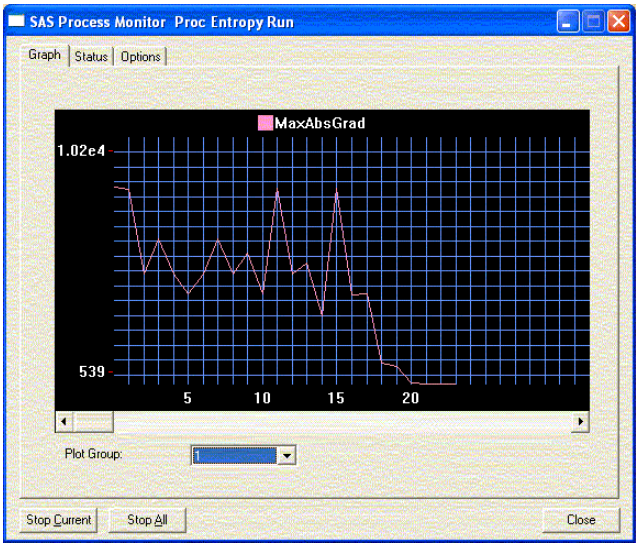


Figure 7.3 Status Tab

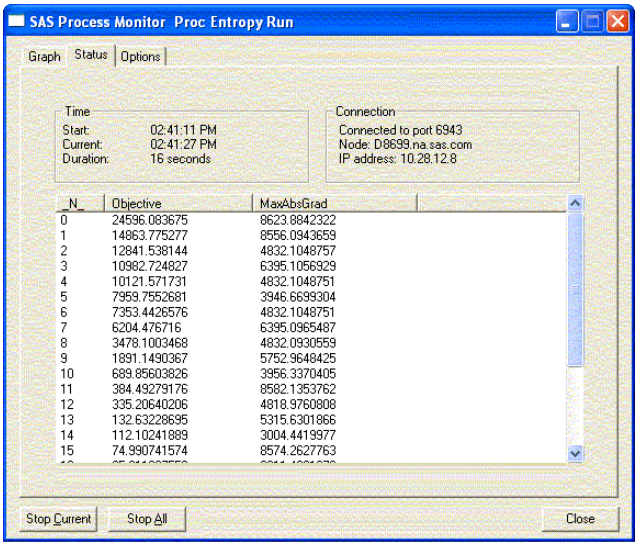
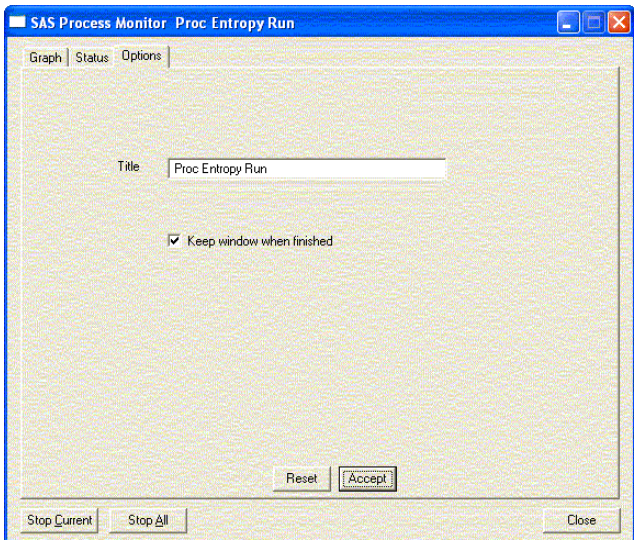


Figure 7.4 Options Tab



ODS Table Names

The NLO subsystem assigns a name to each table it creates. You can use these names when using the Output Delivery System (ODS) to select tables and create output data sets. Not all tables are created by all SAS/ETS procedures that use the NLO subsystem. You should check the procedure chapter for more details. The names are listed in the following table.

Table 7.5 ODS Tables Produced by the NLO Subsystem

ODS Table Name	Description
ConvergenceStatus	Convergence status
InputOptions	Input options
IterHist	Iteration history
IterStart	Iteration start
IterStop	Iteration stop
Lagrange	Lagrange multipliers at the solution
LinCon	Linear constraints
LinConDel	Deleted linear constraints
LinConSol	Linear constraints at the solution
ParameterEstimatesResults	Estimates at the results
ParameterEstimatesStart	Estimates at the start of the iterations
ProblemDescription	Problem description
ProjGrad	Projected gradients

References

Beale, E. M. L. (1972). "A Derivation of Conjugate Gradients." In *Numerical Methods for Nonlinear Optimization*, edited by F. A. Lootsma, 39–43. London: Academic Press.

Dennis, J. E., Gay, D. M., and Welsch, R. E. (1981). "An Adaptive Nonlinear Least-Squares Algorithm." *ACM Transactions on Mathematical Software* 7:348–368.

Dennis, J. E., and Mei, H. H. W. (1979). "Two New Unconstrained Optimization Algorithms Which Use Function and Gradient Values." *Journal of Optimization Theory and Applications* 28:453–482.

Dennis, J. E., and Schnabel, R. B. (1983). *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Englewood Cliffs, NJ: Prentice-Hall.

Fletcher, R. (1987). *Practical Methods of Optimization*. 2nd ed. Chichester, UK: John Wiley & Sons.

Gay, D. M. (1983). "Subroutines for Unconstrained Minimization." *ACM Transactions on Mathematical Software* 9:503–524.

Moré, J. J. (1978). "The Levenberg-Marquardt Algorithm: Implementation and Theory." In *Lecture Notes in Mathematics*, vol. 30, edited by G. A. Watson, 105–116. Berlin: Springer-Verlag.

Moré, J. J., and Sorensen, D. C. (1983). "Computing a Trust-Region Step." *SIAM Journal on Scientific and Statistical Computing* 4:553–572.

Polak, E. (1971). *Computational Methods in Optimization*. New York: Academic Press.

Powell, M. J. D. (1977). "Restart Procedures for the Conjugate Gradient Method." *Mathematical Programming* 12:241–254.

Part II

Procedure Reference

Chapter 8
The ARIMA Procedure

Contents

Overview: ARIMA Procedure **218**
Getting Started: ARIMA Procedure **219**
The Three Stages of ARIMA Modeling 219
Identification Stage 220
Estimation and Diagnostic Checking Stage 225
Forecasting Stage 231
Using ARIMA Procedure Statements 233
General Notation for ARIMA Models 233
Stationarity 236
Differencing 236
Subset, Seasonal, and Factored ARMA Models 238
Input Variables and Regression with ARMA Errors 239
Intervention Models and Interrupted Time Series 242
Rational Transfer Functions and Distributed Lag Models 243
Forecasting with Input Variables 245
Data Requirements 246
Syntax: ARIMA Procedure **247**
Functional Summary 247
PROC ARIMA Statement 250
BY Statement 252
IDENTIFY Statement 253
ESTIMATE Statement 257
OUTLIER Statement 261
FORECAST Statement 262
Details: ARIMA Procedure **263**
The Inverse Autocorrelation Function 263
The Partial Autocorrelation Function 264
The Cross-Correlation Function 264
The ESACF Method 265
The MINIC Method 267
The SCAN Method 268
Stationarity Tests 270
Prewhitening 270
Identifying Transfer Function Models 271
Missing Values and Autocorrelations 271
Estimation Details 272

Specifying Inputs and Transfer Functions 276
Initial Values 277
Stationarity and Invertibility 278
Naming of Model Parameters 279
Missing Values and Estimation and Forecasting 279
Forecasting Details 280
Forecasting Log Transformed Data 281
Specifying Series Periodicity 282
Detecting Outliers 282
OUT= Data Set 284
OUTCOV= Data Set 285
OUTEST= Data Set 286
OUTMODEL= SAS Data Set 289
OUTSTAT= Data Set 290
Printed Output 291
ODS Table Names 294
Statistical Graphics 295
Examples: ARIMA Procedure **299**
Example 8.1: Simulated IMA Model 299
Example 8.2: Seasonal Model for the Airline Series 303
Example 8.3: Model for Series J Data from Box and Jenkins 310
Example 8.4: An Intervention Model for Ozone Data 318
Example 8.5: Using Diagnostics to Identify ARIMA Models 320
Example 8.6: Detection of Level Changes in the Nile River Data 325
Example 8.7: Iterative Outlier Detection 326
References **328**

Overview: ARIMA Procedure

The ARIMA procedure analyzes and forecasts equally spaced univariate time series data, transfer function data, and intervention data by using the autoregressive integrated moving-average (ARIMA) or autoregressive moving-average (ARMA) model. An ARIMA model predicts a value in a response time series as a linear combination of its own past values, past errors (also called shocks or innovations), and current and past values of other time series.

The ARIMA approach was first popularized by Box and Jenkins, and ARIMA models are often referred to as Box-Jenkins models. The general transfer function model employed by the ARIMA procedure was discussed by Box and Tiao (1975). When an ARIMA model includes other time series as input variables, the model is sometimes referred to as an ARIMAX model. Pankratz (1991) refers to the ARIMAX model as *dynamic regression*.

The ARIMA procedure provides a comprehensive set of tools for univariate time series model identification, parameter estimation, and forecasting, and it offers great flexibility in the kinds of ARIMA or ARIMAX

models that can be analyzed. The ARIMA procedure supports seasonal, subset, and factored ARIMA models; intervention or interrupted time series models; multiple regression analysis with ARMA errors; and rational transfer function models of any complexity.

The design of PROC ARIMA closely follows the Box-Jenkins strategy for time series modeling with features for the identification, estimation and diagnostic checking, and forecasting steps of the Box-Jenkins method.

Before you use PROC ARIMA, you should be familiar with Box-Jenkins methods, and you should exercise care and judgment when you use the ARIMA procedure. The ARIMA class of time series models is complex and powerful, and some degree of expertise is needed to use them correctly.

Getting Started: ARIMA Procedure

This section outlines the use of the ARIMA procedure and gives a cursory description of the ARIMA modeling process for readers who are less familiar with these methods.

The Three Stages of ARIMA Modeling

The analysis performed by PROC ARIMA is divided into three stages, corresponding to the stages described by Box and Jenkins (1976).

1. In the *identification* stage, you use the IDENTIFY statement to specify the response series and identify candidate ARIMA models for it. The IDENTIFY statement reads time series that are to be used in later statements, possibly differencing them, and computes autocorrelations, inverse autocorrelations, partial autocorrelations, and cross-correlations. Stationarity tests can be performed to determine if differencing is necessary. The analysis of the IDENTIFY statement output usually suggests one or more ARIMA models that could be fit. Options enable you to test for stationarity and tentative ARMA order identification.

2. In the *estimation and diagnostic checking* stage, you use the ESTIMATE statement to specify the ARIMA model to fit to the variable specified in the previous IDENTIFY statement and to estimate the parameters of that model. The ESTIMATE statement also produces diagnostic statistics to help you judge the adequacy of the model.

   Significance tests for parameter estimates indicate whether some terms in the model might be unnecessary. Goodness-of-fit statistics aid in comparing this model to others. Tests for white noise residuals indicate whether the residual series contains additional information that might be used by a more complex model. The OUTLIER statement provides another useful tool to check whether the currently estimated model accounts for all the variation in the series. If the diagnostic tests indicate problems with the model, you try another model and then repeat the estimation and diagnostic checking stage.

3. In the *forecasting* stage, you use the FORECAST statement to forecast future values of the time series and to generate confidence intervals for these forecasts from the ARIMA model produced by the preceding ESTIMATE statement.

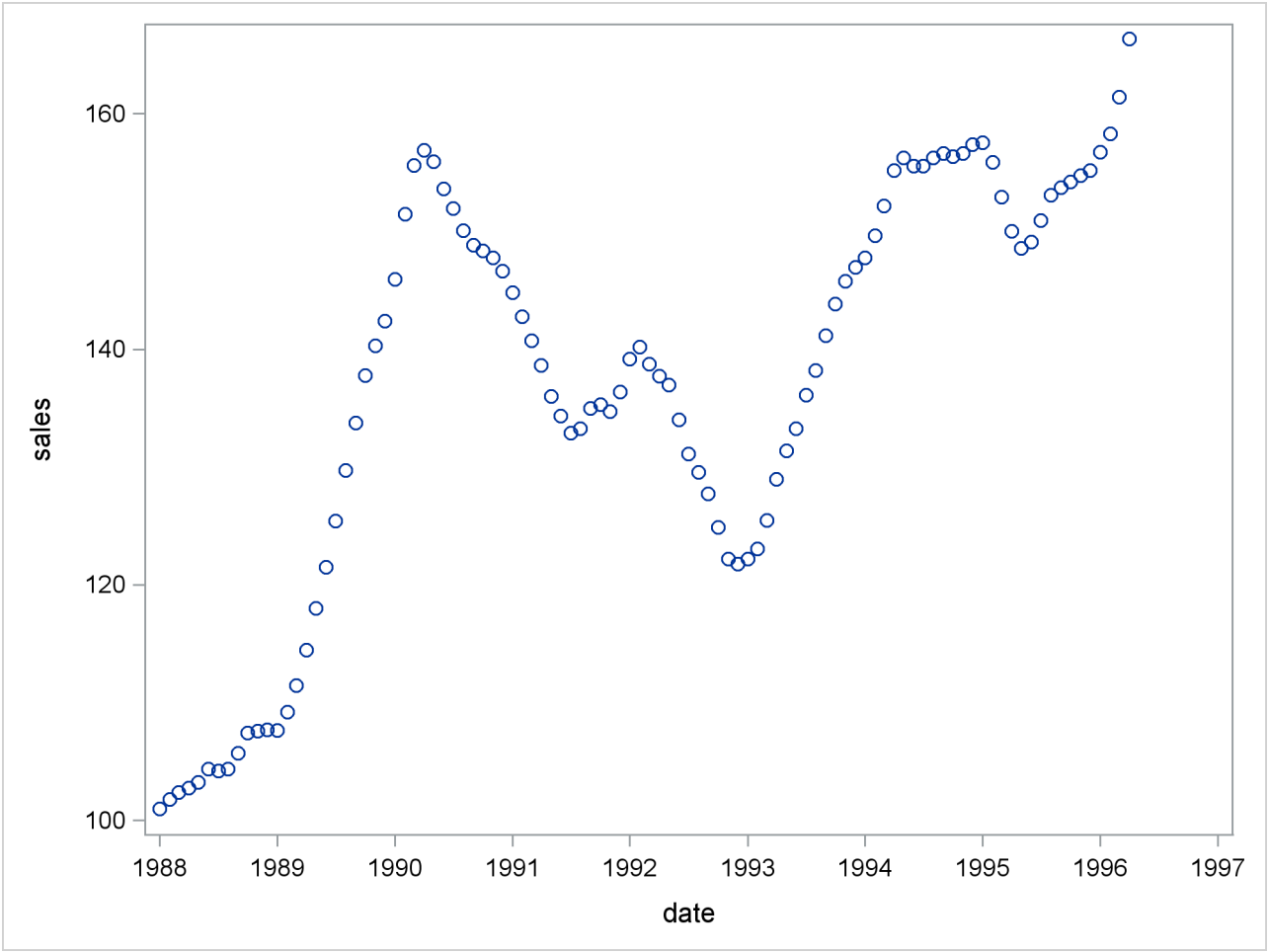
These three steps are explained further and illustrated through an extended example in the following sections.

Identification Stage

Suppose you have a variable called SALES that you want to forecast. The following example illustrates ARIMA modeling and forecasting by using a simulated data set TEST that contains a time series SALES generated by an ARIMA(1,1,1) model. The output produced by this example is explained in the following sections. The simulated SALES series is shown in Figure 8.1.

```
proc sgplot data=test;
   scatter y=sales x=date;
run;
```

Figure 8.1 Simulated ARIMA(1,1,1) Series SALES



Using the IDENTIFY Statement

You first specify the input data set in the PROC ARIMA statement. Then, you use an IDENTIFY statement to read in the SALES series and analyze its correlation properties. You do this by using the following statements:

```
proc arima data=test ;
   identify var=sales nlag=24;
run;
```

Descriptive Statistics

The IDENTIFY statement first prints descriptive statistics for the SALES series. This part of the IDENTIFY statement output is shown in Figure 8.2.

Figure 8.2 IDENTIFY Statement Descriptive Statistics Output

The ARIMA Procedure

Name of Variable = sales	
Mean of Working Series	137.3662
Standard Deviation	17.36385
Number of Observations	100

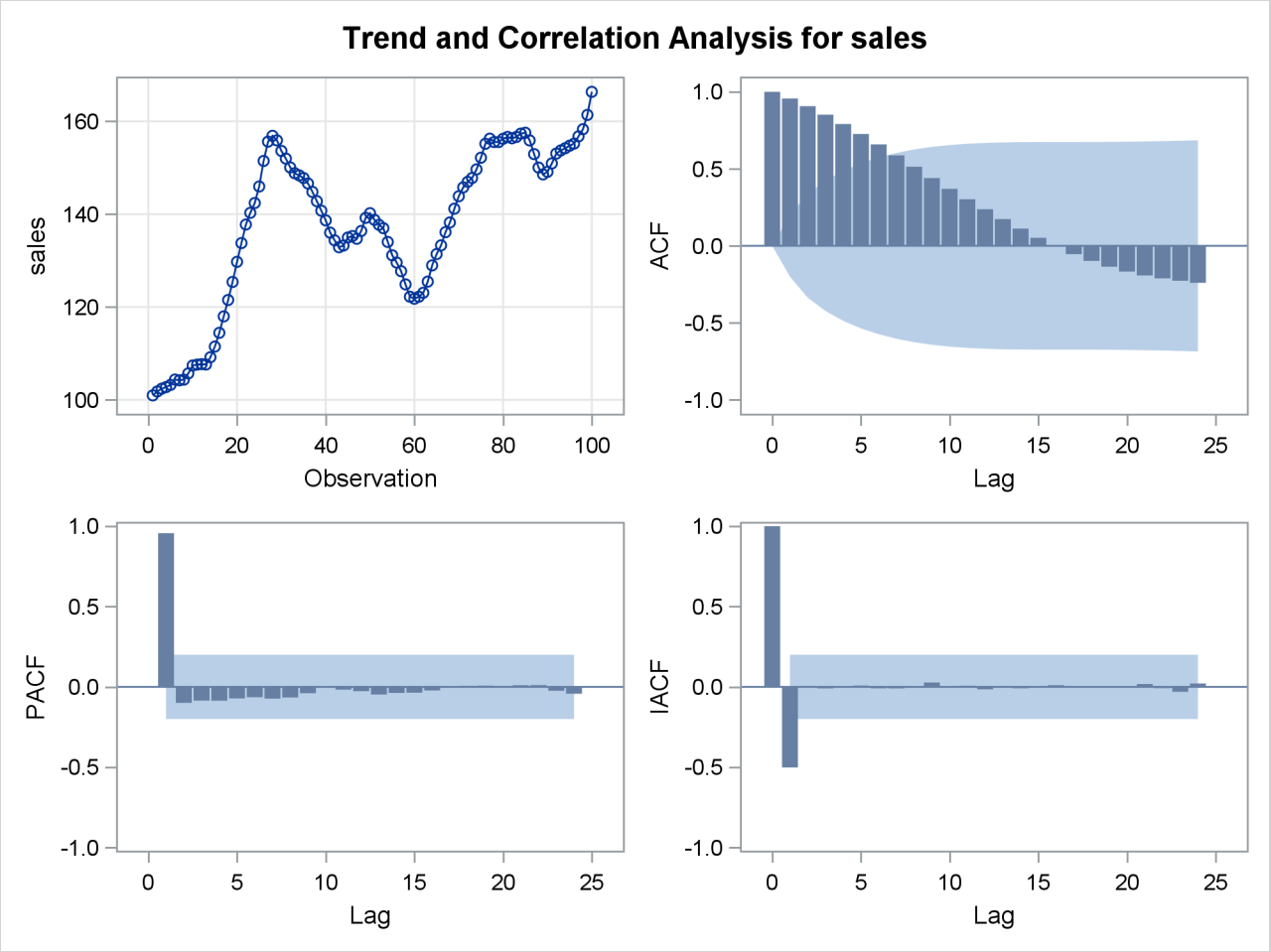
Autocorrelation Function Plots

The IDENTIFY statement next produces a panel of plots used for its autocorrelation and trend analysis. The panel contains the following plots:

- the time series plot of the series
- the sample autocorrelation function plot (ACF)
- the sample inverse autocorrelation function plot (IACF)
- the sample partial autocorrelation function plot (PACF)

This correlation analysis panel is shown in Figure 8.3.

Figure 8.3 Correlation Analysis of SALES



These autocorrelation function plots show the degree of correlation with past values of the series as a function of the number of periods in the past (that is, the lag) at which the correlation is computed.

The NLAG= option controls the number of lags for which the autocorrelations are shown. By default, the autocorrelation functions are plotted to lag 24.

Most books on time series analysis explain how to interpret the autocorrelation and the partial autocorrelation plots. See the section "The Inverse Autocorrelation Function" on page 263 for a discussion of the inverse autocorrelation plots.

By examining these plots, you can judge whether the series is *stationary* or *nonstationary*. In this case, a visual inspection of the autocorrelation function plot indicates that the SALES series is nonstationary, since the ACF decays very slowly. For more formal stationarity tests, use the STATIONARITY= option. (See the section "Stationarity" on page 236.)

White Noise Test

The last part of the default IDENTIFY statement output is the check for white noise. This is an approximate statistical test of the hypothesis that none of the autocorrelations of the series up to a given lag are significantly different from 0. If this is true for all lags, then there is no information in the series to model, and no ARIMA model is needed for the series.

The autocorrelations are checked in groups of six, and the number of lags checked depends on the NLAG= option. The check for white noise output is shown in Figure 8.4.

Figure 8.4 IDENTIFY Statement Check for White Noise

Autocorrelation Check for White Noise

To Lag	Chi-Square	DF	Pr > ChiSq	Autocorrelations					
6	426.44	6	<.0001	0.957	0.907	0.852	0.791	0.726	0.659
12	547.82	12	<.0001	0.588	0.514	0.440	0.370	0.303	0.238
18	554.70	18	<.0001	0.174	0.112	0.052	-0.004	-0.054	-0.098
24	585.73	24	<.0001	-0.135	-0.167	-0.192	-0.211	-0.227	-0.240

In this case, the white noise hypothesis is rejected very strongly, which is expected since the series is nonstationary. The *p*-value for the test of the first six autocorrelations is printed as <0.0001, which means the *p*-value is less than 0.0001.

Identification of the Differenced Series

Since the series is nonstationary, the next step is to transform it to a stationary series by differencing. That is, instead of modeling the SALES series itself, you model the change in SALES from one period to the next. To difference the SALES series, use another IDENTIFY statement and specify that the first difference of SALES be analyzed, as shown in the following statements:

```
proc arima data=test;
   identify var=sales(1);
run;
```

The second IDENTIFY statement produces the same information as the first, but for the change in SALES from one period to the next rather than for the total SALES in each period. The summary statistics output from this IDENTIFY statement is shown in Figure 8.5. Note that the period of differencing is given as 1, and one observation was lost through the differencing operation.

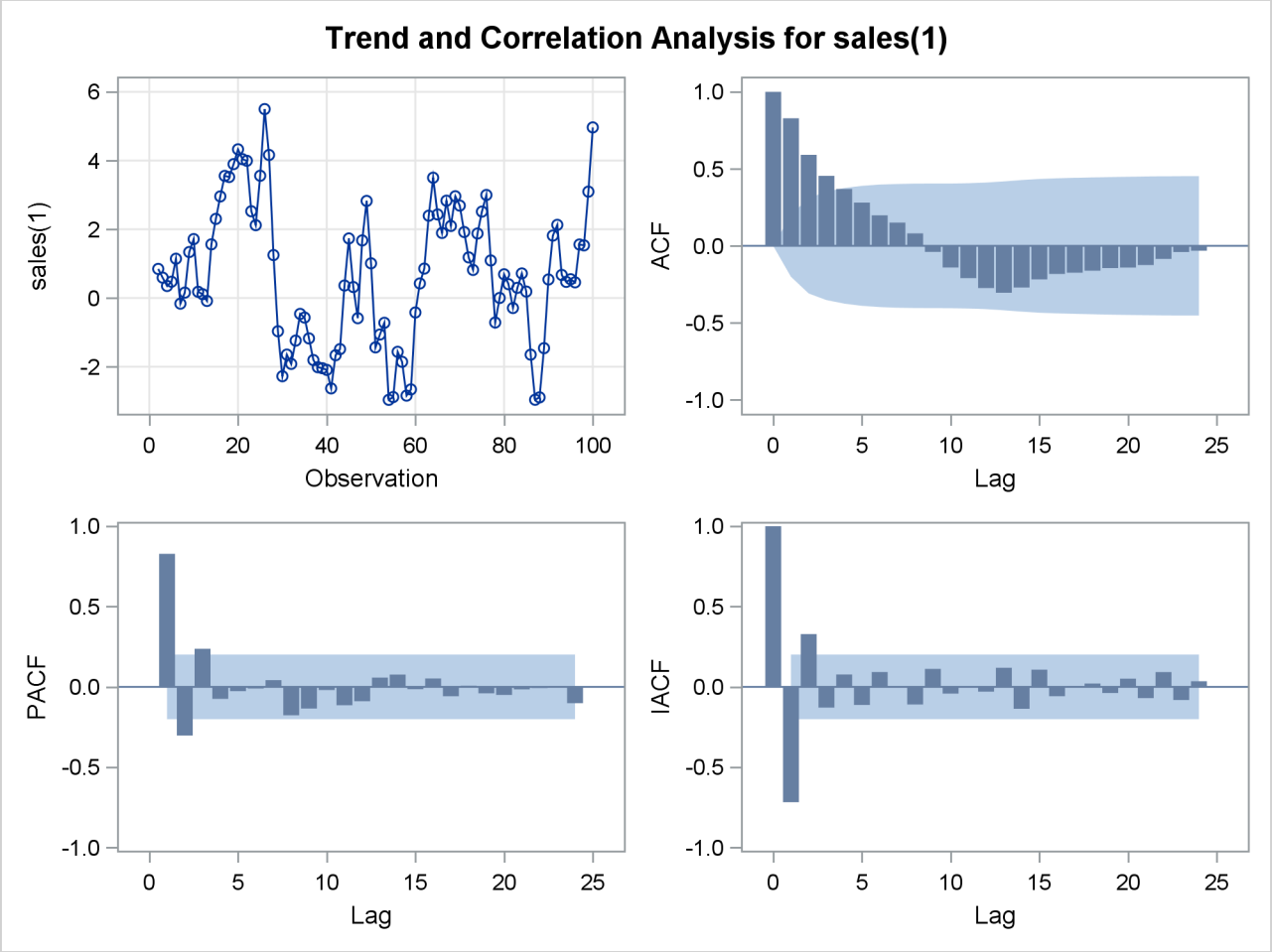
Figure 8.5 IDENTIFY Statement Output for Differenced Series

The ARIMA Procedure

Name of Variable = sales	
Period(s) of Differencing	1
Mean of Working Series	0.660589
Standard Deviation	2.011543
Number of Observations	99
Observation(s) eliminated by differencing	1

The autocorrelation plots for the differenced series are shown in Figure 8.6.

Figure 8.6 Correlation Analysis of the Change in SALES



The autocorrelations decrease rapidly in this plot, indicating that the change in SALES is a stationary time series.

The next step in the Box-Jenkins methodology is to examine the patterns in the autocorrelation plot to choose candidate ARMA models to the series. The partial and inverse autocorrelation function plots are also useful aids in identifying appropriate ARMA models for the series.

In the usual Box-Jenkins approach to ARIMA modeling, the sample autocorrelation function, inverse autocorrelation function, and partial autocorrelation function are compared with the theoretical correlation functions expected from different kinds of ARMA models. This matching of theoretical autocorrelation functions of different ARMA models to the sample autocorrelation functions computed from the response series is the heart of the identification stage of Box-Jenkins modeling. Most textbooks on time series analysis, such as Pankratz (1983), discuss the theoretical autocorrelation functions for different kinds of ARMA models.

Since the input data are only a limited sample of the series, the sample autocorrelation functions computed from the input series only approximate the true autocorrelation function of the process that generates the series. This means that the sample autocorrelation functions do not exactly match the theoretical autocorrelation functions for any ARMA model and can have a pattern similar to that of several different ARMA models. If the series is white noise (a purely random process), then there is no need to fit a model. The check for

white noise, shown in Figure 8.7, indicates that the change in SALES is highly autocorrelated. Thus, an autocorrelation model, for example an AR(1) model, might be a good candidate model to fit to this process.

Figure 8.7 IDENTIFY Statement Check for White Noise

Autocorrelation Check for White Noise

To Lag	Chi-Square	DF	Pr > ChiSq	Autocorrelations					
6	154.44	6	<.0001	0.828	0.591	0.454	0.369	0.281	0.198
12	173.66	12	<.0001	0.151	0.081	-0.039	-0.141	-0.210	-0.274
18	209.64	18	<.0001	-0.305	-0.271	-0.218	-0.183	-0.174	-0.161
24	218.04	24	<.0001	-0.144	-0.141	-0.125	-0.085	-0.040	-0.032

Estimation and Diagnostic Checking Stage

The autocorrelation plots for this series, as shown in the previous section, suggest an AR(1) model for the change in SALES. You should check the diagnostic statistics to see if the AR(1) model is adequate. Other candidate models include an MA(1) model and low-order mixed ARMA models. In this example, the AR(1) model is tried first.

Estimating an AR(1) Model

The following statements fit an AR(1) model (an autoregressive model of order 1), which predicts the change in SALES as an average change, plus some fraction of the previous change, plus a random error. To estimate an AR model, you specify the order of the autoregressive model with the P= option in an ESTIMATE statement:

```
estimate p=1;
run;
```

The ESTIMATE statement fits the model to the data and prints parameter estimates and various diagnostic statistics that indicate how well the model fits the data. The first part of the ESTIMATE statement output, the table of parameter estimates, is shown in Figure 8.8.

Figure 8.8 Parameter Estimates for AR(1) Model

The ARIMA Procedure

Conditional Least Squares Estimation

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|	Lag
MU	0.90280	0.65984	1.37	0.1744	0
AR1,1	0.86847	0.05485	15.83	<.0001	1

The table of parameter estimates is titled "Conditional Least Squares Estimation," which indicates the estimation method used. You can request different estimation methods with the METHOD= option.

The table of parameter estimates lists the parameters in the model; for each parameter, the table shows the estimated value and the standard error and *t* value for the estimate. The table also indicates the lag at which the parameter appears in the model.

In this case, there are two parameters in the model. The mean term is labeled MU; its estimated value is 0.90280. The autoregressive parameter is labeled AR1,1; this is the coefficient of the lagged value of the change in SALES, and its estimate is 0.86847.

The *t* values provide significance tests for the parameter estimates and indicate whether some terms in the model might be unnecessary. In this case, the *t* value for the autoregressive parameter is 15.83, so this term is highly significant. The *t* value for MU indicates that the mean term adds little to the model.

The standard error estimates are based on large sample theory. Thus, the standard errors are labeled as approximate, and the standard errors and *t* values might not be reliable in small samples.

The next part of the ESTIMATE statement output is a table of goodness-of-fit statistics, which aid in comparing this model to other models. This output is shown in Figure 8.9.

Figure 8.9 Goodness-of-Fit Statistics for AR(1) Model

Constant Estimate	0.118749
Variance Estimate	1.15794
Std Error Estimate	1.076076
AIC	297.4469
SBC	302.6372
Number of Residuals	99

The "Constant Estimate" is a function of the mean term MU and the autoregressive parameters. This estimate is computed only for AR or ARMA models, but not for strictly MA models. See the section "General Notation for ARIMA Models" on page 233 for an explanation of the constant estimate.

The "Variance Estimate" is the variance of the residual series, which estimates the innovation variance. The item labeled "Std Error Estimate" is the square root of the variance estimate. In general, when you are comparing candidate models, smaller AIC and SBC statistics indicate the better fitting model. The section "Estimation Details" on page 272 explains the AIC and SBC statistics.

The ESTIMATE statement next prints a table of correlations of the parameter estimates, as shown in Figure 8.10. This table can help you assess the extent to which collinearity might have influenced the results. If two parameter estimates are very highly correlated, you might consider dropping one of them from the model.

Figure 8.10 Correlations of the Estimates for AR(1) Model

Correlations of Parameter Estimates		
Parameter	**MU**	**AR1,1**
MU	1.000	0.114
AR1,1	0.114	1.000

The next part of the ESTIMATE statement output is a check of the autocorrelations of the residuals. This output has the same form as the autocorrelation check for white noise that the IDENTIFY statement prints for the response series. The autocorrelation check of residuals is shown in Figure 8.11.

Figure 8.11 Check for White Noise Residuals for AR(1) Model

Autocorrelation Check of Residuals									
To Lag	Chi-Square	DF	Pr > ChiSq	Autocorrelations					
6	19.09	5	0.0019	0.327	-0.220	-0.128	0.068	-0.002	-0.096
12	22.90	11	0.0183	0.072	0.116	-0.042	-0.066	0.031	-0.091
18	31.63	17	0.0167	-0.233	-0.129	-0.024	0.056	-0.014	-0.008
24	32.83	23	0.0841	0.009	-0.057	-0.057	-0.001	0.049	-0.015

The χ^2 test statistics for the residuals series indicate whether the residuals are uncorrelated (white noise) or contain additional information that might be used by a more complex model. In this case, the test statistics reject the no-autocorrelation hypothesis at a high level of significance ($p = 0.0019$ for the first six lags.) This means that the residuals are not white noise, and so the AR(1) model is not a fully adequate model for this series. The ESTIMATE statement output also includes graphical analysis of the residuals. It is not shown here. The graphical analysis also reveals the inadequacy of the AR(1) model.

The final part of the ESTIMATE statement output is a listing of the estimated model, using the backshift notation. This output is shown in Figure 8.12.

Figure 8.12 Estimated ARIMA(1, 1, 0) Model for SALES

Model for variable sales	
Estimated Mean	0.902799
Period(s) of Differencing	1

Autoregressive Factors	
Factor 1:	1 - 0.86847 B**(1)

This listing combines the differencing specification given in the IDENTIFY statement with the parameter estimates of the model for the change in SALES. Since the AR(1) model is for the change in SALES, the final model for SALES is an ARIMA(1,1,0) model. Using B, the backshift operator, the mathematical form of the estimated model shown in this output is as follows:

$$(1 - B)sales_t = 0.902799 + \frac{1}{(1 - 0.86847B)}a_t$$

See the section "General Notation for ARIMA Models" on page 233 for further explanation of this notation.

Estimating an ARMA(1,1) Model

The IDENTIFY statement plots suggest a mixed autoregressive and moving-average model, and the previous ESTIMATE statement check of residuals indicates that an AR(1) model is not sufficient. You now try estimating an ARMA(1,1) model for the change in SALES.

An ARMA(1,1) model predicts the change in SALES as an average change, plus some fraction of the previous change, plus a random error, plus some fraction of the random error in the preceding period. An ARMA(1,1) model for the change in SALES is the same as an ARIMA(1,1,1) model for the level of SALES.

To estimate a mixed autoregressive moving-average model, you specify the order of the moving-average part of the model with the Q= option in an ESTIMATE statement in addition to specifying the order of the autoregressive part with the P= option. The following statements fit an ARMA(1,1) model to the differenced SALES series:

```
estimate p=1 q=1;
run;
```

The parameter estimates table and goodness-of-fit statistics for this model are shown in Figure 8.13.

Figure 8.13 Estimated ARMA(1, 1) Model for Change in SALES

The ARIMA Procedure

Conditional Least Squares Estimation

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|	Lag
MU	0.89288	0.49391	1.81	0.0738	0
MA1,1	-0.58935	0.08988	-6.56	<.0001	1
AR1,1	0.74755	0.07785	9.60	<.0001	1

Constant Estimate	0.225409
Variance Estimate	0.904034
Std Error Estimate	0.950807
AIC	273.9155
SBC	281.7009
Number of Residuals	99

The moving-average parameter estimate, labeled "MA1,1", is –0.58935. Both the moving-average and the autoregressive parameters have significant *t* values. Note that the variance estimate, AIC, and SBC are all smaller than they were for the AR(1) model, indicating that the ARMA(1,1) model fits the data better without over-parameterizing.

The graphical check of the residuals from this model is shown in Figure 8.14 and Figure 8.15. The residual correlation and white noise test plots show that you cannot reject the hypothesis that the residuals are uncorrelated. The normality plots also show no departure from normality. Thus, you conclude that the ARMA(1,1) model is adequate for the change in SALES series, and there is no point in trying more complex models.

Figure 8.14 White Noise Check of Residuals for the ARMA(1,1) Model

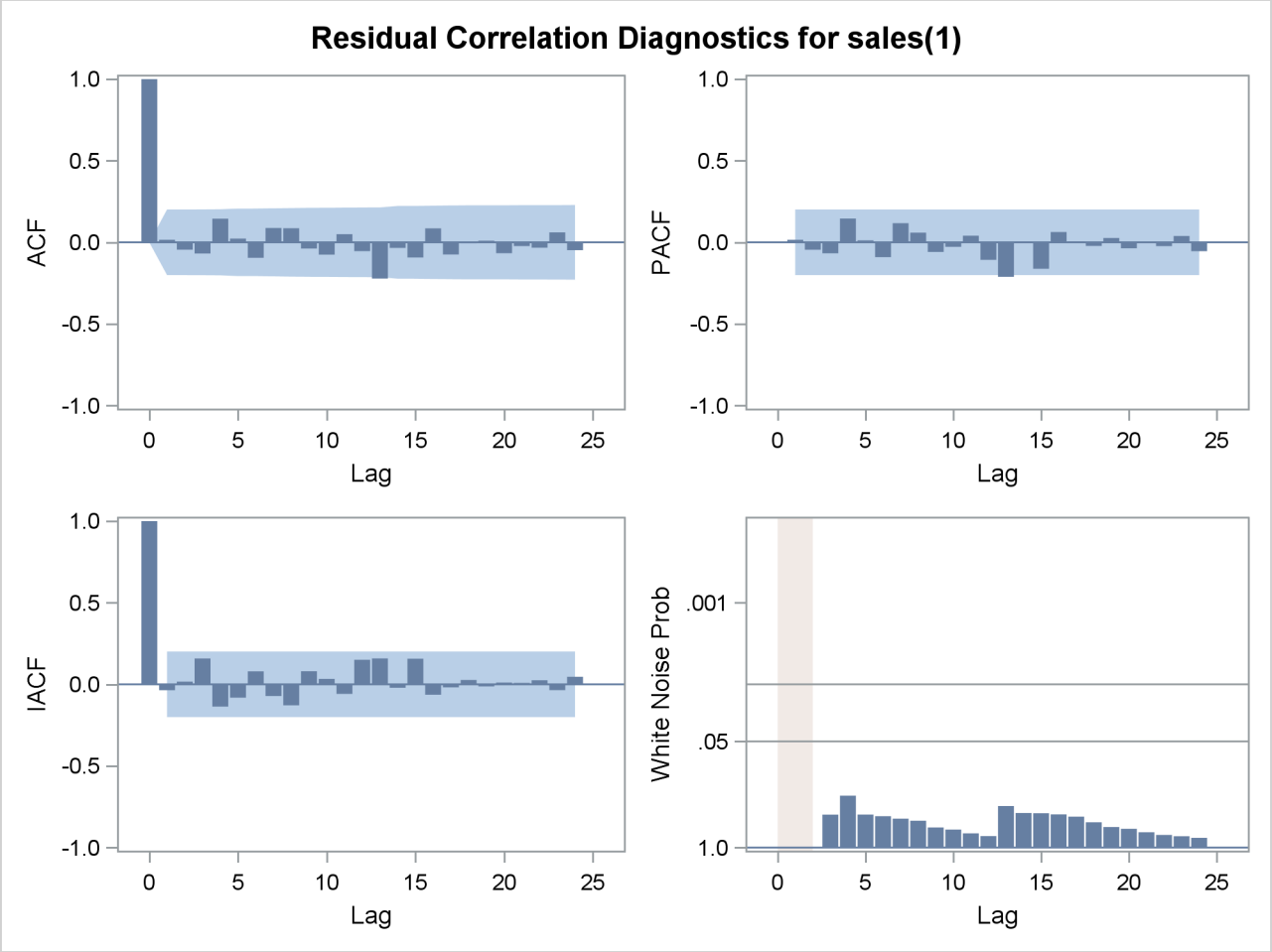
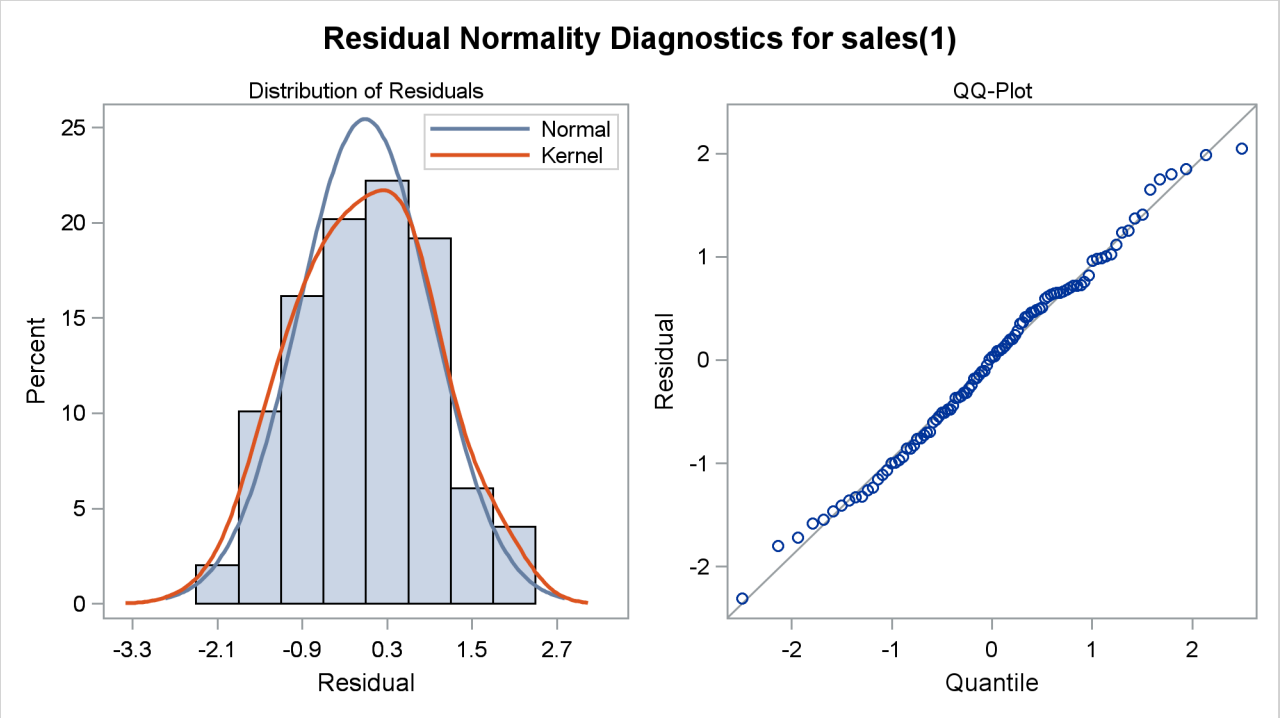


Figure 8.15 Normality Check of Residuals for the ARMA(1,1) Model



The form of the estimated ARIMA(1,1,1) model for SALES is shown in Figure 8.16.

Figure 8.16 Estimated ARIMA(1,1,1) Model for SALES

Model for variable sales	
Estimated Mean	0.892875
Period(s) of Differencing	1

Autoregressive Factors
Factor 1: 1 - 0.74755 B**(1)

Moving Average Factors
Factor 1: 1 + 0.58935 B**(1)

The estimated model shown in this output is

$$(1-B)sales_t = 0.892875 + \frac{(1+0.58935B)}{(1-0.74755B)}a_t$$

In addition to the residual analysis of a model, it is often useful to check whether there are any changes in the time series that are not accounted for by the currently estimated model. The OUTLIER statement enables you to detect such changes. For a long series, this task can be computationally burdensome. Therefore, in general, it is better done after a model that fits the data reasonably well has been found. Figure 8.17 shows the output of the simplest form of the OUTLIER statement:

```
outlier;
run;
```

Two possible outliers have been found for the model in question. See the section "Detecting Outliers" on page 282, and the examples Example 8.6 and Example 8.7, for more details about modeling in the presence of outliers. In this illustration these outliers are not discussed any further.

Figure 8.17 Outliers for the ARIMA(1,1,1) Model for SALES

The ARIMA Procedure

Outlier Detection Summary	
Maximum number searched	2
Number found	2
Significance used	0.05

Outlier Details

Obs	Type	Estimate	Chi-Square	Approx Prob>ChiSq
10	Additive	0.56879	4.20	0.0403
67	Additive	0.55698	4.42	0.0355

Since the model diagnostic tests show that all the parameter estimates are significant and the residual series is white noise, the estimation and diagnostic checking stage is complete. You can now proceed to forecasting the SALES series with this ARIMA(1,1,1) model.

Forecasting Stage

To produce the forecast, use a FORECAST statement after the ESTIMATE statement for the model you decide is best. If the last model fit is not the best, then repeat the ESTIMATE statement for the best model before you use the FORECAST statement.

Suppose that the SALES series is monthly, that you want to forecast one year ahead from the most recently available SALES figure, and that the dates for the observations are given by a variable DATE in the input data set TEST. You use the following FORECAST statement:

```
forecast lead=12 interval=month id=date out=results;
run;
```

The LEAD= option specifies how many periods ahead to forecast (12 months, in this case). The ID= option specifies the ID variable, which is typically a SAS *date*, *time*, or *datetime* variable, used to date the observations of the SALES time series. The INTERVAL= option indicates that data are monthly and enables PROC ARIMA to extrapolate DATE values for forecast periods. The OUT= option writes the forecasts to the output data set RESULTS. See the section "OUT= Data Set" on page 284 for information about the contents of the output data set.

By default, the FORECAST statement also prints and plots the forecast values, as shown in Figure 8.18 and Figure 8.19. The forecast table shows for each forecast period the observation number, forecast value, standard error estimate for the forecast value, and lower and upper limits for a 95% confidence interval for the forecast.

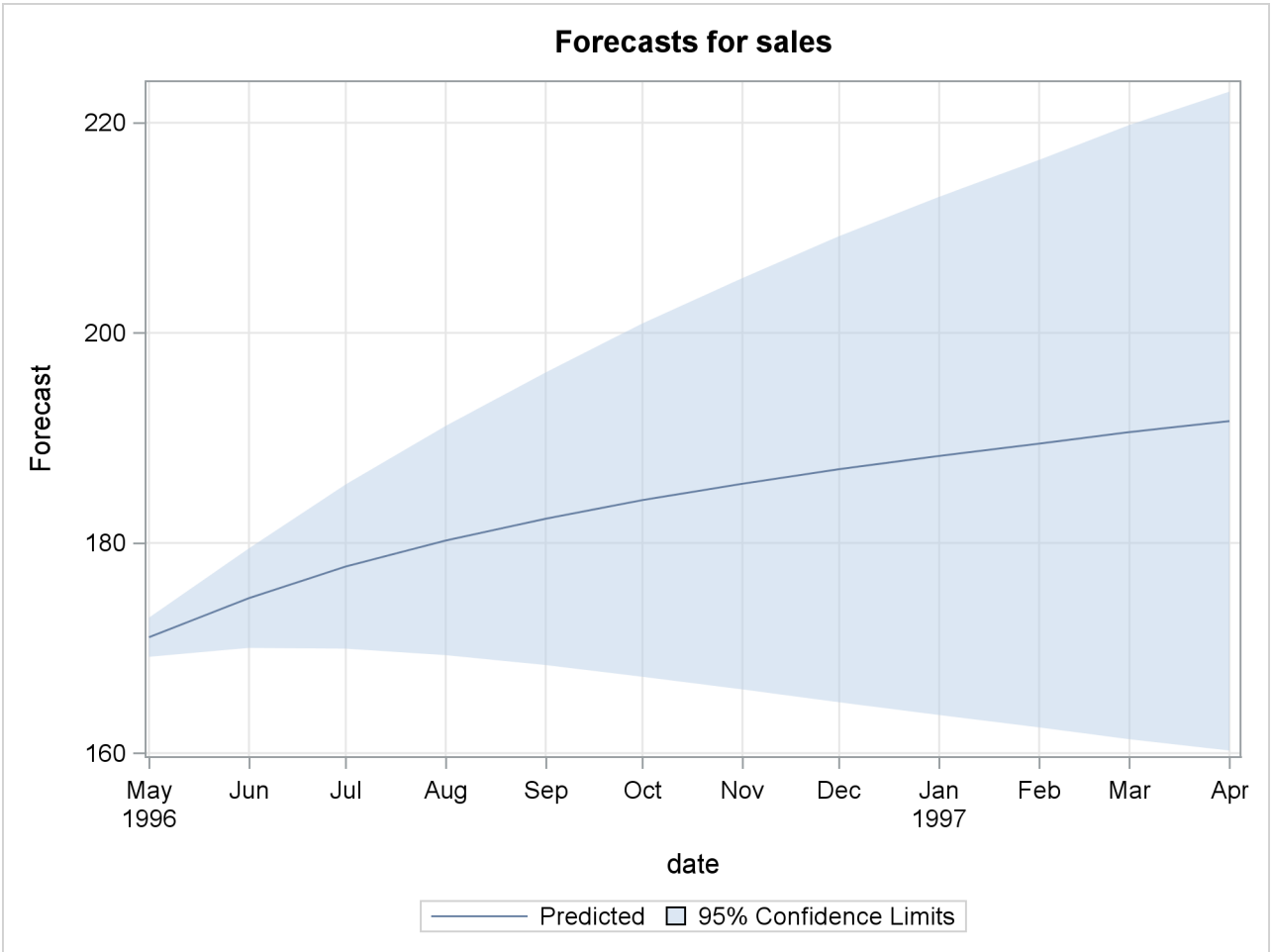
Figure 8.18 Forecasts for ARIMA(1,1,1) Model for SALES

The ARIMA Procedure

Forecasts for variable sales

Obs	Forecast	Std Error	95% Confidence Limits	
101	171.0320	0.9508	169.1684	172.8955
102	174.7534	2.4168	170.0165	179.4903
103	177.7608	3.9879	169.9445	185.5770
104	180.2343	5.5658	169.3256	191.1430
105	182.3088	7.1033	168.3866	196.2310
106	184.0850	8.5789	167.2707	200.8993
107	185.6382	9.9841	166.0698	205.2066
108	187.0247	11.3173	164.8433	209.2061
109	188.2866	12.5807	163.6289	212.9443
110	189.4553	13.7784	162.4501	216.4605
111	190.5544	14.9153	161.3209	219.7879
112	191.6014	15.9964	160.2491	222.9538

Figure 8.19 Forecasts for the ARMA(1,1,1) Model



Normally, you want the forecast values stored in an output data set, and you are not interested in seeing this printed list of the forecast. You can use the NOPRINT option in the FORECAST statement to suppress this output.

Using ARIMA Procedure Statements

The IDENTIFY, ESTIMATE, and FORECAST statements are related in a hierarchy. An IDENTIFY statement brings in a time series to be modeled; several ESTIMATE statements can follow to estimate different ARIMA models for the series; for each model estimated, several FORECAST statements can be used. Thus, a FORECAST statement must be preceded at some point by an ESTIMATE statement, and an ESTIMATE statement must be preceded at some point by an IDENTIFY statement. Additional IDENTIFY statements can be used to switch to modeling a different response series or to change the degree of differencing used.

The ARIMA procedure can be used interactively in the sense that all ARIMA procedure statements can be executed any number of times without reinvoking PROC ARIMA. You can execute ARIMA procedure statements singly or in groups by following the single statement or group of statements with a RUN statement. The output for each statement or group of statements is produced when the RUN statement is entered.

A RUN statement does not terminate the PROC ARIMA step but tells the procedure to execute the statements given so far. You can end PROC ARIMA by submitting a QUIT statement, a DATA step, another PROC step, or an ENDSAS statement.

The example in the preceding section illustrates the interactive use of ARIMA procedure statements. The complete PROC ARIMA program for that example is as follows:

```
proc arima data=test;
   identify var=sales nlag=24;
   run;
   identify var=sales(1);
   run;
   estimate p=1;
   run;
   estimate p=1 q=1;
   run;
   outlier;
   run;
   forecast lead=12 interval=month id=date out=results;
   run;
quit;
```

General Notation for ARIMA Models

The order of an ARIMA (autoregressive integrated moving-average) model is usually denoted by the notation ARIMA(p,d,q), where

p	is the order of the autoregressive part
d	is the order of the differencing

q	is the order of the moving-average process

If no differencing is done ($d = 0$), the models are usually referred to as ARMA(p, q) models. The final model in the preceding example is an ARIMA(1,1,1) model since the IDENTIFY statement specified $d = 1$, and the final ESTIMATE statement specified $p = 1$ and $q = 1$.

Notation for Pure ARIMA Models

Mathematically the pure ARIMA model is written as

$$W_t = \mu + \frac{\theta(B)}{\phi(B)} a_t$$

where

t	indexes time
W_t	is the response series Y_t or a difference of the response series
μ	is the mean term
B	is the backshift operator; that is, $BX_t = X_{t-1}$
$\phi(B)$	is the autoregressive operator, represented as a polynomial in the backshift operator: $\phi(B) = 1 - \phi_1 B - \ldots - \phi_p B^p$
$\theta(B)$	is the moving-average operator, represented as a polynomial in the backshift operator: $\theta(B) = 1 - \theta_1 B - \ldots - \theta_q B^q$
a_t	is the independent disturbance, also called the random error

The series W_t is computed by the IDENTIFY statement and is the series processed by the ESTIMATE statement. Thus, W_t is either the response series Y_t or a difference of Y_t specified by the differencing operators in the IDENTIFY statement.

For simple (nonseasonal) differencing, $W_t = (1 - B)^d Y_t$. For seasonal differencing $W_t = (1 - B)^d (1 - B^s)^D Y_t$, where d is the degree of nonseasonal differencing, D is the degree of seasonal differencing, and s is the length of the seasonal cycle.

For example, the mathematical form of the ARIMA(1,1,1) model estimated in the preceding example is

$$(1 - B)Y_t = \mu + \frac{(1 - \theta_1 B)}{(1 - \phi_1 B)} a_t$$

Model Constant Term

The ARIMA model can also be written as

$$\phi(B)(W_t - \mu) = \theta(B) a_t$$

or

$$\phi(B) W_t = const + \theta(B) a_t$$

where

$$const = \phi(B)\mu = \mu - \phi_1\mu - \phi_2\mu - \ldots - \phi_p\mu$$

Thus, when an autoregressive operator and a mean term are both included in the model, the constant term for the model can be represented as $\phi(B)\mu$. This value is printed with the label "Constant Estimate" in the ESTIMATE statement output.

Notation for Transfer Function Models

The general ARIMA model with input series, also called the ARIMAX model, is written as

$$W_t = \mu + \sum_i \frac{\omega_i(B)}{\delta_i(B)} B^{k_i} X_{i,t} + \frac{\theta(B)}{\phi(B)} a_t$$

where

$X_{i,t}$	is the ith input time series or a difference of the ith input series at time t
k_i	is the pure time delay for the effect of the ith input series
$\omega_i(B)$	is the numerator polynomial of the transfer function for the ith input series
$\delta_i(B)$	is the denominator polynomial of the transfer function for the ith input series.

The model can also be written more compactly as

$$W_t = \mu + \sum_i \Psi_i(B) X_{i,t} + n_t$$

where

$\Psi_i(B)$	is the transfer function for the ith input series modeled as a ratio of the ω and δ polynomials: $\Psi_i(B) = (\omega_i(B)/\delta_i(B))B^{k_i}$
n_t	is the noise series: $n_t = (\theta(B)/\phi(B))a_t$

This model expresses the response series as a combination of past values of the random shocks and past values of other input series. The response series is also called the *dependent series* or *output series*. An input time series is also referred to as an *independent series* or a *predictor series*. Response variable, dependent variable, independent variable, or predictor variable are other terms often used.

Notation for Factored Models

ARIMA models are sometimes expressed in a factored form. This means that the ϕ, θ, ω, or δ polynomials are expressed as products of simpler polynomials. For example, you could express the pure ARIMA model as

$$W_t = \mu + \frac{\theta_1(B)\theta_2(B)}{\phi_1(B)\phi_2(B)} a_t$$

where $\phi_1(B)\phi_2(B) = \phi(B)$ and $\theta_1(B)\theta_2(B) = \theta(B)$.

When an ARIMA model is expressed in factored form, the order of the model is usually expressed by using a factored notation also. The order of an ARIMA model expressed as the product of two factors is denoted as ARIMA(p,d,q)×(P,D,Q).

Notation for Seasonal Models

ARIMA models for time series with regular seasonal fluctuations often use differencing operators and autoregressive and moving-average parameters at lags that are multiples of the length of the seasonal cycle. When all the terms in an ARIMA model factor refer to lags that are a multiple of a constant *s*, the constant is factored out and suffixed to the ARIMA(*p,d,q*) notation.

Thus, the general notation for the order of a seasonal ARIMA model with both seasonal and nonseasonal factors is ARIMA$(p,d,q)\times(P,D,Q)_s$. The term (p,d,q) gives the order of the nonseasonal part of the ARIMA model; the term $(P,D,Q)_s$ gives the order of the seasonal part. The value of *s* is the number of observations in a seasonal cycle: 12 for monthly series, 4 for quarterly series, 7 for daily series with day-of-week effects, and so forth.

For example, the notation ARIMA$(0,1,2)\times(0,1,1)_{12}$ describes a seasonal ARIMA model for monthly data with the following mathematical form:

$$(1-B)(1-B^{12})Y_t = \mu + (1-\theta_{1,1}B-\theta_{1,2}B^2)(1-\theta_{2,1}B^{12})a_t$$

Stationarity

The noise (or residual) series for an ARMA model must be *stationary*, which means that both the expected values of the series and its autocovariance function are independent of time.

The standard way to check for nonstationarity is to plot the series and its autocorrelation function. You can visually examine a graph of the series over time to see if it has a visible trend or if its variability changes noticeably over time. If the series is nonstationary, its autocorrelation function will usually decay slowly.

Another way of checking for stationarity is to use the stationarity tests described in the section "Stationarity Tests" on page 270.

Most time series are nonstationary and must be transformed to a stationary series before the ARIMA modeling process can proceed. If the series has a nonstationary variance, taking the log of the series can help. You can compute the log values in a DATA step and then analyze the log values with PROC ARIMA.

If the series has a trend over time, seasonality, or some other nonstationary pattern, the usual solution is to take the difference of the series from one period to the next and then analyze this differenced series. Sometimes a series might need to be differenced more than once or differenced at lags greater than one period. (If the trend or seasonal effects are very regular, the introduction of explanatory variables can be an appropriate alternative to differencing.)

Differencing

Differencing of the response series is specified with the VAR= option of the IDENTIFY statement by placing a list of differencing periods in parentheses after the variable name. For example, to take a simple first difference of the series SALES, use the statement

```
identify var=sales(1);
```

In this example, the change in SALES from one period to the next is analyzed.

A deterministic seasonal pattern also causes the series to be nonstationary, since the expected value of the series is not the same for all time periods but is higher or lower depending on the season. When the series has a seasonal pattern, you might want to difference the series at a lag that corresponds to the length of the seasonal cycle. For example, if SALES is a monthly series, the statement

```
identify var=sales(12);
```

takes a seasonal difference of SALES, so that the series analyzed is the change in SALES from its value in the same month one year ago.

To take a second difference, add another differencing period to the list. For example, the following statement takes the second difference of SALES:

```
identify var=sales(1,1);
```

That is, SALES is differenced once at lag 1 and then differenced again, also at lag 1. The statement

```
identify var=sales(2);
```

creates a 2-span difference—that is, current period SALES minus SALES from two periods ago. The statement

```
identify var=sales(1,12);
```

takes a second-order difference of SALES, so that the series analyzed is the difference between the current period-to-period change in SALES and the change 12 periods ago. You might want to do this if the series had both a trend over time and a seasonal pattern.

There is no limit to the order of differencing and the degree of lagging for each difference.

Differencing not only affects the series used for the IDENTIFY statement output but also applies to any following ESTIMATE and FORECAST statements. ESTIMATE statements fit ARMA models to the differenced series. FORECAST statements forecast the differences and automatically sum these differences back to undo the differencing operation specified by the IDENTIFY statement, thus producing the final forecast result.

Differencing of input series is specified by the CROSSCORR= option and works just like differencing of the response series. For example, the statement

```
identify var=y(1) crosscorr=(x1(1) x2(1));
```

takes the first difference of Y, the first difference of X1, and the first difference of X2. Whenever X1 and X2 are used in INPUT= options in following ESTIMATE statements, these names refer to the differenced series.

Subset, Seasonal, and Factored ARMA Models

The simplest way to specify an ARMA model is to give the order of the AR and MA parts with the P= and Q= options. When you do this, the model has parameters for the AR and MA parts for all lags through the order specified. However, you can control the form of the ARIMA model exactly as shown in the following section.

Subset Models

You can control which lags have parameters by specifying the P= or Q= option as a list of lags in parentheses. A model that includes parameters for only some lags is sometimes called a *subset* or *additive model*. For example, consider the following two ESTIMATE statements:

```
identify var=sales;
estimate p=4;
estimate p=(1 4);
```

Both specify AR(4) models, but the first has parameters for lags 1, 2, 3, and 4, while the second has parameters for lags 1 and 4, with the coefficients for lags 2 and 3 constrained to 0. The mathematical form of the autoregressive models produced by these two specifications is shown in Table 8.1.

Table 8.1 Saturated versus Subset Models

Option	Autoregressive Operator
P=4	$(1 - \phi_1 B - \phi_2 B^2 - \phi_3 B^3 - \phi_4 B^4)$
P=(1 4)	$(1 - \phi_1 B - \phi_4 B^4)$

Seasonal Models

One particularly useful kind of subset model is a *seasonal model*. When the response series has a seasonal pattern, the values of the series at the same time of year in previous years can be important for modeling the series. For example, if the series SALES is observed monthly, the statements

```
identify var=sales;
estimate p=(12);
```

model SALES as an average value plus some fraction of its deviation from this average value a year ago, plus a random error. Although this is an AR(12) model, it has only one autoregressive parameter.

Factored Models

A factored model (also referred to as a multiplicative model) represents the ARIMA model as a product of simpler ARIMA models. For example, you might model SALES as a combination of an AR(1) process that reflects short term dependencies and an AR(12) model that reflects the seasonal pattern.

It might seem that the way to do this is with the option P=(1 12), but the AR(1) process also operates in past years; you really need autoregressive parameters at lags 1, 12, and 13. You can specify a subset model

with separate parameters at these lags, or you can specify a factored model that represents the model as the product of an AR(1) model and an AR(12) model. Consider the following two ESTIMATE statements:

```
identify var=sales;
estimate p=(1 12 13);
estimate p=(1)(12);
```

The mathematical form of the autoregressive models produced by these two specifications are shown in Table 8.2.

Table 8.2 Subset versus Factored Models

Option	Autoregressive Operator
P=(1 12 13)	$(1 - \phi_1 B - \phi_{12} B^{12} - \phi_{13} B^{13})$
P=(1)(12)	$(1 - \phi_1 B)(1 - \phi_{12} B^{12})$

Both models fit by these two ESTIMATE statements predict SALES from its values 1, 12, and 13 periods ago, but they use different parameterizations. The first model has three parameters, whose meanings may be hard to interpret.

The factored specification P=(1)(12) represents the model as the product of two different AR models. It has only two parameters: one that corresponds to recent effects and one that represents seasonal effects. Thus the factored model is more parsimonious, and its parameter estimates are more clearly interpretable.

Input Variables and Regression with ARMA Errors

In addition to past values of the response series and past errors, you can also model the response series using the current and past values of other series, called *input series*.

Several different names are used to describe ARIMA models with input series. *Transfer function model*, *intervention model*, *interrupted time series model*, *regression model with ARMA errors*, *Box-Tiao model*, and *ARIMAX model* are all different names for ARIMA models with input series. Pankratz (1991) refers to these models as *dynamic regression* models.

Using Input Series

To use input series, list the input series in a CROSSCORR= option on the IDENTIFY statement and specify how they enter the model with an INPUT= option on the ESTIMATE statement. For example, you might use a series called PRICE to help model SALES, as shown in the following statements:

```
proc arima data=a;
   identify var=sales crosscorr=price;
   estimate input=price;
run;
```

This example performs a simple linear regression of SALES on PRICE; it produces the same results as PROC REG or another SAS regression procedure. The mathematical form of the model estimated by these statements is

$$Y_t = \mu + \omega_0 X_t + a_t$$

The parameter estimates table for this example (using simulated data) is shown in Figure 8.20. The intercept parameter is labeled MU. The regression coefficient for PRICE is labeled NUM1. (See the section "Naming of Model Parameters" on page 279 for information about how parameters for input series are named.)

Figure 8.20 Parameter Estimates Table for Regression Model

The ARIMA Procedure

Conditional Least Squares Estimation

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|	Lag	Variable	Shift
MU	199.83602	2.99463	66.73	<.0001	0	sales	0
NUM1	-9.99299	0.02885	-346.38	<.0001	0	price	0

Any number of input variables can be used in a model. For example, the following statements fit a multiple regression of SALES on PRICE and INCOME:

```
proc arima data=a;
   identify var=sales crosscorr=(price income);
   estimate input=(price income);
run;
```

The mathematical form of the regression model estimated by these statements is

$$Y_t = \mu + \omega_1 X_{1,t} + \omega_2 X_{2,t} + a_t$$

Lagging and Differencing Input Series

You can also difference and lag the input series. For example, the following statements regress the change in SALES on the change in PRICE lagged by one period. The difference of PRICE is specified with the CROSSCORR= option and the lag of the change in PRICE is specified by the 1 $ in the INPUT= option.

```
proc arima data=a;
   identify var=sales(1) crosscorr=price(1);
   estimate input=( 1 $ price );
run;
```

These statements estimate the model

$$(1 - B)Y_t = \mu + \omega_0(1 - B)X_{t-1} + a_t$$

Regression with ARMA Errors

You can combine input series with ARMA models for the errors. For example, the following statements regress SALES on INCOME and PRICE but with the error term of the regression model (called the *noise series* in ARIMA modeling terminology) assumed to be an ARMA(1,1) process.

```
proc arima data=a;
   identify var=sales crosscorr=(price income);
   estimate p=1 q=1 input=(price income);
run;
```

These statements estimate the model

$$Y_t = \mu + \omega_1 X_{1,t} + \omega_2 X_{2,t} + \frac{(1-\theta_1 B)}{(1-\phi_1 B)} a_t$$

Stationarity and Input Series

Note that the requirement of stationarity applies to the noise series. If there are no input variables, the response series (after differencing and minus the mean term) and the noise series are the same. However, if there are inputs, the noise series is the residual after the effect of the inputs is removed.

There is no requirement that the input series be stationary. If the inputs are nonstationary, the response series will be nonstationary, even though the noise process might be stationary.

When nonstationary input series are used, you can fit the input variables first with no ARMA model for the errors and then consider the stationarity of the residuals before identifying an ARMA model for the noise part.

Identifying Regression Models with ARMA Errors

Previous sections described the ARIMA modeling identification process that uses the autocorrelation function plots produced by the IDENTIFY statement. This identification process does not apply when the response series depends on input variables. This is because it is the noise process for which you need to identify an ARIMA model, and when input series are involved the response series adjusted for the mean is no longer an estimate of the noise series.

However, if the input series are independent of the noise series, you can use the residuals from the regression model as an estimate of the noise series, then apply the ARIMA modeling identification process to this residual series. This assumes that the noise process is stationary.

The PLOT option in the ESTIMATE statement produces similar plots for the model residuals as the IDENTIFY statement produces for the response series. The PLOT option prints an autocorrelation function plot, an inverse autocorrelation function plot, and a partial autocorrelation function plot for the residual series. Note that these residual correlation plots are produced by default.

The following statements show how the PLOT option is used to identify the ARMA(1,1) model for the noise process used in the preceding example of regression with ARMA errors:

```
proc arima data=a;
   identify var=sales crosscorr=(price income) noprint;
   estimate input=(price income) plot;
   run;
   estimate p=1 q=1 input=(price income);
run;
```

In this example, the IDENTIFY statement includes the NOPRINT option since the autocorrelation plots for the response series are not useful when you know that the response series depends on input series.

The first ESTIMATE statement fits the regression model with no model for the noise process. The PLOT option produces plots of the autocorrelation function, inverse autocorrelation function, and partial autocorrelation function for the residual series of the regression on PRICE and INCOME.

By examining the PLOT option output for the residual series, you verify that the residual series is stationary and identify an ARMA(1,1) model for the noise process. The second ESTIMATE statement fits the final model.

Although this discussion addresses regression models, the same remarks apply to identifying an ARIMA model for the noise process in models that include input series with complex transfer functions.

Intervention Models and Interrupted Time Series

One special kind of ARIMA model with input series is called an *intervention model* or *interrupted time series* model. In an intervention model, the input series is an indicator variable that contains discrete values that flag the occurrence of an event affecting the response series. This event is an intervention in or an interruption of the normal evolution of the response time series, which, in the absence of the intervention, is usually assumed to be a pure ARIMA process.

Intervention models can be used both to model and forecast the response series and also to analyze the impact of the intervention. When the focus is on estimating the effect of the intervention, the process is often called *intervention analysis* or *interrupted time series analysis*.

Impulse Interventions

The intervention can be a one-time event. For example, you might want to study the effect of a short-term advertising campaign on the sales of a product. In this case, the input variable has the value of 1 for the period during which the advertising campaign took place and the value 0 for all other periods. Intervention variables of this kind are sometimes called *impulse functions* or *pulse functions*.

Suppose that SALES is a monthly series, and a special advertising effort was made during the month of March 1992. The following statements estimate the effect of this intervention by assuming an ARMA(1,1) model for SALES. The model is specified just like the regression model, but the intervention variable AD is constructed in the DATA step as a zero-one indicator for the month of the advertising effort.

```
data a;
   set a;
   ad = (date = '1mar1992'd);
run;

proc arima data=a;
   identify var=sales crosscorr=ad;
   estimate p=1 q=1 input=ad;
run;
```

Continuing Interventions

Other interventions can be continuing, in which case the input variable flags periods before and after the intervention. For example, you might want to study the effect of a change in tax rates on some economic measure. Another example is a study of the effect of a change in speed limits on the rate of traffic fatalities. In this case, the input variable has the value 1 after the new speed limit went into effect and the value 0 before. Intervention variables of this kind are called *step functions*.

Another example is the effect of news on product demand. Suppose it was reported in July 1996 that consumption of the product prevents heart disease (or causes cancer), and SALES is consistently higher (or lower) thereafter. The following statements model the effect of this news intervention:

```
data a;
   set a;
   news = (date >= '1jul1996'd);
run;

proc arima data=a;
   identify var=sales crosscorr=news;
   estimate p=1 q=1 input=news;
run;
```

Interaction Effects

You can include any number of intervention variables in the model. Intervention variables can have any pattern—impulse and continuing interventions are just two possible cases. You can mix discrete valued intervention variables and continuous regressor variables in the same model.

You can also form interaction effects by multiplying input variables and including the product variable as another input. Indeed, as long as the dependent measure is continuous and forms a regular time series, you can use PROC ARIMA to fit any general linear model in conjunction with an ARMA model for the error process by using input variables that correspond to the columns of the design matrix of the linear model.

Rational Transfer Functions and Distributed Lag Models

How an input series enters the model is called its *transfer function*. Thus, ARIMA models with input series are sometimes referred to as transfer function models.

In the preceding regression and intervention model examples, the transfer function is a single scale parameter. However, you can also specify complex transfer functions composed of numerator and denominator polynomials in the backshift operator. These transfer functions operate on the input series in the same way that the ARMA specification operates on the error term.

Numerator Factors

For example, suppose you want to model the effect of PRICE on SALES as taking place gradually with the impact distributed over several past lags of PRICE. This is illustrated by the following statements:

```
proc arima data=a;
   identify var=sales crosscorr=price;
   estimate input=( (1 2 3) price );
run;
```

These statements estimate the model

$$Y_t = \mu + (\omega_0 - \omega_1 B - \omega_2 B^2 - \omega_3 B^3) X_t + a_t$$

This example models the effect of PRICE on SALES as a linear function of the current and three most recent values of PRICE. It is equivalent to a multiple linear regression of SALES on PRICE, LAG(PRICE), LAG2(PRICE), and LAG3(PRICE).

This is an example of a transfer function with one *numerator factor*. The numerator factors for a transfer function for an input series are like the MA part of the ARMA model for the noise series.

Denominator Factors

You can also use transfer functions with *denominator factors*. The denominator factors for a transfer function for an input series are like the AR part of the ARMA model for the noise series. Denominator factors introduce exponentially weighted, infinite distributed lags into the transfer function.

To specify transfer functions with denominator factors, place the denominator factors after a slash (/) in the INPUT= option. For example, the following statements estimate the PRICE effect as an infinite distributed lag model with exponentially declining weights:

```
proc arima data=a;
   identify var=sales crosscorr=price;
   estimate input=( / (1) price );
run;
```

The transfer function specified by these statements is as follows:

$$\frac{\omega_0}{(1 - \delta_1 B)} X_t$$

This transfer function also can be written in the following equivalent form:

$$\omega_0 \left(1 + \sum_{i=1}^{\infty} \delta_1^i B^i \right) X_t$$

This transfer function can be used with intervention inputs. When it is used with a pulse function input, the result is an intervention effect that dies out gradually over time. When it is used with a step function input, the result is an intervention effect that increases gradually to a limiting value.

Rational Transfer Functions

By combining various numerator and denominator factors in the INPUT= option, you can specify *rational transfer functions* of any complexity. To specify an input with a general rational transfer function of the form

$$\frac{\omega(B)}{\delta(B)} B^k X_t$$

use an INPUT= option in the ESTIMATE statement of the form

input=(*k* $ (*ω-lags*) / (*δ-lags*) x)

See the section "Specifying Inputs and Transfer Functions" on page 276 for more information.

Identifying Transfer Function Models

The CROSSCORR= option of the IDENTIFY statement prints sample cross-correlation functions that show the correlation between the response series and the input series at different lags. The sample cross-correlation function can be used to help identify the form of the transfer function appropriate for an input series. See textbooks on time series analysis for information about using cross-correlation functions to identify transfer function models.

For the cross-correlation function to be meaningful, the input and response series must be filtered with a prewhitening model for the input series. See the section "Prewhitening" on page 270 for more information about this issue.

Forecasting with Input Variables

To forecast a response series by using an ARIMA model with inputs, you need values of the input series for the forecast periods. You can supply values for the input variables for the forecast periods in the DATA= data set, or you can have PROC ARIMA forecast the input variables.

If you do not have future values of the input variables in the input data set used by the FORECAST statement, the input series must be forecast before the ARIMA procedure can forecast the response series. If you fit an ARIMA model to each of the input series for which you need forecasts before fitting the model for the response series, the FORECAST statement automatically uses the ARIMA models for the input series to generate the needed forecasts of the inputs.

For example, suppose you want to forecast SALES for the next 12 months. In this example, the change in SALES is predicted as a function of the change in PRICE, plus an ARMA(1,1) noise process. To forecast SALES by using PRICE as an input, you also need to fit an ARIMA model for PRICE.

The following statements fit an AR(2) model to the change in PRICE before fitting and forecasting the model for SALES. The FORECAST statement automatically forecasts PRICE using this AR(2) model to get the future inputs needed to produce the forecast of SALES.

```
proc arima data=a;
   identify var=price(1);
   estimate p=2;
   identify var=sales(1) crosscorr=price(1);
   estimate p=1 q=1 input=price;
   forecast lead=12 interval=month id=date out=results;
run;
```

Fitting a model to the input series is also important for identifying transfer functions. (See the section "Prewhitening" on page 270 for more information.)

Input values from the DATA= data set and input values forecast by PROC ARIMA can be combined. For example, a model for SALES might have three input series: PRICE, INCOME, and TAXRATE. For the forecast, you assume that the tax rate will be unchanged. You have a forecast for INCOME from another source but only for the first few periods of the SALES forecast you want to make. You have no future values for PRICE, which needs to be forecast as in the preceding example.

In this situation, you include observations in the input data set for all forecast periods, with SALES and PRICE set to a missing value, with TAXRATE set to its last actual value, and with INCOME set to forecast values for the periods you have forecasts for and set to missing values for later periods. In the PROC ARIMA step, you estimate ARIMA models for PRICE and INCOME before you estimate the model for SALES, as shown in the following statements:

```
proc arima data=a;
   identify var=price(1);
   estimate p=2;
   identify var=income(1);
   estimate p=2;
   identify var=sales(1) crosscorr=( price(1) income(1) taxrate );
   estimate p=1 q=1 input=( price income taxrate );
   forecast lead=12 interval=month id=date out=results;
run;
```

In forecasting SALES, the ARIMA procedure uses as inputs the value of PRICE forecast by its ARIMA model, the value of TAXRATE found in the DATA= data set, and the value of INCOME found in the DATA= data set, or, when the INCOME variable is missing, the value of INCOME forecast by its ARIMA model. (Because SALES is missing for future time periods, the estimation of model parameters is not affected by the forecast values for PRICE, INCOME, or TAXRATE.)

Data Requirements

PROC ARIMA can handle time series of moderate size; there should be at least 30 observations. With fewer than 30 observations, the parameter estimates might be poor. With thousands of observations, the method requires considerable computer time and memory.

Syntax: ARIMA Procedure

The ARIMA procedure uses the following statements:

```
PROC ARIMA options ;
  BY variables ;
  IDENTIFY VAR=variable options ;
  ESTIMATE options ;
  OUTLIER options ;
  FORECAST options ;
```

The PROC ARIMA and IDENTIFY statements are required.

Functional Summary

The statements and options that control the ARIMA procedure are summarized in Table 8.3.

Table 8.3 Functional Summary

Description	Statement	Option
Data Set Options		
specify the input data set	PROC ARIMA	DATA=
	IDENTIFY	DATA=
specify the output data set	PROC ARIMA	OUT=
	FORECAST	OUT=
include only forecasts in the output data set	FORECAST	NOOUTALL
write autocovariances to output data set	IDENTIFY	OUTCOV=
write parameter estimates to an output data set	ESTIMATE	OUTEST=
write correlation of parameter estimates	ESTIMATE	OUTCORR
write covariance of parameter estimates	ESTIMATE	OUTCOV
write estimated model to an output data set	ESTIMATE	OUTMODEL=
write statistics of fit to an output data set	ESTIMATE	OUTSTAT=
Options for Identifying the Series		
difference time series and plot autocorrelations	IDENTIFY	
specify response series and differencing	IDENTIFY	VAR=
specify and cross-correlate input series	IDENTIFY	CROSSCORR=
center data by subtracting the mean	IDENTIFY	CENTER
exclude missing values	IDENTIFY	NOMISS
delete previous models and start	IDENTIFY	CLEAR
specify the significance level for tests	IDENTIFY	ALPHA=
perform tentative ARMA order identification by using the ESACF method	IDENTIFY	ESACF
perform tentative ARMA order identification by using the MINIC method	IDENTIFY	MINIC

Table 8.3 *continued*

Description	Statement	Option
perform tentative ARMA order identification by using the SCAN method	IDENTIFY	SCAN
specify the range of autoregressive model orders for estimating the error series for the MINIC method	IDENTIFY	PERROR=
determine the AR dimension of the SCAN, ESACF, and MINIC tables	IDENTIFY	P=
determine the MA dimension of the SCAN, ESACF, and MINIC tables	IDENTIFY	Q=
perform stationarity tests	IDENTIFY	STATIONARITY=
selection of white noise test statistic in the presence of missing values	IDENTIFY	WHITENOISE=
Options for Defining and Estimating the Model		
specify and estimate ARIMA models	ESTIMATE	
specify autoregressive part of model	ESTIMATE	P=
specify moving-average part of model	ESTIMATE	Q=
specify input variables and transfer functions	ESTIMATE	INPUT=
drop mean term from the model	ESTIMATE	NOINT
specify the estimation method	ESTIMATE	METHOD=
use alternative form for transfer functions	ESTIMATE	ALTPARM
suppress degrees-of-freedom correction in variance estimates	ESTIMATE	NODF
selection of white noise test statistic in the presence of missing values	ESTIMATE	WHITENOISE=
Options for Outlier Detection		
specify the significance level for tests	OUTLIER	ALPHA=
identify detected outliers with variable	OUTLIER	ID=
limit the number of outliers	OUTLIER	MAXNUM=
limit the number of outliers to a percentage of the series	OUTLIER	MAXPCT=
specify the variance estimator used for testing	OUTLIER	SIGMA=
specify the type of level shifts	OUTLIER	TYPE=
Printing Control Options		
limit number of lags shown in correlation plots	IDENTIFY	NLAG=
suppress printed output for identification	IDENTIFY	NOPRINT
plot autocorrelation functions of the residuals	ESTIMATE	PLOT
print log-likelihood around the estimates	ESTIMATE	GRID
control spacing for GRID option	ESTIMATE	GRIDVAL=
print details of the iterative estimation process	ESTIMATE	PRINTALL
suppress printed output for estimation	ESTIMATE	NOPRINT

Table 8.3 *continued*

Description	Statement	Option
suppress printing of the forecast values	FORECAST	NOPRINT
print the one-step forecasts and residuals	FORECAST	PRINTALL
Plotting Control Options		
request plots associated with model identification, residual analysis, and forecasting	PROC ARIMA	PLOTS=
Options to Specify Parameter Values		
specify autoregressive starting values	ESTIMATE	AR=
specify moving-average starting values	ESTIMATE	MA=
specify a starting value for the mean parameter	ESTIMATE	MU=
specify starting values for transfer functions	ESTIMATE	INITVAL=
Options to Control the Iterative Estimation Process		
specify convergence criterion	ESTIMATE	CONVERGE=
specify the maximum number of iterations	ESTIMATE	MAXITER=
specify criterion for checking for singularity	ESTIMATE	SINGULAR=
suppress the iterative estimation process	ESTIMATE	NOEST
omit initial observations from objective	ESTIMATE	BACKLIM=
specify perturbation for numerical derivatives	ESTIMATE	DELTA=
omit stationarity and invertibility checks	ESTIMATE	NOSTABLE
use preliminary estimates as starting values for ML and ULS	ESTIMATE	NOLS
Options for Forecasting		
forecast the response series	FORECAST	
specify how many periods to forecast	FORECAST	LEAD=
specify the ID variable	FORECAST	ID=
specify the periodicity of the series	FORECAST	INTERVAL=
specify size of forecast confidence limits	FORECAST	ALPHA=
start forecasting before end of the input data	FORECAST	BACK=
specify the variance term used to compute forecast standard errors and confidence limits	FORECAST	SIGSQ=
control the alignment of SAS date values	FORECAST	ALIGN=
BY Groups		
specify BY group processing	BY	

PROC ARIMA Statement

PROC ARIMA *options* **;**

The following options can be used in the PROC ARIMA statement.

DATA=*SAS-data-set*
specifies the name of the SAS data set that contains the time series. If different DATA= specifications appear in the PROC ARIMA and IDENTIFY statements, the one in the IDENTIFY statement is used. If the DATA= option is not specified in either the PROC ARIMA or IDENTIFY statement, the most recently created SAS data set is used.

PLOTS<(global-plot-options)> <= plot-request <(options)> >
PLOTS<(global-plot-options)> <= (plot-request <(options)> <... plot-request <(options)> >) >
controls the plots produced through ODS Graphics. When you specify only one plot request, you can omit the parentheses around the plot request.

Here are some examples:

```
plots=none
plots=all
plots(unpack)=series(corr crosscorr)
plots(only)=(series(corr crosscorr) residual(normal smooth))
```

Global Plot Options:

The *global-plot-options* apply to all relevant plots generated by the ARIMA procedure. The following *global-plot-options* are supported:

ONLY
suppresses the default plots. Only the plots specifically requested are produced.

UNPACK
displays each graph separately. (By default, some graphs can appear together in a single panel.)

Specific Plot Options

The following list describes the specific plots and their options.

ALL
produces all plots appropriate for the particular analysis.

NONE
suppresses all plots.

SERIES(< *series-plot-options* **>)**
produces plots associated with the identification stage of the modeling. The panel plots corresponding to the CORR and CROSSCORR options are produced by default. The following *series-plot-options* are available:

ACF
produces the plot of autocorrelations.

ALL
produces all the plots associated with the identification stage.

CORR
produces a panel of plots that are useful in the trend and correlation analysis of the series. The panel consists of the following:

- the time series plot
- the series-autocorrelation plot
- the series-partial-autocorrelation plot
- the series-inverse-autocorrelation plot

CROSSCORR
produces panels of cross-correlation plots.

IACF
produces the plot of inverse-autocorrelations.

PACF
produces the plot of partial-autocorrelations.

RESIDUAL(< *residual-plot-options* **>)**
produces the residuals plots. The residual correlation and normality diagnostic panels are produced by default. The following *residual-plot-options* are available:

ACF
produces the plot of residual autocorrelations.

ALL
produces all the residual diagnostics plots appropriate for the particular analysis.

CORR
produces a summary panel of the residual correlation diagnostics that consists of the following:

- the residual-autocorrelation plot
- the residual-partial-autocorrelation plot
- the residual-inverse-autocorrelation plot
- a plot of Ljung-Box white-noise test p-values at different lags

HIST
produces the histogram of the residuals.

IACF
produces the plot of residual inverse-autocorrelations.

NORMAL
produces a summary panel of the residual normality diagnostics that consists of the following:

- histogram of the residuals
- normal quantile plot of the residuals

PACF
produces the plot of residual partial-autocorrelations.

QQ
produces the normal quantile plot of the residuals.

SMOOTH
produces a scatter plot of the residuals against time, which has an overlaid smooth fit.

WN
produces the plot of Ljung-Box white-noise test p-values at different lags.

FORECAST(< *forecast-plot-options* **>)**
produces the forecast plots in the forecasting stage. The forecast-only plot that shows the multistep forecasts in the forecast region is produced by default.

The following *forecast-plot-options* are available:

ALL
produces the forecast-only plot as well as the forecast plot.

FORECAST
produces a plot that shows the one-step-ahead forecasts as well as the multistep-ahead forecasts.

FORECASTONLY
produces a plot that shows only the multistep-ahead forecasts in the forecast region.

OUT=*SAS-data-set*
specifies a SAS data set to which the forecasts are output. If different OUT= specifications appear in the PROC ARIMA and FORECAST statements, the one in the FORECAST statement is used.

BY Statement

BY *variables* **;**

A BY statement can be used in the ARIMA procedure to process a data set in groups of observations defined by the BY variables. Note that all IDENTIFY, ESTIMATE, and FORECAST statements specified are applied to all BY groups.

Because of the need to make data-based model selections, BY-group processing is not usually done with PROC ARIMA. You usually want to use different models for the different series contained in different BY groups, and the PROC ARIMA BY statement does not let you do this.

Using a BY statement imposes certain restrictions. The BY statement must appear before the first RUN statement. If a BY statement is used, the input data must come from the data set specified in the PROC statement; that is, no input data sets can be specified in IDENTIFY statements.

When a BY statement is used with PROC ARIMA, interactive processing applies only to the first BY group. Once the end of the PROC ARIMA step is reached, all ARIMA statements specified are executed again for each of the remaining BY groups in the input data set.

IDENTIFY Statement

IDENTIFY *VAR=variable options* **;**

The IDENTIFY statement specifies the time series to be modeled, differences the series if desired, and computes statistics to help identify models to fit. Use an IDENTIFY statement for each time series that you want to model.

If other time series are to be used as inputs in a subsequent ESTIMATE statement, they must be listed in a CROSSCORR= list in the IDENTIFY statement.

The following options are used in the IDENTIFY statement. The VAR= option is required.

ALPHA=*significance-level*

The ALPHA= option specifies the significance level for tests in the IDENTIFY statement. The default is 0.05.

CENTER

centers each time series by subtracting its sample mean. The analysis is done on the centered data. Later, when forecasts are generated, the mean is added back. Note that centering is done after differencing. The CENTER option is normally used in conjunction with the NOCONSTANT option of the ESTIMATE statement.

CLEAR

deletes all old models. This option is useful when you want to delete old models so that the input variables are not prewhitened. (See the section "Prewhitening" on page 270 for more information.)

CROSSCORR=*variable* **(***d11, d12, . . . , d1k* **)**

CROSSCORR= (*variable* **(***d11, d12, . . . , d1k* **)...** *variable* **(***d21, d22, . . . , d2k* **))**

names the variables cross-correlated with the response variable given by the VAR= specification.

Each variable name can be followed by a list of differencing lags in parentheses, the same as for the VAR= specification. If differencing is specified for a variable in the CROSSCORR= list, the differenced series is cross-correlated with the VAR= option series, and the differenced series is used when the ESTIMATE statement INPUT= option refers to the variable.

DATA=*SAS-data-set*

specifies the input SAS data set that contains the time series. If the DATA= option is omitted, the DATA= data set specified in the PROC ARIMA statement is used; if the DATA= option is omitted from the PROC ARIMA statement as well, the most recently created data set is used.

ESACF

computes the extended sample autocorrelation function and uses these estimates to tentatively identify the autoregressive and moving-average orders of mixed models.

The ESACF option generates two tables. The first table displays extended sample autocorrelation estimates, and the second table displays probability values that can be used to test the significance of these estimates. The P=($p_{min} : p_{max}$) and Q=($q_{min} : q_{max}$) options determine the size of the table.

The autoregressive and moving-average orders are tentatively identified by finding a triangular pattern in which all values are insignificant. The ARIMA procedure finds these patterns based on the IDENTIFY statement ALPHA= option and displays possible recommendations for the orders.

The following code generates an ESACF table with dimensions of p=(0:7) and q=(0:8).

```
proc arima data=test;
   identify var=x esacf p=(0:7) q=(0:8);
run;
```

See the section "The ESACF Method" on page 265 for more information.

MINIC

uses information criteria or penalty functions to provide tentative ARMA order identification. The MINIC option generates a table that contains the computed information criterion associated with various ARMA model orders. The PERROR=($p_{\epsilon,min} : p_{\epsilon,max}$) option determines the range of the autoregressive model orders used to estimate the error series. The P=($p_{min} : p_{max}$) and Q=($q_{min} : q_{max}$) options determine the size of the table. The ARMA orders are tentatively identified by those orders that minimize the information criterion.

The following statements generate a MINIC table with default dimensions of p=(0:5) and q=(0:5) and with the error series estimated by an autoregressive model with an order, p_ϵ, that minimizes the AIC in the range from 8 to 11.

```
proc arima data=test;
   identify var=x minic perror=(8:11);
run;
```

See the section "The MINIC Method" on page 267 for more information.

NLAG=*number*

indicates the number of lags to consider in computing the autocorrelations and cross-correlations. To obtain preliminary estimates of an ARIMA(p, d, q) model, the NLAG= value must be at least $p + q + d$. The number of observations must be greater than or equal to the NLAG= value. The default value for NLAG= is 24 or one-fourth the number of observations, whichever is less. Even though the NLAG= value is specified, the NLAG= value can be changed according to the data set.

NOMISS
uses only the first continuous sequence of data with no missing values. By default, all observations are used.

NOPRINT
suppresses the normal printout (including the correlation plots) generated by the IDENTIFY statement.

OUTCOV=*SAS-data-set*
writes the autocovariances, autocorrelations, inverse autocorrelations, partial autocorrelations, and cross covariances to an output SAS data set. If the OUTCOV= option is not specified, no covariance output data set is created. See the section "OUTCOV= Data Set" on page 285 for more information.

P=($p_{min} : p_{max}$**)**
see the ESACF, MINIC, and SCAN options for details.

PERROR=($p_{\epsilon,min} : p_{\epsilon,max}$**)**
determines the range of the autoregressive model orders used to estimate the error series in MINIC, a tentative ARMA order identification method. See the section "The MINIC Method" on page 267 for more information. By default $p_{\epsilon,min}$ is set to p_{max} and $p_{\epsilon,max}$ is set to $p_{max} + q_{max}$, where p_{max} and q_{max} are the maximum settings of the P= and Q= options on the IDENTIFY statement.

Q=($q_{min} : q_{max}$**)**
see the ESACF, MINIC, and SCAN options for details.

SCAN
computes estimates of the squared canonical correlations and uses these estimates to tentatively identify the autoregressive and moving-average orders of mixed models.

The SCAN option generates two tables. The first table displays squared canonical correlation estimates, and the second table displays probability values that can be used to test the significance of these estimates. The P=($p_{min} : p_{max}$) and Q=($q_{min} : q_{max}$) options determine the size of each table.

The autoregressive and moving-average orders are tentatively identified by finding a rectangular pattern in which all values are insignificant. The ARIMA procedure finds these patterns based on the IDENTIFY statement ALPHA= option and displays possible recommendations for the orders.

The following code generates a SCAN table with default dimensions of p=(0:5) and q=(0:5). The recommended orders are based on a significance level of 0.1.

```
proc arima data=test;
   identify var=x scan alpha=0.1;
run;
```

See the section "The SCAN Method" on page 268 for more information.

STATIONARITY=
performs stationarity tests. Stationarity tests can be used to determine whether differencing terms should be included in the model specification. In each stationarity test, the autoregressive orders can be specified by a range, *test*= ar_{max}, or as a list of values, *test*= $(ar_1, .., ar_n)$, where *test* is ADF, PP, or RW. The default is (0,1,2).

See the section "Stationarity Tests" on page 270 for more information.

STATIONARITY=(ADF= *AR orders* **DLAG=** *s* **)**

STATIONARITY=(DICKEY= *AR orders* **DLAG=** *s* **)**

performs augmented Dickey-Fuller tests. If the DLAG=*s* option is specified with *s* is greater than one, seasonal Dickey-Fuller tests are performed. The maximum allowable value of *s* is 12. The default value of *s* is 1. The following code performs augmented Dickey-Fuller tests with autoregressive orders 2 and 5.

```
proc arima data=test;
   identify var=x stationarity=(adf=(2,5));
run;
```

STATIONARITY=(PP= *AR orders* **)**

STATIONARITY=(PHILLIPS= *AR orders* **)**

performs Phillips-Perron tests. The following statements perform augmented Phillips-Perron tests with autoregressive orders ranging from 0 to 6.

```
proc arima data=test;
   identify var=x stationarity=(pp=6);
run;
```

STATIONARITY=(RW=*AR orders* **)**

STATIONARITY=(RANDOMWALK=*AR orders* **)**

performs random-walk-with-drift tests. The following statements perform random-walk-with-drift tests with autoregressive orders ranging from 0 to 2.

```
proc arima data=test;
   identify var=x stationarity=(rw);
run;
```

VAR=*variable*

VAR= *variable* **(** *d1, d2, . . . , dk* **)**

names the variable that contains the time series to analyze. The VAR= option is required.

A list of differencing lags can be placed in parentheses after the variable name to request that the series be differenced at these lags. For example, VAR=X(1) takes the first differences of X. VAR=X(1,1) requests that X be differenced twice, both times with lag 1, producing a second difference series, which is
$(X_t - X_{t-1}) - (X_{t-1} - X_{t-2}) = X_t - 2X_{t-1} + X_{t-2}$.

VAR=X(2) differences X once at lag two $(X_t - X_{t-2})$.

If differencing is specified, it is the differenced series that is processed by any subsequent ESTIMATE statement.

WHITENOISE=ST | IGNOREMISS

specifies the type of test statistic that is used in the white noise test of the series when the series contains missing values. If WHITENOISE=IGNOREMISS, the standard Ljung-Box test statistic is used. If WHITENOISE=ST, a modification of this statistic suggested by Stoffer and Toloi (1992) is used. The default is WHITENOISE=ST.

ESTIMATE Statement

< *label:* **>ESTIMATE** *options* **;**

The ESTIMATE statement specifies an ARMA model or transfer function model for the response variable specified in the previous IDENTIFY statement, and produces estimates of its parameters. The ESTIMATE statement also prints diagnostic information by which to check the model. The label in the ESTIMATE statement is optional. Include an ESTIMATE statement for each model that you want to estimate.

Options used in the ESTIMATE statement are described in the following sections.

Options for Defining the Model and Controlling Diagnostic Statistics

The following options are used to define the model to be estimated and to control the output that is printed.

ALTPARM
specifies the alternative parameterization of the overall scale of transfer functions in the model. See the section "Alternative Model Parameterization" on page 277 for details.

INPUT=*variable*

INPUT=(*transfer-function variable* **...)**
specifies input variables and their transfer functions.

The variables used on the INPUT= option must be included in the CROSSCORR= list in the previous IDENTIFY statement. If any differencing is specified in the CROSSCORR= list, then the differenced series is used as the input to the transfer function.

The transfer function specification for an input variable is optional. If no transfer function is specified, the input variable enters the model as a simple regressor. If specified, the transfer function specification has the following syntax:

$$S\$(L_{1,1}, L_{1,2}, \ldots)(L_{2,1}, \ldots)\ldots/(L_{j,1}, \ldots)\ldots$$

Here, S is a shift or lag of the input variable, the terms before the slash (/) are numerator factors, and the terms after the slash (/) are denominator factors of the transfer function. All three parts are optional. See the section "Specifying Inputs and Transfer Functions" on page 276 for details.

METHOD=*value*
specifies the estimation method to use. METHOD=ML specifies the maximum likelihood method. METHOD=ULS specifies the unconditional least squares method. METHOD=CLS specifies the conditional least squares method. METHOD=CLS is the default. See the section "Estimation Details" on page 272 for more information.

NOCONSTANT

NOINT
suppresses the fitting of a constant (or intercept) parameter in the model. (That is, the parameter μ is omitted.)

NODF
estimates the variance by dividing the error sum of squares (SSE) by the number of residuals. The default is to divide the SSE by the number of residuals minus the number of free parameters in the model.

NOPRINT
suppresses the normal printout generated by the ESTIMATE statement. If the NOPRINT option is specified for the ESTIMATE statement, then any error and warning messages are printed to the SAS log.

P=*order*

P=*(lag, . . . , lag) . . . (lag, . . . , lag)*
specifies the autoregressive part of the model. By default, no autoregressive parameters are fit.

P=($l_1, l_2, \ldots, l_k$) defines a model with autoregressive parameters at the specified lags. P= *order* is equivalent to P=(1, 2, . . . , *order*).

A concatenation of parenthesized lists specifies a factored model. For example, P=(1,2,5)(6,12) specifies the autoregressive model

$$(1 - \phi_{1,1}B - \phi_{1,2}B^2 - \phi_{1,3}B^5)(1 - \phi_{2,1}B^6 - \phi_{2,2}B^{12})$$

PLOT
plots the residual autocorrelation functions. The sample autocorrelation, the sample inverse autocorrelation, and the sample partial autocorrelation functions of the model residuals are plotted.

Q=*order*

Q=*(lag, . . . , lag) . . . (lag, . . . , lag)*
specifies the moving-average part of the model. By default, no moving-average part is included in the model.

Q=($l_1, l_2, \ldots, l_k$) defines a model with moving-average parameters at the specified lags. Q= *order* is equivalent to Q=(1, 2, . . . , *order*). A concatenation of parenthesized lists specifies a factored model. The interpretation of factors and lags is the same as for the P= option.

WHITENOISE=ST | IGNOREMISS
specifies the type of test statistic that is used in the white noise test of the series when the series contains missing values. If WHITENOISE=IGNOREMISS, the standard Ljung-Box test statistic is used. If WHITENOISE=ST, a modification of this statistic suggested by Stoffer and Toloi (1992) is used. The default is WHITENOISE=ST.

Options for Output Data Sets

The following options are used to store results in SAS data sets:

OUTEST=*SAS-data-set*
writes the parameter estimates to an output data set. If the OUTCORR or OUTCOV option is used, the correlations or covariances of the estimates are also written to the OUTEST= data set. See the section "OUTEST= Data Set" on page 286 for a description of the OUTEST= output data set.

OUTCORR
writes the correlations of the parameter estimates to the OUTEST= data set.

OUTCOV
writes the covariances of the parameter estimates to the OUTEST= data set.

OUTMODEL=*SAS-data-set*
writes the model and parameter estimates to an output data set. If OUTMODEL= is not specified, no model output data set is created. See the section "OUTMODEL= SAS Data Set" on page 289 for a description of the OUTMODEL= output data set.

OUTSTAT=*SAS-data-set*
writes the model diagnostic statistics to an output data set. If OUTSTAT= is not specified, no statistics output data set is created. See the section "OUTSTAT= Data Set" on page 290 for a description of the OUTSTAT= output data set.

Options to Specify Parameter Values

The following options enable you to specify values for the model parameters. These options can provide starting values for the estimation process, or you can specify fixed parameters for use in the FORECAST stage and suppress the estimation process with the NOEST option. By default, the ARIMA procedure finds initial parameter estimates and uses these estimates as starting values in the iterative estimation process.

If values for any parameters are specified, values for all parameters should be given. The number of values given must agree with the model specifications.

AR=*value* **...**
lists starting values for the autoregressive parameters. See the section "Initial Values" on page 277 for more information.

INITVAL=(*initializer-spec variable* **...)**
specifies starting values for the parameters in the transfer function parts of the model. See the section "Initial Values" on page 277 for more information.

MA=*value* **...**
lists starting values for the moving-average parameters. See the section "Initial Values" on page 277 for more information.

MU=*value*
specifies the MU parameter.

NOEST
uses the values specified with the AR=, MA=, INITVAL=, and MU= options as final parameter values. The estimation process is suppressed except for estimation of the residual variance. The specified parameter values are used directly by the next FORECAST statement. When NOEST is specified, standard errors, *t* values, and the correlations between estimates are displayed as 0 or missing. (The NOEST option is useful, for example, when you want to generate forecasts that correspond to a published model.)

Options to Control the Iterative Estimation Process

The following options can be used to control the iterative process of minimizing the error sum of squares or maximizing the log-likelihood function. These tuning options are not usually needed but can be useful if convergence problems arise.

BACKLIM=*−n*

omits the specified number of initial residuals from the sum of squares or likelihood function. Omitting values can be useful for suppressing transients in transfer function models that are sensitive to start-up values.

CONVERGE=*value*

specifies the convergence criterion. Convergence is assumed when the largest change in the estimate for any parameter is less that the CONVERGE= option value. If the absolute value of the parameter estimate is greater than 0.01, the relative change is used; otherwise, the absolute change in the estimate is used. The default is CONVERGE=0.001.

DELTA=*value*

specifies the perturbation value for computing numerical derivatives. The default is DELTA=0.001.

GRID

prints the error sum of squares (SSE) or concentrated log-likelihood surface in a small grid of the parameter space around the final estimates. For each pair of parameters, the SSE is printed for the nine parameter-value combinations formed by the grid, with a center at the final estimates and with spacing given by the GRIDVAL= specification. The GRID option can help you judge whether the estimates are truly at the optimum, since the estimation process does not always converge. For models with a large number of parameters, the GRID option produces voluminous output.

GRIDVAL=*number*

controls the spacing in the grid printed by the GRID option. The default is GRIDVAL=0.005.

MAXITER=n

MAXIT=n

specifies the maximum number of iterations allowed. The default is MAXITER=50.

NOLS

begins the maximum likelihood or unconditional least squares iterations from the preliminary estimates rather than from the conditional least squares estimates that are produced after four iterations. See the section "Estimation Details" on page 272 for more information.

NOSTABLE

specifies that the autoregressive and moving-average parameter estimates for the noise part of the model not be restricted to the stationary and invertible regions, respectively. See the section "Stationarity and Invertibility" on page 278 for more information.

PRINTALL

prints preliminary estimation results and the iterations in the final estimation process.

NOTFSTABLE

specifies that the parameter estimates for the denominator polynomial of the transfer function part of the model not be restricted to the stability region. See the section "Stationarity and Invertibility" on page 278 for more information.

SINGULAR=*value*

specifies the criterion for checking singularity. If a pivot of a sweep operation is less than the SINGULAR= value, the matrix is deemed singular. Sweep operations are performed on the Jacobian matrix during final estimation and on the covariance matrix when preliminary estimates are obtained. The default is SINGULAR=1E–7.

OUTLIER Statement

OUTLIER *options* **;**

The OUTLIER statement can be used to detect shifts in the level of the response series that are not accounted for by the previously estimated model. An ESTIMATE statement must precede the OUTLIER statement. The following options are used in the OUTLIER statement:

TYPE=ADDITIVE

TYPE=SHIFT

TYPE=TEMP ($d_1, \ldots, d_k$ **)**

TYPE=(< ADDITIVE >< SHIFT > < TEMP ($d_1, \ldots, d_k$ **) >)**

specifies the types of level shifts to search for. The default is TYPE=(ADDITIVE SHIFT), which requests searching for additive outliers and permanent level shifts. The option TEMP($d_1, \ldots, d_k$) requests searching for temporary changes in the level of durations $d_1, \ldots, d_k$. These options can also be abbreviated as AO, LS, and TC.

ALPHA=*significance-level*

specifies the significance level for tests in the OUTLIER statement. The default is 0.05.

SIGMA=ROBUST | MSE

specifies the type of error variance estimate to use in the statistical tests performed during the outlier detection. SIGMA=MSE corresponds to the usual mean squared error (MSE) estimate, and SIGMA=ROBUST corresponds to a robust estimate of the error variance. The default is SIGMA=ROBUST.

MAXNUM=*number*

limits the number of outliers to search. The default is MAXNUM=5.

MAXPCT=*number*

limits the number of outliers to search for according to a percentage of the series length. The default is MAXPCT=2. When both the MAXNUM= and MAXPCT= options are specified, the minimum of the two search numbers is used.

ID=*Date-Time ID variable*

specifies a SAS date, time, or datetime identification variable to label the detected outliers. This variable must be present in the input data set.

The following examples illustrate a few possibilities for the OUTLIER statement.

The most basic usage, shown as follows, sets all the options to their default values.

```
outlier;
```

That is, it is equivalent to

```
outlier type=(ao ls) alpha=0.05 sigma=robust maxnum=5 maxpct=2;
```

The following statement requests a search for permanent level shifts and for temporary level changes of durations 6 and 12. The search is limited to at most three changes and the significance level of the underlying tests is 0.001. MSE is used as the estimate of error variance. It also requests labeling of the detected shifts using an ID variable *date*.

```
outlier type=(ls tc(6 12)) alpha=0.001 sigma=mse maxnum=3 ID=date;
```

FORECAST Statement

FORECAST *options* **;**

The FORECAST statement generates forecast values for a time series by using the parameter estimates produced by the previous ESTIMATE statement. See the section "Forecasting Details" on page 280 for more information about calculating forecasts.

The following options can be used in the FORECAST statement:

ALIGN=*option*

controls the alignment of SAS dates used to identify output observations. The ALIGN= option allows the following values: BEGINNING|BEG|B, MIDDLE|MID|M, and ENDING|END|E. BEGINNING is the default.

ALPHA=*n*

sets the size of the forecast confidence limits. The ALPHA= value must be between 0 and 1. When you specify ALPHA=α, the upper and lower confidence limits have a $1-\alpha$ confidence level. The default is ALPHA=0.05, which produces 95% confidence intervals. ALPHA values are rounded to the nearest hundredth.

BACK=*n*

specifies the number of observations before the end of the data where the multistep forecasts are to begin. The BACK= option value must be less than or equal to the number of observations minus the number of parameters.

The default is BACK=0, which means that the forecast starts at the end of the available data. The end of the data is the last observation for which a noise value can be calculated. If there are no input series, the end of the data is the last nonmissing value of the response time series. If there are input series, this observation can precede the last nonmissing value of the response variable, since there may be missing values for some of the input series.

ID=*variable*

names a variable in the input data set that identifies the time periods associated with the observations. The ID= variable is used in conjunction with the INTERVAL= option to extrapolate ID values from the end of the input data to identify forecast periods in the OUT= data set.

If the INTERVAL= option specifies an interval type, the ID variable must be a SAS date or datetime variable with the spacing between observations indicated by the INTERVAL= value. If the INTERVAL= option is not used, the last input value of the ID= variable is incremented by one for each forecast period to extrapolate the ID values for forecast observations.

INTERVAL=*interval*

INTERVAL=*n*

specifies the time interval between observations. See Chapter 5, "Date Intervals, Formats, and Functions," for information about valid INTERVAL= values.

The value of the INTERVAL= option is used by PROC ARIMA to extrapolate the ID values for forecast observations and to check that the input data are in order with no missing periods. See the section "Specifying Series Periodicity" on page 282 for more details.

LEAD=*n*

specifies the number of multistep forecast values to compute. For example, if LEAD=10, PROC ARIMA forecasts for ten periods beginning with the end of the input series (or earlier if BACK= is specified). It is possible to obtain fewer than the requested number of forecasts if a transfer function model is specified and insufficient data are available to compute the forecast. The default is LEAD=24.

NOOUTALL

includes only the final forecast observations in the OUT= output data set, not the one-step forecasts for the data before the forecast period.

NOPRINT

suppresses the normal printout of the forecast and associated values.

OUT=*SAS-data-set*

writes the forecast (and other values) to an output data set. If OUT= is not specified, the OUT= data set specified in the PROC ARIMA statement is used. If OUT= is also not specified in the PROC ARIMA statement, no output data set is created. See the section "OUT= Data Set" on page 284 for more information.

PRINTALL

prints the FORECAST computation throughout the whole data set. The forecast values for the data before the forecast period (specified by the BACK= option) are one-step forecasts.

SIGSQ=*value*

specifies the variance term used in the formula for computing forecast standard errors and confidence limits. The default value is the variance estimate computed by the preceding ESTIMATE statement. This option is useful when you wish to generate forecast standard errors and confidence limits based on a published model. It would often be used in conjunction with the NOEST option in the preceding ESTIMATE statement.

Details: ARIMA Procedure

The Inverse Autocorrelation Function

The sample inverse autocorrelation function (SIACF) plays much the same role in ARIMA modeling as the sample partial autocorrelation function (SPACF), but it generally indicates subset and seasonal autoregressive models better than the SPACF.

Additionally, the SIACF can be useful for detecting over-differencing. If the data come from a nonstationary or nearly nonstationary model, the SIACF has the characteristics of a noninvertible moving-average. Likewise, if the data come from a model with a noninvertible moving average, then the SIACF has nonstationary characteristics and therefore decays slowly. In particular, if the data have been over-differenced, the SIACF looks like a SACF from a nonstationary process.

The inverse autocorrelation function is not often discussed in textbooks, so a brief description is given here. For more complete discussions, see: Cleveland (1972); Chatfield (1980); Priestley (1981).

Let W_t be generated by the ARMA(p, q) process

$$\phi(B)W_t = \theta(B)a_t$$

where a_t is a white noise sequence. If $\theta(B)$ is invertible (that is, if θ considered as a polynomial in B has no roots less than or equal to 1 in magnitude), then the model

$$\theta(B)Z_t = \phi(B)a_t$$

is also a valid ARMA(q,p) model. This model is sometimes referred to as the dual model. The autocorrelation function (ACF) of this dual model is called the inverse autocorrelation function (IACF) of the original model.

Notice that if the original model is a pure autoregressive model, then the IACF is an ACF that corresponds to a pure moving-average model. Thus, it cuts off sharply when the lag is greater than p; this behavior is similar to the behavior of the partial autocorrelation function (PACF).

The sample inverse autocorrelation function (SIACF) is estimated in the ARIMA procedure by the following steps. A high-order autoregressive model is fit to the data by means of the Yule-Walker equations. The order of the autoregressive model used to calculate the SIACF is the minimum of the NLAG= value and one-half the number of observations after differencing. The SIACF is then calculated as the autocorrelation function that corresponds to this autoregressive operator when treated as a moving-average operator. That is, the autoregressive coefficients are convolved with themselves and treated as autocovariances.

Under certain conditions, the sampling distribution of the SIACF can be approximated by the sampling distribution of the SACF of the dual model (Bhansali 1980). In the plots generated by ARIMA, the confidence limit marks (.) are located at $\pm 2/\sqrt{n}$. These limits bound an approximate 95% confidence interval for the hypothesis that the data are from a white noise process.

The Partial Autocorrelation Function

The approximation for a standard error for the estimated partial autocorrelation function at lag k is based on a null hypothesis that a pure autoregressive Gaussian process of order k–1 generated the time series. This standard error is $1/\sqrt{n}$ and is used to produce the approximate 95% confidence intervals depicted by the dots in the plot.

The Cross-Correlation Function

The autocorrelation and partial and inverse autocorrelation functions described in the preceding sections help when you want to model a series as a function of its past values and past random errors. When you want to

include the effects of past and current values of other series in the model, the correlations of the response series and the other series must be considered.

The CROSSCORR= option in the IDENTIFY statement computes cross-correlations of the VAR= series with other series and makes these series available for use as inputs in models specified by later ESTIMATE statements.

When the CROSSCORR= option is used, PROC ARIMA prints a plot of the cross-correlation function for each variable in the CROSSCORR= list. This plot is similar in format to the other correlation plots, but it shows the correlation between the two series at both lags and leads. For example,

```
identify var=y crosscorr=x ...;
```

plots the cross-correlation function of Y and X, $\mathrm{Cor}(y_t, x_{t-s})$, for $s = -L$ to L, where L is the value of the NLAG= option. Study of the cross-correlation functions can indicate the transfer functions through which the input series should enter the model for the response series.

The cross-correlation function is computed after any specified differencing has been done. If differencing is specified for the VAR= variable or for a variable in the CROSSCORR= list, it is the differenced series that is cross-correlated (and the differenced series is processed by any following ESTIMATE statement).

For example,

```
identify var=y(1) crosscorr=x(1);
```

computes the cross-correlations of the changes in Y with the changes in X. When differencing is specified, the subsequent ESTIMATE statement models changes in the variables rather than the variables themselves.

The ESACF Method

The extended sample autocorrelation function (ESACF) method can tentatively identify the orders of a *stationary or nonstationary* ARMA process based on iterated least squares estimates of the autoregressive parameters. Tsay and Tiao (1984) proposed the technique, and Choi (1992) provides useful descriptions of the algorithm.

Given a stationary or nonstationary time series $\{z_t : 1 \leq t \leq n\}$ with mean corrected form $\tilde{z}_t = z_t - \mu_z$ with a true autoregressive order of $p + d$ and with a true moving-average order of q, you can use the ESACF method to estimate the unknown orders $p + d$ and q by analyzing the autocorrelation functions associated with filtered series of the form

$$w_t^{(m,j)} = \hat{\Phi}_{(m,j)}(B)\tilde{z}_t = \tilde{z}_t - \sum_{i=1}^{m} \hat{\phi}_i^{(m,j)} \tilde{z}_{t-i}$$

where B represents the backshift operator, where $m = p_{min}, \ldots, p_{max}$ are the autoregressive *test* orders, where $j = q_{min} + 1, \ldots, q_{max} + 1$ are the moving-average *test* orders, and where $\hat{\phi}_i^{(m,j)}$ are the autoregressive parameter estimates under the assumption that the series is an ARMA(m, j) process.

For purely autoregressive models ($j = 0$), ordinary least squares (OLS) is used to consistently estimate $\hat{\phi}_i^{(m,0)}$. For ARMA models, consistent estimates are obtained by the iterated least squares recursion formula, which is initiated by the pure autoregressive estimates:

$$\hat{\phi}_i^{(m,j)} = \hat{\phi}_i^{(m+1,j-1)} - \hat{\phi}_{i-1}^{(m,j-1)} \frac{\hat{\phi}_{m+1}^{(m+1,j-1)}}{\hat{\phi}_m^{(m,j-1)}}$$

The jth lag of the sample autocorrelation function of the filtered series $w_t^{(m,j)}$ is the *extended sample autocorrelation function*, and it is denoted as $r_{j(m)} = r_j(w^{(m,j)})$.

The standard errors of $r_{j(m)}$ are computed in the usual way by using Bartlett's approximation of the variance of the sample autocorrelation function, $var(r_{j(m)}) \approx (1 + \sum_{t=1}^{j-1} r_j^2(w^{(m,j)}))$.

If the true model is an ARMA $(p + d, q)$ process, the filtered series $w_t^{(m,j)}$ follows an MA(q) model for $j \geq q$ so that

$$r_{j(p+d)} \approx 0 \quad j > q$$

$$r_{j(p+d)} \neq 0 \quad j = q$$

Additionally, Tsay and Tiao (1984) show that the extended sample autocorrelation satisfies

$$r_{j(m)} \approx 0 \quad j - q > m - p - d \leq 0$$

$$r_{j(m)} \neq c(m - p - d, j - q) \quad 0 \leq j - q \leq m - p - d$$

where $c(m - p - d, j - q)$ is a nonzero constant or a continuous random variable bounded by –1 and 1.

An ESACF table is then constructed by using the $r_{j(m)}$ for $m = p_{min}, \ldots, p_{max}$ and $j = q_{min} + 1, \ldots, q_{max} + 1$ to identify the ARMA orders (see Table 8.4). The orders are tentatively identified by finding a right (maximal) triangular pattern with vertices located at $(p + d, q)$ and $(p + d, q_{max})$ and in which all elements are insignificant (based on asymptotic normality of the autocorrelation function). The vertex $(p + d, q)$ identifies the order. Table 8.5 depicts the theoretical pattern associated with an ARMA(1,2) series.

Table 8.4 ESACF Table

	MA					
AR	0	1	2	3	·	·
0	$r_{1(0)}$	$r_{2(0)}$	$r_{3(0)}$	$r_{4(0)}$	·	·
1	$r_{1(1)}$	$r_{2(1)}$	$r_{3(1)}$	$r_{4(1)}$	·	·
2	$r_{1(2)}$	$r_{2(2)}$	$r_{3(2)}$	$r_{4(2)}$	·	·
3	$r_{1(3)}$	$r_{2(3)}$	$r_{3(3)}$	$r_{4(3)}$	·	·
·	·	·	·	·	·	·
·	·	·	·	·	·	·

Table 8.5 Theoretical ESACF Table for an ARMA(1,2) Series

	MA							
AR	0	1	2	3	4	5	6	7
0	*	X	X	X	X	X	X	X
1	*	X	0	0	0	0	0	0
2	*	X	X	0	0	0	0	0
3	*	X	X	X	0	0	0	0
4	*	X	X	X	X	0	0	0
	X = significant terms 0 = insignificant terms * = no pattern							

The MINIC Method

The minimum information criterion (MINIC) method can tentatively identify the order of a *stationary and invertible* ARMA process. Note that Hannan and Rissanen (1982) proposed this method; for useful descriptions of the algorithm, see: Box, Jenkins, and Reinsel (1994); Choi (1992).

Given a stationary and invertible time series $\{z_t : 1 \leq t \leq n\}$ with mean corrected form $\tilde{z}_t = z_t - \mu_z$ with a true autoregressive order of p and with a true moving-average order of q, you can use the MINIC method to compute information criteria (or penalty functions) for various autoregressive and moving average orders. The following paragraphs provide a brief description of the algorithm.

If the series is a stationary and invertible ARMA(p, q) process of the form

$$\Phi_{(p,q)}(B)\tilde{z}_t = \Theta_{(p,q)}(B)\epsilon_t$$

the error series can be approximated by a high-order AR process

$$\hat{\epsilon}_t = \hat{\Phi}_{(p_\epsilon,q)}(B)\tilde{z}_t \approx \epsilon_t$$

where the parameter estimates $\hat{\Phi}_{(p_\epsilon,q)}$ are obtained from the Yule-Walker estimates. The choice of the autoregressive order p_ϵ is determined by the order that minimizes the Akaike information criterion (AIC) in the range $p_{\epsilon,min} \leq p_\epsilon \leq p_{\epsilon,max}$

$$AIC(p_\epsilon, 0) = \ln(\tilde{\sigma}^2_{(p_\epsilon,0)}) + 2(p_\epsilon + 0)/n$$

where

$$\tilde{\sigma}^2_{(p_\epsilon,0)} = \frac{1}{n}\sum_{t=p_\epsilon+1}^{n} \hat{\epsilon}_t^2$$

Note that Hannan and Rissanen (1982) use the Bayesian information criterion (BIC) to determine the autoregressive order used to estimate the error series while others recommend the AIC (Box, Jenkins, and Reinsel 1994; Choi 1992).

Once the error series has been estimated for autoregressive test order $m = p_{min}, \ldots, p_{max}$ and for moving-average test order $j = q_{min}, \ldots, q_{max}$, the OLS estimates $\hat{\Phi}_{(m,j)}$ and $\hat{\Theta}_{(m,j)}$ are computed from the regression model

$$\tilde{z}_t = \sum_{i=1}^{m} \phi_i^{(m,j)} \tilde{z}_{t-i} + \sum_{k=1}^{j} \theta_k^{(m,j)} \hat{\epsilon}_{t-k} + error$$

From the preceding parameter estimates, the BIC is then computed

$$BIC(m, j) = \ln(\tilde{\sigma}^2_{(m,j)}) + 2(m + j)\ln(n)/n$$

where

$$\tilde{\sigma}^2_{(m,j)} = \frac{1}{n} \sum_{t=t_0}^{n} \left(\tilde{z}_t - \sum_{i=1}^{m} \phi_i^{(m,j)} \tilde{z}_{t-i} + \sum_{k=1}^{j} \theta_k^{(m,j)} \hat{\epsilon}_{t-k} \right)$$

where $t_0 = p_\epsilon + \max(m, j)$.

A MINIC table is then constructed using $BIC(m, j)$; see Table 8.6. If $p_{max} > p_{\epsilon,min}$, the preceding regression might fail due to linear dependence on the estimated error series and the mean-corrected series. Values of $BIC(m, j)$ that cannot be computed are set to missing. For large autoregressive and moving-average test orders with relatively few observations, a nearly perfect fit can result. This condition can be identified by a large negative $BIC(m, j)$ value.

Table 8.6 MINIC Table

	MA					
AR	0	1	2	3	·	·
0	$BIC(0,0)$	$BIC(0,1)$	$BIC(0,2)$	$BIC(0,3)$	·	·
1	$BIC(1,0)$	$BIC(1,1)$	$BIC(1,2)$	$BIC(1,3)$	·	·
2	$BIC(2,0)$	$BIC(2,1)$	$BIC(2,2)$	$BIC(2,3)$	·	·
3	$BIC(3,0)$	$BIC(3,1)$	$BIC(3,2)$	$BIC(3,3)$	·	·
·	·	·	·	·	·	·
·	·	·	·	·	·	·

The SCAN Method

The smallest canonical (SCAN) correlation method can tentatively identify the orders of a *stationary or nonstationary* ARMA process. Tsay and Tiao (1985) proposed the technique, and for useful descriptions of the algorithm, see: Box, Jenkins, and Reinsel (1994); Choi (1992).

Given a stationary or nonstationary time series $\{z_t : 1 \leq t \leq n\}$ with mean corrected form $\tilde{z}_t = z_t - \mu_z$ with a true autoregressive order of $p + d$ and with a true moving-average order of q, you can use the SCAN method to analyze eigenvalues of the correlation matrix of the ARMA process. The following paragraphs provide a brief description of the algorithm.

For autoregressive test order $m = p_{min}, \ldots, p_{max}$ and for moving-average test order $j = q_{min}, \ldots, q_{max}$, perform the following steps.

1. Let $Y_{m,t} = (\tilde{z}_t, \tilde{z}_{t-1}, \ldots, \tilde{z}_{t-m})'$. Compute the following $(m+1) \times (m+1)$ matrix

$$\begin{aligned}
\hat{\beta}(m, j+1) &= \left(\sum_t Y_{m,t-j-1} Y'_{m,t-j-1}\right)^{-1} \left(\sum_t Y_{m,t-j-1} Y'_{m,t}\right) \\
\hat{\beta}^*(m, j+1) &= \left(\sum_t Y_{m,t} Y'_{m,t}\right)^{-1} \left(\sum_t Y_{m,t} Y'_{m,t-j-1}\right) \\
\hat{A}^*(m, j) &= \hat{\beta}^*(m, j+1)\hat{\beta}(m, j+1)
\end{aligned}$$

   where t ranges from $j + m + 2$ to n.

2. Find the *smallest* eigenvalue, $\hat{\lambda}^*(m, j)$, of $\hat{A}^*(m, j)$ and its corresponding *normalized* eigenvector, $\Phi_{m,j} = (1, -\phi_1^{(m,j)}, -\phi_2^{(m,j)}, \ldots, -\phi_m^{(m,j)})$. The squared canonical correlation estimate is $\hat{\lambda}^*(m, j)$.

3. Using the $\Phi_{m,j}$ as AR(m) coefficients, obtain the residuals for $t = j + m + 1$ to n, by following the formula: $w_t^{(m,j)} = \tilde{z}_t - \phi_1^{(m,j)}\tilde{z}_{t-1} - \phi_2^{(m,j)}\tilde{z}_{t-2} - \ldots - \phi_m^{(m,j)}\tilde{z}_{t-m}$.

4. From the sample autocorrelations of the residuals, $r_k(w)$, approximate the standard error of the squared canonical correlation estimate by

   $$var(\hat{\lambda}^*(m, j)^{1/2}) \approx d(m, j)/(n - m - j)$$

   where $d(m, j) = (1 + 2\sum_{i=1}^{j-1} r_k(w^{(m,j)}))$.

The test statistic to be used as an identification criterion is

$$c(m, j) = -(n - m - j)\ln(1 - \hat{\lambda}^*(m, j)/d(m, j))$$

which is asymptotically χ_1^2 if $m = p + d$ and $j \geq q$ or if $m \geq p + d$ and $j = q$. For $m > p$ and $j < q$, there is more than one theoretical zero canonical correlation between $Y_{m,t}$ and $Y_{m,t-j-1}$. Since the $\hat{\lambda}^*(m, j)$ are the smallest canonical correlations for each (m, j), the percentiles of $c(m, j)$ are less than those of a χ_1^2; therefore, Tsay and Tiao (1985) state that it is safe to assume a χ_1^2. For $m < p$ and $j < q$, no conclusions about the distribution of $c(m, j)$ are made.

A SCAN table is then constructed using $c(m, j)$ to determine which of the $\hat{\lambda}^*(m, j)$ are significantly different from zero (see Table 8.7). The ARMA orders are tentatively identified by finding a (maximal) rectangular pattern in which the $\hat{\lambda}^*(m, j)$ are insignificant for all test orders $m \geq p + d$ and $j \geq q$. There may be more than one pair of values $(p + d, q)$ that permit such a rectangular pattern. In this case, parsimony and the number of insignificant items in the rectangular pattern should help determine the model order. Table 8.8 depicts the theoretical pattern associated with an ARMA(2,2) series.

Table 8.7 SCAN Table

	MA					
AR	0	1	2	3	·	·
0	$c(0,0)$	$c(0,1)$	$c(0,2)$	$c(0,3)$	·	·
1	$c(1,0)$	$c(1,1)$	$c(1,2)$	$c(1,3)$	·	·
2	$c(2,0)$	$c(2,1)$	$c(2,2)$	$c(2,3)$	·	·
3	$c(3,0)$	$c(3,1)$	$c(3,2)$	$c(3,3)$	·	·
·	·	·	·	·	·	·
·	·	·	·	·	·	·

Table 8.8 Theoretical SCAN Table for an ARMA(2,2) Series

	MA							
AR	0	1	2	3	4	5	6	7
0	*	X	X	X	X	X	X	X
1	*	X	X	X	X	X	X	X
2	*	X	0	0	0	0	0	0
3	*	X	0	0	0	0	0	0
4	*	X	0	0	0	0	0	0
	X = significant terms 0 = insignificant terms * = no pattern							

Stationarity Tests

When a time series has a unit root, the series is nonstationary and the ordinary least squares (OLS) estimator is not normally distributed. Dickey and Fuller studied the limiting distribution of the OLS estimator of autoregressive models for time series with a simple unit root (Dickey 1976; Dickey and Fuller 1979). Dickey, Hasza, and Fuller obtained the limiting distribution for time series with seasonal unit roots (Dickey, Hasza, and Fuller 1984). Hamilton (1994) discusses the various types of unit root testing.

For a description of Dickey-Fuller tests, see the section "PROBDF Function for Dickey-Fuller Tests" on page 187 in Chapter 6. See Chapter 9, "The AUTOREG Procedure," for a description of Phillips-Perron tests.

The random-walk-with-drift test recommends whether or not an integrated times series has a drift term. Hamilton (1994) discusses this test.

Prewhitening

If, as is usually the case, an input series is autocorrelated, the direct cross-correlation function between the input and response series gives a misleading indication of the relation between the input and response series.

One solution to this problem is called *prewhitening*. You first fit an ARIMA model for the input series sufficient to reduce the residuals to white noise; then, filter the input series with this model to get the white noise residual series. You then filter the response series with the same model and cross-correlate the filtered response with the filtered input series.

The ARIMA procedure performs this prewhitening process automatically when you precede the IDENTIFY statement for the response series with IDENTIFY and ESTIMATE statements to fit a model for the input series. If a model with no inputs was previously fit to a variable specified by the CROSSCORR= option, then that model is used to prewhiten both the input series and the response series before the cross-correlations are computed for the input series.

For example,

```
proc arima data=in;
   identify var=x;
   estimate p=1 q=1;
   identify var=y crosscorr=x;
run;
```

Both X and Y are filtered by the ARMA(1,1) model fit to X before the cross-correlations are computed.

Note that prewhitening is done to estimate the cross-correlation function; the unfiltered series are used in any subsequent ESTIMATE or FORECAST statements, and the correlation functions of Y with its own lags are computed from the unfiltered Y series. But initial values in the ESTIMATE statement are obtained with prewhitened data; therefore, the result with prewhitening can be different from the result without prewhitening.

To suppress prewhitening for all input variables, use the CLEAR option in the IDENTIFY statement to make PROC ARIMA disregard all previous models.

Prewhitening and Differencing

If the VAR= and CROSSCORR= options specify differencing, the series are differenced before the prewhitening filter is applied. When the differencing lists specified in the VAR= option for an input and in the CROSSCORR= option for that input are not the same, PROC ARIMA combines the two lists so that the differencing operators used for prewhitening include all differences in either list (in the least common multiple sense).

Identifying Transfer Function Models

When identifying a transfer function model with multiple input variables, the cross-correlation functions can be misleading if the input series are correlated with each other. Any dependencies among two or more input series will confound their cross-correlations with the response series.

The prewhitening technique assumes that the input variables do not depend on past values of the response variable. If there is feedback from the response variable to an input variable, as evidenced by significant cross-correlation at negative lags, both the input and the response variables need to be prewhitened before meaningful cross-correlations can be computed.

PROC ARIMA cannot handle feedback models. The STATESPACE and VARMAX procedures are more appropriate for models with feedback.

Missing Values and Autocorrelations

To compute the sample autocorrelation function when missing values are present, PROC ARIMA uses only crossproducts that do not involve missing values and employs divisors that reflect the number of crossproducts used rather than the total length of the series. Sample partial autocorrelations and inverse autocorrelations are then computed by using the sample autocorrelation function. If necessary, a taper is employed to transform the sample autocorrelations into a positive definite sequence before calculating the partial autocorrelation and inverse correlation functions. The confidence intervals produced for these functions might not be valid when there are missing values. The distributional properties for sample correlation functions are not clear for finite samples. See Dunsmuir (1984) for some asymptotic properties of the sample correlation functions.

Estimation Details

The ARIMA procedure primarily uses the computational methods outlined by Box and Jenkins. Marquardt's method is used for the nonlinear least squares iterations. Numerical approximations of the derivatives of the sum-of-squares function are taken by using a fixed delta (controlled by the DELTA= option).

The methods do not always converge successfully for a given set of data, particularly if the starting values for the parameters are not close to the least squares estimates.

Back-Forecasting

The unconditional sum of squares is computed exactly; thus, back-forecasting is not performed. Early versions of SAS/ETS software used the back-forecasting approximation and allowed a positive value of the BACKLIM= option to control the extent of the back-forecasting. In the current version, requesting a positive number of back-forecasting steps with the BACKLIM= option has no effect.

Preliminary Estimation

If an autoregressive or moving-average operator is specified with no missing lags, preliminary estimates of the parameters are computed by using the autocorrelations computed in the IDENTIFY stage. Otherwise, the preliminary estimates are arbitrarily set to values that produce stable polynomials.

When preliminary estimation is not performed by PROC ARIMA, then initial values of the coefficients for any given autoregressive or moving-average factor are set to 0.1 if the degree of the polynomial associated with the factor is 9 or less. Otherwise, the coefficients are determined by expanding the polynomial $(1-0.1B)$ to an appropriate power by using a recursive algorithm.

These preliminary estimates are the starting values in an iterative algorithm to compute estimates of the parameters.

Estimation Methods

Maximum Likelihood

The METHOD= ML option produces maximum likelihood estimates. The likelihood function is maximized via nonlinear least squares using Marquardt's method. Maximum likelihood estimates are more expensive to compute than the conditional least squares estimates; however, they may be preferable in some cases (Ansley and Newbold 1980; Davidson 1981).

The maximum likelihood estimates are computed as follows. Let the univariate ARMA model be

$$\phi(B)(W_t - \mu_t) = \theta(B)a_t$$

where a_t is an independent sequence of normally distributed innovations with mean 0 and variance σ^2. Here μ_t is the mean parameter μ plus the transfer function inputs. The log-likelihood function can be written as follows:

$$-\frac{1}{2\sigma^2}\mathbf{x}'\mathbf{\Omega}^{-1}\mathbf{x} - \frac{1}{2}\ln(|\mathbf{\Omega}|) - \frac{n}{2}\ln(\sigma^2)$$

In this equation, n is the number of observations, $\sigma^2\mathbf{\Omega}$ is the variance of $\mathbf{x}$ as a function of the ϕ and θ parameters, and $|\mathbf{\Omega}|$ denotes the determinant. The vector $\mathbf{x}$ is the time series W_t minus the structural part of

the model μ_t, written as a column vector, as follows:

$$\mathbf{x} = \begin{bmatrix} W_1 \\ W_2 \\ \vdots \\ W_n \end{bmatrix} - \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{bmatrix}$$

The maximum likelihood estimate (MLE) of σ^2 is

$$s^2 = \frac{1}{n}\mathbf{x}'\mathbf{\Omega}^{-1}\mathbf{x}$$

Note that the default estimator of the variance divides by $n - r$, where r is the number of parameters in the model, instead of by n. Specifying the NODF option causes a divisor of n to be used.

The log-likelihood concentrated with respect to σ^2 can be taken up to additive constants as

$$-\frac{n}{2}\ln(\mathbf{x}'\mathbf{\Omega}^{-1}\mathbf{x}) - \frac{1}{2}\ln(|\mathbf{\Omega}|)$$

Let $\mathbf{H}$ be the lower triangular matrix with positive elements on the diagonal such that $\mathbf{H}\mathbf{H}' = \mathbf{\Omega}$. Let $\mathbf{e}$ be the vector $\mathbf{H}^{-1}\mathbf{x}$. The concentrated log-likelihood with respect to σ^2 can now be written as

$$-\frac{n}{2}\ln(\mathbf{e}'\mathbf{e}) - \ln(|\mathbf{H}|)$$

or

$$-\frac{n}{2}\ln(|\mathbf{H}|^{1/n}\mathbf{e}'\mathbf{e}|\mathbf{H}|^{1/n})$$

The MLE is produced by using a Marquardt algorithm to minimize the following sum of squares:

$$|\mathbf{H}|^{1/n}\mathbf{e}'\mathbf{e}|\mathbf{H}|^{1/n}$$

The subsequent analysis of the residuals is done by using $\mathbf{e}$ as the vector of residuals.

Unconditional Least Squares

The METHOD=ULS option produces unconditional least squares estimates. The ULS method is also referred to as the *exact least squares* (ELS) method. For METHOD=ULS, the estimates minimize

$$\sum_{t=1}^{n} \tilde{a}_t^2 = \sum_{t=1}^{n} (x_t - \mathbf{C}_t\mathbf{V}_t^{-1}(x_1, \cdots, x_{t-1})')^2$$

where $\mathbf{C}_t$ is the covariance matrix of x_t and $(x_1, \cdots, x_{t-1})$, and $\mathbf{V}_t$ is the variance matrix of $(x_1, \cdots, x_{t-1})$. In fact, $\sum_{t=1}^{n} \tilde{a}_t^2$ is the same as $\mathbf{x}'\mathbf{\Omega}^{-1}\mathbf{x}$, and hence $\mathbf{e}'\mathbf{e}$. Therefore, the unconditional least squares estimates are obtained by minimizing the sum of squared residuals rather than using the log-likelihood as the criterion function.

Conditional Least Squares

The METHOD=CLS option produces conditional least squares estimates. The CLS estimates are conditional on the assumption that the past unobserved errors are equal to 0. The series x_t can be represented in terms of the previous observations, as follows:

$$x_t = a_t + \sum_{i=1}^{\infty} \pi_i x_{t-i}$$

The π weights are computed from the ratio of the ϕ and θ polynomials, as follows:

$$\frac{\phi(B)}{\theta(B)} = 1 - \sum_{i=1}^{\infty} \pi_i B^i$$

The CLS method produces estimates minimizing

$$\sum_{t=1}^{n} \hat{a}_t^2 = \sum_{t=1}^{n} (x_t - \sum_{i=1}^{\infty} \hat{\pi}_i x_{t-i})^2$$

where the unobserved past values of x_t are set to 0 and $\hat{\pi}_i$ are computed from the estimates of ϕ and θ at each iteration.

For METHOD=ULS and METHOD=ML, initial estimates are computed using the METHOD=CLS algorithm.

Start-up for Transfer Functions

When computing the noise series for transfer function and intervention models, the start-up for the transferred variable is done by assuming that past values of the input series are equal to the first value of the series. The estimates are then obtained by applying least squares or maximum likelihood to the noise series. Thus, for transfer function models, the ML option does not generate the full (multivariate ARMA) maximum likelihood estimates, but it uses only the univariate likelihood function applied to the noise series.

Because PROC ARIMA uses all of the available data for the input series to generate the noise series, other start-up options for the transferred series can be implemented by prefixing an observation to the beginning of the real data. For example, if you fit a transfer function model to the variable Y with the single input X, then you can employ a start-up using 0 for the past values by prefixing to the actual data an observation with a missing value for Y and a value of 0 for X.

Information Criteria

PROC ARIMA computes and prints two information criteria, Akaike's information criterion (AIC) (Akaike 1974; Harvey 1981) and Schwarz's Bayesian criterion (SBC) (Schwarz 1978). The AIC and SBC are used to compare competing models fit to the same series. The model with the smaller information criteria is said to fit the data better. The AIC is computed as

$$-2\ln(L) + 2k$$

where L is the likelihood function and k is the number of free parameters. The SBC is computed as

$$-2\ln(L) + \ln(n)k$$

where n is the number of residuals that can be computed for the time series. Sometimes Schwarz's Bayesian criterion is called the Bayesian information criterion (BIC).

If METHOD=CLS is used to do the estimation, an approximation value of L is used, where L is based on the conditional sum of squares instead of the exact sum of squares, and a Jacobian factor is left out.

Tests of Residuals

A table of test statistics for the hypothesis that the model residuals are white noise is printed as part of the ESTIMATE statement output. The chi-square statistics used in the test for lack of fit are computed using the Ljung-Box formula

$$\chi^2_m = n(n+2)\sum_{k=1}^{m}\frac{r_k^2}{(n-k)}$$

where

$$r_k = \frac{\sum_{t=1}^{n-k} a_t a_{t+k}}{\sum_{t=1}^{n} a_t^2}$$

and a_t is the residual series.

This formula has been suggested by Ljung and Box (1978) as yielding a better fit to the asymptotic chi-square distribution than the Box-Pierce Q statistic. Some simulation studies of the finite sample properties of this statistic are given by Davies, Triggs, and Newbold (1977); Ljung and Box (1978). When the time series has missing values, Stoffer and Toloi (1992) suggest a modification of this test statistic that has improved distributional properties over the standard Ljung-Box formula given above. When the series contains missing values, this modified test statistic is used by default.

Each chi-square statistic is computed for all lags up to the indicated lag value and is not independent of the preceding chi-square values. The null hypotheses tested is that the current set of autocorrelations is white noise.

t-values

The t values reported in the table of parameter estimates are approximations whose accuracy depends on the validity of the model, the nature of the model, and the length of the observed series. When the length of the observed series is short and the number of estimated parameters is large with respect to the series length, the t approximation is usually poor. Probability values that correspond to a t distribution should be interpreted carefully because they may be misleading.

Cautions during Estimation

The ARIMA procedure uses a general nonlinear least squares estimation method that can yield problematic results if your data do not fit the model. Output should be examined carefully. The GRID option can be used to ensure the validity and quality of the results. Problems you might encounter include the following:

- Preliminary moving-average estimates might not converge. If this occurs, preliminary estimates are derived as described previously in "Preliminary Estimation" on page 272. You can supply your own preliminary estimates with the ESTIMATE statement options.

- The estimates can lead to an unstable time series process, which can cause extreme forecast values or overflows in the forecast.
- The Jacobian matrix of partial derivatives might be singular; usually, this happens because not all the parameters are identifiable. Removing some of the parameters or using a longer time series might help.
- The iterative process might not converge. PROC ARIMA's estimation method stops after *n* iterations, where *n* is the value of the MAXITER= option. If an iteration does not improve the SSE, the Marquardt parameter is increased by a factor of ten until parameters that have a smaller SSE are obtained or until the limit value of the Marquardt parameter is exceeded.
- For METHOD=CLS, the estimates might converge but not to least squares estimates. The estimates might converge to a local minimum, the numerical calculations might be distorted by data whose sum-of-squares surface is not smooth, or the minimum might lie outside the region of invertibility or stationarity.
- If the data are differenced and a moving-average model is fit, the parameter estimates might try to converge exactly on the invertibility boundary. In this case, the standard error estimates that are based on derivatives might be inaccurate.

Specifying Inputs and Transfer Functions

Input variables and transfer functions for them can be specified using the INPUT= option in the ESTIMATE statement. The variables used in the INPUT= option must be included in the CROSSCORR= list in the previous IDENTIFY statement. If any differencing is specified in the CROSSCORR= list, then the differenced variable is used as the input to the transfer function.

General Syntax of the INPUT= Option

The general syntax of the INPUT= option is

ESTIMATE . . . INPUT=*(transfer-function variable . . .)*

The transfer function for an input variable is optional. The name of a variable by itself can be used to specify a pure regression term for the variable.

If specified, the syntax of the transfer function is

$S \$ (L_{1,1}, L_{1,2}, \ldots)(L_{2,1}, \ldots)\ldots/(L_{i,1}, L_{i,2}, \ldots)(L_{i+1,1}, \ldots)\ldots$

S is the number of periods of time delay (lag) for this input series. Each term in parentheses specifies a polynomial factor with parameters at the lags specified by the $L_{i,j}$ values. The terms before the slash (/) are numerator factors. The terms after the slash (/) are denominator factors. All three parts are optional.

Commas can optionally be used between input specifications to make the INPUT= option more readable. The $ sign after the shift is also optional.

Except for the first numerator factor, each of the terms $L_{i,1}, L_{i,2}, \ldots, L_{i,k}$ indicates a factor of the form

$$(1 - \omega_{i,1}B^{L_{i,1}} - \omega_{i,2}B^{L_{i,2}} - \ldots - \omega_{i,k}B^{L_{i,k}})$$

The form of the first numerator factor depends on the ALTPARM option. By default, the constant 1 in the first numerator factor is replaced with a free parameter ω_0.

Alternative Model Parameterization

When the ALTPARM option is specified, the ω_0 parameter is factored out so that it multiplies the entire transfer function, and the first numerator factor has the same form as the other factors.

The ALTPARM option does not materially affect the results; it just presents the results differently. Some people prefer to see the model written one way, while others prefer the alternative representation. Table 8.9 illustrates the effect of the ALTPARM option.

Table 8.9 The ALTPARM Option

INPUT= Option	ALTPARM	Model
INPUT=((1 2)(12)/(1)X);	No	$(\omega_0 - \omega_1 B - \omega_2 B^2)(1 - \omega_3 B^{12})/(1 - \delta_1 B)X_t$
	Yes	$\omega_0(1 - \omega_1 B - \omega_2 B^2)(1 - \omega_3 B^{12})/(1 - \delta_1 B)X_t$

Differencing and Input Variables

If you difference the response series and use input variables, take care that the differencing operations do not change the meaning of the model. For example, if you want to fit the model

$$Y_t = \frac{\omega_0}{(1 - \delta_1 B)} X_t + \frac{(1 - \theta_1 B)}{(1 - B)(1 - B^{12})} a_t$$

then the IDENTIFY statement must read

```
identify var=y(1,12) crosscorr=x(1,12);
estimate q=1 input=(/(1)x) noconstant;
```

If instead you specify the differencing as

```
identify var=y(1,12) crosscorr=x;
estimate q=1 input=(/(1)x) noconstant;
```

then the model being requested is

$$Y_t = \frac{\omega_0}{(1 - \delta_1 B)(1 - B)(1 - B^{12})} X_t + \frac{(1 - \theta_1 B)}{(1 - B)(1 - B^{12})} a_t$$

which is a very different model.

The point to remember is that a differencing operation requested for the response variable specified by the VAR= option is applied only to that variable and not to the noise term of the model.

Initial Values

The syntax for giving initial values to transfer function parameters in the INITVAL= option parallels the syntax of the INPUT= option. For each transfer function in the INPUT= option, the INITVAL= option should

give an initialization specification followed by the input series name. The initialization specification for each transfer function has the form

$$C \$ (V_{1,1}, V_{1,2}, \ldots)(V_{2,1}, \ldots)\ldots/(V_{i,1}, \ldots)\ldots$$

where C is the lag 0 term in the first numerator factor of the transfer function (or the overall scale factor if the ALTPARM option is specified) and $V_{i,j}$ is the coefficient of the $L_{i,j}$ element in the transfer function.

To illustrate, suppose you want to fit the model

$$Y_t = \mu + \frac{(\omega_0 - \omega_1 B - \omega_2 B^2)}{(1 - \delta_1 B - \delta_2 B^2 - \delta_3 B^3)} X_{t-3} + \frac{1}{(1 - \phi_1 B - \phi_2 B^3)} a_t$$

and start the estimation process with the initial values μ=10, ω_0=1, ω_1=0.5, ω_2=0.03, δ_1=0.8, δ_2=–0.1, δ_3=0.002, ϕ_1=0.1, ϕ_2=0.01. (These are arbitrary values for illustration only.) You would use the following statements:

```
identify var=y crosscorr=x;
estimate p=(1,3) input=(3$(1,2)/(1,2,3)x)
         mu=10 ar=.1 .01
         initval=(1$(.5,.03)/(.8,-.1,.002)x);
```

Note that the lags specified for a particular factor are sorted, so initial values should be given in sorted order. For example, if the P= option had been entered as P=(3,1) instead of P=(1,3), the model would be the same and so would the AR= option. Sorting is done within all factors, including transfer function factors, so initial values should always be given in order of increasing lags.

Here is another illustration, showing initialization for a factored model with multiple inputs. The model is

$$\begin{aligned} Y_t = \mu \quad &+ \quad \frac{\omega_{1,0}}{(1 - \delta_{1,1} B)} W_t + (\omega_{2,0} - \omega_{2,1} B) X_{t-3} \\ &+ \quad \frac{1}{(1 - \phi_1 B)(1 - \phi_2 B^6 - \phi_3 B^{12})} a_t \end{aligned}$$

and the initial values are μ=10, $\omega_{1,0}$=5, $\delta_{1,1}$=0.8, $\omega_{2,0}$=1, $\omega_{2,1}$=0.5, ϕ_1=0.1, ϕ_2=0.05, and ϕ_3=0.01. You would use the following statements:

```
identify var=y crosscorr=(w x);
estimate p=(1)(6,12) input=(/(1)w, 3$(1)x)
         mu=10 ar=.1 .05 .01
         initval=(5$/(.8)w 1$(.5)x);
```

Stationarity and Invertibility

By default, PROC ARIMA requires that the parameter estimates for the AR and MA parts of the model always remain in the stationary and invertible regions, respectively. The NOSTABLE option removes this restriction and for high-order models can save some computer time. Note that using the NOSTABLE option does not necessarily result in an unstable model being fit, since the estimates can leave the stable region for some iterations but still ultimately converge to stable values. Similarly, by default, the parameter estimates for the denominator polynomial of the transfer function part of the model are also restricted to be stable. The NOTFSTABLE option can be used to remove this restriction.

Naming of Model Parameters

In the table of parameter estimates produced by the ESTIMATE statement, model parameters are referred to by using the naming convention described in this section.

The parameters in the noise part of the model are named as AR*i,j* or MA*i,j*, where AR refers to autoregressive parameters and MA to moving-average parameters. The subscript *i* refers to the particular polynomial factor, and the subscript *j* refers to the *j*th term within the *i*th factor. These terms are sorted in order of increasing lag within factors, so the subscript *j* refers to the *j*th term after sorting.

When inputs are used in the model, the parameters of each transfer function are named NUM*i,j* and DEN*i,j*. The *j*th term in the *i*th factor of a numerator polynomial is named NUM*i,j*. The *j*th term in the *i*th factor of a denominator polynomial is named DEN*i,j*.

This naming process is repeated for each input variable, so if there are multiple inputs, parameters in transfer functions for different input series have the same name. The table of parameter estimates shows in the "Variable" column the input with which each parameter is associated. The parameter name shown in the "Parameter" column and the input variable name shown in the "Variable" column must be combined to fully identify transfer function parameters.

The lag 0 parameter in the first numerator factor for the first input variable is named NUM1. For subsequent input variables, the lag 0 parameter in the first numerator factor is named NUM*k*, where *k* is the position of the input variable in the INPUT= option list. If the ALTPARM option is specified, the NUM*k* parameter is replaced by an overall scale parameter named SCALE*k*.

For the mean and noise process parameters, the response series name is shown in the "Variable" column. The lag and shift for each parameter are also shown in the table of parameter estimates when inputs are used.

Missing Values and Estimation and Forecasting

Estimation and forecasting are carried out in the presence of missing values by forecasting the missing values with the current set of parameter estimates. The maximum likelihood algorithm employed was suggested by Jones (1980) and is used for both unconditional least squares (ULS) and maximum likelihood (ML) estimation.

The CLS algorithm simply fills in missing values with infinite memory forecast values, computed by forecasting ahead from the nonmissing past values as far as required by the structure of the missing values. These artificial values are then employed in the nonmissing value CLS algorithm. Artificial values are updated at each iteration along with parameter estimates.

For models with input variables, embedded missing values (that is, missing values other than at the beginning or end of the series) are not generally supported. Embedded missing values in input variables are supported for the special case of a multiple regression model that has ARIMA errors. A multiple regression model is specified by an INPUT= option that simply lists the input variables (possibly with lag shifts) without any numerator or denominator transfer function factors. One-step-ahead forecasts are not available for the response variable when one or more of the input variables have missing values.

When embedded missing values are present for a model with complex transfer functions, PROC ARIMA uses the first continuous nonmissing piece of each series to do the analysis. That is, PROC ARIMA skips observations at the beginning of each series until it encounters a nonmissing value and then uses the data from

there until it encounters another missing value or until the end of the data is reached. This makes the current version of PROC ARIMA compatible with earlier releases that did not allow embedded missing values.

Forecasting Details

If the model has input variables, a forecast beyond the end of the data for the input variables is possible only if univariate ARIMA models have previously been fit to the input variables or future values for the input variables are included in the DATA= data set.

If input variables are used, the forecast standard errors and confidence limits of the response depend on the estimated forecast error variance of the predicted inputs. If several input series are used, the forecast errors for the inputs should be independent; otherwise, the standard errors and confidence limits for the response series will not be accurate. If future values for the input variables are included in the DATA= data set, the standard errors of the forecasts will be underestimated since these values are assumed to be known with certainty.

The forecasts are generated using forecasting equations consistent with the method used to estimate the model parameters. Thus, the estimation method specified in the ESTIMATE statement also controls the way forecasts are produced by the FORECAST statement. If METHOD=CLS is used, the forecasts are *infinite memory forecasts*, also called *conditional forecasts*. If METHOD=ULS or METHOD=ML, the forecasts are *finite memory forecasts*, also called *unconditional forecasts*. A complete description of the steps to produce the series forecasts and their standard errors by using either of these methods is quite involved, and only a brief explanation of the algorithm is given in the next two sections. Additional details about the finite and infinite memory forecasts can be found in Brockwell and Davis (1991). The prediction of stationary ARMA processes is explained in Chapter 5, and the prediction of nonstationary ARMA processes is given in Chapter 9 of Brockwell and Davis (1991).

Infinite Memory Forecasts

If METHOD=CLS is used, the forecasts are *infinite memory forecasts*, also called *conditional forecasts*. The term *conditional* is used because the forecasts are computed by assuming that the unknown values of the response series before the start of the data are equal to the mean of the series. Thus, the forecasts are conditional on this assumption.

The series x_t can be represented as

$$x_t = a_t + \sum_{i=1}^{\infty} \pi_i x_{t-i}$$

where $\phi(B)/\theta(B) = 1 - \sum_{i=1}^{\infty} \pi_i B^i$.

The k -step forecast of x_{t+k} is computed as

$$\hat{x}_{t+k} = \sum_{i=1}^{k-1} \hat{\pi}_i \hat{x}_{t+k-i} + \sum_{i=k}^{\infty} \hat{\pi}_i x_{t+k-i}$$

where unobserved past values of x_t are set to zero and $\hat{\pi}_i$ is obtained from the estimated parameters $\hat{\phi}$ and $\hat{\theta}$.

Finite Memory Forecasts

For METHOD=ULS or METHOD=ML, the forecasts are *finite memory forecasts*, also called *unconditional forecasts*. For finite memory forecasts, the covariance function of the ARMA model is used to derive the best linear prediction equation.

That is, the k-step forecast of x_{t+k}, given $(x_1, \cdots, x_{t-1})$, is

$$\tilde{x}_{t+k} = \mathbf{C}_{k,t}\mathbf{V}_t^{-1}(x_1, \cdots, x_{t-1})'$$

where $\mathbf{C}_{k,t}$ is the covariance of x_{t+k} and $(x_1, \cdots, x_{t-1})$ and $\mathbf{V}_t$ is the covariance matrix of the vector $(x_1, \cdots, x_{t-1})$. $\mathbf{C}_{k,t}$ and $\mathbf{V}_t$ are derived from the estimated parameters.

Finite memory forecasts minimize the mean squared error of prediction if the parameters of the ARMA model are known exactly. (In most cases, the parameters of the ARMA model are estimated, so the predictors are not true best linear forecasts.)

If the response series is differenced, the final forecast is produced by summing the forecast of the differenced series. This summation and the forecast are conditional on the initial values of the series. Thus, when the response series is differenced, the final forecasts are not true finite memory forecasts because they are derived by assuming that the differenced series begins in a steady-state condition. Thus, they fall somewhere between finite memory and infinite memory forecasts. In practice, there is seldom any practical difference between these forecasts and true finite memory forecasts.

Forecasting Log Transformed Data

The log transformation is often used to convert time series that are nonstationary with respect to the innovation variance into stationary time series. The usual approach is to take the log of the series in a DATA step and then apply PROC ARIMA to the transformed data. A DATA step is then used to transform the forecasts of the logs back to the original units of measurement. The confidence limits are also transformed by using the exponential function.

As one alternative, you can simply exponentiate the forecast series. This procedure gives a forecast for the median of the series, but the antilog of the forecast log series underpredicts the mean of the original series. If you want to predict the expected value of the series, you need to take into account the standard error of the forecast, as shown in the following example, which uses an AR(2) model to forecast the log of a series Y:

```
data in;
   set in;
   ylog = log( y );
run;

proc arima data=in;
   identify var=ylog;
   estimate p=2;
   forecast lead=10 out=out;
run;

data out;
   set out;
   y   = exp( ylog );
```

```
   l95 = exp( l95 );
   u95 = exp( u95 );
   forecast = exp( forecast + std*std/2 );
run;
```

Specifying Series Periodicity

The INTERVAL= option is used together with the ID= variable to describe the observations that make up the time series. For example, INTERVAL=MONTH specifies a monthly time series in which each observation represents one month. See Chapter 5, "Date Intervals, Formats, and Functions," for details about the interval values supported.

The variable specified by the ID= option in the PROC ARIMA statement identifies the time periods associated with the observations. Usually, SAS date, time, or datetime values are used for this variable. PROC ARIMA uses the ID= variable in the following ways:

- to validate the data periodicity. When the INTERVAL= option is specified, PROC ARIMA uses the ID variable to check the data and verify that successive observations have valid ID values that correspond to successive time intervals. When the INTERVAL= option is not used, PROC ARIMA verifies that the ID values are nonmissing and in ascending order.
- to check for gaps in the input observations. For example, if INTERVAL=MONTH and an input observation for April 1970 follows an observation for January 1970, there is a gap in the input data with two omitted observations (namely February and March 1970). A warning message is printed when a gap in the input data is found.
- to label the forecast observations in the output data set. PROC ARIMA extrapolates the values of the ID variable for the forecast observations from the ID value at the end of the input data according to the frequency specifications of the INTERVAL= option. If the INTERVAL= option is not specified, PROC ARIMA extrapolates the ID variable by incrementing the ID variable value for the last observation in the input data by 1 for each forecast period. Values of the ID variable over the range of the input data are copied to the output data set.

The ALIGN= option is used to align the ID variable to the beginning, middle, or end of the time ID interval specified by the INTERVAL= option.

Detecting Outliers

You can use the OUTLIER statement to detect changes in the level of the response series that are not accounted for by the estimated model. The types of changes considered are additive outliers (AO), level shifts (LS), and temporary changes (TC).

Let η_t be a regression variable that describes some type of change in the mean response. In time series literature η_t is called a shock signature. An additive outlier at some time point s corresponds to a shock signature η_t such that $\eta_s = 1.0$ and η_t is 0.0 at all other points. Similarly a permanent level shift that

originates at time s has a shock signature such that η_t is 0.0 for $t < s$ and 1.0 for $t \geq s$. A temporary level shift of duration d that originates at time s has η_t equal to 1.0 between s and $s + d$ and 0.0 otherwise.

Suppose that you are estimating the ARIMA model

$$D(B)Y_t = \mu_t + \frac{\theta(B)}{\phi(B)}a_t$$

where Y_t is the response series, $D(B)$ is the differencing polynomial in the backward shift operator B (possibly identity), μ_t is the transfer function input, $\phi(B)$ and $\theta(B)$ are the AR and MA polynomials, respectively, and a_t is the Gaussian white noise series.

The problem of detection of level shifts in the OUTLIER statement is formulated as a problem of sequential selection of shock signatures that improve the model in the ESTIMATE statement. This is similar to the forward selection process in the stepwise regression procedure. The selection process starts with considering shock signatures of the type specified in the TYPE= option, originating at each nonmissing measurement. This involves testing $H_0: \beta = 0$ versus $H_a: \beta \neq 0$ in the model

$$D(B)(Y_t - \beta\eta_t) = \mu_t + \frac{\theta(B)}{\phi(B)}a_t$$

for each of these shock signatures. The most significant shock signature, if it also satisfies the significance criterion in ALPHA= option, is included in the model. If no significant shock signature is found, then the outlier detection process stops; otherwise this augmented model, which incorporates the selected shock signature in its transfer function input, becomes the null model for the subsequent selection process. This iterative process stops if at any stage no more significant shock signatures are found or if the number of iterations exceeds the maximum search number that results due to the MAXNUM= and MAXPCT= settings. In all these iterations, the parameters of the ARIMA model in the ESTIMATE statement are held fixed.

The precise details of the testing procedure for a given shock signature η_t are as follows:

The preceding testing problem is equivalent to testing $H_0: \beta = 0$ versus $H_a: \beta \neq 0$ in the following "regression with ARMA errors" model

$$N_t = \beta\zeta_t + \frac{\theta(B)}{\phi(B)}a_t$$

where $N_t = (D(B)Y_t - \mu_t)$ is the "noise" process and $\zeta_t = D(B)\eta_t$ is the "effective" shock signature.

In this setting, under H_0, $N = (N_1, N_2, \ldots, N_n)^T$ is a mean zero Gaussian vector with variance covariance matrix $\sigma^2\mathbf{\Omega}$. Here σ^2 is the variance of the white noise process a_t and $\mathbf{\Omega}$ is the variance-covariance matrix associated with the ARMA model. Moreover, under H_a, N has $\beta\zeta$ as the mean vector where $\zeta = (\zeta_1, \zeta_2, \ldots, \zeta_n)^T$. Additionally, the generalized least squares estimate of β and its variance is given by

$$\begin{aligned} \hat{\beta} &= \delta/\kappa \\ \mathrm{Var}(\hat{\beta}) &= \sigma^2/\kappa \end{aligned}$$

where $\delta = \zeta^T\mathbf{\Omega}^{-1}N$ and $\kappa = \zeta^T\mathbf{\Omega}^{-1}\zeta$. The test statistic $\tau^2 = \delta^2/(\sigma^2\kappa)$ is used to test the significance of β, which has an approximate chi-squared distribution with 1 degree of freedom under H_0. The type of estimate of σ^2 used in the calculation of τ^2 can be specified by the SIGMA= option. The default setting is SIGMA=ROBUST, which corresponds to a robust estimate suggested in an outlier detection procedure in

X-12-ARIMA, the Census Bureau's time series analysis program; see Findley et al. (1998) for additional information. The robust estimate of σ^2 is computed by the formula

$$\hat{\sigma}^2 = (1.49 \times \text{Median}(|\hat{a}_t|))^2$$

where $\hat{a}_t$ are the standardized residuals of the null ARIMA model. The setting SIGMA=MSE corresponds to the usual mean squared error estimate (MSE) computed the same way as in the ESTIMATE statement with the NODF option.

The quantities δ and κ are efficiently computed by a method described in De Jong and Penzer (1998); see also Kohn and Ansley (1985).

Modeling in the Presence of Outliers

In practice, modeling and forecasting time series data in the presence of outliers is a difficult problem for several reasons. The presence of outliers can adversely affect the model identification and estimation steps. Their presence close to the end of the observation period can have a serious impact on the forecasting performance of the model. In some cases, level shifts are associated with changes in the mechanism that drives the observation process, and separate models might be appropriate to different sections of the data. In view of all these difficulties, diagnostic tools such as outlier detection and residual analysis are essential in any modeling process.

The following modeling strategy, which incorporates level shift detection in the familiar Box-Jenkins modeling methodology, seems to work in many cases:

1. Proceed with model identification and estimation as usual. Suppose this results in a tentative ARIMA model, say M.
2. Check for additive and permanent level shifts unaccounted for by the model M by using the OUTLIER statement. In this step, unless there is evidence to justify it, the number of level shifts searched should be kept small.
3. Augment the original dataset with the regression variables that correspond to the detected outliers.
4. Include the first few of these regression variables in M, and call this model M1. Reestimate all the parameters of M1. It is important not to include too many of these outlier variables in the model in order to avoid the danger of over-fitting.
5. Check the adequacy of M1 by examining the parameter estimates, residual analysis, and outlier detection. Refine it more if necessary.

OUT= Data Set

The output data set produced by the OUT= option of the PROC ARIMA or FORECAST statements contains the following:

- the BY variables
- the ID variable

- the variable specified by the VAR= option in the IDENTIFY statement, which contains the actual values of the response series
- FORECAST, a numeric variable that contains the one-step-ahead predicted values and the multistep forecasts
- STD, a numeric variable that contains the standard errors of the forecasts
- a numeric variable that contains the lower confidence limits of the forecast. This variable is named L95 by default but has a different name if the ALPHA= option specifies a different size for the confidence limits.
- RESIDUAL, a numeric variable that contains the differences between actual and forecast values
- a numeric variable that contains the upper confidence limits of the forecast. This variable is named U95 by default but has a different name if the ALPHA= option specifies a different size for the confidence limits.

The ID variable, the BY variables, and the response variable are the only ones copied from the input to the output data set. In particular, the input variables are not copied to the OUT= data set.

Unless the NOOUTALL option is specified, the data set contains the whole time series. The FORECAST variable has the one-step forecasts (predicted values) for the input periods, followed by *n* forecast values, where *n* is the LEAD= value. The actual and RESIDUAL values are missing beyond the end of the series.

If you specify the same OUT= data set in different FORECAST statements, the latter FORECAST statements overwrite the output from the previous FORECAST statements. If you want to combine the forecasts from different FORECAST statements in the same output data set, specify the OUT= option once in the PROC ARIMA statement and omit the OUT= option in the FORECAST statements.

When a global output data set is created by the OUT= option in the PROC ARIMA statement, the variables in the OUT= data set are defined by the first FORECAST statement that is executed. The results of subsequent FORECAST statements are vertically concatenated onto the OUT= data set. Thus, if no ID variable is specified in the first FORECAST statement that is executed, no ID variable appears in the output data set, even if one is specified in a later FORECAST statement. If an ID variable is specified in the first FORECAST statement that is executed but not in a later FORECAST statement, the value of the ID variable is the same as the last value processed for the ID variable for all observations created by the later FORECAST statement. Furthermore, even if the response variable changes in subsequent FORECAST statements, the response variable name in the output data set is that of the first response variable analyzed.

OUTCOV= Data Set

The output data set produced by the OUTCOV= option of the IDENTIFY statement contains the following variables:

- LAG, a numeric variable that contains the lags that correspond to the values of the covariance variables. The values of LAG range from 0 to N for covariance functions and from –N to N for cross-covariance functions, where N is the value of the NLAG= option.
- VAR, a character variable that contains the name of the variable specified by the VAR= option.

- CROSSVAR, a character variable that contains the name of the variable specified in the CROSSCORR= option, which labels the different cross-covariance functions. The CROSSVAR variable is blank for the autocovariance observations. When there is no CROSSCORR= option, this variable is not created.
- N, a numeric variable that contains the number of observations used to calculate the current value of the covariance or cross-covariance function.
- COV, a numeric variable that contains the autocovariance or cross-covariance function values. COV contains the autocovariances of the VAR= variable when the value of the CROSSVAR variable is blank. Otherwise COV contains the cross covariances between the VAR= variable and the variable named by the CROSSVAR variable.
- CORR, a numeric variable that contains the autocorrelation or cross-correlation function values. CORR contains the autocorrelations of the VAR= variable when the value of the CROSSVAR variable is blank. Otherwise CORR contains the cross-correlations between the VAR= variable and the variable named by the CROSSVAR variable.
- STDERR, a numeric variable that contains the standard errors of the autocorrelations. The standard error estimate is based on the hypothesis that the process that generates the time series is a pure moving-average process of order LAG–1. For the cross-correlations, STDERR contains the value $1/\sqrt{n}$, which approximates the standard error under the hypothesis that the two series are uncorrelated.
- INVCORR, a numeric variable that contains the inverse autocorrelation function values of the VAR= variable. For cross-correlation observations (that is, when the value of the CROSSVAR variable is not blank), INVCORR contains missing values.
- PARTCORR, a numeric variable that contains the partial autocorrelation function values of the VAR= variable. For cross-correlation observations (that is, when the value of the CROSSVAR variable is not blank), PARTCORR contains missing values.

OUTEST= Data Set

PROC ARIMA writes the parameter estimates for a model to an output data set when the OUTEST= option is specified in the ESTIMATE statement. The OUTEST= data set contains the following:

- the BY variables
- _MODLABEL_, a character variable that contains the model label, if it is provided by using the label option in the ESTIMATE statement (otherwise this variable is not created).
- _NAME_, a character variable that contains the name of the parameter for the covariance or correlation observations or is blank for the observations that contain the parameter estimates. (This variable is not created if neither OUTCOV nor OUTCORR is specified.)
- _TYPE_, a character variable that identifies the type of observation. A description of the _TYPE_ variable values is given below.
- variables for model parameters

The variables for the model parameters are named as follows:

ERRORVAR — This numeric variable contains the variance estimate. The _TYPE_=EST observation for this variable contains the estimated error variance, and the remaining observations are missing.

MU — This numeric variable contains values for the mean parameter for the model. (This variable is not created if NOCONSTANT is specified.)

MA*j* _*k* — These numeric variables contain values for the moving-average parameters. The variables for moving-average parameters are named MA*j* _*k*, where *j* is the factor-number and *k* is the index of the parameter within a factor.

AR*j* _*k* — These numeric variables contain values for the autoregressive parameters. The variables for autoregressive parameters are named AR*j* _*k*, where *j* is the factor number and *k* is the index of the parameter within a factor.

I*j* _*k* — These variables contain values for the transfer function parameters. Variables for transfer function parameters are named I*j* _*k*, where *j* is the number of the INPUT variable associated with the transfer function component and *k* is the number of the parameter for the particular INPUT variable. INPUT variables are numbered according to the order in which they appear in the INPUT= list.

STATUS — This variable describes the convergence status of the model. A value of 0_CONVERGED indicates that the model converged.

The value of the _TYPE_ variable for each observation indicates the kind of value contained in the variables for model parameters for the observation. The OUTEST= data set contains observations with the following _TYPE_ values:

EST — The observation contains parameter estimates.

STD — The observation contains approximate standard errors of the estimates.

CORR — The observation contains correlations of the estimates. OUTCORR must be specified to get these observations.

COV — The observation contains covariances of the estimates. OUTCOV must be specified to get these observations.

FACTOR — The observation contains values that identify for each parameter the factor that contains it. Negative values indicate denominator factors in transfer function models.

LAG — The observation contains values that identify the lag associated with each parameter.

SHIFT — The observation contains values that identify the shift associated with the input series for the parameter.

The values given for _TYPE_=FACTOR, _TYPE_=LAG, or _TYPE_=SHIFT observations enable you to reconstruct the model employed when provided with only the OUTEST= data set.

OUTEST= Examples

This section clarifies how model parameters are stored in the OUTEST= data set with two examples.

Consider the following example:

```
proc arima data=input;
   identify var=y cross=(x1 x2);
   estimate p=(1)(6) q=(1,3)(12) input=(x1 x2) outest=est;
run;

proc print data=est;
run;
```

The model specified by these statements is

$$Y_t = \mu + \omega_{1,0}X_{1,t} + \omega_{2,0}X_{2,t} + \frac{(1-\theta_{11}B-\theta_{12}B^3)(1-\theta_{21}B^{12})}{(1-\phi_{11}B)(1-\phi_{21}B^6)}a_t$$

The OUTEST= data set contains the values shown in Table 8.10.

Table 8.10 OUTEST= Data Set for First Example

Obs	_TYPE_	Y	MU	MA1_1	MA1_2	MA2_1	AR1_1	AR2_1	I1_1	I2_1
1	EST	σ^2	μ	θ_{11}	θ_{12}	θ_{21}	ϕ_{11}	ϕ_{21}	$\omega_{1,0}$	$\omega_{2,0}$
2	STD	.	se μ	se θ_{11}	se θ_{12}	se θ_{21}	se ϕ_{11}	se ϕ_{21}	se $\omega_{1,0}$	se $\omega_{2,0}$
3	FACTOR	.	0	1	1	2	1	2	1	1
4	LAG	.	0	1	3	12	1	6	0	0
5	SHIFT	.	0	0	0	0	0	0	0	0

Note that the symbols in the rows for _TYPE_=EST and _TYPE_=STD in Table 8.10 would be numeric values in a real data set.

Next, consider the following example:

```
proc arima data=input;
   identify var=y cross=(x1 x2);
   estimate p=1 q=1 input=(2 $ (1)/(1,2)x1 1 $ /(1)x2) outest=est;
run;

proc print data=est;
run;
```

The model specified by these statements is

$$Y_t = \mu + \frac{\omega_{10}-\omega_{11}B}{1-\delta_{11}B-\delta_{12}B^2}X_{1,t-2} + \frac{\omega_{20}}{1-\delta_{21}B}X_{2,t-1} + \frac{(1-\theta_1 B)}{(1-\phi_1 B)}a_t$$

The OUTEST= data set contains the values shown in Table 8.11.

Table 8.11 OUTEST= Data Set for Second Example

Obs	_TYPE_	Y	MU	MA1_1	AR1_1	I1_1	I1_2	I1_3	I1_4	I2_1	I2_2
1	EST	σ^2	μ	θ_1	ϕ_1	ω_{10}	ω_{11}	δ_{11}	δ_{12}	ω_{20}	δ_{21}
2	STD	.	se μ	se θ_1	se ϕ_1	se ω_{10}	se ω_{11}	se δ_{11}	se δ_{12}	se ω_{20}	se δ_{21}
3	FACTOR	.	0	1	1	1	1	-1	-1	1	-1
4	LAG	.	0	1	1	0	1	1	2	0	1
5	SHIFT	.	0	0	0	2	2	2	2	1	1

OUTMODEL= SAS Data Set

The OUTMODEL= option in the ESTIMATE statement writes an output data set that enables you to reconstruct the model. The OUTMODEL= data set contains much the same information as the OUTEST= data set but in a transposed form that might be more useful for some purposes. In addition, the OUTMODEL= data set includes the differencing operators.

The OUTMODEL data set contains the following:

- the BY variables
- _MODLABEL_, a character variable that contains the model label, if it is provided by using the label option in the ESTIMATE statement (otherwise this variable is not created).
- _NAME_, a character variable that contains the name of the response or input variable for the observation.
- _TYPE_, a character variable that contains the estimation method that was employed. The value of _TYPE_ can be CLS, ULS, or ML.
- _STATUS_, a character variable that describes the convergence status of the model. A value of 0_CONVERGED indicates that the model converged.
- _PARM_, a character variable that contains the name of the parameter given by the observation. _PARM_ takes on the values ERRORVAR, MU, AR, MA, NUM, DEN, and DIF.
- _VALUE_, a numeric variable that contains the value of the estimate defined by the _PARM_ variable.
- _STD_, a numeric variable that contains the standard error of the estimate.
- _FACTOR_, a numeric variable that indicates the number of the factor to which the parameter belongs.
- _LAG_, a numeric variable that contains the number of the term within the factor that contains the parameter.
- _SHIFT_, a numeric variable that contains the shift value for the input variable associated with the current parameter.

The values of _FACTOR_ and _LAG_ identify which particular MA, AR, NUM, or DEN parameter estimate is given by the _VALUE_ variable. The _NAME_ variable contains the response variable name for the MU, AR, or MA parameters. Otherwise, _NAME_ contains the input variable name associated with NUM or DEN parameter estimates. The _NAME_ variable contains the appropriate variable name associated with the current DIF observation as well. The _VALUE_ variable is 1 for all DIF observations, and the _LAG_ variable indicates the degree of differencing employed.

The observations contained in the OUTMODEL= data set are identified by the _PARM_ variable. A description of the values of the _PARM_ variable follows:

NUMRESID	_VALUE_ contains the number of residuals.
NPARMS	_VALUE_ contains the number of parameters in the model.

NDIFS	_VALUE_ contains the sum of the differencing lags employed for the response variable.
ERRORVAR	_VALUE_ contains the estimate of the innovation variance.
MU	_VALUE_ contains the estimate of the mean term.
AR	_VALUE_ contains the estimate of the autoregressive parameter indexed by the _FACTOR_ and _LAG_ variable values.
MA	_VALUE_ contains the estimate of a moving-average parameter indexed by the _FACTOR_ and _LAG_ variable values.
NUM	_VALUE_ contains the estimate of the parameter in the numerator factor of the transfer function of the input variable indexed by the _FACTOR_, _LAG_, and _SHIFT_ variable values.
DEN	_VALUE_ contains the estimate of the parameter in the denominator factor of the transfer function of the input variable indexed by the _FACTOR_, _LAG_, and _SHIFT_ variable values.
DIF	_VALUE_ contains the difference operator defined by the difference lag given by the value in the _LAG_ variable.

OUTSTAT= Data Set

PROC ARIMA writes the diagnostic statistics for a model to an output data set when the OUTSTAT= option is specified in the ESTIMATE statement. The OUTSTAT data set contains the following:

- the BY variables.
- _MODLABEL_, a character variable that contains the model label, if it is provided by using the label option in the ESTIMATE statement (otherwise this variable is not created).
- _TYPE_, a character variable that contains the estimation method used. _TYPE_ can have the value CLS, ULS, or ML.
- _STAT_, a character variable that contains the name of the statistic given by the _VALUE_ variable in this observation. _STAT_ takes on the values AIC, SBC, LOGLIK, SSE, NUMRESID, NPARMS, NDIFS, ERRORVAR, MU, CONV, and NITER.
- _VALUE_, a numeric variable that contains the value of the statistic named by the _STAT_ variable.

The observations contained in the OUTSTAT= data set are identified by the _STAT_ variable. A description of the values of the _STAT_ variable follows:

AIC	Akaike's information criterion
SBC	Schwarz's Bayesian criterion
LOGLIK	the log-likelihood, if METHOD=ML or METHOD=ULS is specified
SSE	the sum of the squared residuals
NUMRESID	the number of residuals

NPARMS	the number of parameters in the model
NDIFS	the sum of the differencing lags employed for the response variable
ERRORVAR	the estimate of the innovation variance
MU	the estimate of the mean term
CONV	tells if the estimation converged. The value of 0 signifies that estimation converged. Nonzero values reflect convergence problems.
NITER	the number of iterations

Remark. CONV takes an integer value that corresponds to the error condition of the parameter estimation process. The value of 0 signifies that estimation process has converged. The higher values signify convergence problems of increasing severity. Specifically:

- CONV= 0 indicates that the estimation process has converged.
- CONV= 1 or 2 indicates that the estimation process has run into numerical problems (such as encountering an unstable model or a ridge) during the iterations.
- CONV= 3 or greater indicates that the estimation process has failed to converge.

Printed Output

The ARIMA procedure produces printed output for each of the IDENTIFY, ESTIMATE, and FORECAST statements. The output produced by each ARIMA statement is described in the following sections.

IDENTIFY Statement Printed Output

The printed output of the IDENTIFY statement consists of the following:

- a table of summary statistics, including the name of the response variable, any specified periods of differencing, the mean and standard deviation of the response series after differencing, and the number of observations after differencing
- a plot of the sample autocorrelation function for lags up to and including the NLAG= option value. Standard errors of the autocorrelations also appear to the right of the autocorrelation plot if the value of LINESIZE= option is sufficiently large. The standard errors are derived using Bartlett's approximation (Box and Jenkins 1976, p. 177). The approximation for a standard error for the estimated autocorrelation function at lag k is based on a null hypothesis that a pure moving-average Gaussian process of order k–1 generated the time series. The relative position of an approximate 95% confidence interval under this null hypothesis is indicated by the dots in the plot, while the asterisks represent the relative magnitude of the autocorrelation value.
- a plot of the sample inverse autocorrelation function. See the section "The Inverse Autocorrelation Function" on page 263 for more information about the inverse autocorrelation function.
- a plot of the sample partial autocorrelation function

- a table of test statistics for the hypothesis that the series is white noise. These test statistics are the same as the tests for white noise residuals produced by the ESTIMATE statement and are described in the section "Estimation Details" on page 272.
- a plot of the sample cross-correlation function for each series specified in the CROSSCORR= option. If a model was previously estimated for a variable in the CROSSCORR= list, the cross-correlations for that series are computed for the prewhitened input and response series. For each input variable with a prewhitening filter, the cross-correlation report for the input series includes the following:
  - a table of test statistics for the hypothesis of no cross-correlation between the input and response series
  - the prewhitening filter used for the prewhitening transformation of the predictor and response variables
- ESACF tables if the ESACF option is used
- MINIC table if the MINIC option is used
- SCAN table if the SCAN option is used
- STATIONARITY test results if the STATIONARITY option is used

ESTIMATE Statement Printed Output

The printed output of the ESTIMATE statement consists of the following:

- if the PRINTALL option is specified, the preliminary parameter estimates and an iteration history that shows the sequence of parameter estimates tried during the fitting process
- a table of parameter estimates that show the following for each parameter: the parameter name, the parameter estimate, the approximate standard error, t value, approximate probability ($Pr > |t|$), the lag for the parameter, the input variable name for the parameter, and the lag or "Shift" for the input variable
- the estimates of the constant term, the innovation variance (variance estimate), the innovation standard deviation (Std Error Estimate), Akaike's information criterion (AIC), Schwarz's Bayesian criterion (SBC), and the number of residuals
- the correlation matrix of the parameter estimates
- a table of test statistics for hypothesis that the residuals of the model are white noise. The table is titled "Autocorrelation Check of Residuals."
- if the PLOT option is specified, autocorrelation, inverse autocorrelation, and partial autocorrelation function plots of the residuals
- if an INPUT variable has been modeled in such a way that prewhitening is performed in the IDENTIFY step, a table of test statistics titled "Cross-correlation Check of Residuals." The test statistic is based on the chi-square approximation suggested by Box and Jenkins (1976, pp. 395–396). The cross-correlation function is computed by using the residuals from the model as one series and the prewhitened input variable as the other series.

- if the GRID option is specified, the sum-of-squares or likelihood surface over a grid of parameter values near the final estimates
- a summary of the estimated model that shows the autoregressive factors, moving-average factors, and transfer function factors in backshift notation with the estimated parameter values.

OUTLIER Statement Printed Output

The printed output of the OUTLIER statement consists of the following:

- a summary that contains the information about the maximum number of outliers searched, the number of outliers actually detected, and the significance level used in the outlier detection.
- a table that contains the results of the outlier detection process. The outliers are listed in the order in which they are found. This table contains the following columns:
  - The Obs column contains the observation number of the start of the level shift.
  - If an ID= option is specified, then the Time ID column contains the time identification labels of the start of the outlier.
  - The Type column lists the type of the outlier.
  - The Estimate column contains $\hat{\beta}$, the estimate of the regression coefficient of the shock signature.
  - The Chi-Square column lists the value of the test statistic τ^2.
  - The Approx Prob > ChiSq column lists the approximate p-value of the test statistic.

FORECAST Statement Printed Output

The printed output of the FORECAST statement consists of the following:

- a summary of the estimated model
- a table of forecasts with following columns:
  - The Obs column contains the observation number.
  - The Forecast column contains the forecast values.
  - The Std Error column contains the forecast standard errors.
  - The Lower and Uppers columns contain the approximate 95% confidence limits. The ALPHA= option can be used to change the confidence interval for forecasts.
  - If the PRINTALL option is specified, the forecast table also includes columns for the actual values of the response series (Actual) and the residual values (Residual).

ODS Table Names

PROC ARIMA assigns a name to each table it creates. You can use these names to reference the table when you use the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in Table 8.12.

Table 8.12 ODS Tables Produced by PROC ARIMA

ODS Table Name	Description	Statement	Option
ChiSqAuto	chi-square statistics table for autocorrelation	IDENTIFY	
ChiSqCross	chi-square statistics table for cross-correlations	IDENTIFY	CROSSCORR
AutoCorrGraph	Correlations graph	IDENTIFY	
CrossCorrGraph	Cross-Correlations graph	IDENTIFY	
DescStats	Descriptive statistics	IDENTIFY	
ESACF	Extended sample autocorrelation function	IDENTIFY	ESACF
ESACFPValues	ESACF probability values	IDENTIFY	ESACF
IACFGraph	Inverse autocorrelations graph	IDENTIFY	
InputDescStats	Input descriptive statistics	IDENTIFY	
MINIC	Minimum information criterion	IDENTIFY	MINIC
PACFGraph	Partial autocorrelations graph	IDENTIFY	
SCAN	Squared canonical correlation estimates	IDENTIFY	SCAN
SCANPValues	SCAN chi-square probability values	IDENTIFY	SCAN
StationarityTests	Stationarity tests	IDENTIFY	STATIONARITY
TentativeOrders	Tentative order selection	IDENTIFY	MINIC, ESACF, or SCAN
ARPolynomial	Filter equations	ESTIMATE	
ChiSqAuto	chi-square statistics table for autocorrelation	ESTIMATE	
ChiSqCross	chi-square statistics table for cross-correlations	ESTIMATE	
CorrB	Correlations of the estimates	ESTIMATE	
DenPolynomial	Filter equations	ESTIMATE	
FitStatistics	Fit statistics	ESTIMATE	
IterHistory	Iteration history	ESTIMATE	PRINTALL
InitialAREstimates	Initial autoregressive parameter estimates	ESTIMATE	
InitialMAEstimates	Initial moving-average parameter estimates	ESTIMATE	
InputDescription	Input description	ESTIMATE	
MAPolynomial	Filter equations	ESTIMATE	

Table 8.12 *continued*

ODS Table Name	Description	Statement	Option
ModelDescription	Model description	ESTIMATE	
NumPolynomial	Filter equations	ESTIMATE	
ParameterEstimates	Parameter estimates	ESTIMATE	
PrelimEstimates	Preliminary estimates	ESTIMATE	
ObjectiveGrid	Objective function grid matrix	ESTIMATE	GRID
OptSummary	ARIMA estimation optimization	ESTIMATE	PRINTALL
OutlierDetails	Detected outliers	OUTLIER	
Forecasts	Forecast	FORECAST	

Statistical Graphics

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*).

Before you create graphs, ODS Graphics must be enabled (for example, with the ODS GRAPHICS ON statement). For more information about enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" in that chapter.

The overall appearance of graphs is controlled by ODS styles. Styles and other aspects of using ODS Graphics are discussed in the section "A Primer on ODS Statistical Graphics" in that chapter.

This section provides information about the graphics produced by the ARIMA procedure. (See Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*), for more information about ODS statistical graphics.) The main types of plots available are as follows:

- plots useful in the trend and correlation analysis of the dependent and input series
- plots useful for the residual analysis of an estimated model
- forecast plots

You can obtain most plots relevant to the specified model by default. For finer control of the graphics, you can use the PLOTS= option in the PROC ARIMA statement. The following example is a simple illustration of how to use the PLOTS= option.

Airline Series: Illustration of ODS Graphics

The series in this example, the monthly airline passenger series, is also discussed later, in Example 8.2.

The following statements specify an ARIMA$(0,1,1)\times(0,1,1)_{12}$ model without a mean term to the logarithms of the airline passengers series, xlog. Notice the use of the global plot option **ONLY** in the PLOTS= option

of the PROC ARIMA statement. It suppresses the production of default graphics and produces only the plots specified by the subsequent RESIDUAL and FORECAST plot options. The **RESIDUAL(SMOOTH)** plot specification produces a time series plot of residuals that has an overlaid loess fit; see Figure 8.21. The **FORECAST(FORECAST)** option produces a plot that shows the one-step-ahead forecasts, as well as the multistep-ahead forecasts; see Figure 8.22.

```
proc arima data=seriesg
   plots(only)=(residual(smooth) forecast(forecasts));
   identify var=xlog(1,12);
   estimate q=(1)(12) noint method=ml;
   forecast id=date interval=month;
run;
```

Figure 8.21 Residual Plot of the Airline Model

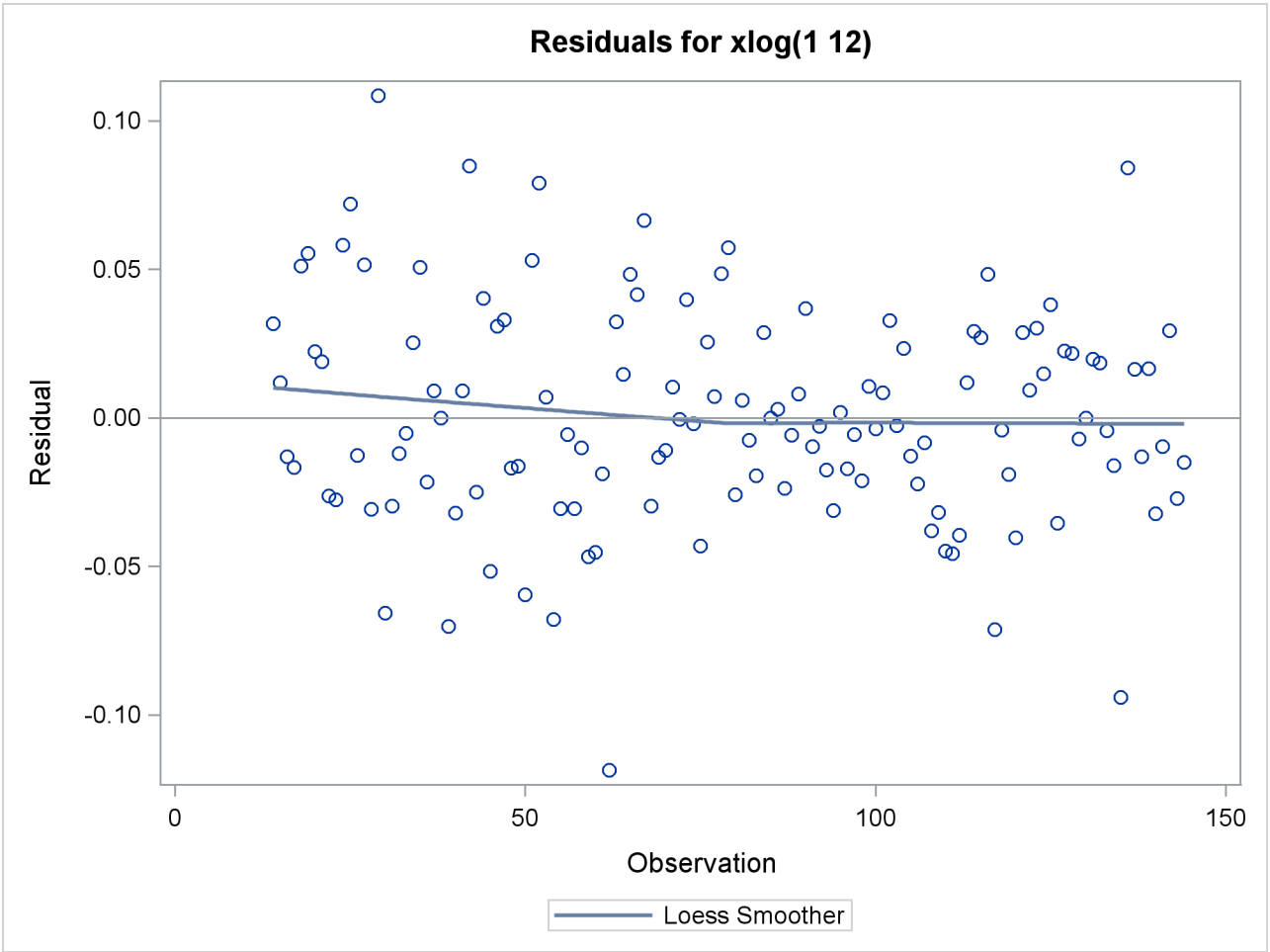
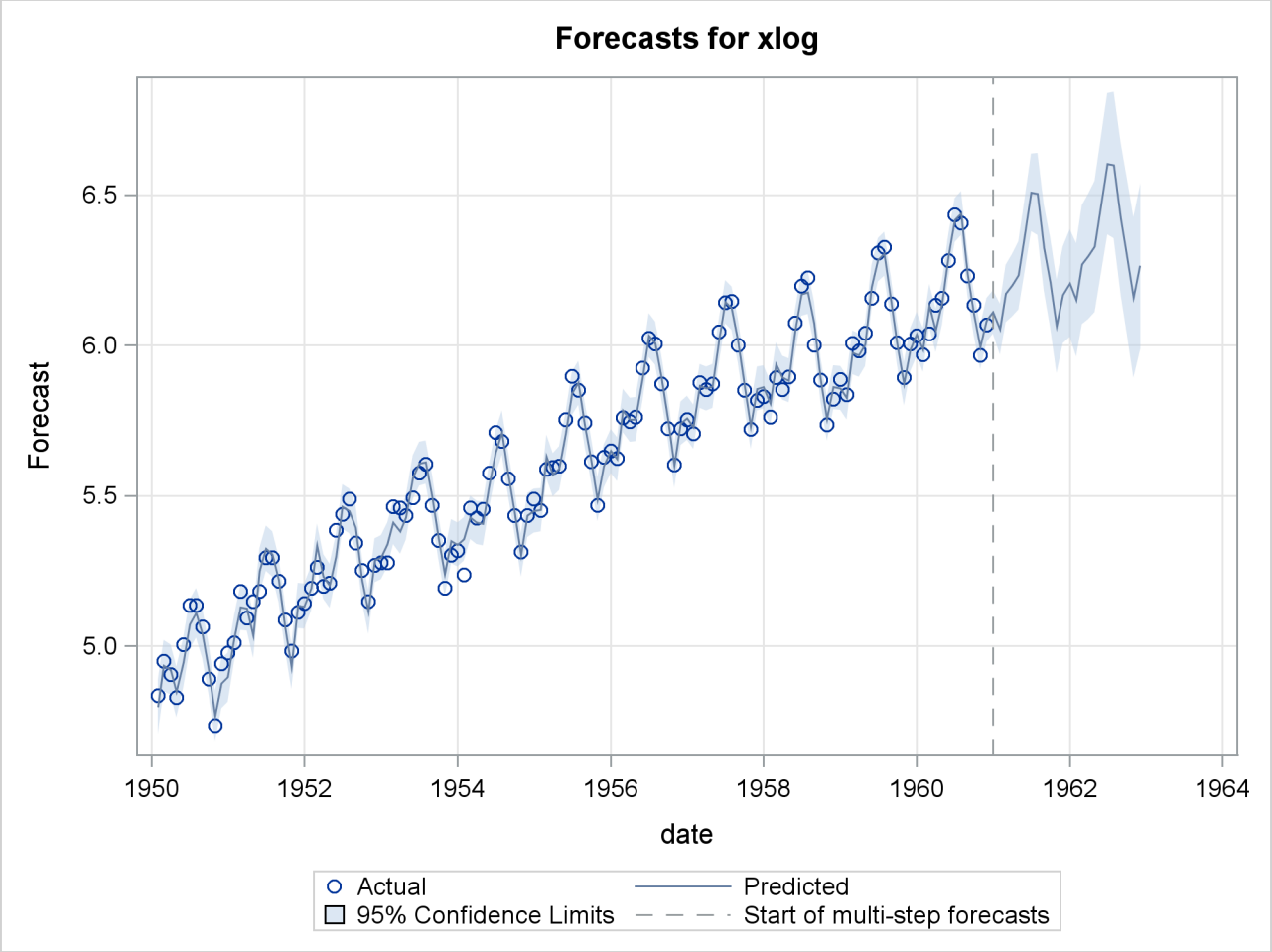


Figure 8.22 Forecast Plot of the Airline Model



ODS Graph Names

PROC ARIMA assigns a name to each graph it creates by using ODS. You can use these names to reference the graphs when you use ODS. The names are listed in Table 8.13.

Table 8.13 ODS Graphics Produced by PROC ARIMA

ODS Graph Name	Plot Description	Option
SeriesPlot	Time series plot of the dependent series	PLOTS(UNPACK)
SeriesACFPlot	Autocorrelation plot of the dependent series	PLOTS(UNPACK)
SeriesPACFPlot	Partial-autocorrelation plot of the dependent series	PLOTS(UNPACK)
SeriesIACFPlot	Inverse-autocorrelation plot of the dependent series	PLOTS(UNPACK)
SeriesCorrPanel	Series trend and correlation analysis panel	Default
CrossCorrPanel	Cross-correlation plots, either individual or paneled. They are numbered 1, 2, and so on as needed.	Default
ResidualACFPlot	Residual-autocorrelation plot	PLOTS(UNPACK)
ResidualPACFPlot	Residual-partial-autocorrelation plot	PLOTS(UNPACK)
ResidualIACFPlot	Residual-inverse-autocorrelation plot	PLOTS(UNPACK)
ResidualWNPlot	Residual-white-noise-probability plot	PLOTS(UNPACK)
ResidualHistogram	Residual histogram	PLOTS(UNPACK)
ResidualQQPlot	Residual normal Q-Q Plot	PLOTS(UNPACK)
ResidualPlot	Time series plot of residuals with a superimposed smoother	PLOTS=RESIDUAL(SMOOTH)
ForecastsOnlyPlot	Time series plot of multistep forecasts	Default
ForecastsPlot	Time series plot of one-step-ahead as well as multistep forecasts	PLOTS=FORECAST(FORECAST)

Examples: ARIMA Procedure

Example 8.1: Simulated IMA Model

This example illustrates the ARIMA procedure results for a case where the true model is known. An integrated moving-average model is used for this illustration.

The following DATA step generates a pseudo-random sample of 100 periods from the ARIMA(0,1,1) process $u_t = u_{t-1} + a_t - 0.8a_{t-1}$, a_t iid $\mathrm{N}(0, 1)$:

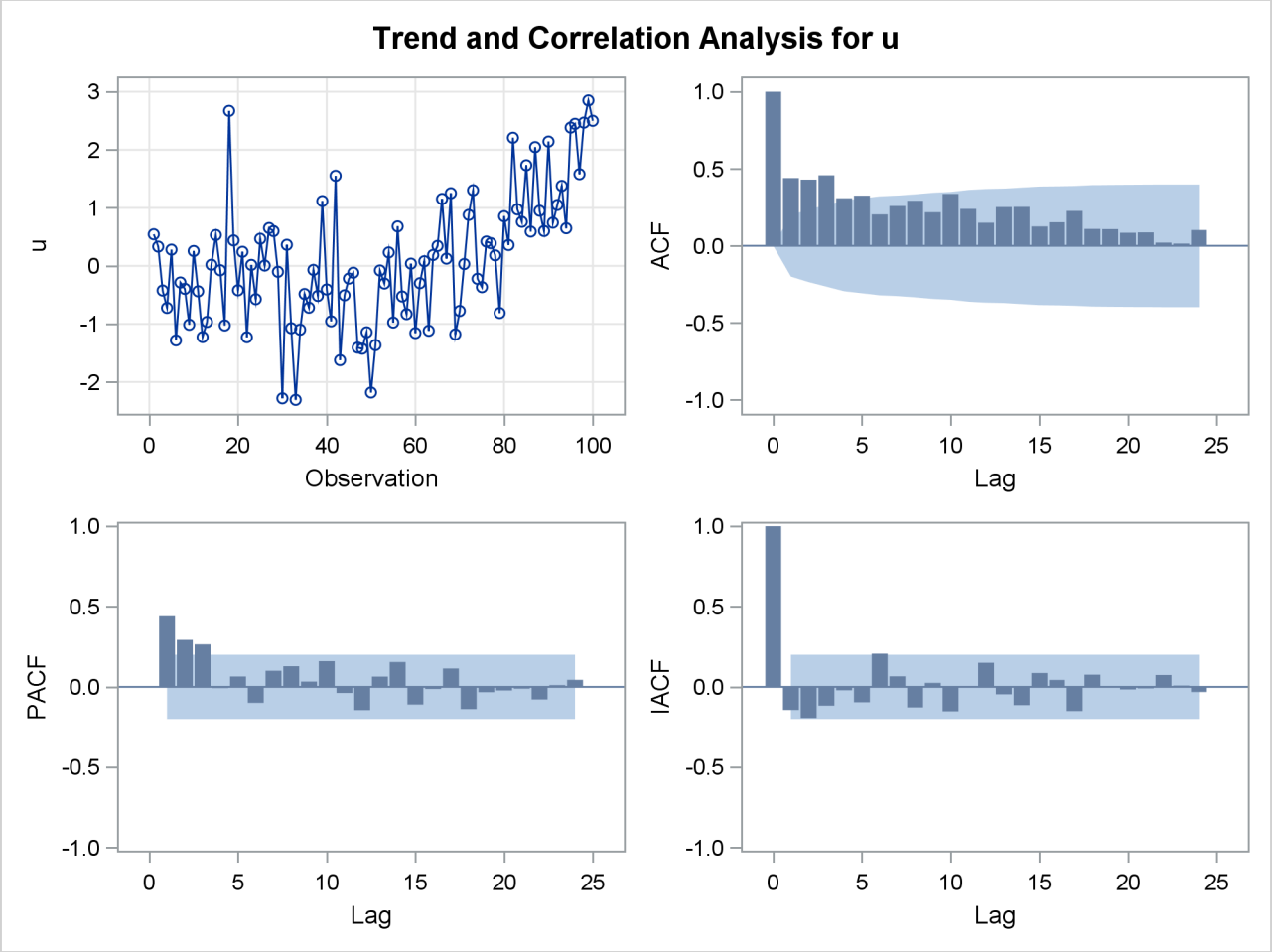
```
title1 'Simulated IMA(1,1) Series';
data a;
  u1 = 0.9; a1 = 0;
  do i = -50 to 100;
     a = rannor( 32565 );
     u = u1 + a - .8 * a1;
     if i > 0 then output;
     a1 = a;
     u1 = u;
  end;
run;
```

The following ARIMA procedure statements identify and estimate the model:

```
/*-- Simulated IMA Model --*/
proc arima data=a;
  identify var=u;
  run;
  identify var=u(1);
  run;
  estimate q=1 ;
  run;
quit;
```

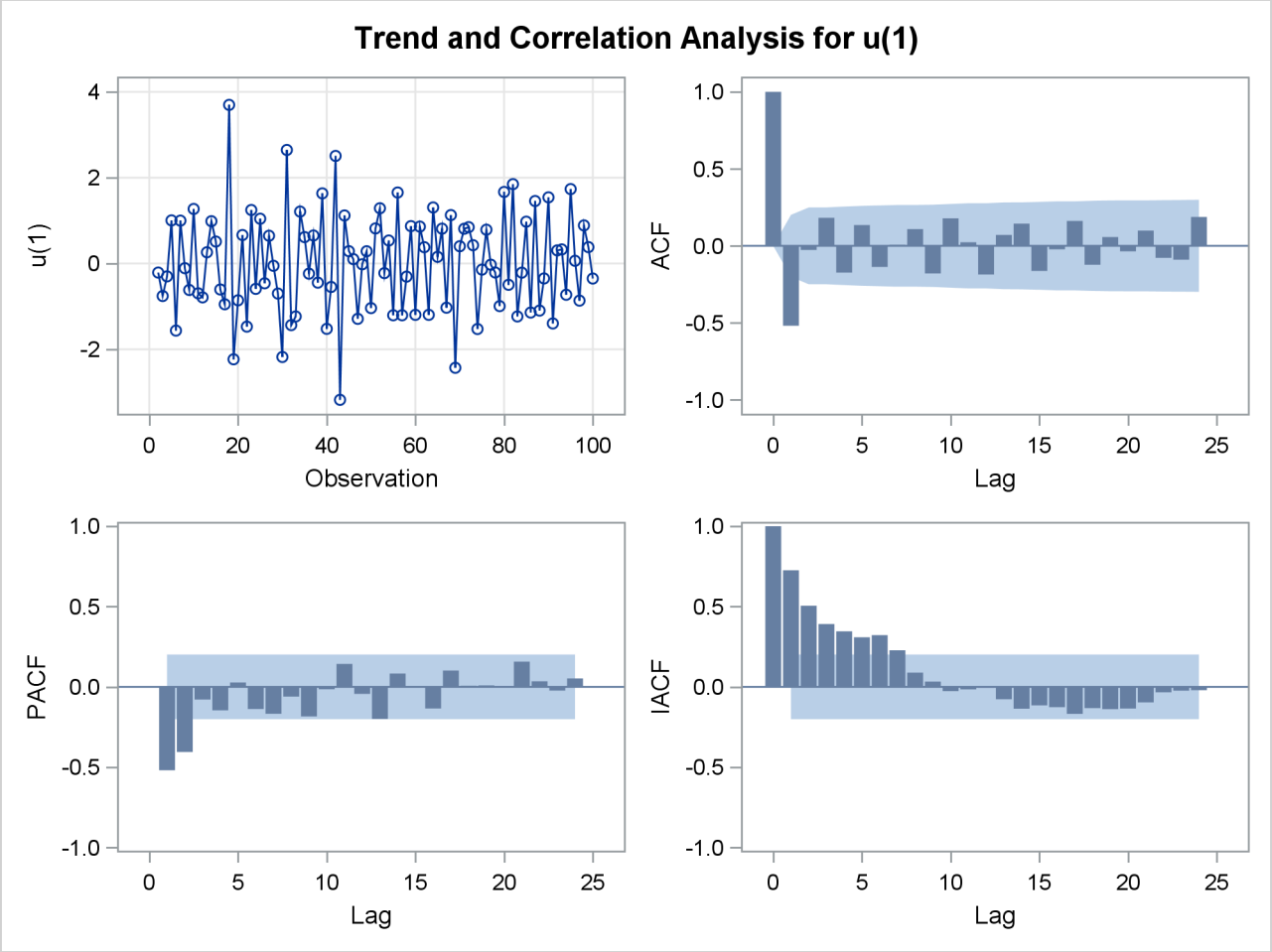
The graphical series correlation analysis output of the first IDENTIFY statement is shown in Output 8.1.1. The output shows the behavior of the sample autocorrelation function when the process is nonstationary. Note that in this case the estimated autocorrelations are not very high, even at small lags. Nonstationarity is reflected in a pattern of significant autocorrelations that do not decline quickly with increasing lag, not in the size of the autocorrelations.

Output 8.1.1 Correlation Analysis from the First IDENTIFY Statement



The second IDENTIFY statement differences the series. The results of the second IDENTIFY statement are shown in Output 8.1.2. This output shows autocorrelation, inverse autocorrelation, and partial autocorrelation functions typical of MA(1) processes.

Output 8.1.2 Correlation Analysis from the Second IDENTIFY Statement



The ESTIMATE statement fits an ARIMA(0,1,1) model to the simulated data. Note that in this case the parameter estimates are reasonably close to the values used to generate the simulated data. ($\mu = 0$, $\hat{\mu} = 0.02$; $\theta_1 = 0.8$, $\hat{\theta}_1 = 0.79$; $\sigma^2 = 1$, $\hat{\sigma}^2 = 0.82$.) Moreover, the graphical analysis of the residuals shows no model inadequacies (see Output 8.1.4 and Output 8.1.5).

The ESTIMATE statement results are shown in Output 8.1.3.

Output 8.1.3 Output from Fitting ARIMA(0,1,1) Model

Conditional Least Squares Estimation

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|	Lag
MU	0.02056	0.01972	1.04	0.2997	0
MA1,1	0.79142	0.06474	12.22	<.0001	1

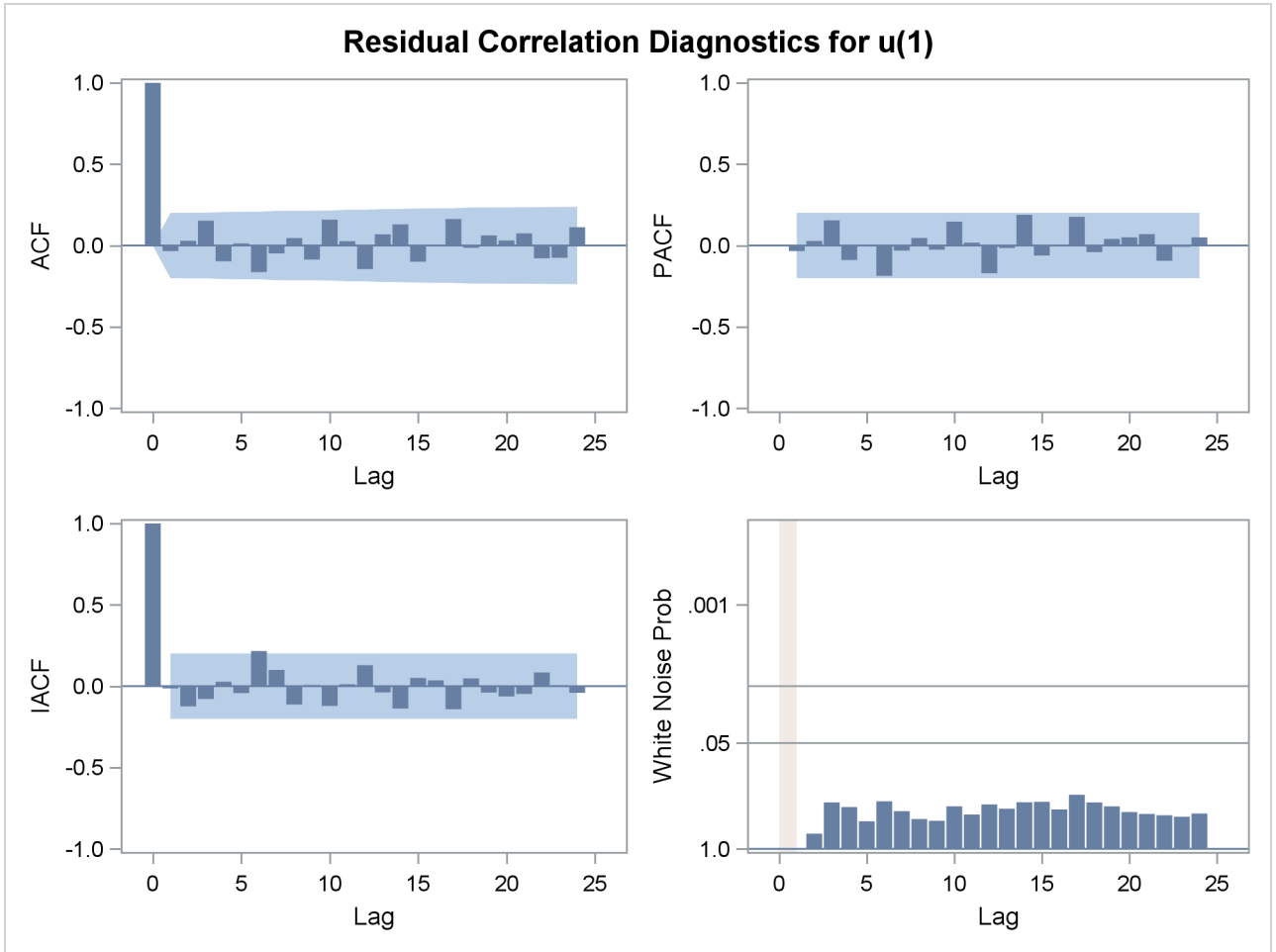
Output 8.1.3 *continued*

Constant Estimate	0.020558
Variance Estimate	0.819807
Std Error Estimate	0.905432
AIC	263.2594
SBC	268.4497
Number of Residuals	99

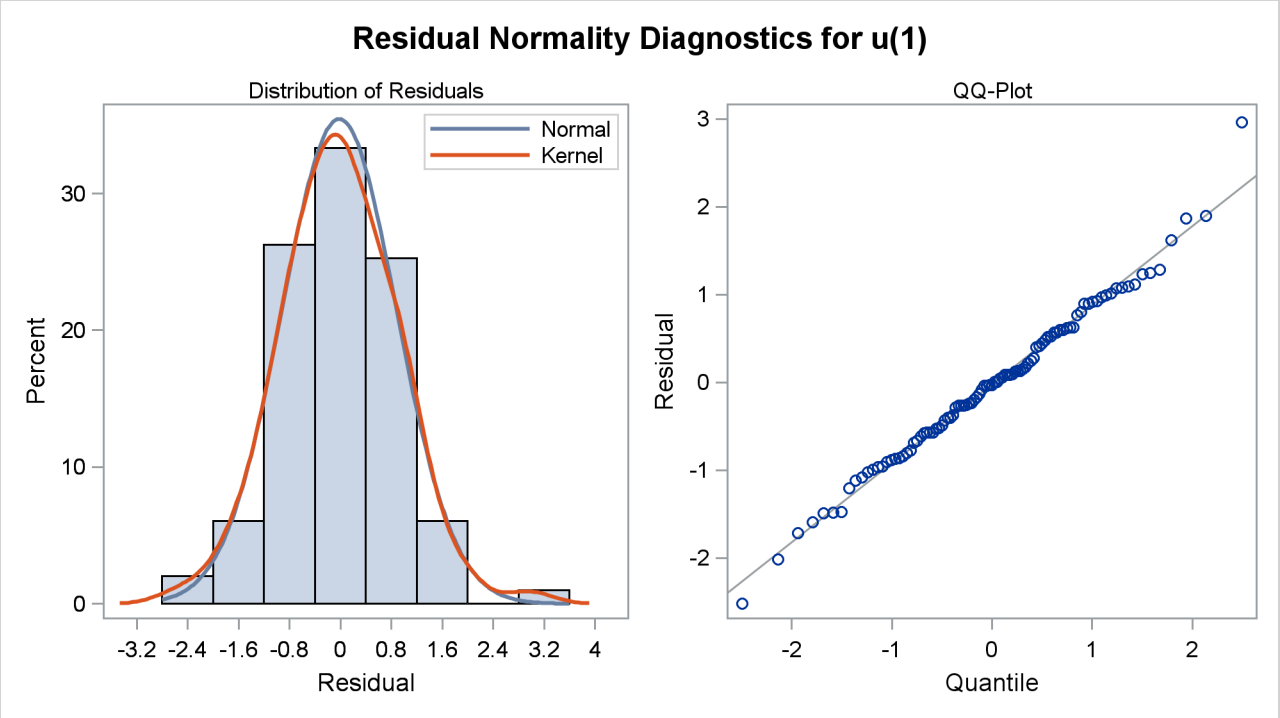
Model for variable u	
Estimated Mean	0.020558
Period(s) of Differencing	1

Moving Average Factors
Factor 1: 1 - 0.79142 B**(1)

Output 8.1.4 Residual Correlation Analysis of the ARIMA(0,1,1) Model



Output 8.1.5 Residual Normality Analysis of the ARIMA(0,1,1) Model



Example 8.2: Seasonal Model for the Airline Series

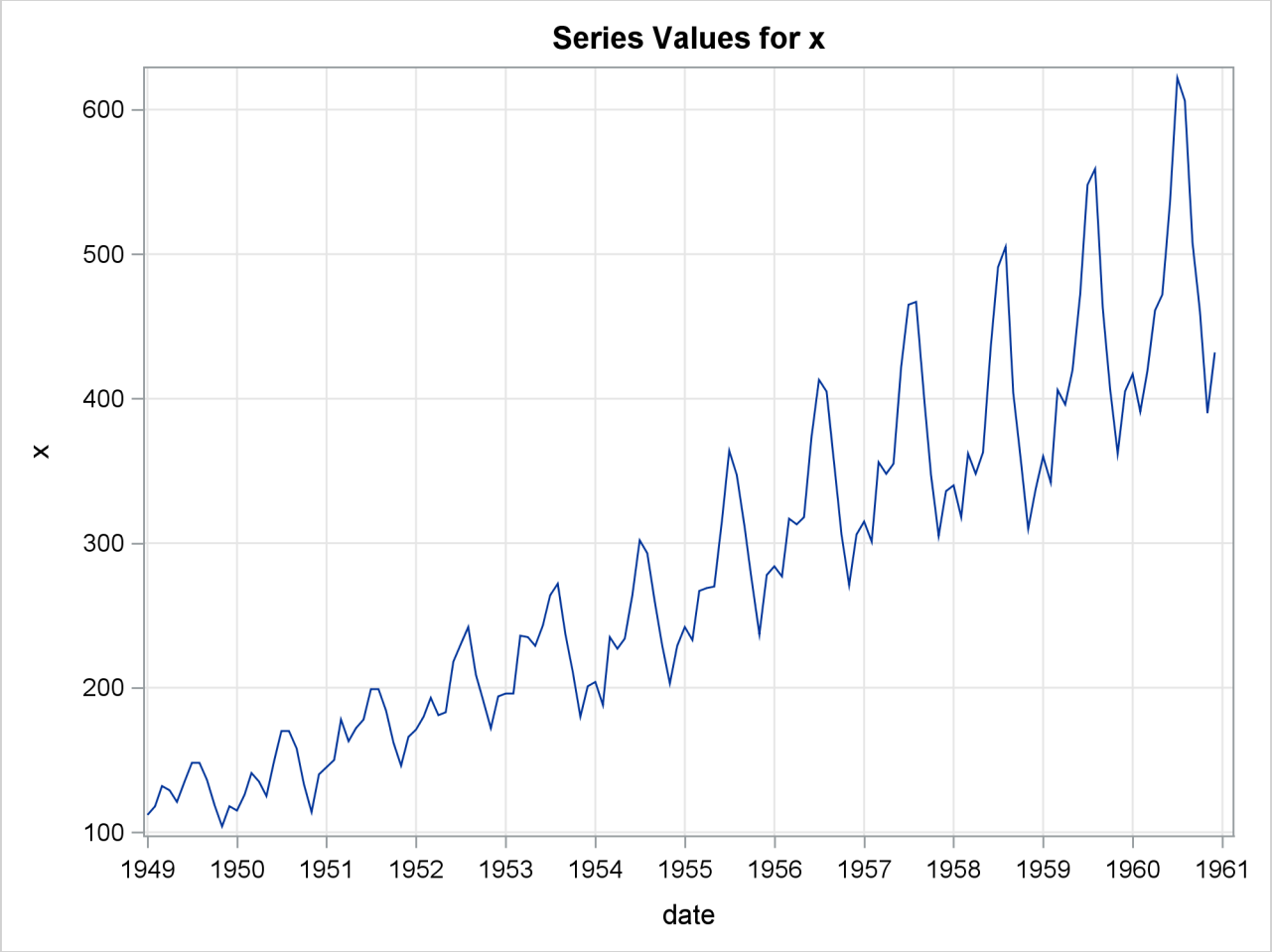
The airline passenger data, given as Series G in Box and Jenkins (1976), have been used in time series analysis literature as an example of a nonstationary seasonal time series. This example uses PROC ARIMA to fit the airline model, ARIMA$(0,1,1)\times(0,1,1)_{12}$, to Box and Jenkins' Series G. The following statements read the data and log-transform the series:

```
title1 'International Airline Passengers';
title2 '(Box and Jenkins Series-G)';
data seriesg;
   input x @@;
   xlog = log( x );
   date = intnx( 'month', '31dec1948'd, _n_ );
   format date monyy.;
datalines;
112 118 132 129 121 135 148 148 136 119 104 118

   ... more lines ...
```

The following PROC TIMESERIES step plots the series, as shown in Output 8.2.1:

```
proc timeseries data=seriesg plot=series;
   id date interval=month;
   var x;
run;
```

Output 8.2.1 Time Series Plot of the Airline Passenger Series



The following statements specify an ARIMA(0,1,1)×(0,1,1)$_{12}$ model without a mean term to the logarithms of the airline passengers series, xlog. The model is forecast, and the results are stored in the data set B.

```
/*-- Seasonal Model for the Airline Series --*/
proc arima data=seriesg;
   identify var=xlog(1,12);
   estimate q=(1)(12) noint method=ml;
   forecast id=date interval=month printall out=b;
run;
```

The output from the IDENTIFY statement is shown in Output 8.2.2. The autocorrelation plots shown are for the twice differenced series $(1-B)(1-B^{12})XLOG$. Note that the autocorrelation functions have the pattern characteristic of a first-order moving-average process combined with a seasonal moving-average process with lag 12.

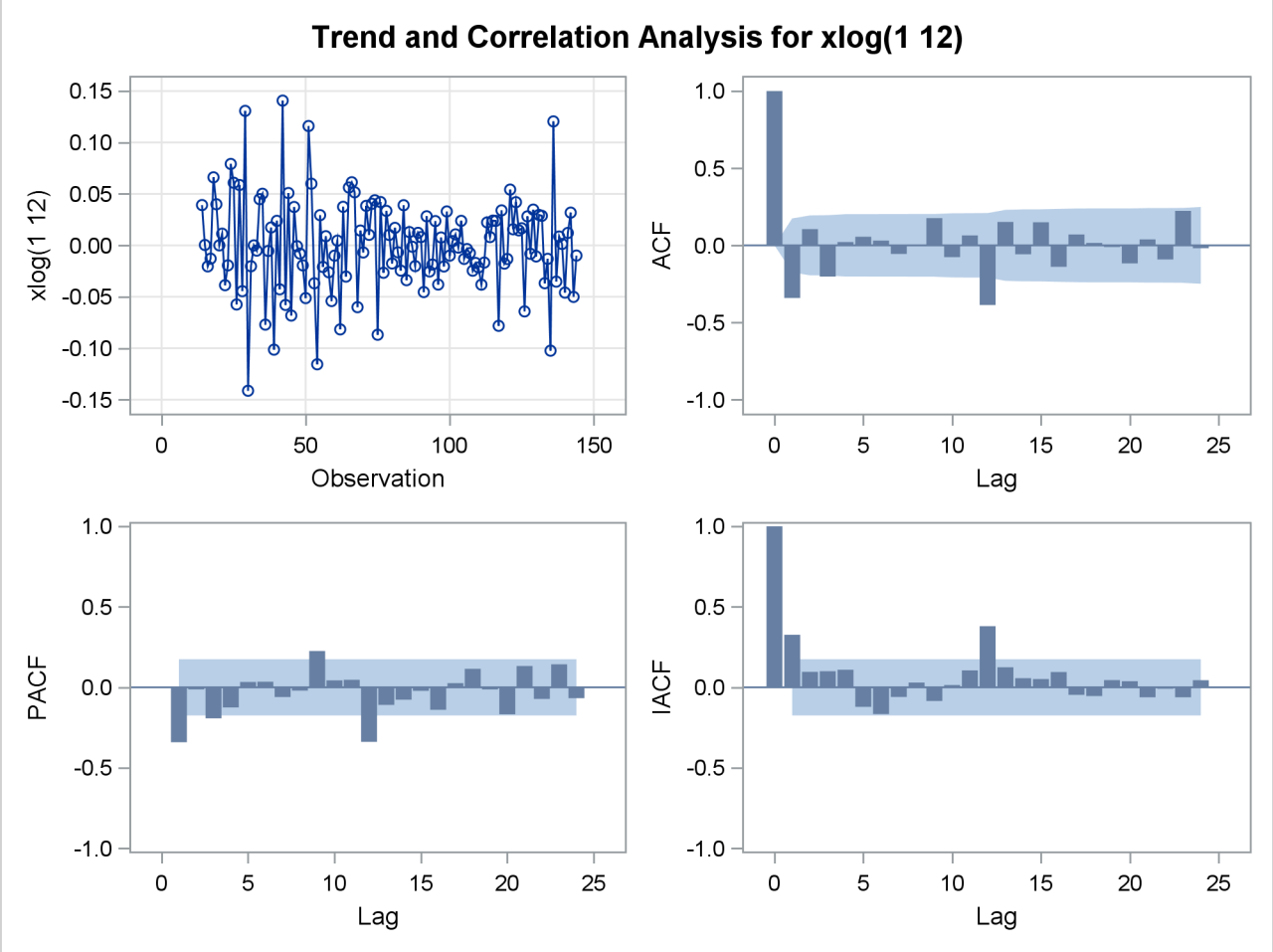
Output 8.2.2 IDENTIFY Statement Output

International Airline Passengers
(Box and Jenkins Series-G)

The ARIMA Procedure

Name of Variable = xlog	
Period(s) of Differencing	1,12
Mean of Working Series	0.000291
Standard Deviation	0.045673
Number of Observations	131
Observation(s) eliminated by differencing	13

Output 8.2.3 Trend and Correlation Analysis for the Twice Differenced Series



The results of the ESTIMATE statement are shown in Output 8.2.4, Output 8.2.5, and Output 8.2.6. The model appears to fit the data quite well.

Output 8.2.4 ESTIMATE Statement Output

Maximum Likelihood Estimation

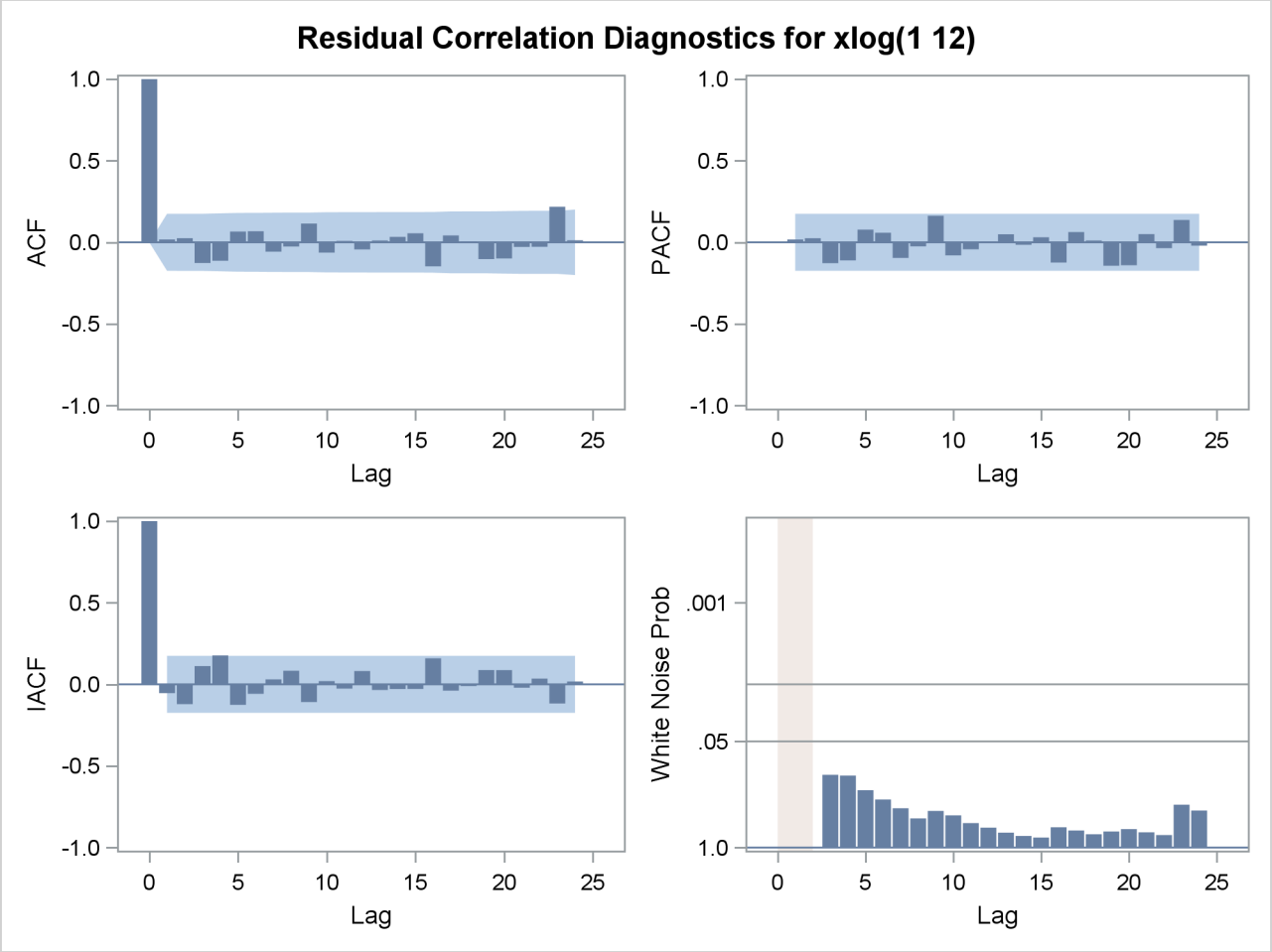
Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|	Lag
MA1,1	0.40194	0.07988	5.03	<.0001	1
MA2,1	0.55686	0.08403	6.63	<.0001	12

Variance Estimate	0.001369
Std Error Estimate	0.037
AIC	-485.393
SBC	-479.643
Number of Residuals	131

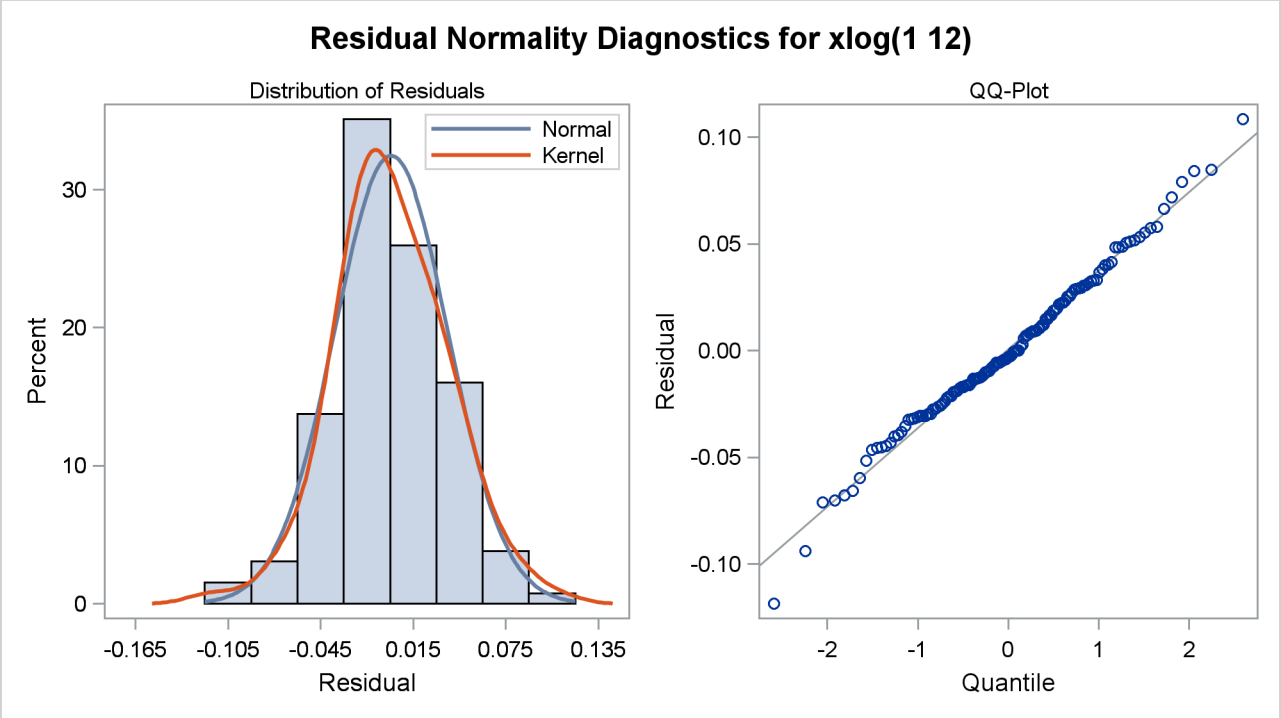
Model for variable xlog	
Period(s) of Differencing	1,12

Moving Average Factors	
Factor 1:	1 - 0.40194 B**(1)
Factor 2:	1 - 0.55686 B**(12)

Output 8.2.5 Residual Analysis of the Airline Model: Correlation

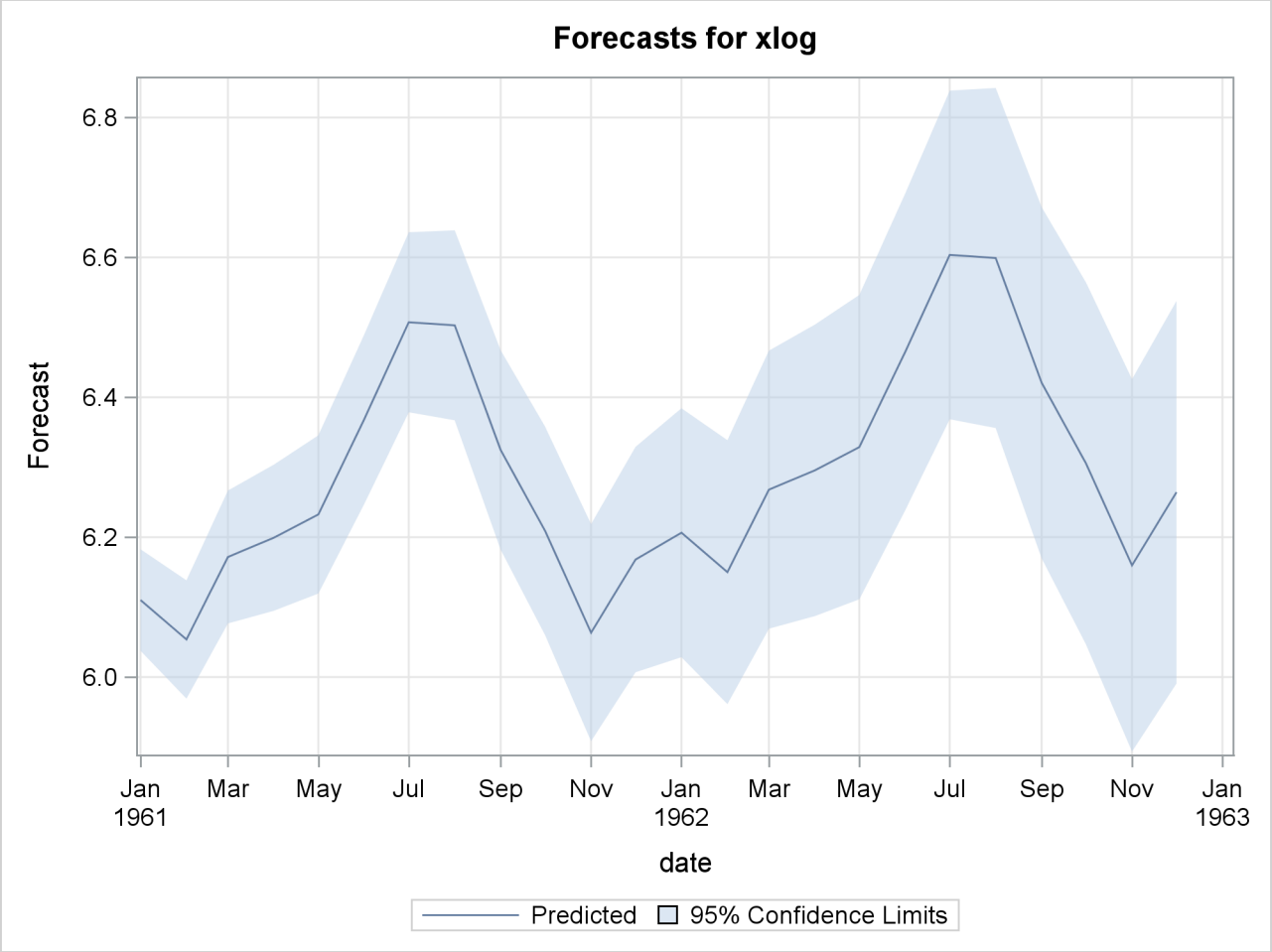


Output 8.2.6 Residual Analysis of the Airline Model: Normality



The forecasts and their confidence limits for the transformed series are shown in Output 8.2.7.

Output 8.2.7 Forecast Plot for the Transformed Series



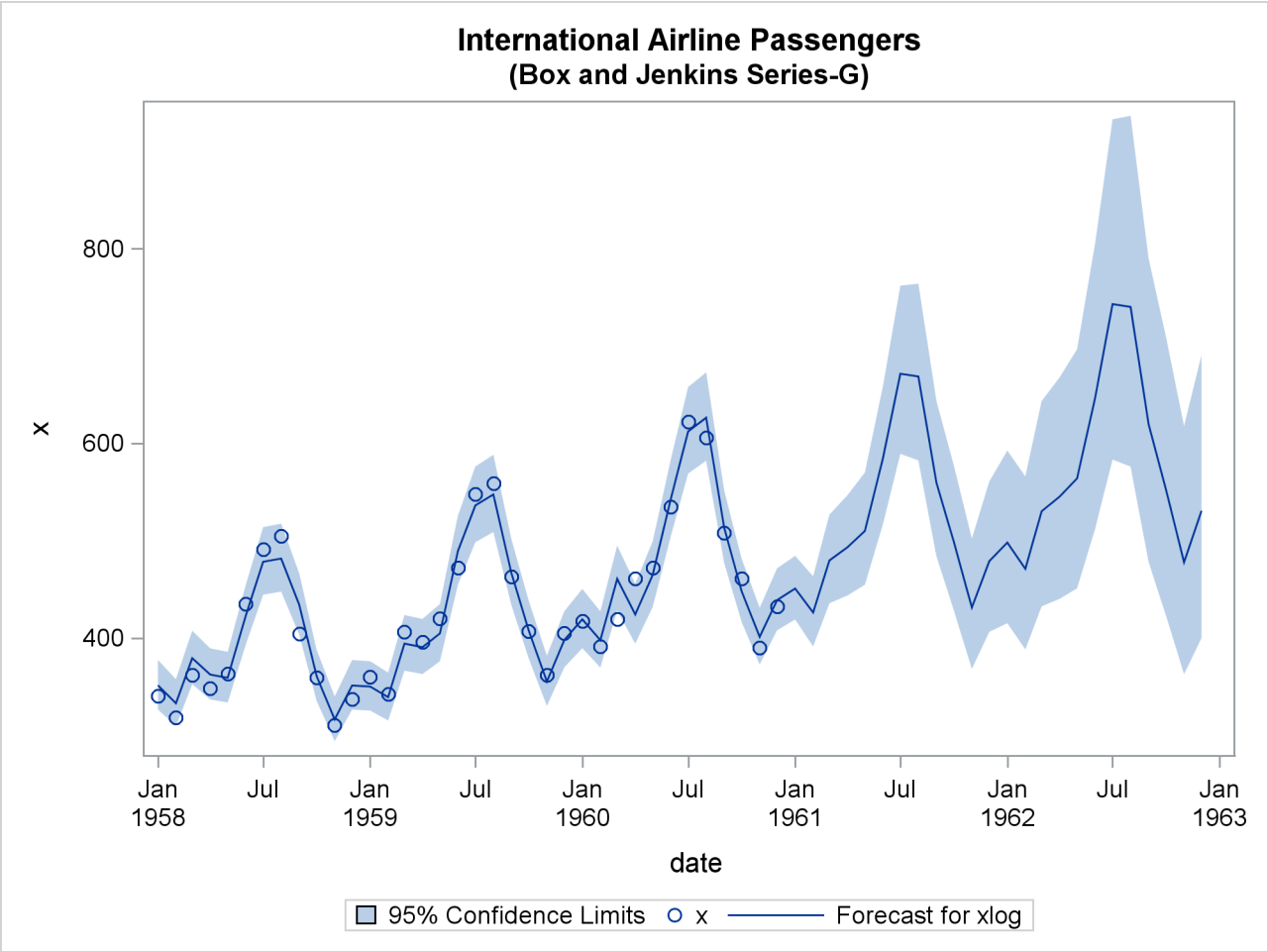
The following statements retransform the forecast values to get forecasts in the original scales. See the section "Forecasting Log Transformed Data" on page 281 for more information.

```
data c;
   set b;
   x        = exp( xlog );
   forecast = exp( forecast + std*std/2 );
   l95      = exp( l95 );
   u95      = exp( u95 );
run;
```

The forecasts and their confidence limits are plotted by using the following PROC SGPLOT step. The plot is shown in Output 8.2.8.

```
proc sgplot data=c;
   where date >= '1jan58'd;
   band Upper=u95 Lower=l95 x=date
      / LegendLabel="95% Confidence Limits";
   scatter x=date y=x;
   series x=date y=forecast;
run;
```

Output 8.2.8 Plot of the Forecast for the Original Series



Example 8.3: Model for Series J Data from Box and Jenkins

This example uses the Series J data from Box and Jenkins (1976). First, the input series X is modeled with a univariate ARMA model. Next, the dependent series Y is cross-correlated with the input series. Since a model has been fit to X, both Y and X are prewhitened by this model before the sample cross-correlations are computed. Next, a transfer function model is fit with no structure on the noise term. The residuals from this model are analyzed; then, the full model, transfer function and noise, is fit to the data.

The following statements read `'Input Gas Rate'` and `'Output CO'`$_2$ from a gas furnace. (Data values are not shown. The full example including data is in the SAS/ETS sample library.)

```
title1 'Gas Furnace Data';
title2 '(Box and Jenkins, Series J)';
data seriesj;
   input x y @@;
   label x = 'Input Gas Rate'
         y = 'Output CO2';
datalines;
-0.109  53.8  0.000  53.6  0.178  53.5  0.339  53.5

   ... more lines ...
```

The following statements produce Output 8.3.1 through Output 8.3.11:

```
proc arima data=seriesj;

   /*--- Look at the input process ----------------------------*/
   identify var=x;
   run;

   /*--- Fit a model for the input ----------------------------*/
   estimate p=3 plot;
   run;

   /*--- Cross-correlation of prewhitened series ---------------*/
   identify var=y crosscorr=(x) nlag=12;
   run;

   /*--- Fit a simple transfer function - look at residuals ---*/
   estimate input=( 3 $ (1,2)/(1) x );
   run;

   /*--- Final Model - look at residuals ----------------------*/
   estimate p=2 input=( 3 $ (1,2)/(1) x );
   run;

quit;
```

The results of the first IDENTIFY statement for the input series X are shown in Output 8.3.1. The correlation analysis suggests an AR(3) model.

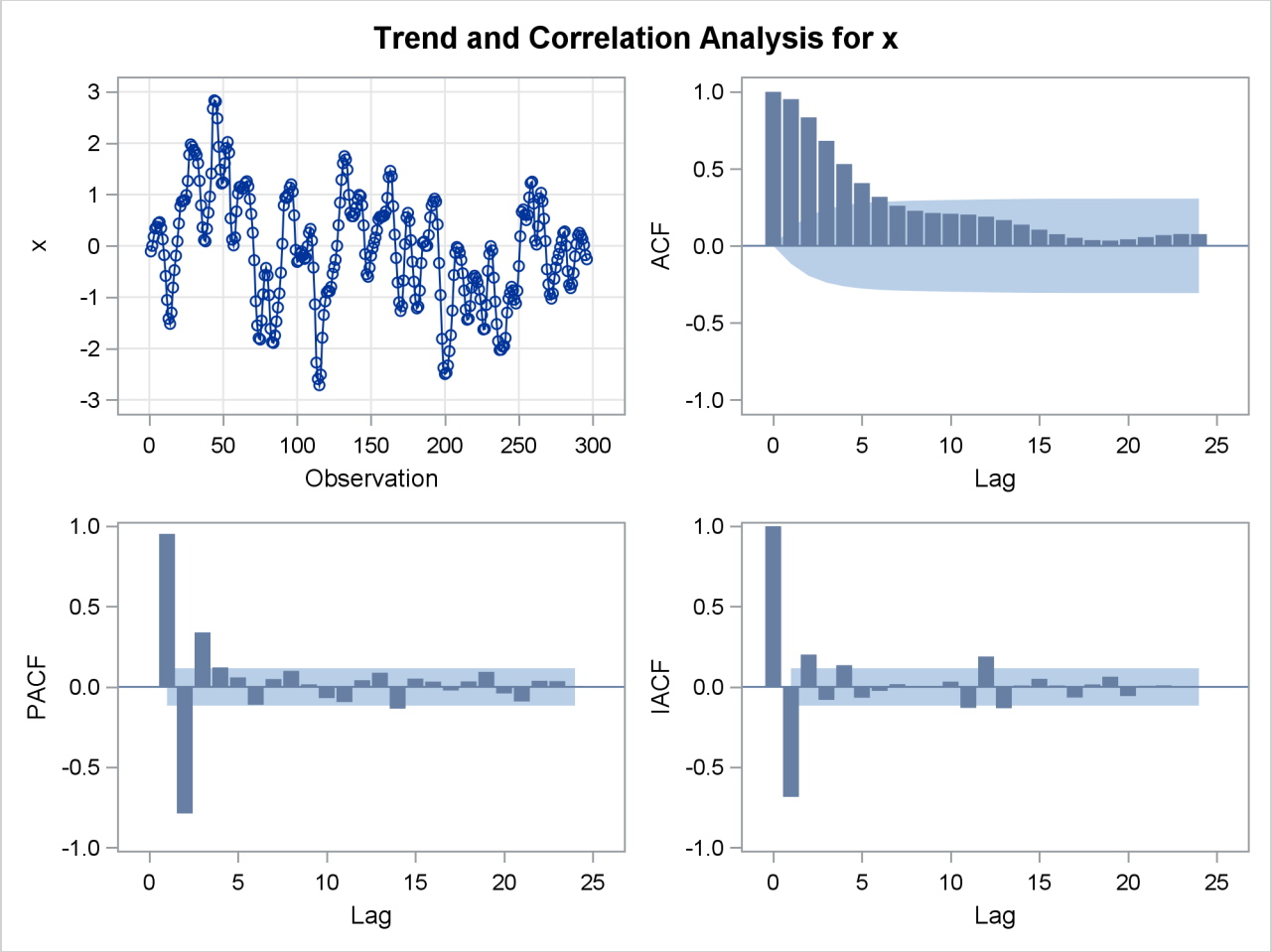
Output 8.3.1 IDENTIFY Statement Results for X

Gas Furnace Data
(Box and Jenkins, Series J)

The ARIMA Procedure

Name of Variable = x	
Mean of Working Series	-0.05683
Standard Deviation	1.070952
Number of Observations	296

Output 8.3.2 IDENTIFY Statement Results for X: Trend and Correlation



The ESTIMATE statement results for the AR(3) model for the input series X are shown in Output 8.3.3.

Output 8.3.3 Estimates of the AR(3) Model for X

Conditional Least Squares Estimation

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|	Lag
MU	-0.12280	0.10902	-1.13	0.2609	0
AR1,1	1.97607	0.05499	35.94	<.0001	1
AR1,2	-1.37499	0.09967	-13.80	<.0001	2
AR1,3	0.34336	0.05502	6.24	<.0001	3

Constant Estimate	-0.00682
Variance Estimate	0.035797
Std Error Estimate	0.1892
AIC	-141.667
SBC	-126.906
Number of Residuals	296

Output 8.3.3 *continued*

Model for variable x	
Estimated Mean	-0.1228

Autoregressive Factors	
Factor 1:	1 - 1.97607 B**(1) + 1.37499 B**(2) - 0.34336 B**(3)

The IDENTIFY statement results for the dependent series Y cross-correlated with the input series X are shown in Output 8.3.4, Output 8.3.5, Output 8.3.6, and Output 8.3.7. Since a model has been fit to X, both Y and X are prewhitened by this model before the sample cross-correlations are computed.

Output 8.3.4 Summary Table: Y Cross-Correlated with X

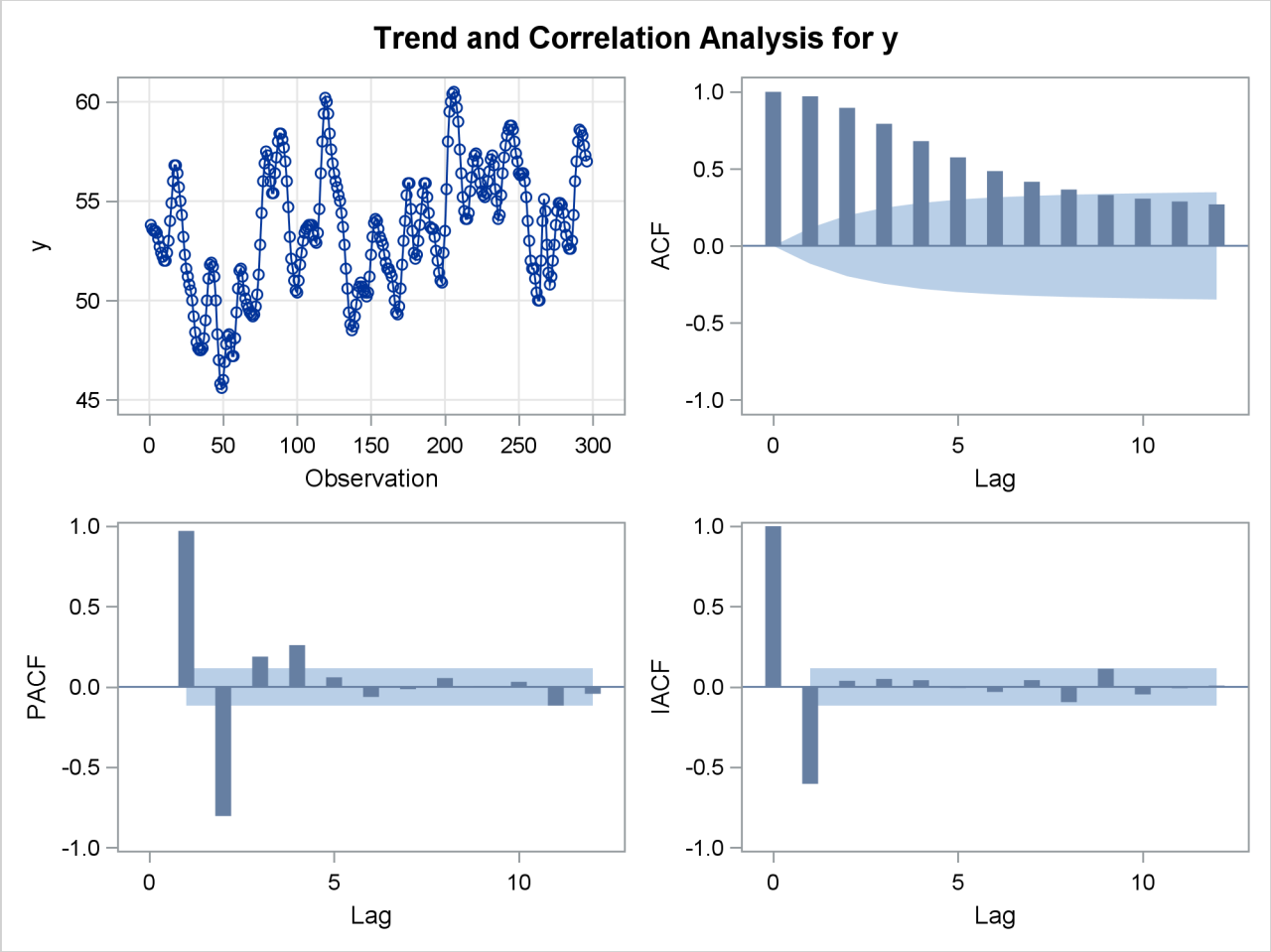
Correlation of y and x	
Number of Observations	296
Variance of transformed series y	0.131438
Variance of transformed series x	0.035357

Both series have been prewhitened.

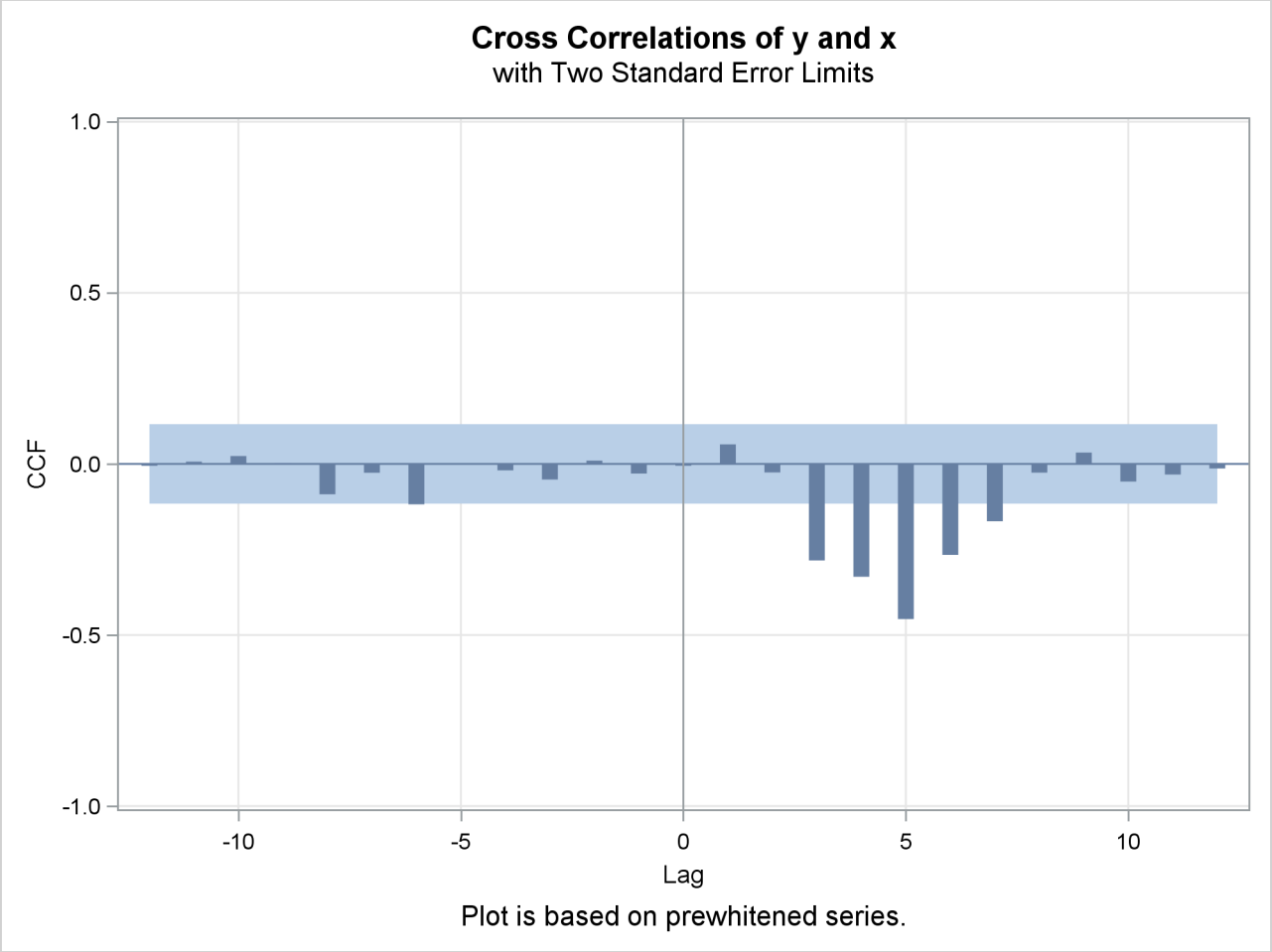
Output 8.3.5 Prewhitening Filter

Autoregressive Factors	
Factor 1:	1 - 1.97607 B**(1) + 1.37499 B**(2) - 0.34336 B**(3)

Output 8.3.6 IDENTIFY Statement Results for Y: Trend and Correlation



Output 8.3.7 IDENTIFY Statement for Y Cross-Correlated with X



The ESTIMATE statement results for the transfer function model with no structure on the noise term are shown in Output 8.3.8, Output 8.3.9, and Output 8.3.10.

Output 8.3.8 Estimation Output of the First Transfer Function Model

Conditional Least Squares Estimation

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|	Lag	Variable	Shift
MU	53.32256	0.04926	1082.51	<.0001	0	y	0
NUM1	-0.56467	0.22405	-2.52	0.0123	0	x	3
NUM1,1	0.42623	0.46472	0.92	0.3598	1	x	3
NUM1,2	0.29914	0.35506	0.84	0.4002	2	x	3
DEN1,1	0.60073	0.04101	14.65	<.0001	1	x	3

Constant Estimate	53.32256
Variance Estimate	0.702625
Std Error Estimate	0.838227
AIC	728.0754
SBC	746.442
Number of Residuals	291

Output 8.3.9 Model Summary: First Transfer Function Model

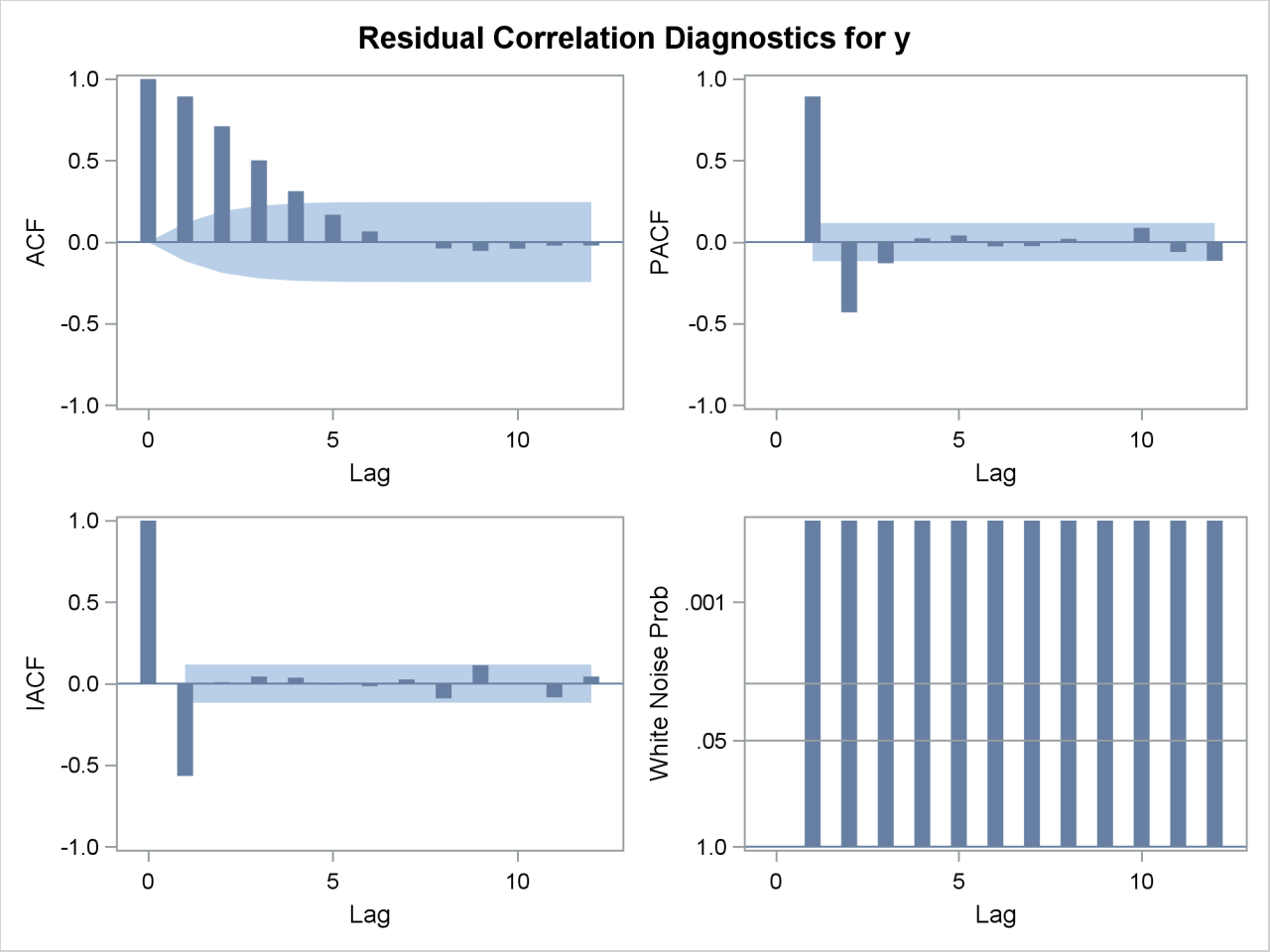
Model for variable y	
Estimated Intercept	53.32256

Input Number 1	
Input Variable	x
Shift	3

Numerator Factors
Factor 1: -0.5647 - 0.42623 B**(1) - 0.29914 B**(2)

Denominator Factors
Factor 1: 1 - 0.60073 B**(1)

Output 8.3.10 Residual Analysis: First Transfer Function Model



The residual correlation analysis suggests an AR(2) model for the noise part of the model. The ESTIMATE statement results for the final transfer function model with AR(2) noise are shown in Output 8.3.11.

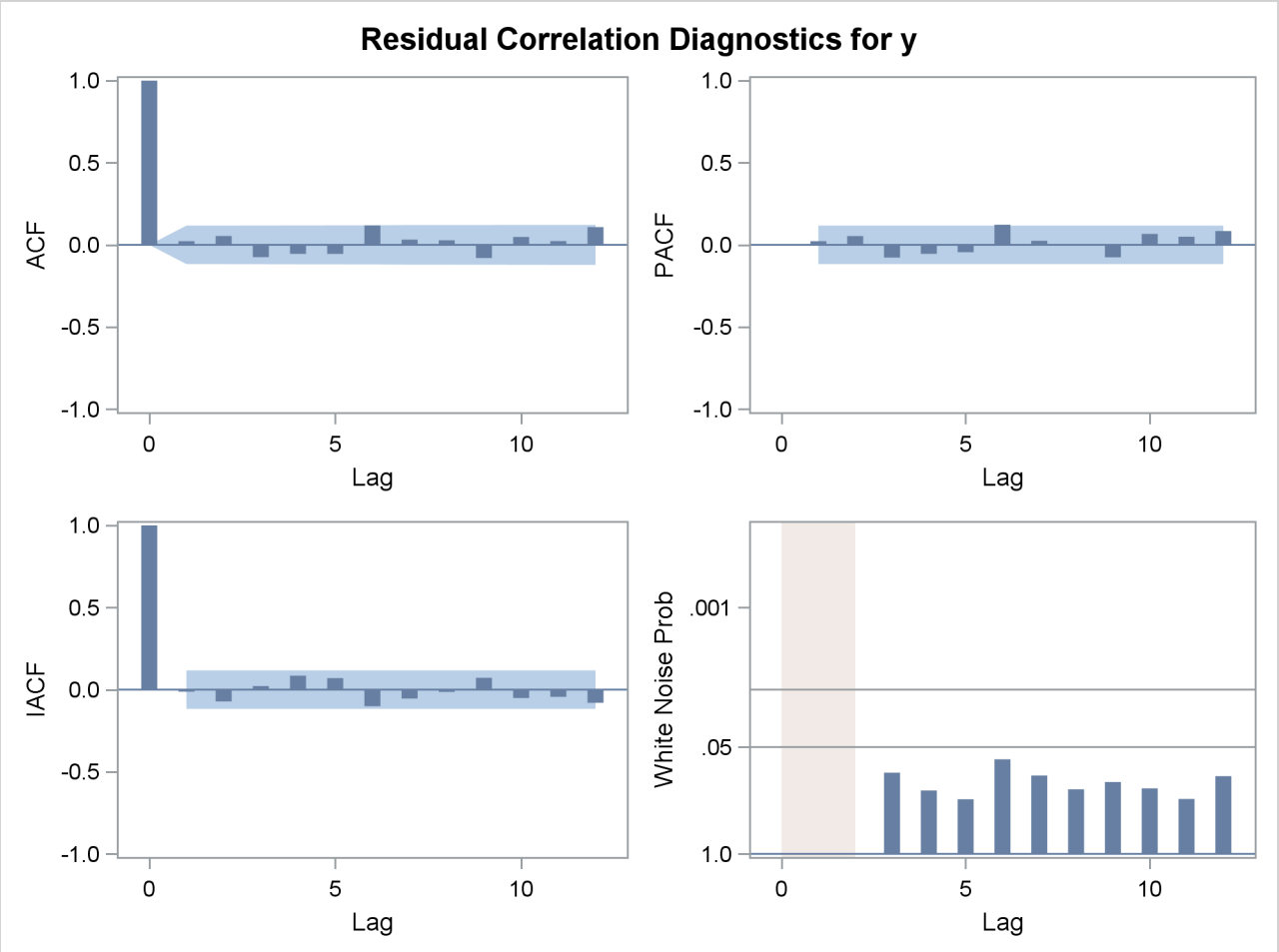
Output 8.3.11 Estimation Output of the Final Model

Conditional Least Squares Estimation

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|	Lag	Variable	Shift
MU	53.26304	0.11929	446.48	<.0001	0	y	0
AR1,1	1.53291	0.04754	32.25	<.0001	1	y	0
AR1,2	-0.63297	0.05006	-12.64	<.0001	2	y	0
NUM1	-0.53522	0.07482	-7.15	<.0001	0	x	3
NUM1,1	0.37603	0.10287	3.66	0.0003	1	x	3
NUM1,2	0.51895	0.10783	4.81	<.0001	2	x	3
DEN1,1	0.54841	0.03822	14.35	<.0001	1	x	3

Constant Estimate	5.329425
Variance Estimate	0.058828
Std Error Estimate	0.242544
AIC	8.292809
SBC	34.00607
Number of Residuals	291

Output 8.3.12 Residual Analysis of the Final Model



Output 8.3.13 Model Summary of the Final Model

Model for variable y	
Estimated Intercept	53.26304

Autoregressive Factors	
Factor 1:	1 - 1.53291 B**(1) + 0.63297 B**(2)

Input Number 1	
Input Variable	x
Shift	3

Numerator Factors	
Factor 1:	-0.5352 - 0.37603 B**(1) - 0.51895 B**(2)

Denominator Factors	
Factor 1:	1 - 0.54841 B**(1)

Example 8.4: An Intervention Model for Ozone Data

This example fits an intervention model to ozone data as suggested by Box and Tiao (1975). Notice that the response variable, OZONE, and the innovation, X1, are seasonally differenced. The final model for the differenced data is a multiple regression model with a moving-average structure assumed for the residuals.

The model is fit by maximum likelihood. The seasonal moving-average parameter and its standard error are fairly sensitive to which method is chosen to fit the model (Ansley and Newbold 1980; Davidson 1981); thus, fitting the model by the unconditional or conditional least squares method produces somewhat different estimates for these parameters.

Some missing values are appended to the end of the input data to generate additional values for the independent variables. Since the independent variables are not modeled, values for them must be available for any times at which predicted values are desired. In this case, predicted values are requested for 12 periods beyond the end of the data. Thus, values for X1, WINTER, and SUMMER must be given for 12 periods ahead.

The following statements read in the data and compute dummy variables for use as intervention inputs:

```
title1 'Intervention Data for Ozone Concentration';
title2 '(Box and Tiao, JASA 1975 P.70)';
data air;
   input ozone @@;
   label ozone  = 'Ozone Concentration'
         x1     = 'Intervention for post 1960 period'
         summer = 'Summer Months Intervention'
         winter = 'Winter Months Intervention';
   date = intnx( 'month', '31dec1954'd, _n_ );
   format date monyy.;
   month = month( date );
   year = year( date );
   x1 = year >= 1960;
   summer = ( 5 < month < 11 ) * ( year > 1965 );
   winter = ( year > 1965 ) - summer;
datalines;
```

```
2.7  2.0  3.6  5.0  6.5  6.1  5.9  5.0  6.4  7.4  8.2  3.9
4.1  4.5  5.5  3.8  4.8  5.6  6.3  5.9  8.7  5.3  5.7  5.7
3.0  3.4  4.9  4.5  4.0  5.7  6.3  7.1  8.0  5.2  5.0  4.7
3.7  3.1  2.5  4.0  4.1  4.6  4.4  4.2  5.1  4.6  4.4  4.0

   ... more lines ...
```

The following statements produce Output 8.4.1 through Output 8.4.3:

```
proc arima data=air;

   /* Identify and seasonally difference ozone series */
   identify var=ozone(12)
            crosscorr=( x1(12) summer winter ) noprint;

   /* Fit a multiple regression with a seasonal MA model */
   /*     by the maximum likelihood method               */
   estimate q=(1)(12) input=( x1 summer winter )
            noconstant method=ml;

   /* Forecast */
   forecast  lead=12 id=date interval=month;

run;
```

The ESTIMATE statement results are shown in Output 8.4.1 and Output 8.4.2.

Output 8.4.1 Parameter Estimates

Intervention Data for Ozone Concentration (Box and Tiao, JASA 1975 P.70)

The ARIMA Procedure

Maximum Likelihood Estimation

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|	Lag	Variable	Shift
MA1,1	-0.26684	0.06710	-3.98	<.0001	1	ozone	0
MA2,1	0.76665	0.05973	12.83	<.0001	12	ozone	0
NUM1	-1.33062	0.19236	-6.92	<.0001	0	x1	0
NUM2	-0.23936	0.05952	-4.02	<.0001	0	summer	0
NUM3	-0.08021	0.04978	-1.61	0.1071	0	winter	0

Variance Estimate	0.634506
Std Error Estimate	0.796559
AIC	501.7696
SBC	518.3602
Number of Residuals	204

Output 8.4.2 Model Summary

Model for variable ozone	
Period(s) of Differencing	12

Moving Average Factors	
Factor 1:	1 + 0.26684 B**(1)
Factor 2:	1 - 0.76665 B**(12)

Input Number 1	
Input Variable	x1
Period(s) of Differencing	12
Overall Regression Factor	-1.33062

The FORECAST statement results are shown in Output 8.4.3.

Output 8.4.3 Forecasts

Forecasts for variable ozone				
Obs	Forecast	Std Error	95% Confidence Limits	
217	1.4205	0.7966	-0.1407	2.9817
218	1.8446	0.8244	0.2287	3.4604
219	2.4567	0.8244	0.8408	4.0725
220	2.8590	0.8244	1.2431	4.4748
221	3.1501	0.8244	1.5342	4.7659
222	2.7211	0.8244	1.1053	4.3370
223	3.3147	0.8244	1.6989	4.9306
224	3.4787	0.8244	1.8629	5.0946
225	2.9405	0.8244	1.3247	4.5564
226	2.3587	0.8244	0.7429	3.9746
227	1.8588	0.8244	0.2429	3.4746
228	1.2898	0.8244	-0.3260	2.9057

Example 8.5: Using Diagnostics to Identify ARIMA Models

Fitting ARIMA models is as much an art as it is a science. The ARIMA procedure has diagnostic options to help tentatively identify the orders of both stationary and nonstationary ARIMA processes.

Consider the Series A in Box, Jenkins, and Reinsel (1994), which consists of 197 concentration readings taken every two hours from a chemical process. Let Series A be a data set that contains these readings in a variable named X. The following SAS statements use the SCAN option of the IDENTIFY statement to generate Output 8.5.1 and Output 8.5.2. See "The SCAN Method" on page 268 for details of the SCAN method.

```
/*-- Order Identification Diagnostic with SCAN Method --*/
proc arima data=SeriesA;
   identify var=x scan;
run;
```

Output 8.5.1 Example of SCAN Tables

SERIES A: Chemical Process Concentration Readings

The ARIMA Procedure

Squared Canonical Correlation Estimates

Lags	MA 0	MA 1	MA 2	MA 3	MA 4	MA 5
AR 0	0.3263	0.2479	0.1654	0.1387	0.1183	0.1417
AR 1	0.0643	0.0012	0.0028	<.0001	0.0051	0.0002
AR 2	0.0061	0.0027	0.0021	0.0011	0.0017	0.0079
AR 3	0.0072	<.0001	0.0007	0.0005	0.0019	0.0021
AR 4	0.0049	0.0010	0.0014	0.0014	0.0039	0.0145
AR 5	0.0202	0.0009	0.0016	<.0001	0.0126	0.0001

SCAN Chi-Square[1] Probability Values

Lags	MA 0	MA 1	MA 2	MA 3	MA 4	MA 5
AR 0	<.0001	<.0001	<.0001	0.0007	0.0037	0.0024
AR 1	0.0003	0.6649	0.5194	0.9235	0.3993	0.8528
AR 2	0.2754	0.5106	0.5860	0.7346	0.6782	0.2766
AR 3	0.2349	0.9812	0.7667	0.7861	0.6810	0.6546
AR 4	0.3297	0.7154	0.7113	0.6995	0.5807	0.2205
AR 5	0.0477	0.7254	0.6652	0.9576	0.2660	0.9168

In Output 8.5.1, there is one (maximal) rectangular region in which all the elements are insignificant with 95% confidence. This region has a vertex at (1,1). Output 8.5.2 gives recommendations based on the significance level specified by the ALPHA=*siglevel* option.

Output 8.5.2 Example of SCAN Option Tentative Order Selection

ARMA(p+d,q) Tentative Order Selection Tests

SCAN

p+d	q
1	1

(5% Significance Level)

Another order identification diagnostic is the extended sample autocorrelation function or ESACF method. See "The ESACF Method" on page 265 for details of the ESACF method.

The following statements generate Output 8.5.3 and Output 8.5.4:

```
/*-- Order Identification Diagnostic with ESACF Method --*/
proc arima data=SeriesA;
   identify var=x esacf;
run;
```

Output 8.5.3 Example of ESACF Tables

SERIES A: Chemical Process Concentration Readings

The ARIMA Procedure

Extended Sample Autocorrelation Function						
Lags	**MA 0**	**MA 1**	**MA 2**	**MA 3**	**MA 4**	**MA 5**
AR 0	0.5702	0.4951	0.3980	0.3557	0.3269	0.3498
AR 1	-0.3907	0.0425	-0.0605	-0.0083	-0.0651	-0.0127
AR 2	-0.2859	-0.2699	-0.0449	0.0089	-0.0509	-0.0140
AR 3	-0.5030	-0.0106	0.0946	-0.0137	-0.0148	-0.0302
AR 4	-0.4785	-0.0176	0.0827	-0.0244	-0.0149	-0.0421
AR 5	-0.3878	-0.4101	-0.1651	0.0103	-0.1741	-0.0231

ESACF Probability Values						
Lags	**MA 0**	**MA 1**	**MA 2**	**MA 3**	**MA 4**	**MA 5**
AR 0	<.0001	<.0001	0.0001	0.0014	0.0053	0.0041
AR 1	<.0001	0.5974	0.4622	0.9198	0.4292	0.8768
AR 2	<.0001	0.0002	0.6106	0.9182	0.5683	0.8592
AR 3	<.0001	0.9022	0.2400	0.8713	0.8930	0.7372
AR 4	<.0001	0.8380	0.3180	0.7737	0.8913	0.6213
AR 5	<.0001	<.0001	0.0765	0.9142	0.1038	0.8103

In Output 8.5.3, there are three right-triangular regions in which all elements are insignificant at the 5% level. The triangles have vertices (1,1), (3,1), and (4,1). Since the triangle at (1,1) covers more insignificant terms, it is recommended first. Similarly, the remaining recommendations are ordered by the number of insignificant terms contained in the triangle. Output 8.5.4 gives recommendations based on the significance level specified by the ALPHA=*siglevel* option.

Output 8.5.4 Example of ESACF Option Tentative Order Selection

ARMA(p+d,q) Tentative Order Selection Tests

ESACF

p+d	**q**
1	1
3	1
4	1

(5% Significance Level)

If you also specify the SCAN option in the same IDENTIFY statement, the two recommendations are printed side by side:

```
/*-- Combination of SCAN and ESACF Methods --*/
proc arima data=SeriesA;
   identify var=x scan esacf;
run;
```

Output 8.5.5 shows the results.

Output 8.5.5 Example of SCAN and ESACF Option Combined

SERIES A: Chemical Process Concentration Readings

The ARIMA Procedure

ARMA(p+d,q) Tentative Order Selection Tests			
SCAN		ESACF	
p+d	q	p+d	q
1	1	1	1
		3	1
		4	1

(5% Significance Level)

From Output 8.5.5, the autoregressive and moving-average orders are tentatively identified by both SCAN and ESACF tables to be $(p + d, q)$=(1,1). Because both the SCAN and ESACF indicate a $p + d$ term of 1, a unit root test should be used to determine whether this autoregressive term is a unit root. Since a moving-average term appears to be present, a large autoregressive term is appropriate for the augmented Dickey-Fuller test for a unit root.

Submitting the following statements generates Output 8.5.6:

```
/*-- Augmented Dickey-Fuller Unit Root Tests --*/
proc arima data=SeriesA;
   identify var=x stationarity=(adf=(5,6,7,8));
run;
```

Output 8.5.6 Example of STATIONARITY Option Output

SERIES A: Chemical Process Concentration Readings

The ARIMA Procedure

Augmented Dickey-Fuller Unit Root Tests							
Type	Lags	Rho	Pr < Rho	Tau	Pr < Tau	F	Pr > F
Zero Mean	5	0.0403	0.6913	0.42	0.8024		
	6	0.0479	0.6931	0.63	0.8508		
	7	0.0376	0.6907	0.49	0.8200		
	8	0.0354	0.6901	0.48	0.8175		
Single Mean	5	-18.4550	0.0150	-2.67	0.0821	3.67	0.1367
	6	-10.8939	0.1043	-2.02	0.2767	2.27	0.4931
	7	-10.9224	0.1035	-1.93	0.3172	2.00	0.5605
	8	-10.2992	0.1208	-1.83	0.3650	1.81	0.6108
Trend	5	-18.4360	0.0871	-2.66	0.2561	3.54	0.4703
	6	-10.8436	0.3710	-2.01	0.5939	2.04	0.7694
	7	-10.7427	0.3773	-1.90	0.6519	1.91	0.7956
	8	-10.0370	0.4236	-1.79	0.7081	1.74	0.8293

The preceding test results show that a unit root is very likely given that none of the *p*-values are small enough to cause you to reject the null hypothesis that the series has a unit root. Based on this test and the previous results, the series should be differenced, and an ARIMA(0,1,1) would be a good choice for a tentative model for Series A.

Using the recommendation that the series be differenced, the following statements generate Output 8.5.7:

```
/*-- Minimum Information Criterion --*/
proc arima data=SeriesA;
   identify var=x(1) minic;
run;
```

Output 8.5.7 Example of MINIC Table

SERIES A: Chemical Process Concentration Readings

The ARIMA Procedure

Minimum Information Criterion						
Lags	**MA 0**	**MA 1**	**MA 2**	**MA 3**	**MA 4**	**MA 5**
AR 0	-2.05761	-2.3497	-2.32358	-2.31298	-2.30967	-2.28528
AR 1	-2.23291	-2.32345	-2.29665	-2.28644	-2.28356	-2.26011
AR 2	-2.23947	-2.30313	-2.28084	-2.26065	-2.25685	-2.23458
AR 3	-2.25092	-2.28088	-2.25567	-2.23455	-2.22997	-2.20769
AR 4	-2.25934	-2.2778	-2.25363	-2.22983	-2.20312	-2.19531
AR 5	-2.2751	-2.26805	-2.24249	-2.21789	-2.19667	-2.17426

The error series is estimated by using an AR(7) model, and the minimum of this MINIC table is $BIC(0, 1)$. This diagnostic confirms the previous result which indicates that an ARIMA(0,1,1) is a tentative model for Series A.

If you also specify the SCAN or MINIC option in the same IDENTIFY statement as follows, the BIC associated with the SCAN table and ESACF table recommendations is listed. Output 8.5.8 shows the results.

```
/*-- Combination of MINIC, SCAN and ESACF Options --*/
proc arima data=SeriesA;
   identify var=x(1) minic scan esacf;
run;
```

Output 8.5.8 Example of SCAN, ESACF, MINIC Options Combined

SERIES A: Chemical Process Concentration Readings

The ARIMA Procedure

ARMA(p+d,q) Tentative Order Selection Tests					
SCAN			**ESACF**		
p+d	**q**	**BIC**	**p+d**	**q**	**BIC**
0	1	-2.3497	0	1	-2.3497
			1	1	-2.32345

(5% Significance Level)

Example 8.6: Detection of Level Changes in the Nile River Data

This example shows how to use the OUTLIER statement to detect changes in the dynamics of the time series being modeled. The time series used here is discussed in De Jong and Penzer (1998). The data consist of readings of the annual flow volume of the Nile River at Aswan from 1871 to 1970. These data have also been studied by Cobb (1978). These studies indicate that river flow levels in the years 1877 and 1913 are strong candidates for additive outliers and that there was a shift in the flow levels starting from the year 1899. This shift in 1899 is attributed partly to the weather changes and partly to the start of construction work for a new dam at Aswan. The following DATA step statements create the input data set.

```
data nile;
   input level @@;
   year = intnx( 'year', '1jan1871'd, _n_-1 );
   format year year4.;
datalines;
1120  1160  963  1210  1160  1160  813  1230   1370  1140
995   935   1110 994   1020  960   1180 799    958   1140
1100  1210  1150 1250  1260  1220  1030 1100   774   840

   ... more lines ...
```

The following program fits an ARIMA model, ARIMA(0,1,1), similar to the structural model suggested in De Jong and Penzer (1998). This model is also suggested by the usual correlation analysis of the series. By default, the OUTLIER statement requests detection of additive outliers and level shifts, assuming that the series follows the estimated model.

```
/*-- ARIMA(0, 1, 1) Model --*/
proc arima data=nile;
   identify var=level(1);
   estimate q=1 noint method=ml;
   outlier maxnum= 5 id=year;
run;
```

The outlier detection output is shown in Output 8.6.1.

Output 8.6.1 ARIMA(0, 1, 1) Model

The ARIMA Procedure

Outlier Detection Summary	
Maximum number searched	5
Number found	5
Significance used	0.05

Output 8.6.1 *continued*

Outlier Details

Obs	Time ID	Type	Estimate	Chi-Square	Approx Prob>ChiSq
29	1899	Shift	-315.75346	13.13	0.0003
43	1913	Additive	-403.97105	11.83	0.0006
7	1877	Additive	-335.49351	7.69	0.0055
94	1964	Additive	305.03568	6.16	0.0131
18	1888	Additive	-287.81484	6.00	0.0143

Note that the first three outliers detected are indeed the ones discussed earlier. You can include the shock signatures that correspond to these three outliers in the Nile data set as follows:

```
data nile;
   set nile;
   AO1877 = ( year = '1jan1877'd );
   AO1913 = ( year = '1jan1913'd );
   LS1899 = ( year >= '1jan1899'd );
run;
```

Now you can refine the earlier model by including these outliers. After examining the parameter estimates and residuals (not shown) of the ARIMA(0,1,1) model with these regressors, the following stationary MA1 model (with regressors) appears to fit the data well:

```
/*-- MA1 Model with Outliers --*/
proc arima data=nile;
   identify var=level
            crosscorr=( AO1877 AO1913 LS1899 );
   estimate q=1
            input=( AO1877 AO1913 LS1899 )
            method=ml;
   outlier maxnum=5 alpha=0.01 id=year;
run;
```

The relevant outlier detection process output is shown in Output 8.6.2. No outliers, at significance level 0.01, were detected.

Output 8.6.2 MA1 Model with Outliers

The ARIMA Procedure

Outlier Detection Summary	
Maximum number searched	5
Number found	0
Significance used	0.01

Example 8.7: Iterative Outlier Detection

This example illustrates the iterative nature of the outlier detection process. This is done by using a simple test example where an additive outlier at observation number 50 and a level shift at observation number 100 are artificially introduced in the international airline passenger data used in Example 8.2. The following

DATA step shows the modifications introduced in the data set:

```
data airline;
   set sashelp.air;
   logair = log(air);
   if _n_ = 50 then logair = logair - 0.25;
   if _n_ >= 100 then logair = logair + 0.5;
run;
```

In Example 8.2 the airline model, ARIMA$(0,1,1)\times(0,1,1)_{12}$, was seen to be a good fit to the unmodified log-transformed airline passenger series. The preliminary identification steps (not shown) again suggest the airline model as a suitable initial model for the modified data. The following statements specify the airline model and request an outlier search.

```
/*-- Outlier Detection --*/
proc arima data=airline;
   identify var=logair( 1, 12 )  noprint;
   estimate q= (1)(12) noint method= ml;
   outlier maxnum=3 alpha=0.01;
run;
```

The outlier detection output is shown in Output 8.7.1.

Output 8.7.1 Initial Model

The ARIMA Procedure

Outlier Detection Summary	
Maximum number searched	3
Number found	3
Significance used	0.01

Outlier Details

Obs	Type	Estimate	Chi-Square	Approx Prob>ChiSq
100	Shift	0.49325	199.36	<.0001
50	Additive	-0.27508	104.78	<.0001
135	Additive	-0.10488	13.08	0.0003

Clearly the level shift at observation number 100 and the additive outlier at observation number 50 are the dominant outliers. Moreover, the corresponding regression coefficients seem to correctly estimate the size and sign of the change. You can augment the airline data with these two regressors, as follows:

```
data airline;
   set airline;
   if _n_ = 50 then AO = 1;
   else AO = 0.0;
   if _n_ >= 100 then LS  = 1;
   else LS = 0.0;
run;
```

You can now refine the previous model by including these regressors, as follows. Note that the differencing order of the dependent series is matched to the differencing orders of the outlier regressors to get the correct "effective" outlier signatures.

```
/*-- Airline Model with Outliers --*/
proc arima data=airline;
   identify var=logair(1, 12)
            crosscorr=( AO(1, 12) LS(1, 12) )
            noprint;
   estimate q= (1)(12) noint
            input=( AO LS )
            method=ml plot;
   outlier maxnum=3 alpha=0.01;
run;
```

The outlier detection results are shown in Output 8.7.2.

Output 8.7.2 Airline Model with Outliers

The ARIMA Procedure

Outlier Detection Summary	
Maximum number searched	3
Number found	3
Significance used	0.01

Outlier Details

Obs	Type	Estimate	Chi-Square	Approx Prob>ChiSq
135	Additive	-0.10310	12.63	0.0004
62	Additive	-0.08872	12.33	0.0004
29	Additive	0.08686	11.66	0.0006

The output shows that a few outliers still remain to be accounted for and that the model could be refined further.

References

Akaike, H. (1974). "A New Look at the Statistical Model Identification." *IEEE Transactions on Automatic Control* AC-19:716–723.

Anderson, T. W. (1971). *The Statistical Analysis of Time Series*. New York: John Wiley & Sons.

Andrews, D. F., and Herzberg, A. M. (1985). *A Collection of Problems from Many Fields for the Student and Research Worker*. New York: Springer-Verlag.

Ansley, C. F. (1979). "An Algorithm for the Exact Likelihood of a Mixed Autoregressive–Moving Average Process." *Biometrika* 66:59–65.

Ansley, C. F., and Newbold, P. (1980). "Finite Sample Properties of Estimators for Autoregressive Moving-Average Models." *Journal of Econometrics* 13:159–183.

Bhansali, R. J. (1980). "Autoregressive and Window Estimates of the Inverse Correlation Function." *Biometrika* 67:551–566.

Box, G. E. P., and Jenkins, G. M. (1976). *Time Series Analysis: Forecasting and Control*. Rev. ed. San Francisco: Holden-Day.

Box, G. E. P., Jenkins, G. M., and Reinsel, G. C. (1994). *Time Series Analysis: Forecasting and Control*. 3rd ed. Englewood Cliffs, NJ: Prentice-Hall.

Box, G. E. P., and Tiao, G. C. (1975). "Intervention Analysis with Applications to Economic and Environmental Problems." *Journal of the American Statistical Association* 70:70–79.

Brocklebank, J. C., and Dickey, D. A. (2003). *SAS for Forecasting Time Series*. 2nd ed. Cary, NC: SAS Institute Inc.

Brockwell, P. J., and Davis, R. A. (1991). *Time Series: Theory and Methods*. 2nd ed. New York: Springer-Verlag.

Chatfield, C. (1980). "Inverse Autocorrelations." *Journal of the Royal Statistical Society, Series A* 142:363–377.

Choi, B. (1992). *ARMA Model Identification*. New York: Springer-Verlag.

Cleveland, W. S. (1972). "The Inverse Autocorrelations of a Time Series and Their Applications." *Technometrics* 14:277.

Cobb, G. W. (1978). "The Problem of the Nile: Conditional Solution to a Change Point Problem." *Biometrika* 65:243–251.

Davidson, J. (1981). "Problems with the Estimation of Moving Average Models." *Journal of Econometrics* 16:295.

Davies, N., Triggs, C. M., and Newbold, P. (1977). "Significance Levels of the Box-Pierce Portmanteau Statistic in Finite Samples." *Biometrika* 64:517–522.

De Jong, P., and Penzer, J. (1998). "Diagnosing Shocks in Time Series." *Journal of the American Statistical Association* 93:796–806.

Dickey, D. A. (1976). *Estimation and Testing of Nonstationary Time Series*. Ph.D. diss., Iowa State University.

Dickey, D. A., and Fuller, W. A. (1979). "Distribution of the Estimators for Autoregressive Time Series with a Unit Root." *Journal of the American Statistical Association* 74:427–431.

Dickey, D. A., Hasza, D. P., and Fuller, W. A. (1984). "Testing for Unit Roots in Seasonal Time Series." *Journal of the American Statistical Association* 79:355–367.

Dunsmuir, W. (1984). "Large Sample Properties of Estimation in Time Series Observed at Unequally Spaced Times." In *Time Series Analysis of Irregularly Observed Data*, edited by E. Parzen, 58–77. Vol. 25 of Lecture Notes in Statistics. New York: Springer-Verlag.

Findley, D. F., Monsell, B. C., Bell, W. R., Otto, M. C., and Chen, B. C. (1998). "New Capabilities and Methods of the X-12-ARIMA Seasonal Adjustment Program." *Journal of Business and Economic Statistics* 16:127–176.

Fuller, W. A. (1976). *Introduction to Statistical Time Series*. New York: John Wiley & Sons.

Hamilton, J. D. (1994). *Time Series Analysis*. Princeton, NJ: Princeton University Press.

Hannan, E. J., and Rissanen, J. (1982). "Recursive Estimation of Mixed Autoregressive Moving Average Order." *Biometrika* 69:81–94.

Harvey, A. C. (1981). *Time Series Models*. New York: John Wiley & Sons.

Jones, R. H. (1980). "Maximum Likelihood Fitting of ARMA Models to Time Series with Missing Observations." *Technometrics* 22:389–396.

Kohn, R., and Ansley, C. F. (1985). "Efficient Estimation and Prediction in Time Series Regression Models." *Biometrika* 72:694–697.

Ljung, G. M., and Box, G. E. P. (1978). "On a Measure of Lack of Fit in Time Series Models." *Biometrika* 65:297–303.

Montgomery, D. C., and Johnson, L. A. (1976). *Forecasting and Time Series Analysis*. New York: McGraw-Hill.

Morf, M., Sidhu, G. S., and Kailath, T. (1974). "Some New Algorithms for Recursive Estimation on Constant Linear Discrete Time Systems." *IEEE Transactions on Automatic Control* 19:315–323.

Nelson, C. R. (1973). *Applied Time Series for Managerial Forecasting*. San Francisco: Holden-Day.

Newbold, P. (1981). "Some Recent Developments in Time Series Analysis." *International Statistical Review* 49:53–66.

Newton, H. J., and Pagano, M. (1983). "The Finite Memory Prediction of Covariance Stationary Time Series." *SIAM Journal on Scientific and Statistical Computing* 4:330–339.

Pankratz, A. (1983). *Forecasting with Univariate Box-Jenkins Models: Concepts and Cases*. New York: John Wiley & Sons.

Pankratz, A. (1991). *Forecasting with Dynamic Regression Models*. New York: John Wiley & Sons.

Pearlman, J. G. (1980). "An Algorithm for the Exact Likelihood of a High-Order Autoregressive–Moving Average Process." *Biometrika* 67:232–233.

Priestley, M. B. (1981). *Spectral Analysis and Time Series*. London: Academic Press.

Schwarz, G. (1978). "Estimating the Dimension of a Model." *Annals of Statistics* 6:461–464.

Stoffer, D. S., and Toloi, C. M. C. (1992). "A Note on the Ljung-Box-Pierce Portmanteau Statistic with Missing Data." *Statistics and Probability Letters* 13:391–396.

Tsay, R. S., and Tiao, G. C. (1984). "Consistent Estimates of Autoregressive Parameters and Extended Sample Autocorrelation Function for Stationary and Nonstationary ARMA Models." *Journal of the American Statistical Association* 79:84–96.

Tsay, R. S., and Tiao, G. C. (1985). "Use of Canonical Analysis in Time Series Model Identification." *Biometrika* 72:299–315.

Woodfield, T. J. (1987). "Time Series Intervention Analysis Using SAS Software." In *Proceedings of the Twelfth Annual SAS Users Group International Conference*, 331–339. Cary, NC: SAS Institute Inc. http://www.sascommunity.org/sugi/SUGI87/Sugi-12-57%20Woodfield.pdf.

Chapter 9
The AUTOREG Procedure

Contents

Overview: AUTOREG Procedure **332**
Getting Started: AUTOREG Procedure **334**
Regression with Autocorrelated Errors 334
Forecasting Autoregressive Error Models 341
Testing for Autocorrelation 342
Stepwise Autoregression 345
Testing for Heteroscedasticity 347
Heteroscedasticity and GARCH Models 351
Syntax: AUTOREG Procedure **355**
Functional Summary 355
PROC AUTOREG Statement 359
BY Statement 360
CLASS Statement (Experimental) 360
MODEL Statement 361
HETERO Statement 380
NLOPTIONS Statement 382
OUTPUT Statement 382
RESTRICT Statement 384
TEST Statement 385
Details: AUTOREG Procedure **387**
Missing Values 387
Autoregressive Error Model 387
Alternative Autocorrelation Correction Methods 391
GARCH Models 392
Heteroscedasticity- and Autocorrelation-Consistent Covariance Matrix Estimator . . . 397
Goodness-of-Fit Measures and Information Criteria 401
Testing 403
Predicted Values 427
OUT= Data Set 431
OUTEST= Data Set 431
Printed Output 433
ODS Table Names 434
ODS Graphics 437
Examples: AUTOREG Procedure **438**
Example 9.1: Analysis of Real Output Series 438
Example 9.2: Comparing Estimates and Models 442

Example 9.3: Lack-of-Fit Study 446
Example 9.4: Missing Values 450
Example 9.5: Money Demand Model 454
Example 9.6: Estimation of ARCH(2) Process 458
Example 9.7: Estimation of GARCH-Type Models 461
Example 9.8: Illustration of ODS Graphics 466
References **475**

Overview: AUTOREG Procedure

The AUTOREG procedure estimates and forecasts linear regression models for time series data when the errors are autocorrelated or heteroscedastic. The autoregressive error model is used to correct for autocorrelation, and the generalized autoregressive conditional heteroscedasticity (GARCH) model and its variants are used to model and correct for heteroscedasticity.

When time series data are used in regression analysis, often the error term is not independent through time. Instead, the errors are *serially correlated* (*autocorrelated*). If the error term is autocorrelated, the efficiency of ordinary least squares (OLS) parameter estimates is adversely affected and standard error estimates are biased.

The autoregressive error model corrects for serial correlation. The AUTOREG procedure can fit autoregressive error models of any order and can fit subset autoregressive models. You can also specify stepwise autoregression to select the autoregressive error model automatically.

To diagnose autocorrelation, the AUTOREG procedure produces generalized Durbin-Watson (DW) statistics and their marginal probabilities. Exact p-values are reported for generalized DW tests to any specified order. For models with lagged dependent regressors, PROC AUTOREG performs the Durbin t test and the Durbin h test for first-order autocorrelation and reports their marginal significance levels.

Ordinary regression analysis assumes that the error variance is the same for all observations. When the error variance is not constant, the data are said to be *heteroscedastic*, and ordinary least squares estimates are inefficient. Heteroscedasticity also affects the accuracy of forecast confidence limits. More efficient use of the data and more accurate prediction error estimates can be made by models that take the heteroscedasticity into account.

To test for heteroscedasticity, the AUTOREG procedure uses the portmanteau Q test statistics (McLeod and Li 1983), Engle's Lagrange multiplier tests (Engle 1982), tests from Lee and King (1993), and tests from Wong and Li (1995). Test statistics and significance p-values are reported for conditional heteroscedasticity at lags 1 through 12. The Bera-Jarque normality test statistic and its significance level are also reported to test for conditional nonnormality of residuals. The following tests for independence are also supported by the AUTOREG procedure for residual analysis and diagnostic checking: Brock-Dechert-Scheinkman (BDS) test, runs test, turning point test, and the rank version of the von Neumann ratio test.

The family of GARCH models provides a means of estimating and correcting for the changing variability of the data. The GARCH process assumes that the errors, although uncorrelated, are not independent, and it models the conditional error variance as a function of the past realizations of the series.

The AUTOREG procedure supports the following variations of the GARCH models:

- generalized ARCH (GARCH)
- integrated GARCH (IGARCH)
- exponential GARCH (EGARCH)
- quadratic GARCH (QGARCH)
- threshold GARCH (TGARCH)
- power GARCH (PGARCH)
- GARCH-in-mean (GARCH-M)

For GARCH-type models, the AUTOREG procedure produces the conditional prediction error variances in addition to parameter and covariance estimates.

The AUTOREG procedure can also analyze models that combine autoregressive errors and GARCH-type heteroscedasticity. PROC AUTOREG can output predictions of the conditional mean and variance for models with autocorrelated disturbances and changing conditional error variances over time.

Four estimation methods are supported for the autoregressive error model:

- Yule-Walker
- iterated Yule-Walker
- unconditional least squares
- exact maximum likelihood

The maximum likelihood method is used for GARCH models and for mixed AR-GARCH models.

The AUTOREG procedure produces forecasts and forecast confidence limits when future values of the independent variables are included in the input data set. PROC AUTOREG is a useful tool for forecasting because it uses the time series part of the model in addition to the systematic part in generating predicted values. The autoregressive error model takes into account recent departures from the trend in producing forecasts.

The AUTOREG procedure permits embedded missing values for the independent or dependent variables. The procedure should be used only for ordered and equally spaced time series data.

Getting Started: AUTOREG Procedure

Regression with Autocorrelated Errors

Ordinary regression analysis is based on several statistical assumptions. One key assumption is that the errors are independent of each other. However, with time series data, the ordinary regression residuals usually are correlated over time. It is not desirable to use ordinary regression analysis for time series data since the assumptions on which the classical linear regression model is based will usually be violated.

Violation of the independent errors assumption has three important consequences for ordinary regression. First, statistical tests of the significance of the parameters and the confidence limits for the predicted values are not correct. Second, the estimates of the regression coefficients are not as efficient as they would be if the autocorrelation were taken into account. Third, since the ordinary regression residuals are not independent, they contain information that can be used to improve the prediction of future values.

The AUTOREG procedure solves this problem by augmenting the regression model with an autoregressive model for the random error, thereby accounting for the autocorrelation of the errors. Instead of the usual regression model, the following autoregressive error model is used:

$$y_t = \mathbf{x}_t'\beta + \nu_t$$
$$\nu_t = -\varphi_1\nu_{t-1} - \varphi_2\nu_{t-2} - \ldots - \varphi_m\nu_{t-m} + \epsilon_t$$
$$\epsilon_t \sim \mathrm{IN}(0, \sigma^2)$$

The notation $\epsilon_t \sim \mathrm{IN}(0, \sigma^2)$ indicates that each ϵ_t is normally and independently distributed with mean 0 and variance σ^2.

By simultaneously estimating the regression coefficients β and the autoregressive error model parameters φ_i, the AUTOREG procedure corrects the regression estimates for autocorrelation. Thus, this kind of regression analysis is often called *autoregressive error correction* or *serial correlation correction*.

Example of Autocorrelated Data

A simulated time series is used to introduce the AUTOREG procedure. The following statements generate a simulated time series Y with second-order autocorrelation:

```
/* Regression with Autocorrelated Errors */
data a;
   ul = 0; ull = 0;
   do time = -10 to 36;
      u = + 1.3 * ul - .5 * ull + 2*rannor(12346);
      y = 10 + .5 * time + u;
      if time > 0 then output;
      ull = ul; ul = u;
   end;
run;
```

The series Y is a time trend plus a second-order autoregressive error. The model simulated is

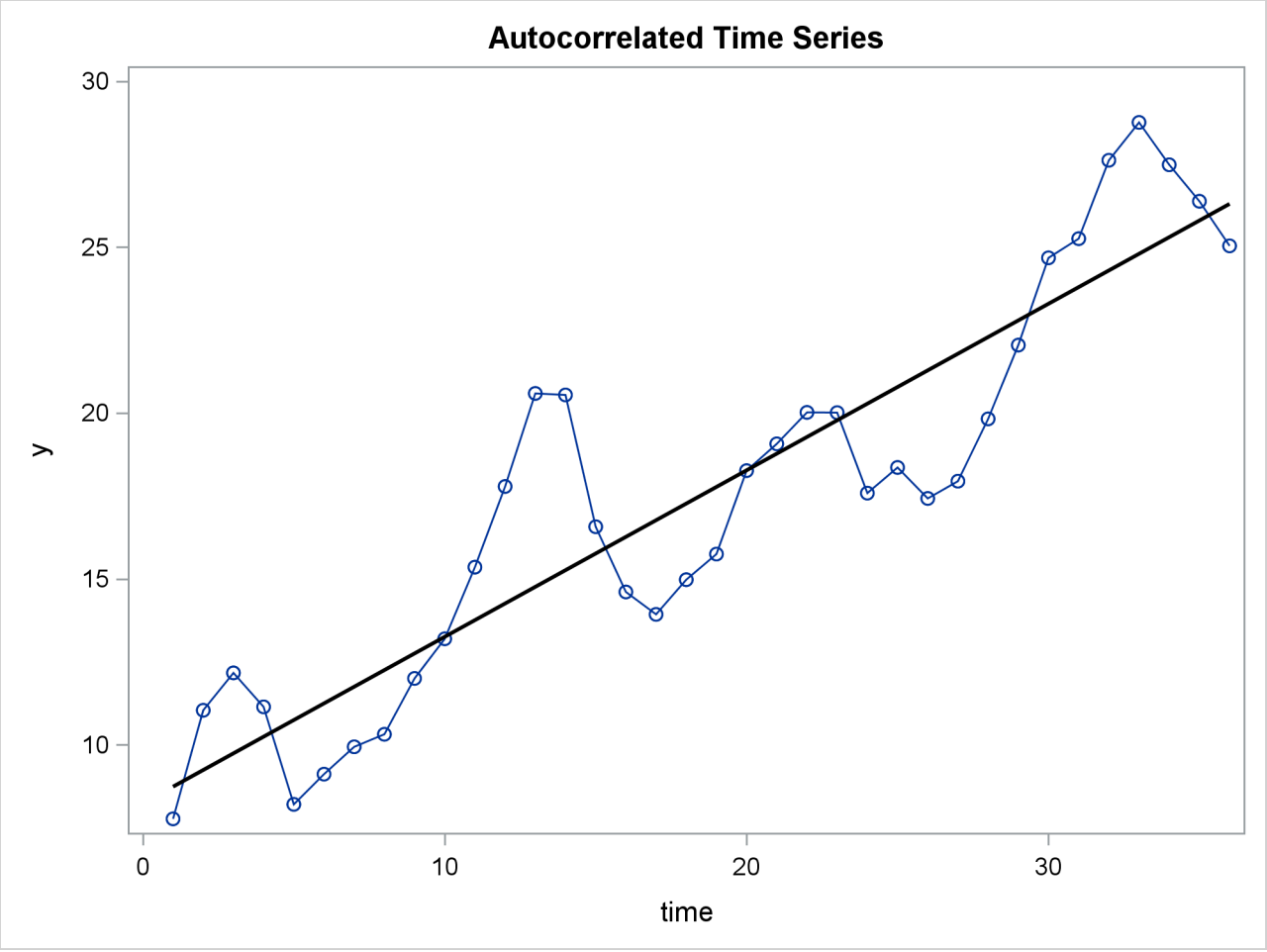
$$y_t = 10 + 0.5t + v_t$$
$$v_t = 1.3v_{t-1} - 0.5v_{t-2} + \epsilon_t$$
$$\epsilon_t \sim \text{IN}(0, 4)$$

The following statements plot the simulated time series Y. A linear regression trend line is shown for reference.

```
title 'Autocorrelated Time Series';
proc sgplot data=a noautolegend;
   series x=time y=y / markers;
   reg x=time y=y/  lineattrs=(color=black);
run;
```

The plot of series Y and the regression line are shown in Figure 9.1.

Figure 9.1 Autocorrelated Time Series



Note that when the series is above (or below) the OLS regression trend line, it tends to remain above (below) the trend for several periods. This pattern is an example of *positive autocorrelation*.

Time series regression usually involves independent variables other than a time trend. However, the simple time trend model is convenient for illustrating regression with autocorrelated errors, and the series Y shown in Figure 9.1 is used in the following introductory examples.

Ordinary Least Squares Regression

To use the AUTOREG procedure, specify the input data set in the PROC AUTOREG statement and specify the regression model in a MODEL statement. Specify the model by first naming the dependent variable and then listing the regressors after an equal sign, as is done in other SAS regression procedures. The following statements regress Y on TIME by using ordinary least squares:

```
proc autoreg data=a;
   model y = time;
run;
```

The AUTOREG procedure output is shown in Figure 9.2.

Figure 9.2 PROC AUTOREG Results for OLS Estimation

Autocorrelated Time Series

The AUTOREG Procedure

Dependent Variable	y

Ordinary Least Squares Estimates			
SSE	214.953429	DFE	34
MSE	6.32216	Root MSE	2.51439
SBC	173.659101	AIC	170.492063
MAE	2.01903356	AICC	170.855699
MAPE	12.5270666	HQC	171.597444
Durbin-Watson	0.4752	Total R-Square	0.8200

Parameter Estimates

Variable	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	8.2308	0.8559	9.62	<.0001
time	1	0.5021	0.0403	12.45	<.0001

The output first shows statistics for the model residuals. The model root mean square error (Root MSE) is 2.51, and the model R^2 (Total R-Square) is 0.82.

Other statistics shown are the sum of square errors (SSE), mean square error (MSE), mean absolute error (MAE), mean absolute percentage error (MAPE), error degrees of freedom (DFE, the number of observations minus the number of parameters), the information criteria SBC, HQC, AIC, and AICC, and the Durbin-Watson statistic. (Durbin-Watson statistics, MAE, MAPE, SBC, HQC, AIC, and AICC are discussed in the section "Goodness-of-Fit Measures and Information Criteria" on page 401.)

The output then shows a table of regression coefficients, with standard errors and t tests. The estimated model is

$$y_t = 8.23 + 0.502t + \epsilon_t$$
$$Est.\ Var(\epsilon_t) = 6.32$$

The OLS parameter estimates are reasonably close to the true values, but the estimated error variance, 6.32, is much larger than the true value, 4.

Autoregressive Error Model

The following statements regress Y on TIME with the errors assumed to follow a second-order autoregressive process. The order of the autoregressive model is specified by the NLAG=2 option. The Yule-Walker estimation method is used by default. The example uses the METHOD=ML option to specify the exact maximum likelihood method instead.

```
proc autoreg data=a;
   model y = time / nlag=2 method=ml;
run;
```

The first part of the results is shown in Figure 9.3. The initial OLS results are produced first, followed by estimates of the autocorrelations computed from the OLS residuals. The autocorrelations are also displayed graphically.

Figure 9.3 Preliminary Estimate for AR(2) Error Model

Autocorrelated Time Series

The AUTOREG Procedure

Dependent Variable	y

Ordinary Least Squares Estimates			
SSE	214.953429	DFE	34
MSE	6.32216	Root MSE	2.51439
SBC	173.659101	AIC	170.492063
MAE	2.01903356	AICC	170.855699
MAPE	12.5270666	HQC	171.597444
Durbin-Watson	0.4752	Total R-Square	0.8200

Parameter Estimates

Variable	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	8.2308	0.8559	9.62	<.0001
time	1	0.5021	0.0403	12.45	<.0001

Estimates of Autocorrelations

Lag	Covariance	Correlation	-1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1
0	5.9709	1.000000	||********************|
1	4.5169	0.756485	||*************** |
2	2.0241	0.338995	||******* |

Preliminary MSE	1.7943

The maximum likelihood estimates are shown in Figure 9.4. Figure 9.4 also shows the preliminary Yule-Walker estimates used as starting values for the iterative computation of the maximum likelihood estimates.

Figure 9.4 Maximum Likelihood Estimates of AR(2) Error Model

Estimates of Autoregressive Parameters

Lag	Coefficient	Standard Error	t Value
1	-1.169057	0.148172	-7.89
2	0.545379	0.148172	3.68

Algorithm converged.

Maximum Likelihood Estimates

SSE	54.7493022	**DFE**	32
MSE	1.71092	**Root MSE**	1.30802
SBC	133.476508	**AIC**	127.142432
MAE	0.98307236	**AICC**	128.432755
MAPE	6.45517689	**HQC**	129.353194
Log Likelihood	-59.571216	**Transformed Regression R-Square**	0.7280
Durbin-Watson	2.2761	**Total R-Square**	0.9542
		Observations	36

Parameter Estimates

Variable	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	7.8833	1.1693	6.74	<.0001
time	1	0.5096	0.0551	9.25	<.0001
AR1	1	-1.2464	0.1385	-9.00	<.0001
AR2	1	0.6283	0.1366	4.60	<.0001

Autoregressive parameters assumed given

Variable	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	7.8833	1.1678	6.75	<.0001
time	1	0.5096	0.0551	9.26	<.0001

The diagnostic statistics and parameter estimates tables in Figure 9.4 have the same form as in the OLS output, but the values shown are for the autoregressive error model. The MSE for the autoregressive model is 1.71, which is much smaller than the true value of 4. In small samples, the autoregressive error model tends to underestimate σ^2, while the OLS MSE overestimates σ^2.

Notice that the total R^2 statistic computed from the autoregressive model residuals is 0.954, reflecting the improved fit from the use of past residuals to help predict the next Y value. The transformed regression R^2 0.728 is the R^2 statistic for a regression of transformed variables adjusted for the estimated autocorrelation. (This is not the R^2 for the estimated trend line. For details, see the section "Goodness-of-Fit Measures and Information Criteria" on page 401 later in this chapter.)

The parameter estimates table shows the ML estimates of the regression coefficients and includes two additional rows for the estimates of the autoregressive parameters, labeled AR(1) and AR(2).

The estimated model is

$$y_t = 7.88 + 0.5096t + v_t$$

$$v_t = 1.25v_{t-1} - 0.628v_{t-2} + \epsilon_t$$

$$Est.\ Var(\epsilon_t) = 1.71$$

Note that the signs of the autoregressive parameters shown in this equation for v_t are the reverse of the estimates shown in the AUTOREG procedure output. Figure 9.4 also shows the estimates of the regression coefficients with the standard errors recomputed on the assumption that the autoregressive parameter estimates equal the true values.

Predicted Values and Residuals

The AUTOREG procedure can produce two kinds of predicted values and corresponding residuals and confidence limits. The first kind of predicted value is obtained from only the structural part of the model, $\mathbf{x}_t'\mathbf{b}$. This is an estimate of the unconditional mean of the response variable at time t. For the time trend model, these predicted values trace the estimated trend. The second kind of predicted value includes both the structural part of the model and the predicted values of the autoregressive error process. The full model (conditional) predictions are used to forecast future values.

Use the OUTPUT statement to store predicted values and residuals in a SAS data set and to output other values such as confidence limits and variance estimates. The P= option specifies an output variable to contain the full model predicted values. The PM= option names an output variable for the predicted mean. The R= and RM= options specify output variables for the corresponding residuals, computed as the actual value minus the predicted value.

The following statements store both kinds of predicted values in the output data set. (The printed output is the same as previously shown in Figure 9.3 and Figure 9.4.)

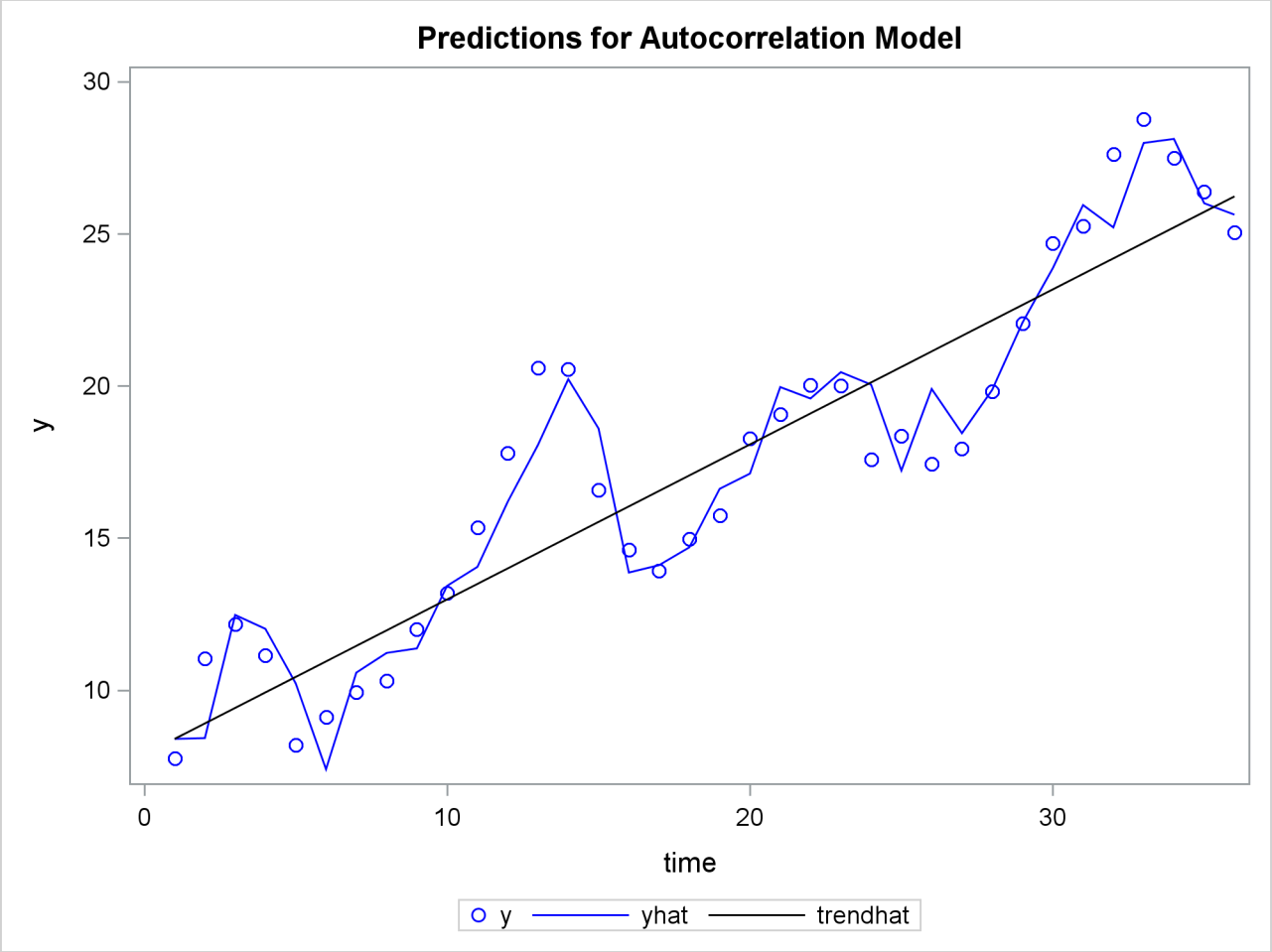
```
proc autoreg data=a;
   model y = time / nlag=2 method=ml;
   output out=p p=yhat pm=trendhat;
run;
```

The following statements plot the predicted values from the regression trend line and from the full model together with the actual values:

```
title 'Predictions for Autocorrelation Model';
proc sgplot data=p;
   scatter x=time y=y / markerattrs=(color=blue);
   series x=time y=yhat / lineattrs=(color=blue);
   series x=time y=trendhat / lineattrs=(color=black);
run;
```

The plot of predicted values is shown in Figure 9.5.

Figure 9.5 PROC AUTOREG Predictions



In Figure 9.5 the straight line is the autocorrelation corrected regression line, traced out by the structural predicted values TRENDHAT. The jagged line traces the full model prediction values. The actual values are marked by asterisks. This plot graphically illustrates the improvement in fit provided by the autoregressive error process for highly autocorrelated data.

Forecasting Autoregressive Error Models

To produce forecasts for future periods, include observations for the forecast periods in the input data set. The forecast observations must provide values for the independent variables and have missing values for the response variable.

For the time trend model, the only regressor is time. The following statements add observations for time periods 37 through 46 to the data set A to produce an augmented data set B:

```
data b;
   y = .;
   do time = 37 to 46; output; end;
run;

data b;
   merge a b;
   by time;
run;
```

To produce the forecast, use the augmented data set as input to PROC AUTOREG, and specify the appropriate options in the OUTPUT statement. The following statements produce forecasts for the time trend with autoregressive error model. The output data set includes all the variables in the input data set, the forecast values (YHAT), the predicted trend (YTREND), and the upper (UCL) and lower (LCL) 95% confidence limits.

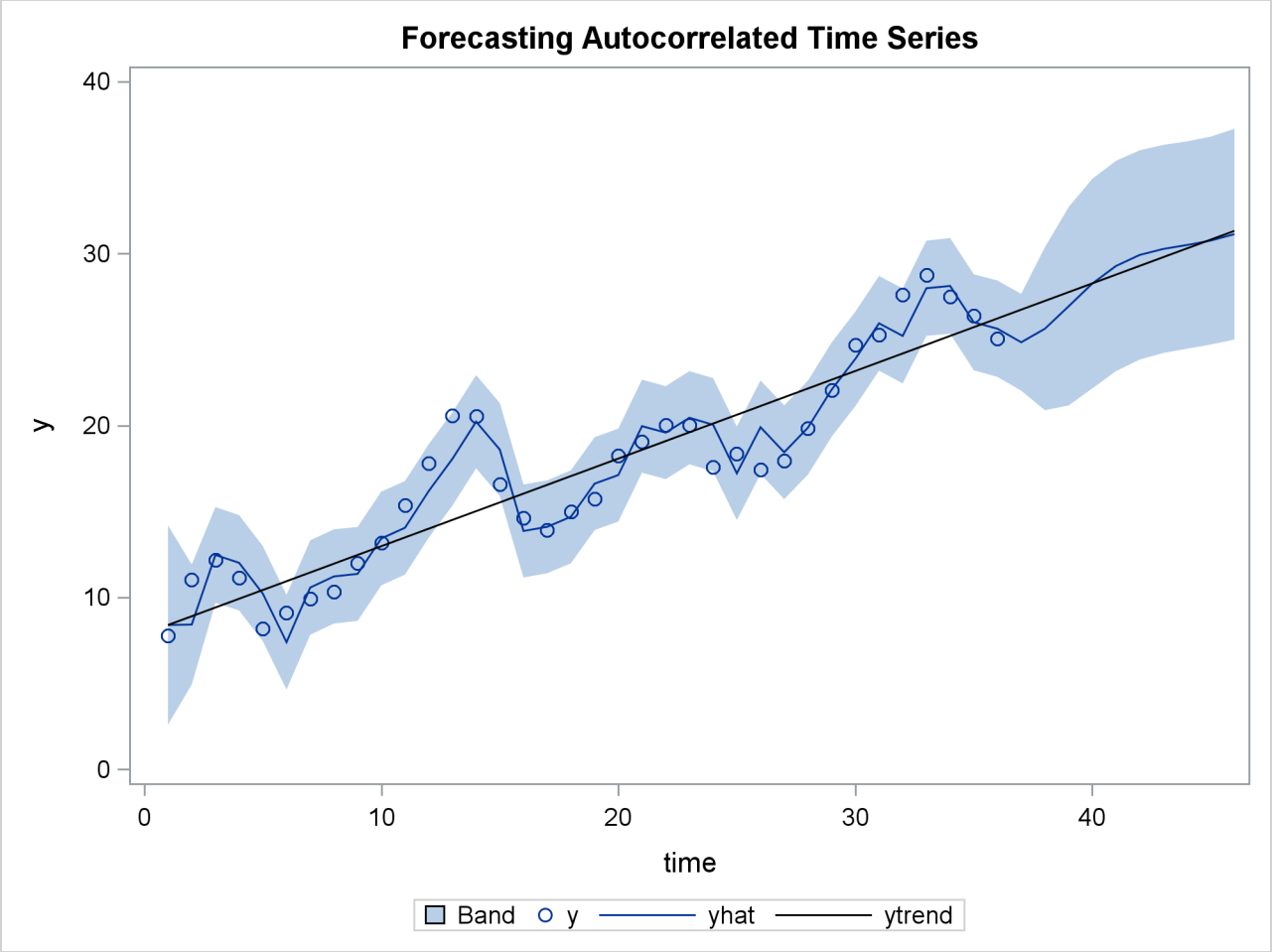
```
proc autoreg data=b;
   model y = time / nlag=2 method=ml;
   output out=p p=yhat pm=ytrend
                lcl=lcl ucl=ucl;
run;
```

The following statements plot the predicted values and confidence limits, and they also plot the trend line for reference. The actual observations are shown for periods 16 through 36, and a reference line is drawn at the start of the out-of-sample forecasts.

```
title 'Forecasting Autocorrelated Time Series';
proc sgplot data=p;
   band x=time upper=ucl lower=lcl;
   scatter x=time y=y;
   series x=time y=yhat;
   series x=time y=ytrend / lineattrs=(color=black);
run;
```

The plot is shown in Figure 9.6. Notice that the forecasts take into account the recent departures from the trend but converge back to the trend line for longer forecast horizons.

Figure 9.6 PROC AUTOREG Forecasts



Testing for Autocorrelation

In the preceding section, it is assumed that the order of the autoregressive process is known. In practice, you need to test for the presence of autocorrelation.

The Durbin-Watson test is a widely used method of testing for autocorrelation. The first-order Durbin-Watson statistic is printed by default. This statistic can be used to test for first-order autocorrelation. Use the DWPROB option to print the significance level (p-values) for the Durbin-Watson tests. (Since the Durbin-Watson p-values are computationally expensive, they are not reported by default.)

You can use the DW= option to request higher-order Durbin-Watson statistics. Since the ordinary Durbin-Watson statistic tests only for first-order autocorrelation, the Durbin-Watson statistics for higher-order autocorrelation are called *generalized Durbin-Watson statistics*.

The following statements perform the Durbin-Watson test for autocorrelation in the OLS residuals for orders 1 through 4. The DWPROB option prints the marginal significance levels (p-values) for the Durbin-Watson statistics.

```
/*-- Durbin-Watson test for autocorrelation --*/
proc autoreg data=a;
   model y = time / dw=4 dwprob;
run;
```

The AUTOREG procedure output is shown in Figure 9.7. In this case, the first-order Durbin-Watson test is highly significant, with $p < .0001$ for the hypothesis of no first-order autocorrelation. Thus, autocorrelation correction is needed.

Figure 9.7 Durbin-Watson Test Results for OLS Residuals

Forecasting Autocorrelated Time Series

The AUTOREG Procedure

Dependent Variable	y

Ordinary Least Squares Estimates			
SSE	214.953429	DFE	34
MSE	6.32216	Root MSE	2.51439
SBC	173.659101	AIC	170.492063
MAE	2.01903356	AICC	170.855699
MAPE	12.5270666	HQC	171.597444
		Total R-Square	0.8200

Durbin-Watson Statistics			
Order	DW	Pr < DW	Pr > DW
1	0.4752	<.0001	1.0000
2	1.2935	0.0137	0.9863
3	2.0694	0.6545	0.3455
4	2.5544	0.9818	0.0182

NOTE: Pr<DW is the p-value for testing positive autocorrelation, and Pr>DW is the p-value for testing negative autocorrelation.

Parameter Estimates					
Variable	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	8.2308	0.8559	9.62	<.0001
time	1	0.5021	0.0403	12.45	<.0001

Using the Durbin-Watson test, you can decide if autocorrelation correction is needed. However, generalized Durbin-Watson tests should not be used to decide on the autoregressive order. The higher-order tests assume the absence of lower-order autocorrelation. If the ordinary Durbin-Watson test indicates no first-order autocorrelation, you can use the second-order test to check for second-order autocorrelation. Once autocorrelation is detected, further tests at higher orders are not appropriate. In Figure 9.7, since the first-order Durbin-Watson test is significant, the order 2, 3, and 4 tests can be ignored.

When using Durbin-Watson tests to check for autocorrelation, you should specify an order at least as large as the order of any potential seasonality, since seasonality produces autocorrelation at the seasonal lag. For example, for quarterly data use DW=4, and for monthly data use DW=12.

Lagged Dependent Variables

The Durbin-Watson tests are not valid when the lagged dependent variable is used in the regression model. In this case, the Durbin *h* test or Durbin *t* test can be used to test for first-order autocorrelation.

For the Durbin *h* test, specify the name of the lagged dependent variable in the LAGDEP= option. For the Durbin *t* test, specify the LAGDEP option without giving the name of the lagged dependent variable.

For example, the following statements add the variable YLAG to the data set A and regress Y on YLAG instead of TIME:

```
data b;
   set a;
   ylag = lag1( y );
run;

proc autoreg data=b;
   model y = ylag / lagdep=ylag;
run;
```

The results are shown in Figure 9.8. The Durbin *h* statistic 2.78 is significant with a *p*-value of 0.0027, indicating autocorrelation.

Figure 9.8 Durbin *h* Test with a Lagged Dependent Variable

Forecasting Autocorrelated Time Series

The AUTOREG Procedure

Dependent Variable	y

Ordinary Least Squares Estimates			
SSE	97.711226	DFE	33
MSE	2.96095	Root MSE	1.72074
SBC	142.369787	AIC	139.259091
MAE	1.29949385	AICC	139.634091
MAPE	8.1922836	HQC	140.332903
		Total R-Square	0.9109

Miscellaneous Statistics			
Statistic	Value	Prob	Label
Durbin h	2.7814	0.0027	Pr > h

Parameter Estimates					
Variable	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	1.5742	0.9300	1.69	0.0999
ylag	1	0.9376	0.0510	18.37	<.0001

Stepwise Autoregression

Once you determine that autocorrelation correction is needed, you must select the order of the autoregressive error model to use. One way to select the order of the autoregressive error model is *stepwise autoregression*. The stepwise autoregression method initially fits a high-order model with many autoregressive lags and then sequentially removes autoregressive parameters until all remaining autoregressive parameters have significant *t* tests.

To use stepwise autoregression, specify the BACKSTEP option, and specify a large order with the NLAG= option. The following statements show the stepwise feature, using an initial order of 5:

```
/*-- stepwise autoregression --*/
proc autoreg data=a;
   model y = time / method=ml nlag=5 backstep;
run;
```

The results are shown in Figure 9.9.

Figure 9.9 Stepwise Autoregression

Forecasting Autocorrelated Time Series

The AUTOREG Procedure

Dependent Variable	y

Ordinary Least Squares Estimates			
SSE	214.953429	**DFE**	34
MSE	6.32216	**Root MSE**	2.51439
SBC	173.659101	**AIC**	170.492063
MAE	2.01903356	**AICC**	170.855699
MAPE	12.5270666	**HQC**	171.597444
Durbin-Watson	0.4752	**Total R-Square**	0.8200

Parameter Estimates

Variable	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	8.2308	0.8559	9.62	<.0001
time	1	0.5021	0.0403	12.45	<.0001

Estimates of Autocorrelations

Lag	Covariance	Correlation	-1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1
0	5.9709	1.000000	|********************|
1	4.5169	0.756485	|*************** |
2	2.0241	0.338995	|******* |
3	-0.4402	-0.073725	|* |
4	-2.1175	-0.354632	|******* |
5	-2.8534	-0.477887	|********** |

Figure 9.9 *continued*

Backward Elimination of Autoregressive Terms			
Lag	Estimate	t Value	Pr > |t|
4	-0.052908	-0.20	0.8442
3	0.115986	0.57	0.5698
5	0.131734	1.21	0.2340

The estimates of the autocorrelations are shown for 5 lags. The backward elimination of autoregressive terms report shows that the autoregressive parameters at lags 3, 4, and 5 were insignificant and eliminated, resulting in the second-order model shown previously in Figure 9.4. By default, retained autoregressive parameters must be significant at the 0.05 level, but you can control this with the SLSTAY= option. The remainder of the output from this example is the same as that in Figure 9.3 and Figure 9.4, and it is not repeated here.

The stepwise autoregressive process is performed using the Yule-Walker method. The maximum likelihood estimates are produced after the order of the model is determined from the significance tests of the preliminary Yule-Walker estimates.

When using stepwise autoregression, it is a good idea to specify an NLAG= option value larger than the order of any potential seasonality, since seasonality produces autocorrelation at the seasonal lag. For example, for monthly data use NLAG=13, and for quarterly data use NLAG=5.

Subset and Factored Models

In the previous example, the BACKSTEP option dropped lags 3, 4, and 5, leaving a second-order model. However, in other cases a parameter at a longer lag may be kept while some smaller lags are dropped. For example, the stepwise autoregression method might drop lags 2, 3, and 5 but keep lags 1 and 4. This is called a *subset model*, since the number of estimated autoregressive parameters is lower than the order of the model.

Subset models are common for seasonal data and often correspond to *factored* autoregressive models. A factored model is the product of simpler autoregressive models. For example, the best model for seasonal monthly data may be the combination of a first-order model for recent effects with a 12th-order subset model for the seasonality, with a single parameter at lag 12. This results in a 13th-order subset model with nonzero parameters at lags 1, 12, and 13. See Chapter 8, "The ARIMA Procedure," for further discussion of subset and factored autoregressive models.

You can specify subset models with the NLAG= option. List the lags to include in the autoregressive model within parentheses. The following statements show an example of specifying the subset model resulting from the combination of a first-order process for recent effects with a fourth-order seasonal process:

```
/*-- specifying the lags --*/
proc autoreg data=a;
   model y = time / nlag=(1 4 5);
run;
```

The MODEL statement specifies the following fifth-order autoregressive error model:

$$y_t = a + bt + v_t$$

$$v_t = -\varphi_1 v_{t-1} - \varphi_4 v_{t-4} - \varphi_5 v_{t-5} + \epsilon_t$$

Testing for Heteroscedasticity

One of the key assumptions of the ordinary regression model is that the errors have the same variance throughout the sample. This is also called the *homoscedasticity* model. If the error variance is not constant, the data are said to be *heteroscedastic*.

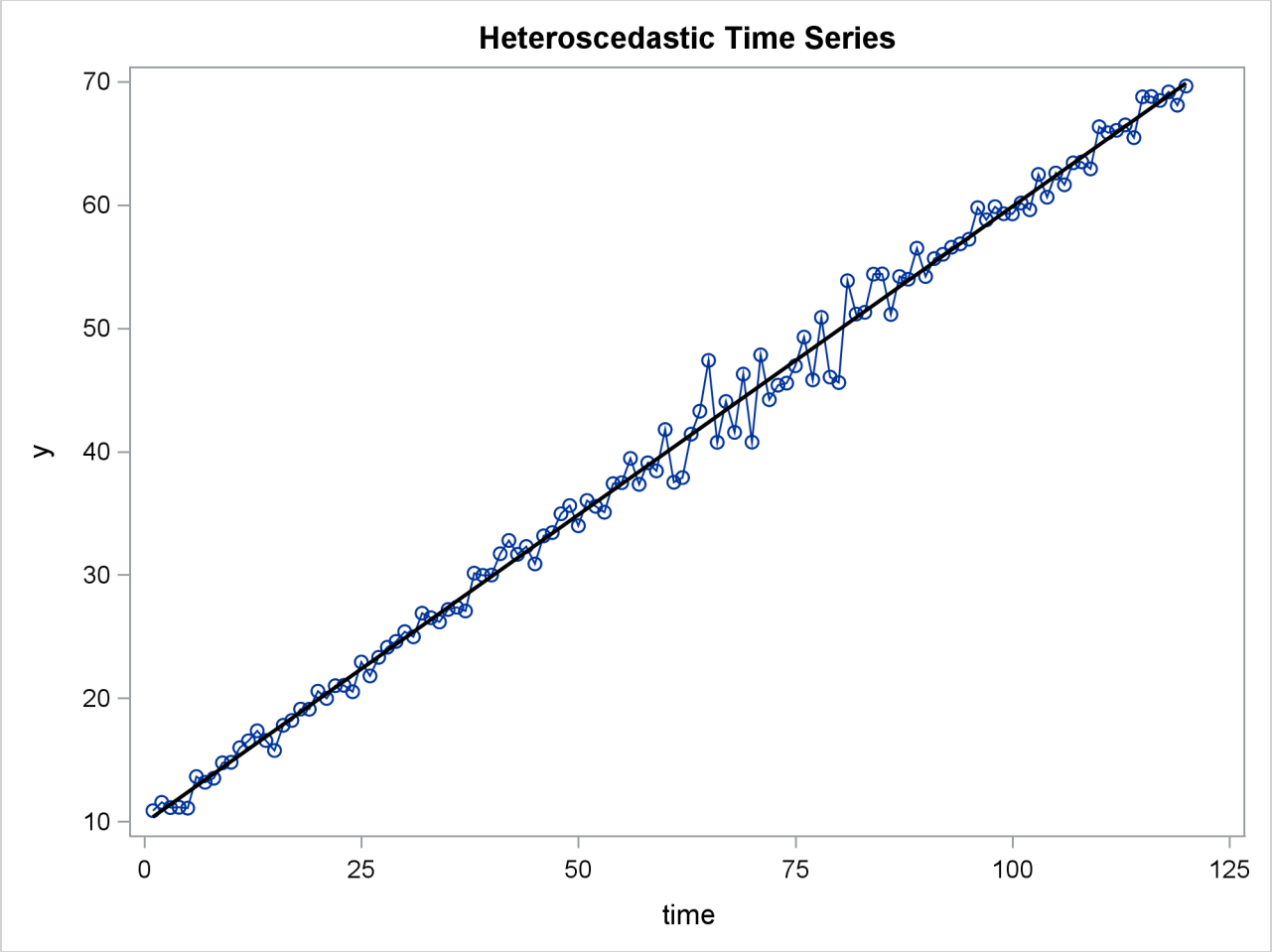
Since ordinary least squares regression assumes constant error variance, heteroscedasticity causes the OLS estimates to be inefficient. Models that take into account the changing variance can make more efficient use of the data. Also, heteroscedasticity can make the OLS forecast error variance inaccurate because the predicted forecast variance is based on the average variance instead of on the variability at the end of the series.

To illustrate heteroscedastic time series, the following statements create the simulated series Y. The variable Y has an error variance that changes from 1 to 4 in the middle part of the series.

```
data a;
   do time = -10 to 120;
      s = 1 + (time >= 60 & time < 90);
      u = s*rannor(12346);
      y = 10 + .5 * time + u;
      if time > 0 then output;
      end;
run;

title 'Heteroscedastic Time Series';
proc sgplot data=a noautolegend;
   series x=time y=y / markers;
   reg x=time y=y / lineattrs=(color=black);
run;
```

The simulated series is plotted in Figure 9.10.

Figure 9.10 Heteroscedastic and Autocorrelated Series



To test for heteroscedasticity with PROC AUTOREG, specify the ARCHTEST option. The following statements regress Y on TIME and use the ARCHTEST= option to test for heteroscedastic OLS residuals:

```
/*-- test for heteroscedastic OLS residuals --*/
proc autoreg data=a;
   model y = time / archtest;
   output out=r r=yresid;
run;
```

The PROC AUTOREG output is shown in Figure 9.11. The Q statistics test for changes in variance across time by using lag windows that range from 1 through 12. (See the section "Testing for Nonlinear Dependence: Heteroscedasticity Tests" on page 422 for details.) The p-values for the test statistics strongly indicate heteroscedasticity, with $p < 0.0001$ for all lag windows.

The Lagrange multiplier (LM) tests also indicate heteroscedasticity. These tests can also help determine the order of the ARCH model that is appropriate for modeling the heteroscedasticity, assuming that the changing variance follows an autoregressive conditional heteroscedasticity model.

Figure 9.11 Heteroscedasticity Tests

Heteroscedastic Time Series

The AUTOREG Procedure

Dependent Variable	y

Ordinary Least Squares Estimates			
SSE	223.645647	**DFE**	118
MSE	1.89530	**Root MSE**	1.37670
SBC	424.828766	**AIC**	419.253783
MAE	0.97683599	**AICC**	419.356347
MAPE	2.73888672	**HQC**	421.517809
Durbin-Watson	2.4444	**Total R-Square**	0.9938

Tests for ARCH Disturbances Based on OLS Residuals

Order	Q	Pr > Q	LM	Pr > LM
1	19.4549	<.0001	19.1493	<.0001
2	21.3563	<.0001	19.3057	<.0001
3	28.7738	<.0001	25.7313	<.0001
4	38.1132	<.0001	26.9664	<.0001
5	52.3745	<.0001	32.5714	<.0001
6	54.4968	<.0001	34.2375	<.0001
7	55.3127	<.0001	34.4726	<.0001
8	58.3809	<.0001	34.4850	<.0001
9	68.3075	<.0001	38.7244	<.0001
10	73.2949	<.0001	38.9814	<.0001
11	74.9273	<.0001	39.9395	<.0001
12	76.0254	<.0001	40.8144	<.0001

Parameter Estimates

Variable	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	9.8684	0.2529	39.02	<.0001
time	1	0.5000	0.003628	137.82	<.0001

The tests of Lee and King (1993) and Wong and Li (1995) can also be applied to check the absence of ARCH effects. The following example shows that Wong and Li's test is robust to detect the presence of ARCH effects with the existence of outliers.

```
/*-- data with outliers at observation 10 --*/
data b;
   do time = -10 to 120;
      s = 1 + (time >= 60 & time < 90);
      u = s*rannor(12346);
      y = 10 + .5 * time + u;
      if time = 10 then
         do; y = 200; end;
      if time > 0 then output;
   end;
run;
/*-- test for heteroscedastic OLS residuals --*/
proc autoreg data=b;
   model y = time / archtest=(qlm) ;
   model y = time / archtest=(lk,wl) ;
run;
```

As shown in Figure 9.12, the *p*-values of Q or LM statistics for all lag windows are above 90%, which fails to reject the null hypothesis of the absence of ARCH effects. Lee and King's test, which rejects the null hypothesis for lags more than 8 at 10% significance level, works better. Wong and Li's test works best, rejecting the null hypothesis and detecting the presence of ARCH effects for all lag windows.

Figure 9.12 Heteroscedasticity Tests

Heteroscedastic Time Series

The AUTOREG Procedure

Tests for ARCH Disturbances Based on OLS Residuals

Order	Q	Pr > Q	LM	Pr > LM
1	0.0076	0.9304	0.0073	0.9319
2	0.0150	0.9925	0.0143	0.9929
3	0.0229	0.9991	0.0217	0.9992
4	0.0308	0.9999	0.0290	0.9999
5	0.0367	1.0000	0.0345	1.0000
6	0.0442	1.0000	0.0413	1.0000
7	0.0522	1.0000	0.0485	1.0000
8	0.0612	1.0000	0.0565	1.0000
9	0.0701	1.0000	0.0643	1.0000
10	0.0701	1.0000	0.0742	1.0000
11	0.0701	1.0000	0.0838	1.0000
12	0.0702	1.0000	0.0939	1.0000

Figure 9.12 *continued*

Tests for ARCH Disturbances Based on OLS Residuals				
Order	LK	Pr > |LK|	WL	Pr > WL
1	-0.6377	0.5236	34.9984	<.0001
2	-0.8926	0.3721	72.9542	<.0001
3	-1.0979	0.2723	104.0322	<.0001
4	-1.2705	0.2039	139.9328	<.0001
5	-1.3824	0.1668	176.9830	<.0001
6	-1.5125	0.1304	200.3388	<.0001
7	-1.6385	0.1013	238.4844	<.0001
8	-1.7695	0.0768	267.8882	<.0001
9	-1.8881	0.0590	304.5706	<.0001
10	-2.2349	0.0254	326.3658	<.0001
11	-2.2380	0.0252	348.8036	<.0001
12	-2.2442	0.0248	371.9596	<.0001

Heteroscedasticity and GARCH Models

There are several approaches to dealing with heteroscedasticity. If the error variance at different times is known, weighted regression is a good method. If, as is usually the case, the error variance is unknown and must be estimated from the data, you can model the changing error variance.

The *generalized autoregressive conditional heteroscedasticity* (GARCH) model is one approach to modeling time series with heteroscedastic errors. The GARCH regression model with autoregressive errors is

$$y_t = \mathbf{x}_t'\beta + v_t$$

$$v_t = \epsilon_t - \varphi_1 v_{t-1} - \ldots - \varphi_m v_{t-m}$$

$$\epsilon_t = \sqrt{h_t}e_t$$

$$h_t = \omega + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2 + \sum_{j=1}^{p} \gamma_j h_{t-j}$$

$$e_t \sim \mathrm{IN}(0,1)$$

This model combines the mth-order autoregressive error model with the GARCH(p,q) variance model. It is denoted as the AR(m)-GARCH(p,q) regression model.

The tests for the presence of ARCH effects (namely, Q and LM tests, tests from Lee and King (1993) and tests from Wong and Li (1995)) can help determine the order of the ARCH model appropriate for the data. For example, the Lagrange multiplier (LM) tests shown in Figure 9.11 are significant ($p < 0.0001$) through order 12, which indicates that a very high-order ARCH model is needed to model the heteroscedasticity.

The basic ARCH(q) model ($p = 0$) is a *short memory* process in that only the most recent q squared residuals are used to estimate the changing variance. The GARCH model ($p > 0$) allows *long memory* processes, which use all the past squared residuals to estimate the current variance. The LM tests in Figure 9.11 suggest the use of the GARCH model ($p > 0$) instead of the ARCH model.

The GARCH(p, q) model is specified with the GARCH=(P=p, Q=q) option in the MODEL statement. The basic ARCH(q) model is the same as the GARCH($0, q$) model and is specified with the GARCH=(Q=q) option.

The following statements fit an AR(2)-GARCH(1, 1) model for the Y series that is regressed on TIME. The GARCH=(P=1,Q=1) option specifies the GARCH(1, 1) conditional variance model. The NLAG=2 option specifies the AR(2) error process. Only the maximum likelihood method is supported for GARCH models; therefore, the METHOD= option is not needed. The CEV= option in the OUTPUT statement stores the estimated conditional error variance at each time period in the variable VHAT in an output data set named OUT. The data set is the same as in the section "Testing for Heteroscedasticity" on page 347.

```
data c;
   ul=0; ull=0;
   do time = -10 to 120;
      s = 1 + (time >= 60 & time < 90);
      u = + 1.3 * ul - .5 * ull + s*rannor(12346);
      y = 10 + .5 * time + u;
      if time > 0 then output;
      ull = ul; ul = u;
      end;
run;
title 'AR(2)-GARCH(1,1) model for the Y series regressed on TIME';
proc autoreg data=c;
   model y = time / nlag=2 garch=(q=1,p=1) maxit=50;
   output out=out cev=vhat;
run;
```

The results for the GARCH model are shown in Figure 9.13. (The preliminary estimates are not shown.)

Figure 9.13 AR(2)-GARCH(1, 1) Model

AR(2)-GARCH(1,1) model for the Y series regressed on TIME

The AUTOREG Procedure

GARCH Estimates			
SSE	218.861036	**Observations**	120
MSE	1.82384	**Uncond Var**	1.6299733
Log Likelihood	-187.44013	**Total R-Square**	0.9941
SBC	408.392693	**AIC**	388.88025
MAE	0.97051406	**AICC**	389.88025
MAPE	2.75945337	**HQC**	396.804343
		Normality Test	0.0838
		Pr > ChiSq	0.9590

Figure 9.13 *continued*

Parameter Estimates

Variable	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	8.9301	0.7456	11.98	<.0001
time	1	0.5075	0.0111	45.90	<.0001
AR1	1	-1.2301	0.1111	-11.07	<.0001
AR2	1	0.5023	0.1090	4.61	<.0001
ARCH0	1	0.0850	0.0780	1.09	0.2758
ARCH1	1	0.2103	0.0873	2.41	0.0159
GARCH1	1	0.7375	0.0989	7.46	<.0001

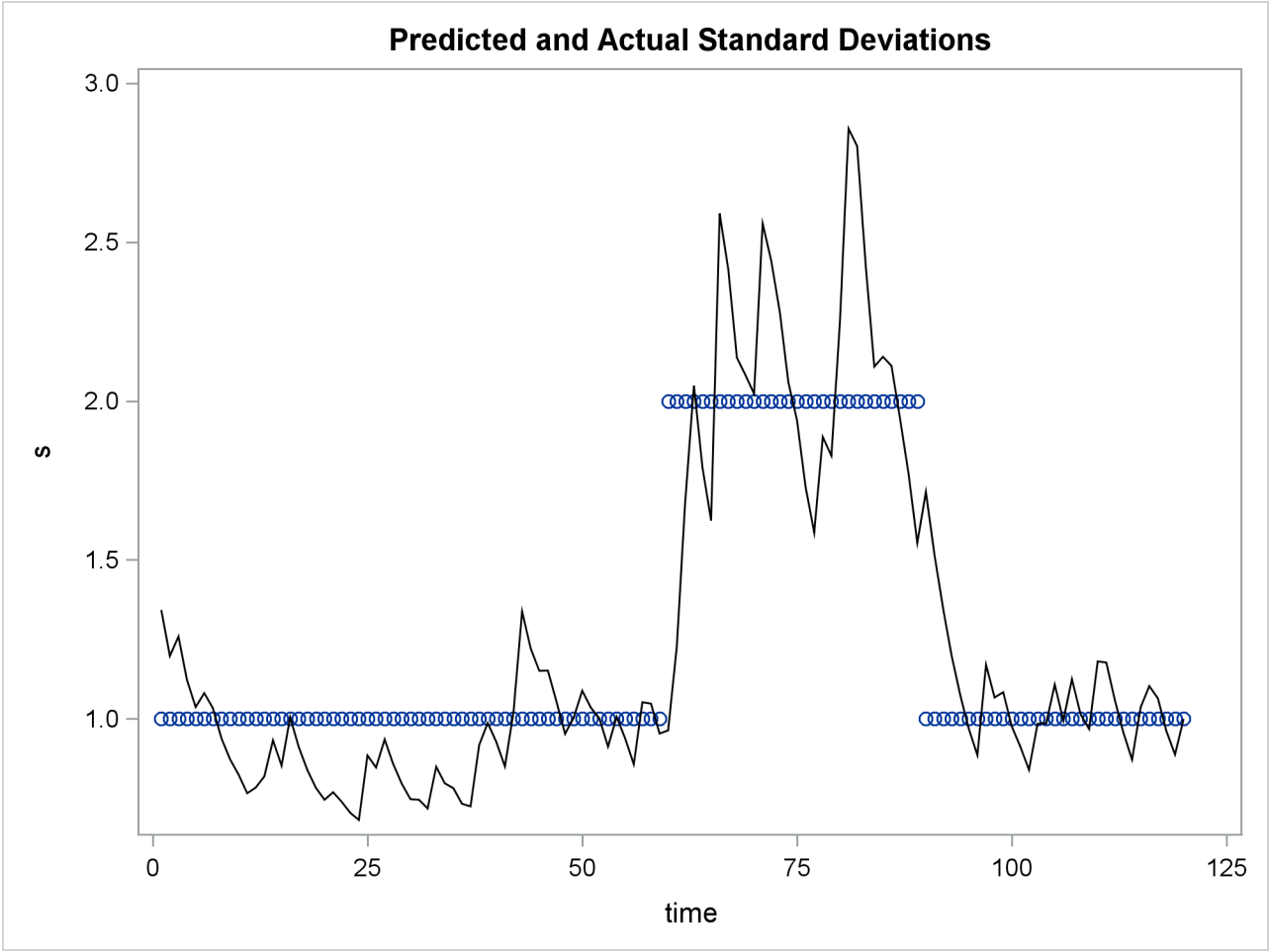
The normality test is not significant ($p = 0.959$), which is consistent with the hypothesis that the residuals from the GARCH model, $\epsilon_t / \sqrt{h_t}$, are normally distributed. The parameter estimates table includes rows for the GARCH parameters. ARCH0 represents the estimate for the parameter ω, ARCH1 represents α_1, and GARCH1 represents γ_1.

The following statements transform the estimated conditional error variance series VHAT to the estimated standard deviation series SHAT. Then, they plot SHAT together with the true standard deviation S used to generate the simulated data.

```
data out;
   set out;
   shat = sqrt( vhat );
run;

title 'Predicted and Actual Standard Deviations';
proc sgplot data=out noautolegend;
   scatter x=time y=s;
   series x=time y=shat/ lineattrs=(color=black);
run;
```

The plot is shown in Figure 9.14.

Figure 9.14 Estimated and Actual Error Standard Deviation Series



In this example note that the form of heteroscedasticity used in generating the simulated series Y does not fit the GARCH model. The GARCH model assumes *conditional* heteroscedasticity, with homoscedastic unconditional error variance. That is, the GARCH model assumes that the changes in variance are a function of the realizations of preceding errors and that these changes represent temporary and random departures from a constant unconditional variance. The data-generating process used to simulate series Y, contrary to the GARCH model, has exogenous unconditional heteroscedasticity that is independent of past errors.

Nonetheless, as shown in Figure 9.14, the GARCH model does a reasonably good job of approximating the error variance in this example, and some improvement in the efficiency of the estimator of the regression parameters can be expected.

The GARCH model might perform better in cases where theory suggests that the data-generating process produces true autoregressive conditional heteroscedasticity. This is the case in some economic theories of asset returns, and GARCH-type models are often used for analysis of financial market data.

GARCH Models

The AUTOREG procedure supports several variations of GARCH models.

Using the TYPE= option along with the GARCH= option enables you to control the constraints placed on the estimated GARCH parameters. You can specify unconstrained, nonnegativity-constrained (default), stationarity-constrained, or integration-constrained models. The integration constraint produces the integrated GARCH (IGARCH) model.

You can also use the TYPE= option to specify the exponential form of the GARCH model, called the EGARCH model, or other types of GARCH models, namely the quadratic GARCH (QGARCH), threshold GARCH (TGARCH), and power GARCH (PGARCH) models. The MEAN= option along with the GARCH= option specifies the GARCH-in-mean (GARCH-M) model.

The following statements illustrate the use of the TYPE= option to fit an AR(2)-EGARCH(1, 1) model to the series Y. (Output is not shown.)

```
/*-- AR(2)-EGARCH(1,1) model --*/
proc autoreg data=a;
   model y = time / nlag=2 garch=(p=1,q=1,type=exp);
run;
```

See the section "GARCH Models" on page 392 for details.

Syntax: AUTOREG Procedure

The AUTOREG procedure is controlled by the following statements:

PROC AUTOREG *options* ;
BY *variables* ;
CLASS *variables* ;
MODEL *dependent = regressors / options* ;
HETERO *variables / options* ;
NLOPTIONS *options* ;
OUTPUT < **OUT=***SAS-data-set* > < *options* > < *keyword=name* > ;
RESTRICT *equation , . . . , equation* ;
TEST *equation , . . . , equation / option* ;

At least one MODEL statement must be specified. One OUTPUT statement can follow each MODEL statement. One HETERO statement can follow each MODEL statement.

Functional Summary

The statements and options used with the AUTOREG procedure are summarized in the following table.

Table 9.1 AUTOREG Functional Summary

Description	Statement	Option
Data Set Options		
Specify the input data set	AUTOREG	DATA=
Write parameter estimates to an output data set	AUTOREG	OUTEST=

Table 9.1 *continued*

Description	Statement	Option
Include covariances in the OUTEST= data set	AUTOREG	COVOUT
Requests that the procedure produce graphics via the Output Delivery System	AUTOREG	PLOTS=
Write predictions, residuals, and confidence limits to an output data set	OUTPUT	OUT=
Declaring the Role of Variables		
Specify BY-group processing	BY	
Specify classification variables	CLASS	
Printing Control Options		
Request all printing options	MODEL	ALL
Print transformed coefficients	MODEL	COEF
Print correlation matrix of the estimates	MODEL	CORRB
Print covariance matrix of the estimates	MODEL	COVB
Print DW statistics up to order *j*	MODEL	DW=*j*
Print marginal probability of the generalized Durbin-Watson test statistics for large sample sizes	MODEL	DWPROB
Print the *p*-values for the Durbin-Watson test be computed using a linearized approximation of the design matrix	MODEL	LDW
Print inverse of Toeplitz matrix	MODEL	GINV
Print the Godfrey LM serial correlation test	MODEL	GODFREY=
Print details at each iteration step	MODEL	ITPRINT
Print the Durbin *t* statistic	MODEL	LAGDEP
Print the Durbin *h* statistic	MODEL	LAGDEP=
Print the log-likelihood value of the regression model	MODEL	LOGLIKL
Print the Jarque-Bera normality test	MODEL	NORMAL
Print the tests for the absence of ARCH effects	MODEL	ARCHTEST=
Print BDS tests for independence	MODEL	BDS=
Print rank version of von Neumann ratio test for independence	MODEL	VNRRANK=
Print runs test for independence	MODEL	RUNS=
Print the turning point test for independence	MODEL	TP=
Print the Lagrange multiplier test	HETERO	TEST=LM
Print Bai-Perron tests for multiple structural changes	MODEL	BP=
Print the Chow test for structural change	MODEL	CHOW=
Print the predictive Chow test for structural change	MODEL	PCHOW=
Suppress printed output	MODEL	NOPRINT
Print partial autocorrelations	MODEL	PARTIAL
Print Ramsey's RESET test	MODEL	RESET

Table 9.1 *continued*

Description	Statement	Option
Print Augmented Dickey-Fuller tests for stationarity or unit roots	MODEL	STATIONARITY=(ADF=)
Print ERS tests for stationarity or unit roots	MODEL	STATIONARITY=(ERS=)
Print KPSS tests or Shin tests for stationarity or cointegration	MODEL	STATIONARITY=(KPSS=)
Print Ng-Perron tests for stationarity or unit roots	MODEL	STATIONARITY=(NP=)
Print Phillips-Perron tests for stationarity or unit roots	MODEL	STATIONARITY=(PHILLIPS=)
Print tests of linear hypotheses	TEST	
Specify the test statistics to use	TEST	TYPE=
Print the uncentered regression R^2	MODEL	URSQ
Options to Control the Optimization Process		
Specify the optimization options	NLOPTIONS	see Chapter 7, "Nonlinear Optimization Methods,"
Model Estimation Options		
Specify the order of autoregressive process	MODEL	NLAG=
Center the dependent variable	MODEL	CENTER
Suppress the intercept parameter	MODEL	NOINT
Remove nonsignificant AR parameters	MODEL	BACKSTEP
Specify significance level for BACKSTEP	MODEL	SLSTAY=
Specify the convergence criterion	MODEL	CONVERGE=
Specify the type of covariance matrix	MODEL	COVEST=
Set the initial values of parameters used by the iterative optimization algorithm	MODEL	INITIAL=
Specify iterative Yule-Walker method	MODEL	ITER
Specify maximum number of iterations	MODEL	MAXITER=
Specify the estimation method	MODEL	METHOD=
Use only first sequence of nonmissing data	MODEL	NOMISS
Specify the optimization technique	MODEL	OPTMETHOD=
Imposes restrictions on the regression estimates	RESTRICT	
Estimate and test heteroscedasticity models	HETERO	
GARCH Related Options		
Specify order of GARCH process	MODEL	GARCH=(Q=,P=)
Specify type of GARCH model	MODEL	GARCH=(. . .,TYPE=)
Specify various forms of the GARCH-M model	MODEL	GARCH=(. . .,MEAN=)
Suppress GARCH intercept parameter	MODEL	GARCH=(. . .,NOINT)
Specify the trust region method	MODEL	GARCH=(. . .,TR)

Table 9.1 *continued*

Description	Statement	Option
Estimate the GARCH model for the conditional *t* distribution	MODEL	GARCH=(. . .) DIST=
Estimate the start-up values for the conditional variance equation	MODEL	GARCH=(. . .,STARTUP=)
Specify the functional form of the heteroscedasticity model	HETERO	LINK=
Specify that the heteroscedasticity model does not include the unit term	HETERO	NOCONST
Impose constraints on the estimated parameters in the heteroscedasticity model	HETERO	COEF=
Impose constraints on the estimated standard deviation of the heteroscedasticity model	HETERO	STD=
Output conditional error variance	OUTPUT	CEV=
Output conditional prediction error variance	OUTPUT	CPEV=
Specify the flexible conditional variance form of the GARCH model	HETERO	
Output Control Options		
Specify confidence limit size	OUTPUT	ALPHACLI=
Specify confidence limit size for structural predicted values	OUTPUT	ALPHACLM=
Specify the significance level for the upper and lower bounds of the CUSUM and CUSUMSQ statistics	OUTPUT	ALPHACSM=
Specify the name of a variable to contain the values of the Theil's BLUS residuals	OUTPUT	BLUS=
Output the value of the error variance σ_t^2	OUTPUT	CEV=
Output transformed intercept variable	OUTPUT	CONSTANT=
Specify the name of a variable to contain the CUSUM statistics	OUTPUT	CUSUM=
Specify the name of a variable to contain the CUSUMSQ statistics	OUTPUT	CUSUMSQ=
Specify the name of a variable to contain the upper confidence bound for the CUSUM statistic	OUTPUT	CUSUMUB=
Specify the name of a variable to contain the lower confidence bound for the CUSUM statistic	OUTPUT	CUSUMLB=
Specify the name of a variable to contain the upper confidence bound for the CUSUMSQ statistic	OUTPUT	CUSUMSQUB=
Specify the name of a variable to contain the lower confidence bound for the CUSUMSQ statistic	OUTPUT	CUSUMSQLB=
Output lower confidence limit	OUTPUT	LCL=

Table 9.1 *continued*

Description	Statement	Option
Output lower confidence limit for structural predicted values	OUTPUT	LCLM=
Output predicted values	OUTPUT	P=
Output predicted values of structural part	OUTPUT	PM=
Output residuals	OUTPUT	R=
Output residuals from structural predictions	OUTPUT	RM=
Specify the name of a variable to contain the part of the predictive error variance (v_t)	OUTPUT	RECPEV=
Specify the name of a variable to contain recursive residuals	OUTPUT	RECRES=
Output transformed variables	OUTPUT	TRANSFORM=
Output upper confidence limit	OUTPUT	UCL=
Output upper confidence limit for structural predicted values	OUTPUT	UCLM=

PROC AUTOREG Statement

PROC AUTOREG *options* **;**

The following options can be used in the PROC AUTOREG statement:

DATA=*SAS-data-set*
specifies the input SAS data set. If the DATA= option is not specified, PROC AUTOREG uses the most recently created SAS data set.

OUTEST=*SAS-data-set*
writes the parameter estimates to an output data set. See the section "OUTEST= Data Set" on page 431 for information on the contents of these data set.

COVOUT
writes the covariance matrix for the parameter estimates to the OUTEST= data set. This option is valid only if the OUTEST= option is specified.

PLOTS<(*global-plot-options***)> < = *(specific plot options)*>**
requests that the AUTOREG procedure produce statistical graphics via the Output Delivery System, provided that the ODS GRAPHICS statement has been specified. For general information about ODS Graphics, see Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*). The *global-plot-options* apply to all relevant plots generated by the AUTOREG procedure. The *global-plot-options* supported by the AUTOREG procedure follow.

Global Plot Options

ONLY suppresses the default plots. Only the plots specifically requested are produced.

UNPACKPANEL | UNPACK displays each graph separately. (By default, some graphs can appear together in a single panel.)

Specific Plot Options

ALL requests that all plots appropriate for the particular analysis be produced.

ACF produces the autocorrelation function plot.

IACF produces the inverse autocorrelation function plot of residuals.

PACF produces the partial autocorrelation function plot of residuals.

FITPLOT plots the predicted and actual values.

COOKSD produces the Cook's *D* plot.

QQ Q-Q plot of residuals.

RESIDUAL | RES plots the residuals.

STUDENTRESIDUAL plots the studentized residuals. For the models with the NLAG= or GARCH= options in the MODEL statement or with the HETERO statement, this option is replaced by the STANDARDRESIDUAL option.

STANDARDRESIDUAL plots the standardized residuals.

WHITENOISE plots the white noise probabilities.

RESIDUALHISTOGRAM | RESIDHISTOGRAM plots the histogram of residuals.

NONE suppresses all plots.

In addition, any of the following MODEL statement options can be specified in the PROC AUTOREG statement, which is equivalent to specifying the option for every MODEL statement: ALL, ARCHTEST, BACKSTEP, CENTER, COEF, CONVERGE=, CORRB, COVB, DW=, DWPROB, GINV, ITER, ITPRINT, MAXITER=, METHOD=, NOINT, NOMISS, NOPRINT, and PARTIAL.

BY Statement

BY *variables* **;**

A BY statement can be used with PROC AUTOREG to obtain separate analyses on observations in groups defined by the BY variables.

CLASS Statement (Experimental)

CLASS *variables* **;**

The CLASS statement names the classification variables to be used in the analysis. Classification variables can be either character or numeric.

In PROC AUTOREG, the CLASS statement enables you to output class variables to a data set that contains a copy of the original data.

Class levels are determined from the formatted values of the CLASS variables. Thus, you can use formats to group values into levels. See the discussion of the FORMAT procedure in *SAS Language Reference: Dictionary* for details.

MODEL Statement

MODEL *dependent = regressors / options* **;**

The MODEL statement specifies the dependent variable and independent regressor variables for the regression model. If no independent variables are specified in the MODEL statement, only the mean is fitted. (This is a way to obtain autocorrelations of a series.)

Models can be given labels of up to eight characters. Model labels are used in the printed output to identify the results for different models. The model label is specified as follows:

label **: MODEL . . . ;**

The following options can be used in the MODEL statement after a slash (/).

CENTER
centers the dependent variable by subtracting its mean and suppresses the intercept parameter from the model. This option is valid only when the model does not have regressors (explanatory variables).

NOINT
suppresses the intercept parameter.

Autoregressive Error Options

NLAG=*number*

NLAG=*(number-list)*
specifies the order of the autoregressive error process or the subset of autoregressive error lags to be fitted. Note that NLAG=3 is the same as NLAG=(1 2 3). If the NLAG= option is not specified, PROC AUTOREG does not fit an autoregressive model.

GARCH Estimation Options

DIST=*value*
specifies the distribution assumed for the error term in GARCH-type estimation. If no GARCH= option is specified, the option is ignored. If EGARCH is specified, the distribution is always the normal distribution. The values of the DIST= option are as follows:

T	specifies Student's *t* distribution.
NORMAL	specifies the standard normal distribution. The default is DIST=NORMAL.

GARCH=*(option-list)*

specifies a GARCH-type conditional heteroscedasticity model. The GARCH= option in the MODEL statement specifies the family of ARCH models to be estimated. The GARCH(1, 1) regression model is specified in the following statement:

```
model y = x1 x2 / garch=(q=1,p=1);
```

When you want to estimate the subset of ARCH terms, such as ARCH(1, 3), you can write the SAS statement as follows:

```
model y = x1 x2 / garch=(q=(1 3));
```

With the TYPE= option, you can specify various GARCH models. The IGARCH(2, 1) model without trend in variance is estimated as follows:

```
model y = / garch=(q=2,p=1,type=integ,noint);
```

The following options can be used in the GARCH=( ) option. The options are listed within parentheses and separated by commas.

Q=*number*

Q=*(number-list)*

specifies the order of the process or the subset of ARCH terms to be fitted.

P=*number*

P=*(number-list)*

specifies the order of the process or the subset of GARCH terms to be fitted. If only the P= option is specified, P= option is ignored and Q=1 is assumed.

TYPE=*value*

specifies the type of GARCH model. The values of the TYPE= option are as follows:

EXP | EGARCH specifies the exponential GARCH, or EGARCH, model.

INTEGRATED | IGARCH specifies the integrated GARCH, or IGARCH, model.

NELSON | NELSONCAO specifies the Nelson-Cao inequality constraints.

NOCONSTRAINT specifies the GARCH model with no constraints.

NONNEG specifies the GARCH model with nonnegativity constraints.

POWER | PGARCH specifies the power GARCH, or PGARCH, model.

QUADR | QUADRATIC | QGARCH specifies the quadratic GARCH, or QGARCH, model.

STATIONARY constrains the sum of GARCH coefficients to be less than 1.

THRES | THRESHOLD | TGARCH | GJR | GJRGARCH specifies the threshold GARCH, or TGARCH, model.

The default is TYPE=NELSON.

MEAN=*value*
specifies the functional form of the GARCH-M model. You can specify the following *values*:

LINEAR specifies the linear function:

$$y_t = \mathbf{x}_t'\beta + \delta h_t + \epsilon_t$$

LOG specifies the log function:

$$y_t = \mathbf{x}_t'\beta + \delta \ln(h_t) + \epsilon_t$$

SQRT specifies the square root function:

$$y_t = \mathbf{x}_t'\beta + \delta \sqrt{h_t} + \epsilon_t$$

NOINT
suppresses the intercept parameter in the conditional variance model. This option is valid only when you also specify the TYPE=INTEG option.

STARTUP=MSE | ESTIMATE
requests that the positive constant c for the start-up values of the GARCH conditional error variance process be estimated. By default, or if you specify STARTUP=MSE, the value of the mean squared error is used as the default constant.

TR
uses the trust region method for GARCH estimation. This algorithm is numerically stable, although computation is expensive. The double quasi-Newton method is the default.

Printing Options

ALL
requests all printing options.

ARCHTEST

ARCHTEST=*(option-list)*
specifies tests for the absence of ARCH effects. The following options can be used in the ARCHTEST=( ) option. The options are listed within parentheses and separated by commas.

QLM | QLMARCH
requests the Q and Engle's LM tests.

LK | LKARCH
requests Lee and King's ARCH tests.

WL | WLARCH
requests Wong and Li's ARCH tests.

ALL
requests all ARCH tests, namely Q and Engle's LM tests, Lee and King's tests, and Wong and Li's tests.

If ARCHTEST is defined without additional suboptions, it requests the Q and Engle's LM tests. That is,the statement

```
model return = x1 x2 / archtest;
```

is equivalent to the statement

```
model return = x1 x2 / archtest=(qlm);
```

The following statement requests Lee and King's tests and Wong and Li's tests:

```
model return = / archtest=(lk,wl);
```

BDS

BDS=*(option-list)*
specifies Brock-Dechert-Scheinkman (BDS) tests for independence. The following options can be used in the BDS=( ) option. The options are listed within parentheses and separated by commas.

M=*number*
specifies the maximum number of the embedding dimension. The BDS tests with embedding dimension from 2 to M are calculated. M must be an integer between 2 and 20. The default value of the M= suboption is 20.

D=*number*
specifies the parameter to determine the radius for BDS test. The BDS test sets up the radius as $r = D * \sigma$, where σ is the standard deviation of the time series to be tested. By default, D=1.5.

PVALUE=DIST | SIM
specifies the way to calculate the p-values. By default or if PVALUE=DIST is specified, the p-values are calculated according to the asymptotic distribution of BDS statistics (that is, the standard normal distribution). Otherwise, for samples of size less than 500, the p-values are obtained though Monte Carlo simulation.

Z=*value*
specifies the type of the time series (residuals) to be tested. You can specify the following *values*:

Y	specifies the regressand.
RO	specifies the OLS residuals.
R	specifies the residuals of the final model.
RM	specifies the structural residuals of the final model.
SR	specifies the standardized residuals of the final model, defined by residuals over the square root of the conditional variance.

The default is Z=Y.

If BDS is defined without additional suboptions, all suboptions are set as default values. That is, the following two statements are equivalent:

```
model return = x1 x2 / nlag=1 BDS;
```

```
model return = x1 x2 / nlag=1 BDS=(M=20, D=1.5, PVALUE=DIST, Z=Y);
```

To do the specification check of a GARCH(1,1) model, you can write the SAS statement as follows:

```
model return = / garch=(p=1,q=1) BDS=(Z=SR);
```

BP

BP=*(option-list)*

specifies Bai-Perron (BP) tests for multiple structural changes, introduced in Bai and Perron (1998). You can specify the following *options* in the BP=( ) option, in parentheses and separated by commas.

EPS=*number*

specifies the minimum length of regime; that is, if EPS=ε, for any $i, i = 1, ..., M$, $T_i - T_{i-1} \geq T\varepsilon$, where T is the sample size; $(T_1 ... T_M)$ are the break dates; and $T_0 = 0$ and $T_{M+1} = T$. The default is EPS=0.05.

ETA=*number*

specifies that the second method is to be used in the calculation of the $supF(l+1|l)$ test, and the minimum length of regime for the new additional break date is $(T_i - T_{i-1})\eta$ if ETA=η and the new break date is in regime i for the given break dates $(T_1 ... T_l)$. The default value of the ETA= suboption is the missing value; i.e., the first method is to be used in the calculation of the $supF(l+1|l)$ test and, no matter which regime the new break date is in, the minimum length of regime for the new additional break date is $T\varepsilon$ when EPS=ε.

HAC*<(option-list)>*

specifies that the heteroscedasticity- and autocorrelation-consistent estimator be applied in the estimation of the variance covariance matrix and the confidence intervals of break dates. When the HAC option is specified, you can specify the following options within parentheses and separated by commas:

KERNEL=*value*

specifies the type of kernel function. You can specify the following *values*:

BARTLETT specifies the Bartlett kernel function.

PARZEN specifies the Parzen kernel function.

QUADRATICSPECTRAL | QS specifies the quadratic spectral kernel function.

TRUNCATED specifies the truncated kernel function.

TUKEYHANNING | TUKEY | TH specifies the Tukey-Hanning kernel function.

The default is KERNEL=QUADRATICSPECTRAL.

KERNELLB=*number*
specifies the lower bound of the kernel weight value. Any kernel weight less than this lower bound is regarded as zero, which accelerates the calculation for big samples, especially for the quadratic spectral kernel. The default is KERNELLB=0.

BANDWIDTH=*value*
specifies the fixed bandwidth value or bandwidth selection method to use in the kernel function. You can specify the following *values*:

ANDREWS91 | ANDREWS specifies the Andrews (1991) bandwidth selection method.

NEWEYWEST94 | NW94 <(C=*number*)> specifies the Newey and West (1994) bandwidth selection method. You can specify the C= option in parentheses to calculate the lag selection parameter; the default is C=12.

SAMPLESIZE | SS <(*option-list*)> specifies that the bandwidth be calculated according to the following equation, based on the sample size:

$$b = \gamma T^r + c$$

where b is the bandwidth parameter and T is the sample size, and γ, r, and c are values specified by the following options within parentheses and separated by commas.

GAMMA=*number*
specifies the coefficient γ in the equation. The default is $\gamma = 0.75$.

RATE=*number*
specifies the growth rate r in the equation. The default is $r = 0.3333$.

CONSTANT=*number*
specifies the constant c in the equation. The default is $c = 0.5$.

INT
specifies that the bandwidth parameter must be integer; that is, $b = \lfloor \gamma T^r + c \rfloor$, where $\lfloor x \rfloor$ denotes the largest integer less than or equal to x.

number specifies the fixed value of the bandwidth parameter.

The default is BANDWIDTH=ANDREWS91.

PREWHITENING
specifies that prewhitening is required in the calculation.

In the calculation of the HAC estimator, the adjustment for degrees of freedom is always applied. See the section "Heteroscedasticity- and Autocorrelation-Consistent Covariance Matrix Estimator" on page 397 for more information about the HAC estimator.

HE
specifies that the errors are assumed to have heterogeneous distribution across regimes in the estimation of covariance matrix.

HO

specifies that Ω_is in the calculation of confidence intervals of break dates are different across regimes.

HQ

specifies that Q_is in the calculation of confidence intervals of break dates are different across regimes.

HR

specifies that the regressors are assumed to have heterogeneous distribution across regimes in the estimation of covariance matrix.

M=*number*

specifies the number of breaks. For a given M, the following tests are to be performed: (1) the $supF$ tests of no break versus the alternative hypothesis that there are i breaks, $i = 1, ..., M$; (2) the $UDmaxF$ and $WDmaxF$ double maximum tests of no break versus the alternative hypothesis that there are unknown number of breaks up to M; and (3) the $supF(l+1|l)$ tests of l versus $l+1$ breaks, $l = 0, ..., M$. The default is M=5.

NTHREADS=*number*

specifies the number of threads to be used for parallel computing. The default is the number of CPUs available.

P=*number*

specifies the number of covariates that have coefficients unchanged over time in the partial structural change model. The first P=p independent variables that are specified in the MODEL statement have unchanged coefficients; the rest of the independent variables have coefficients that change across regimes. The default is P=0; i.e., the pure structural change model is estimated.

PRINTEST=ALL | **BIC** | **LWZ** | **NONE** | **SEQ***<(number)>* | *number*

specifies in which structural change models the parameter estimates are to be printed. You can specify the following option values:

ALL	specifies that the parameter estimates in all structural change models with m breaks, $m = 0, ..., M$, be printed.
BIC	specifies that the parameter estimates in the structural change model that minimizes the BIC information criterion be printed.
LWZ	specifies that the parameter estimates in the structural change model that minimizes the LWZ information criterion be printed.
NONE	specifies that none of the parameter estimates be printed.
SEQ	specifies that the parameter estimates in the structural change model that is chosen by sequentially applying $supF(l+1|l)$ tests, l from 0 to M, be printed. If you specify the SEQ option, you can also specify the significance level in the parentheses, for example, SEQ(0.10). The first l such that the p-value of $supF(l+1|l)$ test is greater than the significance level is selected as the number of breaks in the structural change model. By default, the significance level 5% is used for the SEQ option; i.e., specifying SEQ is equivalent to specifying SEQ(0.05).

number specifies that the parameter estimates in the structural change model with the specified number of breaks be printed. If the specified number is greater than the number specified in the M= option, none of the parameter estimates are printed; that is, it is equivalent to specifying the NONE option.

The default is PRINTEST=ALL.

If you define the BP option without additional suboptions, all suboptions are set as default values. That is, the following two statements are equivalent:

```
model y = z1 z2 / BP;
```

```
model y = z1 z2 / BP=(M=5, P=0, EPS=0.05, PRINTEST=ALL);
```

To apply the HAC estimator with the Bartlett kernel function and print only the parameter estimates in the structural change model selected by the LWZ information criterion, you can write the SAS statement as follows:

```
model y = z1 z2 / BP=(HAC(KERNEL=BARTLETT), PRINTEST=LWZ);
```

To specify a partial structural change model, you can write the SAS statement as follows:

```
model y = x1 x2 x3 z1 z2 / NOINT BP=(P=3);
```

CHOW=$(obs_1 \ldots obs_n)$

computes Chow tests to evaluate the stability of the regression coefficient. The Chow test is also called the analysis-of-variance test.

Each value obs_i listed on the CHOW= option specifies a break point of the sample. The sample is divided into parts at the specified break point, with observations before obs_i in the first part and obs_i and later observations in the second part, and the fits of the model in the two parts are compared to whether both parts of the sample are consistent with the same model.

The break points obs_i refer to observations within the time range of the dependent variable, ignoring missing values before the start of the dependent series. Thus, CHOW=20 specifies the 20th observation after the first nonmissing observation for the dependent variable. For example, if the dependent variable Y contains 10 missing values before the first observation with a nonmissing Y value, then CHOW=20 actually refers to the 30th observation in the data set.

When you specify the break point, you should note the number of presample missing values.

COEF

prints the transformation coefficients for the first p observations. These coefficients are formed from a scalar multiplied by the inverse of the Cholesky root of the Toeplitz matrix of autocovariances.

CORRB
prints the estimated correlations of the parameter estimates.

COVB
prints the estimated covariances of the parameter estimates.

COVEST= OP | **HESSIAN** | **QML** | **HC0** | **HC1** | **HC2** | **HC3** | **HC4** | **HAC***<(. . .)>* | **NEWEYWEST***<(. . .)>*
specifies the type of covariance matrix.

When COVEST=OP is specified, the outer product matrix is used to compute the covariance matrix of the parameter estimates; by default, COVEST=OP. The COVEST=HESSIAN option produces the covariance matrix by using the Hessian matrix. The quasi–maximum likelihood estimates are computed with COVEST=QML, which is equivalent to COVEST=HC0. When the final model is an OLS or AR error model, COVEST=OP, HESSIAN, or QML is ignored; the method to calculate the estimate of covariance matrix is illustrated in the section "Variance Estimates and Standard Errors" on page 390.

When you specify COVEST=HC*n*, where $n = 0, 1, 2, 3, 4$, the corresponding heteroscedasticity-consistent covariance matrix estimator (HCCME) is calculated.

The HAC option specifies the heteroscedasticity- and autocorrelation-consistent (HAC) covariance matrix estimator. When you specify the HAC option, you can specify the following options in parentheses and separate them with commas:

KERNEL=*value*
specifies the type of kernel function. You can specify the following *values*:

BARTLETT specifies the Bartlett kernel function.

PARZEN specifies the Parzen kernel function.

QUADRATICSPECTRAL | QS specifies the quadratic spectral kernel function.

TRUNCATED specifies the truncated kernel function.

TUKEYHANNING | TUKEY | TH specifies the Tukey-Hanning kernel function.

The default is KERNEL=QUADRATICSPECTRAL.

KERNELLB=*number*
specifies the lower bound of the kernel weight value. Any kernel weight less than this lower bound is regarded as zero, which accelerates the calculation for big samples, especially for the quadratic spectral kernel. The default is KERNELLB=0.

BANDWIDTH=*value*
specifies the fixed bandwidth value or bandwidth selection method to use in the kernel function. You can specify the following *values*:

ANDREWS91 | ANDREWS specifies the Andrews (1991) bandwidth selection method.

NEWEYWEST94 | NW94 <(C=*number*)> specifies the Newey and West (1994) bandwidth selection method. You can specify the C= option in the parentheses to calculate the lag selection parameter; the default is C=12.

SAMPLESIZE | SS <(*option-list*)> specifies that the bandwidth be calculated according to the following equation, based on the sample size:

$$b = \gamma T^r + c$$

where b is the bandwidth parameter and T is the sample size, and γ, r, and c are values specified by the following options within parentheses and separated by commas.

GAMMA=*number*
specifies the coefficient γ in the equation. The default is $\gamma = 0.75$.

RATE=*number*
specifies the growth rate r in the equation. The default is $r = 0.3333$.

CONSTANT=*number*
specifies the constant c in the equation. The default is $c = 0.5$.

INT
specifies that the bandwidth parameter must be integer; that is, $b = \lfloor \gamma T^r + c \rfloor$, where $\lfloor x \rfloor$ denotes the largest integer less than or equal to x.

number specifies the fixed value of the bandwidth parameter.

The default is BANDWIDTH=ANDREWS91.

PREWHITENING
specifies that prewhitening is required in the calculation.

ADJUSTDF
specifies that the adjustment for degrees of freedom be required in the calculation.

The COVEST=NEWEYWEST option specifies the well-known Newey-West estimator, a special HAC estimator with (1) the Bartlett kernel; (2) the bandwidth parameter determined by the equation based on the sample size, $b = \lfloor \gamma T^r + c \rfloor$; and (3) no adjustment for degrees of freedom and no prewhitening. By default the bandwidth parameter for Newey-West estimator is $\lfloor 0.75T^{0.3333} + 0.5 \rfloor$, as shown in equation (15.17) in Stock and Watson (2002). When you specify COVEST=NEWEYWEST, you can specify the following options in parentheses and separate them with commas:

GAMMA= *number*
specifies the coefficient γ in the equation. The default is $\gamma = 0.75$.

RATE= *number*
specifies the growth rate r in the equation. The default is $r = 0.3333$.

CONSTANT= *number*
specifies the constant c in the equation. The default is $c = 0.5$.

The following two statements are equivalent:

```
model y = x / COVEST=NEWEYWEST;
```

```
model y = x / COVEST=HAC(KERNEL=BARTLETT,
                         BANDWIDTH=SAMPLESIZE(GAMMA=0.75,
                                              RATE=0.3333,
                                              CONSTANT=0.5,
                                              INT));
```

Another popular sample-size-dependent bandwidth, $\left\lfloor T^{1/4} + 1.5 \right\rfloor$, as mentioned in Newey and West (1987), can be specified by the following statement:

```
model y = x / COVEST=NEWEYWEST(GAMMA=1,RATE=0.25,CONSTANT=1.5);
```

See the section "Heteroscedasticity- and Autocorrelation-Consistent Covariance Matrix Estimator" on page 397 for more information about HC0 to HC4, HAC, and Newey-West estimators.

DW=*n*

prints Durbin-Watson statistics up to the order *n*. The default is DW=1. When the LAGDEP option is specified, the Durbin-Watson statistic is not printed unless the DW= option is explicitly specified.

DWPROB

now produces *p*-values for the generalized Durbin-Watson test statistics for large sample sizes. Previously, the Durbin-Watson probabilities were calculated only for small sample sizes. The new method of calculating Durbin-Watson probabilities is based on the algorithm of Ansley, Kohn, and Shively (1992).

GINV

prints the inverse of the Toeplitz matrix of autocovariances for the Yule-Walker solution. See the section "Computational Methods" on page 389 later in this chapter for more information.

GODFREY

GODFREY=*r*

produces Godfrey's general Lagrange multiplier test against ARMA errors.

ITPRINT

prints the objective function and parameter estimates at each iteration. The objective function is the full log likelihood function for the maximum likelihood method, while the error sum of squares is produced as the objective function of unconditional least squares. For the ML method, the ITPRINT option prints the value of the full log likelihood function, not the concentrated likelihood.

LAGDEP

LAGDV

prints the Durbin *t* statistic, which is used to detect residual autocorrelation in the presence of lagged dependent variables. See the section "Generalized Durbin-Watson Tests" on page 418 for details.

LAGDEP=*name*

LAGDV=*name*

prints the Durbin h statistic for testing the presence of first-order autocorrelation when regressors contain the lagged dependent variable whose name is specified as LAGDEP=*name*. If the Durbin h statistic cannot be computed, the asymptotically equivalent t statistic is printed instead. See the section "Generalized Durbin-Watson Tests" on page 418 for details.

When the regression model contains several lags of the dependent variable, specify the lagged dependent variable for the smallest lag in the LAGDEP= option. For example:

```
model y = x1 x2 ylag2 ylag3 / lagdep=ylag2;
```

LOGLIKL

prints the log likelihood value of the regression model, assuming normally distributed errors.

NOPRINT

suppresses all printed output.

NORMAL

specifies the Jarque-Bera's normality test statistic for regression residuals.

PARTIAL

prints partial autocorrelations.

PCHOW=*(obs$_1$ ... obs$_n$)*

computes the predictive Chow test. The form of the PCHOW= option is the same as the CHOW= option; see the discussion of the CHOW= option earlier in this chapter.

RESET

produces Ramsey's RESET test statistics. The RESET option tests the null model

$$y_t = \mathbf{x}_t \beta + u_t$$

against the alternative

$$y_t = \mathbf{x}_t \beta + \sum_{j=2}^{p} \phi_j \hat{y}_t^j + u_t$$

where $\hat{y}_t$ is the predicted value from the OLS estimation of the null model. The RESET option produces three RESET test statistics for $p = 2$, 3, and 4.

RUNS

RUNS=*(Z=value)*

specifies the runs test for independence. The Z= suboption specifies the type of the time series or residuals to be tested. The values of the Z= suboption are as follows:

Y	specifies the regressand. The default is Z=Y.
RO	specifies the OLS residuals.
R	specifies the residuals of the final model.
RM	specifies the structural residuals of the final model.
SR	specifies the standardized residuals of the final model, defined by residuals over the square root of the conditional variance.

STATIONARITY=(*test*< =(*test-options*) > < , *test*< =(*test-options*) > > . . . < , *test*< =(*test-options*) > >)

specifies tests of stationarity or unit roots. You can specify one of more of the following *tests* along with their *test-options*. For example, the following statement tests the stationarity of a variable by using the augmented Dickey-Fuller unit root test and the KPSS test in which the quadratic spectral kernel is applied:

```
model y=  / stationarity = (adf, kpss=(kernel=qs));
```

STATIONARITY=(ADF)

STATIONARITY=(ADF=(*value . . . value***)**

produces the augmented Dickey-Fuller unit root test (Dickey and Fuller 1979). As in the Phillips-Perron test, three regression models can be specified for the null hypothesis for the augmented Dickey-Fuller test (zero mean, single mean, and trend). These models assume that the disturbances are distributed as white noise. The augmented Dickey-Fuller test can account for the serial correlation between the disturbances in some way. The model, with the time trend specification for example, is

$$y_t = \mu + \rho y_{t-1} + \delta t + \gamma_1 \Delta y_{t-1} + \ldots + \gamma_p \Delta y_{t-p} + u_t$$

This formulation has the advantage that it can accommodate higher-order autoregressive processes in u_t. The test statistic follows the same distribution as the Dickey-Fuller test statistic. For more information, see the section "PROBDF Function for Dickey-Fuller Tests" on page 187.

In the presence of regressors, the ADF option tests the cointegration relation between the dependent variable and the regressors. Following Engle and Granger (1987), a two-step estimation and testing procedure is carried out, in a fashion similar to the Phillips-Ouliaris test. The OLS residuals of the regression in the MODEL statement are used to compute the t statistic of the augmented Dickey-Fuller regression in a second step. Three cases arise based on which type of deterministic terms are included in the first step of regression. Only the constant term and linear trend cases are practically useful (Davidson and MacKinnon 1993, page 721), and therefore are computed and reported. The test statistic, as shown in Phillips and Ouliaris (1990), follows the same distribution as the $\hat{Z}_t$ statistic in the Phillips-Ouliaris cointegration test. The asymptotic distribution is tabulated in tables IIa–IIc of Phillips and Ouliaris (1990), and the finite sample distribution is obtained in Table 2 and Table 3 in Engle and Yoo (1987) by Monte Carlo simulation.

STATIONARITY=(ERS)

STATIONARITY=(ERS=(*value***)**

STATIONARITY=(NP)

STATIONARITY=(NP=(*value***)**

provides a class of *efficient unit root tests*, because they reduce the size distortion and improve the power compared with traditional unit root tests such as the augmented Dickey-Fuller and Phillips-Perron tests. Two test statistics are reported with the ERS= suboption: the point optimal test and the DF-GLS test, which are originally proposed in Elliott, Rothenberg, and Stock (1996). Elliott, Rothenberg, and Stock suggest using the Schwarz Bayesian information criterion to select the optimal lag length in the augmented Dickey-Fuller regression. The maximum lag length can be specified by ERS=*value*. The minimum lag length is 3 and the default maximum lag length is 8.

Six tests, namely MZ_α, MSB, MZ_t, the modified point optimal test, the point optimal test, and the DF-GLS test, which are discussed in Ng and Perron (2001), are reported with the NP= suboption. Ng and Perron suggest using the modified AIC to select the optimal lag length in the augmented Dickey-Fuller regression by using GLS detrended data. The maximum lag length can be specified by NP=*value*. The default maximum lag length is 8. The maximum lag length in the ERS tests and Ng-Perron tests cannot exceed $T/2 - 2$, where T is the sample size.

STATIONARITY=(KPSS)
STATIONARITY=(KPSS=(KERNEL=(*type*))
STATIONARITY=(KPSS=(KERNEL=(*type* TRUNCPOINTMETHOD))

produce the Kwiatkowski, Phillips, Schmidt, and Shin (1992) (KPSS) unit root test or Shin (1994) cointegration test.

Unlike the null hypothesis of the Dickey-Fuller and Phillips-Perron tests, the null hypothesis of the KPSS states that the time series is stationary. As a result, it tends to reject a random walk more often. If the model does not have an intercept, the KPSS option performs the KPSS test for three null hypothesis cases: zero mean, single mean, and deterministic trend. Otherwise, it reports the single mean and deterministic trend only. It computes a test statistic and provides p-value (Hobijn, Franses, and Ooms 2004) for the hypothesis that the random walk component of the time series is equal to zero in the following cases (for more information, see "Kwiatkowski, Phillips, Schmidt, and Shin (KPSS) Unit Root Test and Shin Cointegration Test" on page 413):

Zero mean — computes the KPSS test statistic based on the zero mean autoregressive model.

$$y_t = u_t$$

Single mean — computes the KPSS test statistic based on the autoregressive model with a constant term.

$$y_t = \mu + u_t$$

Trend — computes the KPSS test statistic based on the autoregressive model with constant and time trend terms.

$$y_t = \mu + \delta t + u_t$$

This test depends on the long-run variance of the series being defined as

$$\sigma^2_{Tl} = \frac{1}{T}\sum_{i=1}^{T}\hat{u}_i^2 + \frac{2}{T}\sum_{s=1}^{l} w_{sl} \sum_{t=s+1}^{T} \hat{u}_t \hat{u}_{t-s}$$

where w_{sl} is a kernel, s is a maximum lag (truncation point), and $\hat{u}_t$ are OLS residuals or original data series. You can specify two types of the kernel:

KERNEL=NW | BART — Newey-West (or Bartlett) kernel

$$w(s,l) = 1 - \frac{s}{l+1}$$

KERNEL=QS — Quadratic spectral kernel

$$w(s/l) = w(x) = \frac{25}{12\pi^2 x^2}\left(\frac{\sin(6\pi x/5)}{6\pi x/5} - \cos(6\pi x/5)\right)$$

You can set the truncation point l by using three different methods:

SCHW=c Schwert maximum lag formula

$$l = \max\left\{1, \text{floor}\left[c\left(\frac{T}{100}\right)^{1/4}\right]\right\}$$

LAG=*l* LAG=*l* manually defined number of lags.

AUTO Automatic bandwidth selection (Hobijn, Franses, and Ooms 2004) (for details, see "Kwiatkowski, Phillips, Schmidt, and Shin (KPSS) Unit Root Test and Shin Cointegration Test" on page 413).

If STATIONARITY=KPSS is defined without additional parameters, the Newey-West kernel is used. For the Newey-West kernel the default is the Schwert truncation point method with $c = 12$. For the quadratic spectral kernel the default is AUTO.

The KPSS test can be used in general time series models because its limiting distribution is derived in the context of a class of weakly dependent and heterogeneously distributed data. The limiting probability for the KPSS test is computed assuming that error disturbances are normally distributed. The *p*-values that are reported are based on the simulation of the limiting probability for the KPSS test.

To test for stationarity of a variable, *y*, by using default KERNEL= NW and SCHW= 12, you can use the following statements:

```
/*-- test for stationarity of regression residuals --*/
proc autoreg data=a;
   model y=  / stationarity = (KPSS);
run;
```

To test for stationarity of a variable, *y*, by using quadratic spectral kernel and automatic bandwidth selection, you can use the following statements:

```
/*-- test for stationarity using quadratic
     spectral kernel and automatic bandwidth selection --*/
proc autoreg data=a;
   model y= /
   stationarity = (KPSS=(KERNEL=QS AUTO));
run;
```

If there are regressors in the MODEL statement except for the intercept, the Shin (1994) cointegration test, an extension of the KPSS test, is carried out. The limiting distribution of the tests, and then the reported *p*-values, are different from those in the KPSS tests. For more information, see the section "Kwiatkowski, Phillips, Schmidt, and Shin (KPSS) Unit Root Test and Shin Cointegration Test" on page 413.

STATIONARITY=(PHILLIPS)

STATIONARITY=(PHILLIPS=(*value . . . value***)**

produces the Phillips-Perron unit root test when there are no regressors in the MODEL statement. When the model includes regressors, the PHILLIPS option produces the Phillips-Ouliaris cointegration test. The PHILLIPS option can be abbreviated as PP.

The PHILLIPS option performs the Phillips-Perron test for three null hypothesis cases: zero mean, single mean, and deterministic trend. For each case, the PHILLIPS option computes two test statistics, $\hat{Z}_\rho$ and $\hat{Z}_t$—in the original paper, Phillips and Perron (1988), they are referred to as $\hat{Z}_\alpha$ and $\hat{Z}_t$)— and reports their p-values. These test statistics have the same limiting distributions as the corresponding Dickey-Fuller tests.

The three types of the Phillips-Perron unit root test reported by the PHILLIPS option are as follows:

Zero mean — computes the Phillips-Perron test statistic based on the zero mean autoregressive model:

$$y_t = \rho y_{t-1} + u_t$$

Single mean — computes the Phillips-Perron test statistic based on the autoregressive model with a constant term:

$$y_t = \mu + \rho y_{t-1} + u_t$$

Trend — computes the Phillips-Perron test statistic based on the autoregressive model with constant and time trend terms:

$$y_t = \mu + \rho y_{t-1} + \delta t + u_t$$

You can specify several truncation points l for weighted variance estimators by using the PHILLIPS=($l_1 \ldots l_n$) specification. The statistic for each truncation point l is computed as

$$\sigma^2_{Tl} = \frac{1}{T}\sum_{i=1}^{T} \hat{u}_i^2 + \frac{2}{T}\sum_{s=1}^{l} w_{sl} \sum_{t=s+1}^{T} \hat{u}_t \hat{u}_{t-s}$$

where $w_{sl} = 1 - s/(l+1)$ and $\hat{u}_t$ are OLS residuals. If you specify the PHILLIPS option without specifying truncation points, the default truncation point is $\max(1, \sqrt{T}/5)$, where T is the number of observations.

The Phillips-Perron test can be used in general time series models because its limiting distribution is derived in the context of a class of weakly dependent and heterogeneously distributed data. The marginal probability for the Phillips-Perron test is computed assuming that error disturbances are normally distributed.

When there are regressors in the MODEL statement, the PHILLIPS option computes the Phillips-Ouliaris cointegration test statistic by using the least squares residuals. The normalized cointegrating vector is estimated using OLS regression. Therefore, the cointegrating vector estimates might vary with the regressand (normalized element) unless the regression R-square is 1. You can define the truncation points in the calculation of weighted variance estimators, $\sigma^2_{Tl}, l = l_1 \ldots l_n$, in the same way as you define the truncation points for the Phillips-Perron test—by using the PHILLIPS=($l_1 \ldots l_n$) option.

The marginal probabilities for cointegration testing are not produced. You can refer to Phillips and Ouliaris (1990) tables Ia–Ic for the $\hat{Z}_\alpha$ test and tables IIa–IIc for the $\hat{Z}_t$ test. The standard residual-based cointegration test can be obtained using the NOINT option in the MODEL

statement, and the demeaned test is computed by including the intercept term. To obtain the demeaned and detrended cointegration tests, you should include the time trend variable in the regressors. Refer to Phillips and Ouliaris (1990) or Hamilton (1994, Tbl. 19.1) for information about the Phillips-Ouliaris cointegration test. Note that Hamilton (1994, Tbl. 19.1) uses Z_ρ and Z_t instead of the original Phillips and Ouliaris (1990) notation. This chapter adopts the notation introduced in Hamilton. To distinguish from Student's t distribution, these two statistics are named accordingly as ρ (rho) and τ (tau).

TP

TP=*(***Z=***value)*

specifies the turning point test for independence. The Z= suboption specifies the type of the time series or residuals to be tested. You can specify the following *values*:

Y	specifies the regressand. The default is Z=Y.
RO	specifies the OLS residuals.
R	specifies the residuals of the final model.
RM	specifies the structural residuals of the final model.
SR	specifies the standardized residuals of the final model, defined by residuals over the square root of the conditional variance.

URSQ

prints the uncentered regression R^2. The uncentered regression R^2 is useful to compute Lagrange multiplier test statistics, since most LM test statistics are computed as T *URSQ, where T is the number of observations used in estimation.

VNRRANK

VNRRANK=*(option-list)*

specifies the rank version of the von Neumann ratio test for independence. You can specify the following options in the VNRRANK=( ) option. The options are listed within parentheses and separated by commas.

PVALUE=DIST | **SIM**

specifies how to calculate the p-value. You can specify the following values:

DIST	calculates the p-value according to the asymptotic distribution of the statistic (that is, the standard normal distribution).
SIM	calculates the p-value as follows:

- If the sample size is less than or equal to 10, the p-value is calculated according to the exact CDF of the statistic.
- If the sample size is between 11 and 100, the p-value is calculated according to Monte Carlo simulation of the distribution of the statistic.
- If the sample size is more than 100, the p-value is calculated according to the standard normal distribution because the simulated distribution of the statistic in this case is almost the same as the standard normal distribution.

By default, PVALUE=DIST.

Z=*value*

specifies the type of the time series or residuals to be tested. You can specify the following *values*:

Y	specifies the regressand.
RO	specifies the OLS residuals.
R	specifies the residuals of the final model.
RM	specifies the structural residuals of the final model.
SR	specifies the standardized residuals of the final model, defined by residuals over the square root of the conditional variance.

By default, Z=Y.

Stepwise Selection Options

BACKSTEP

removes insignificant autoregressive parameters. The parameters are removed in order of least significance. This backward elimination is done only once on the Yule-Walker estimates computed after the initial ordinary least squares estimation. You can use the BACKSTEP option with all estimation methods because the initial parameter values for other estimation methods are estimated by using the Yule-Walker method.

SLSTAY=*value*

specifies the significance level criterion to be used by the BACKSTEP option. By default, SLSTAY=.05.

Estimation Control Options

CONVERGE=*value*

specifies the convergence criterion. If the maximum absolute value of the change in the autoregressive parameter estimates between iterations is less than this criterion, then convergence is assumed. By default, CONVERGE=.001.

If you specify the GARCH= option or the HETERO statement, convergence is assumed when the absolute maximum gradient is smaller than the value specified by the CONVERGE= option or when the relative gradient is smaller than 1E–8. By default, CONVERGE=1E–5.

INITIAL=*(initial-values)*

START=*(initial-values)*

specifies initial values for some or all of the parameter estimates. This option is not applicable when the Yule-Walker method or iterative Yule-Walker method is used. The specified values are assigned to model parameters in the same order in which the parameter estimates are printed in the AUTOREG procedure output. The order of values in the INITIAL= or START= option is as follows: the intercept, the regressor coefficients, the autoregressive parameters, the ARCH parameters, the GARCH parameters, the inverted degrees of freedom for Student's t distribution, the start-up value for conditional variance, and the heteroscedasticity model parameters η specified by the HETERO statement.

The following is an example of specifying initial values for an AR(1)-GARCH(1, 1) model with regressors X1 and X2:

```
/*-- specifying initial values --*/
model y = w x / nlag=1 garch=(p=1,q=1)
                initial=(1 1 1 .5 .8 .1 .6);
```

The model that is specified by this MODEL statement is

$$y_t = \beta_0 + \beta_1 w_t + \beta_2 x_t + \nu_t$$

$$\nu_t = \epsilon_t - \phi_1 \nu_{t-1}$$

$$\epsilon_t = \sqrt{h_t} e_t$$

$$h_t = \omega + \alpha_1 \epsilon_{t-1}^2 + \gamma_1 h_{t-1}$$

$$\epsilon_t \ \mathrm{N}(0, \sigma_t^2)$$

The initial values for the regression parameters, INTERCEPT (β_0), X1 (β_1), and X2 (β_2), are specified as 1. The initial value of the AR(1) coefficient (ϕ_1) is specified as 0.5. The initial value of ARCH0 (ω) is 0.8, the initial value of ARCH1 (α_1) is 0.1, and the initial value of GARCH1 (γ_1) is 0.6.

When you use the RESTRICT statement, the initial values that you specify in the INITIAL= option should satisfy the restrictions specified for the parameter estimates. If they do not, these initial values are adjusted to satisfy the restrictions.

LDW

specifies that *p*-values for the Durbin-Watson test be computed by using a linearized approximation of the design matrix when the model is nonlinear because an autoregressive error process is present. (The Durbin-Watson tests of the OLS linear regression model residuals are not affected by the LDW option.) See White (1992) for information about Durbin-Watson testing of nonlinear models.

MAXITER=*number*

sets the maximum number of iterations allowed. The default is MAXITER=50. When you specify both the GARCH= option in the MODEL statement and the MAXITER= option in the NLOPTIONS statement, the MAXITER= option in the MODEL statement is ignored. This option is not applicable when the Yule-Walker method is used.

METHOD=*value*

requests the type of estimates to be computed. You can specify the following *values*:

ML	specifies maximum likelihood estimates.
ULS	specifies unconditional least squares estimates.
YW	specifies Yule-Walker estimates.
ITYW	specifies iterative Yule-Walker estimates.

The default is defined as follows:

- When the GARCH= option or the HETERO statement is specified, METHOD=ML by default.
- When the GARCH= option and the HETERO statement are not specified but the NLAG= option and the LAGDEP option are specified, METHOD=ML by default.
- When the GARCH= option, the LAGDEP option, and the HETERO statement are not specified, but the NLAG= option is specified, METHOD=YW by default.

- When none of the NLAG= option, the GARCH= option, and the HETERO statement is specified (that is, only the OLS model is to be estimated), then the estimates are calculated through the OLS method and the METHOD= option is ignored.

NOMISS
requests the estimation to the first contiguous sequence of data with no missing values. Otherwise, all complete observations are used.

OPTMETHOD=QN | TR
specifies the optimization technique when the GARCH or heteroscedasticity model is estimated. The OPTMETHOD=QN option specifies the quasi-Newton method. The OPTMETHOD=TR option specifies the trust region method. The default is OPTMETHOD=QN.

HETERO Statement

HETERO *variables / options* **;**

The HETERO statement specifies variables that are related to the heteroscedasticity of the residuals and the way these variables are used to model the error variance of the regression.

The heteroscedastic regression model supported by the HETERO statement is

$$y_t = \mathbf{x}_t \beta + \epsilon_t$$

$$\epsilon_t \sim \mathrm{N}(0, \sigma_t^2)$$

$$\sigma_t^2 = \sigma^2 h_t$$

$$h_t = l(\mathbf{z}'_t \eta)$$

The HETERO statement specifies a model for the conditional variance h_t. The vector $\mathbf{z}_t$ is composed of the variables listed in the HETERO statement, η is a parameter vector, and $l(\cdot)$ is a link function that depends on the value of the LINK= option. In the printed output, $HET0$ represents the estimate of sigma, while $HET1$ - $HETn$ are the estimates of parameters in the η vector.

The keyword XBETA can be used in the *variables* list to refer to the model predicted value $\mathbf{x}'_t \beta$. If XBETA is specified in the *variables* list, other variables in the HETERO statement will be ignored. In addition, XBETA cannot be specified in the GARCH process.

For heteroscedastic regression models without GARCH effects, the errors ϵ_t are assumed to be uncorrelated — the heteroscedasticity models specified by the HETERO statement cannot be combined with an autoregressive model for the errors. Thus, when a HETERO statement is used, the NLAG= option cannot be specified unless the GARCH= option is also specified.

You can specify the following options in the HETERO statement.

LINK=*value*
specifies the functional form of the heteroscedasticity model. By default, LINK=EXP. If you specify a GARCH model with the HETERO statement, the model is estimated using LINK= LINEAR only. For details, see the section "Using the HETERO Statement with GARCH Models" on page 394. Values of the LINK= option are as follows:

EXP specifies the exponential link function. The following model is estimated when you specify LINK=EXP:

$$h_t = \exp(\mathbf{z}'_t \eta)$$

SQUARE specifies the square link function. The following model is estimated when you specify LINK=SQUARE:

$$h_t = (1 + \mathbf{z}'_t \eta)^2$$

LINEAR specifies the linear function; that is, the HETERO statement variables predict the error variance linearly. The following model is estimated when you specify LINK=LINEAR:

$$h_t = (1 + \mathbf{z}'_t \eta)$$

COEF=*value*

imposes constraints on the estimated parameters η of the heteroscedasticity model. You can specify the following *values*:

NONNEG specifies that the estimated heteroscedasticity parameters η must be nonnegative.

UNIT constrains all heteroscedasticity parameters η to equal 1.

ZERO constrains all heteroscedasticity parameters η to equal 0.

UNREST specifies unrestricted estimation of η.

If you specify the GARCH= option in the MODEL statement, the default is COEF=NONNEG. If you do not specify the GARCH= option in the MODEL statement, the default is COEF=UNREST.

STD=*value*

imposes constraints on the estimated standard deviation σ of the heteroscedasticity model. You can specify the following *values*:

NONNEG specifies that the estimated standard deviation parameter σ must be nonnegative.

UNIT constrains the standard deviation parameter σ to equal 1.

UNREST specifies unrestricted estimation of σ.

The default is STD=UNREST.

TEST=LM

produces a Lagrange multiplier test for heteroscedasticity. The null hypothesis is homoscedasticity; the alternative hypothesis is heteroscedasticity of the form specified by the HETERO statement. The power of the test depends on the variables specified in the HETERO statement.

The test may give different results depending on the functional form specified by the LINK= option. However, in many cases the test does not depend on the LINK= option. The test is invariant to the form of h_t when $h_t(0) = 1$ and $h'_t(0) \neq 0$. (The condition $h_t(0) = 1$ is satisfied except when the NOCONST option is specified with LINK=SQUARE or LINK=LINEAR.)

NOCONST
specifies that the heteroscedasticity model does not include the unit term for the LINK=SQUARE and LINK=LINEAR options. For example, the following model is estimated when you specify the options LINK=SQUARE NOCONST:

$$h_t = (\mathbf{z}'_t \eta)^2$$

NLOPTIONS Statement

NLOPTIONS < *options* > **;**

PROC AUTOREG uses the nonlinear optimization (NLO) subsystem to perform nonlinear optimization tasks when the GARCH= option is specified. If the GARCH= option is not specified, the NLOPTIONS statement is ignored. For a list of all the options of the NLOPTIONS statement, see Chapter 7, "Nonlinear Optimization Methods."

OUTPUT Statement

OUTPUT < **OUT=***SAS-data-set* > < *options* > < *keyword=name* > **;**

The OUTPUT statement creates an output SAS data set as specified by the following options.

OUT=*SAS-data-set*
names the output SAS data set to contain the predicted and transformed values. If the OUT= option is not specified, the new data set is named according to the DATA*n* convention.

You can specify any of the following *options*.

ALPHACLI=*number*
sets the confidence limit size for the estimates of future values of the response time series. The ALPHACLI= value must be between 0 and 1. The resulting confidence interval has 1–*number* confidence. The default is ALPHACLI=0.05, which corresponds to a 95% confidence interval.

ALPHACLM=*number*
sets the confidence limit size for the estimates of the structural or regression part of the model. The ALPHACLM= value must be between 0 and 1. The resulting confidence interval has 1–*number* confidence. The default is ALPHACLM=0.05, which corresponds to a 95% confidence interval.

ALPHACSM=0.01 | 0.05 | 0.10
specifies the significance level for the upper and lower bounds of the CUSUM and CUSUMSQ statistics output by the CUSUMLB=, CUSUMUB=, CUSUMSQLB=, and CUSUMSQUB= options. The significance level specified by the ALPHACSM= option can be 0.01, 0.05, or 0.10. Other values are not supported.

You can specify the following values for *keyword=name*, where *keyword* specifies the statistic to include in the output data set and *name* gives the name of the variable in the OUT= data set to contain the statistic.

BLUS=*variable*
specifies the name of a variable to contain the values of the Theil's BLUS residuals. Refer to Theil (1971) for more information on BLUS residuals.

CEV=*variable*

HT=*variable*
writes to the output data set the value of the error variance σ_t^2 from the heteroscedasticity model specified by the HETERO statement or the value of the conditional error variance h_t by the GARCH= option in the MODEL statement.

CPEV=*variable*
writes the conditional prediction error variance to the output data set. The value of conditional prediction error variance is equal to that of the conditional error variance when there are no autoregressive parameters. See the section "Predicted Values" on page 427 for details.

CONSTANT=*variable*
writes the transformed intercept to the output data set. The details of the transformation are described in "Computational Methods" on page 389.

CUSUM=*variable*
specifies the name of a variable to contain the CUSUM statistics.

CUSUMSQ=*variable*
specifies the name of a variable to contain the CUSUMSQ statistics.

CUSUMUB=*variable*
specifies the name of a variable to contain the upper confidence bound for the CUSUM statistic.

CUSUMLB=*variable*
specifies the name of a variable to contain the lower confidence bound for the CUSUM statistic.

CUSUMSQUB=*variable*
specifies the name of a variable to contain the upper confidence bound for the CUSUMSQ statistic.

CUSUMSQLB=*variable*
specifies the name of a variable to contain the lower confidence bound for the CUSUMSQ statistic.

LCL=*name*
writes the lower confidence limit for the predicted value (specified in the PREDICTED= option) to the output data set. The size of the confidence interval is set by the ALPHACLI= option. See the section "Predicted Values" on page 427 for details.

LCLM=*name*
writes the lower confidence limit for the structural predicted value (specified in the PREDICTEDM= option) to the output data set under the name given. The size of the confidence interval is set by the ALPHACLM= option.

PREDICTED=*name*

P=*name*
writes the predicted values to the output data set. These values are formed from both the structural and autoregressive parts of the model. See the section "Predicted Values" on page 427 for details.

PREDICTEDM=*name*

PM=*name*

writes the structural predicted values to the output data set. These values are formed from only the structural part of the model. See the section "Predicted Values" on page 427 for details.

RECPEV=*variable*

specifies the name of a variable to contain the part of the predictive error variance (v_t) that is used to compute the recursive residuals.

RECRES=*variable*

specifies the name of a variable to contain recursive residuals. The recursive residuals are used to compute the CUSUM and CUSUMSQ statistics.

RESIDUAL=*name*

R=*name*

writes the residuals from the predicted values based on both the structural and time series parts of the model to the output data set.

RESIDUALM=*name*

RM=*name*

writes the residuals from the structural prediction to the output data set.

TRANSFORM=*variables*

transforms the specified variables from the input data set by the autoregressive model and writes the transformed variables to the output data set. The details of the transformation are described in the section "Computational Methods" on page 389. If you need to reproduce the data suitable for re-estimation, you must also transform an intercept variable. To do this, transform a variable that is all 1s or use the CONSTANT= option.

UCL=*name*

writes the upper confidence limit for the predicted value (specified in the PREDICTED= option) to the output data set. The size of the confidence interval is set by the ALPHACLI= option. For more information, see the section "Predicted Values" on page 427.

UCLM=*name*

writes the upper confidence limit for the structural predicted value (specified in the PREDICTEDM= option) to the output data set. The size of the confidence interval is set by the ALPHACLM= option.

RESTRICT Statement

RESTRICT *equation , . . . , equation* **;**

The RESTRICT statement provides constrained estimation and places restrictions on the parameter estimates for covariates in the preceding MODEL statement. The AR, GARCH, and HETERO parameters are also supported in the RESTRICT statement when you specify the GARCH= option. Any number of RESTRICT statements can follow a MODEL statement. To specify more than one restriction in a single RESTRICT statement, separate them with commas.

Each restriction is written as a linear equation composed of constants and parameter names. Refer to model parameters by the name of the corresponding regressor variable. Each name that is used in the equation must be a regressor in the preceding MODEL statement. Use the keyword INTERCEPT to refer to the intercept parameter in the model. For the names of these parameters, see the section "OUTEST= Data Set" on page 431. Inequality constraints are supported only when you specify the GARCH= option. For non-GARCH models, if inequality signs are specified, they are treated as equality signs.

The following is an example of a RESTRICT statement:

```
model y = a b c d;
restrict a+b=0, 2*d-c=0;
```

When restricting a linear combination of parameters to be 0, you can omit the equal sign. For example, the following RESTRICT statement is equivalent to the preceding example:

```
restrict a+b, 2*d-c;
```

The following RESTRICT statement constrains the parameters estimates for three regressors (X1, X2, and X3) to be equal:

```
restrict x1 = x2, x2 = x3;
```

The preceding restriction can be abbreviated as follows:

```
restrict x1 = x2 = x3;
```

The following example shows how to specify AR, GARCH, and HETERO parameters in the RESTRICT statement:

```
model y = a b / nlag=2 garch=(p=2,q=3,mean=sqrt);
hetero c d;
restrict _A_1=0,_AH_2=0.2,_HET_2=1,_DELTA_=0.1;
```

You can specify only simple linear combinations of parameters in RESTRICT statement expressions. You cannot specify complex expressions that involve parentheses, division, functions, or complex products.

TEST Statement

The AUTOREG procedure supports a TEST statement for linear hypothesis tests. The syntax of the TEST statement is

TEST *equation , . . . , equation / option* **;**

The TEST statement tests hypotheses about the covariates in the model that are estimated by the preceding MODEL statement. The AR, GARCH, and HETERO parameters are also supported in the TEST statement when you specify the GARCH= option. Each equation specifies a linear hypothesis to be tested. If you specify more than one equation, separate them with commas.

Each test is written as a linear equation composed of constants and parameter names. Refer to parameters by the name of the corresponding regressor variable. Each name that is used in the equation must be a regressor in the preceding MODEL statement. Use the keyword INTERCEPT to refer to the intercept parameter in the model. For the names of these parameters, see the section "OUTEST= Data Set" on page 431.

You can specify the following options in the TEST statement:

TYPE=*value*
specifies the test statistics to use. The default is TYPE=F. The following values for TYPE= option are available:

F	produces an *F* test. This option is supported for all models specified in MODEL statement.
WALD	produces a Wald test. This option is supported for all models specified in MODEL statement.
LM	produces a Lagrange multiplier test. This option is supported only when the GARCH= option is specified (for example, when there is a statement like MODEL Y = C D I / GARCH=(Q=2)).
LR	produces a likelihood ratio test. This option is supported only when the GARCH= option is specified (for example, when there is a statement like MODEL Y = C D I / GARCH=(Q=2)).
ALL	produces all tests applicable for a particular model. For non-GARCH-type models, only *F* and Wald tests are output. For all other models, all four tests (LR, LM, *F*, and Wald) are computed.

The following example of a TEST statement tests the hypothesis that the coefficients of two regressors A and B are equal:

```
model y = a b c d;
test a = b;
```

To test separate null hypotheses, use separate TEST statements. To test a joint hypothesis, specify the component hypotheses on the same TEST statement, separated by commas.

For example, consider the following linear model:

$$y_t = \beta_0 + \beta_1 x1_t + \beta_2 x2_t + \epsilon_t$$

The following statements test the two hypotheses $H_0 : \beta_0 = 1$ and $H_0 : \beta_1 + \beta_2 = 0$:

```
model y = x1 x2;
test intercept = 1;
test x1 + x2 = 0;
```

The following statements test the joint hypothesis $H_0 : \beta_0 = 1$ and $\beta_1 + \beta_2 = 0$:

```
model y = x1 x2;
test intercept = 1, x1 + x2 = 0;
```

To illustrate the TYPE= option, consider the following examples.

```
model Y = C D I / garch=(q=2);
test C + D = 1;
```

The preceding statements produce only one default test, the *F* test.

```
model Y = C D I / garch=(q=2);
test C + D = 1 / type = LR;
```

The preceding statements produce one of four tests applicable for GARCH-type models, the likelihood ratio test.

```
model Y = C D I / nlag = 2;
test C + D = 1 / type = LM;
```

The preceding statements produce the warning and do not output any test because the Lagrange multiplier test is not applicable for non-GARCH models.

```
model Y = C D I / nlag=2;
test C + D = 1 / type = ALL;
```

The preceding statements produce all tests that are applicable for non-GARCH models (namely, the *F* and Wald tests). The TYPE= prefix is optional. Thus the test statement in the previous example could also have been written as:

```
test C + D = 1 / ALL;
```

The following example shows how to test AR, GARCH, and HETERO parameters:

```
model y = a b / nlag=2 garch=(p=2,q=3,mean=sqrt);
hetero c d;
test _A_1=0,_AH_2=0.2,_HET_2=1,_DELTA_=0.1;
```

Details: AUTOREG Procedure

Missing Values

PROC AUTOREG skips any missing values at the beginning of the data set. If the NOMISS option is specified, the first contiguous set of data with no missing values is used; otherwise, all data with nonmissing values for the independent and dependent variables are used. Note, however, that the observations containing missing values are still needed to maintain the correct spacing in the time series. PROC AUTOREG can generate predicted values when the dependent variable is missing.

Autoregressive Error Model

The regression model with autocorrelated disturbances is as follows:

$$y_t = \mathbf{x}_t'\beta + \nu_t$$

$$\nu_t = \epsilon_t - \varphi_1 \nu_{t-1} - \ldots - \varphi_m \nu_{t-m}$$

$$\epsilon_t \ \mathrm{N}(0, \sigma^2)$$

In these equations, y_t are the dependent values, $\mathbf{x}_t$ is a column vector of regressor variables, β is a column vector of structural parameters, and ϵ_t is normally and independently distributed with a mean of 0 and a variance of σ^2. Note that in this parameterization, the signs of the autoregressive parameters are reversed from the parameterization documented in most of the literature.

PROC AUTOREG offers four estimation methods for the autoregressive error model. The default method, Yule-Walker (YW) estimation, is the fastest computationally. The Yule-Walker method used by PROC AUTOREG is described in Gallant and Goebel (1976). Harvey (1981) calls this method the *two-step full transform method*. The other methods are iterated YW, unconditional least squares (ULS), and maximum likelihood (ML). The ULS method is also referred to as nonlinear least squares (NLS) or exact least squares (ELS).

You can use all of the methods with data containing missing values, but you should use ML estimation if the missing values are plentiful. See the section "Alternative Autocorrelation Correction Methods" on page 391 later in this chapter for further discussion of the advantages of different methods.

The Yule-Walker Method

Let $\boldsymbol{\varphi}$ represent the vector of autoregressive parameters,

$$\boldsymbol{\varphi} = (\varphi_1, \varphi_2, \ldots, \varphi_m)'$$

and let the variance matrix of the error vector $\boldsymbol{\nu} = (\nu_1, \ldots, \nu_N)'$ be $\boldsymbol{\Sigma}$,

$$E(\nu\nu') = \boldsymbol{\Sigma} = \sigma^2\mathbf{V}$$

If the vector of autoregressive parameters $\boldsymbol{\varphi}$ is known, the matrix $\mathbf{V}$ can be computed from the autoregressive parameters. $\boldsymbol{\Sigma}$ is then $\sigma^2\mathbf{V}$. Given $\boldsymbol{\Sigma}$, the efficient estimates of regression parameters β can be computed using generalized least squares (GLS). The GLS estimates then yield the unbiased estimate of the variance σ^2,

The Yule-Walker method alternates estimation of β using generalized least squares with estimation of $\boldsymbol{\varphi}$ using the Yule-Walker equations applied to the sample autocorrelation function. The YW method starts by forming the OLS estimate of β. Next, $\boldsymbol{\varphi}$ is estimated from the sample autocorrelation function of the OLS residuals by using the Yule-Walker equations. Then $\mathbf{V}$ is estimated from the estimate of $\boldsymbol{\varphi}$, and $\boldsymbol{\Sigma}$ is estimated from $\mathbf{V}$ and the OLS estimate of σ^2. The autocorrelation corrected estimates of the regression parameters β are then computed by GLS, using the estimated $\boldsymbol{\Sigma}$ matrix. These are the Yule-Walker estimates.

If the ITER option is specified, the Yule-Walker residuals are used to form a new sample autocorrelation function, the new autocorrelation function is used to form a new estimate of $\boldsymbol{\varphi}$ and $\mathbf{V}$, and the GLS estimates are recomputed using the new variance matrix. This alternation of estimates continues until either the maximum change in the $\widehat{\varphi}$ estimate between iterations is less than the value specified by the CONVERGE= option or the maximum number of allowed iterations is reached. This produces the iterated Yule-Walker estimates. Iteration of the estimates may not yield much improvement.

The Yule-Walker equations, solved to obtain $\widehat{\varphi}$ and a preliminary estimate of σ^2, are

$$\mathbf{R}\hat{\varphi} = -\mathbf{r}$$

Here $\mathbf{r} = (r_1, \ldots, r_m)'$, where r_i is the lag i sample autocorrelation. The matrix $\mathbf{R}$ is the Toeplitz matrix whose i,jth element is $r_{|i-j|}$. If you specify a subset model, then only the rows and columns of $\mathbf{R}$ and $\mathbf{r}$ corresponding to the subset of lags specified are used.

If the BACKSTEP option is specified, for purposes of significance testing, the matrix $[\mathbf{R}\ \mathbf{r}]$ is treated as a sum-of-squares-and-crossproducts matrix arising from a simple regression with $N - k$ observations, where k is the number of estimated parameters.

The Unconditional Least Squares and Maximum Likelihood Methods

Define the transformed error, $\mathbf{e}$, as

$$\mathbf{e} = \mathbf{L}^{-1}\mathbf{n}$$

where $\mathbf{n} = \mathbf{y} - \mathbf{X}\beta$.

The unconditional sum of squares for the model, S, is

$$S = \mathbf{n}'\mathbf{V}^{-1}\mathbf{n} = \mathbf{e}'\mathbf{e}$$

The ULS estimates are computed by minimizing S with respect to the parameters β and φ_i.

The full log likelihood function for the autoregressive error model is

$$l = -\frac{N}{2}\ln(2\pi) - \frac{N}{2}\ln(\sigma^2) - \frac{1}{2}\ln(|\mathbf{V}|) - \frac{S}{2\sigma^2}$$

where $|\mathbf{V}|$ denotes determinant of $\mathbf{V}$. For the ML method, the likelihood function is maximized by minimizing an equivalent sum-of-squares function.

Maximizing l with respect to σ^2 (and concentrating σ^2 out of the likelihood) and dropping the constant term $-\frac{N}{2}\ln(2\pi) + 1 - \ln(N)$ produces the concentrated log likelihood function

$$l_c = -\frac{N}{2}\ln(S|\mathbf{V}|^{1/N})$$

Rewriting the variable term within the logarithm gives

$$S_{ml} = |\mathbf{L}|^{1/N}\mathbf{e}'\mathbf{e}|\mathbf{L}|^{1/N}$$

PROC AUTOREG computes the ML estimates by minimizing the objective function $S_{ml} = |\mathbf{L}|^{1/N}\mathbf{e}'\mathbf{e}|\mathbf{L}|^{1/N}$.

The maximum likelihood estimates may not exist for some data sets (Anderson and Mentz 1980). This is the case for very regular data sets, such as an exact linear trend.

Computational Methods

Sample Autocorrelation Function

The sample autocorrelation function is computed from the structural residuals or noise $\mathbf{n_t} = y_t - \mathbf{x}_t'\mathbf{b}$, where $\mathbf{b}$ is the current estimate of β. The sample autocorrelation function is the sum of all available lagged products of $\mathbf{n}_t$ of order j divided by $\ell + j$, where ℓ is the number of such products.

If there are no missing values, then $\ell + j = N$, the number of observations. In this case, the Toeplitz matrix of autocorrelations, $\mathbf{R}$, is at least positive semidefinite. If there are missing values, these autocorrelation estimates of r can yield an $\mathbf{R}$ matrix that is not positive semidefinite. If such estimates occur, a warning message is printed, and the estimates are tapered by exponentially declining weights until $\mathbf{R}$ is positive definite.

Data Transformation and the Kalman Filter

The calculation of $\mathbf{V}$ from $\boldsymbol{\varphi}$ for the general AR(m) model is complicated, and the size of $\mathbf{V}$ depends on the number of observations. Instead of actually calculating $\mathbf{V}$ and performing GLS in the usual way, in practice a Kalman filter algorithm is used to transform the data and compute the GLS results through a recursive process.

In all of the estimation methods, the original data are transformed by the inverse of the Cholesky root of $\mathbf{V}$. Let $\mathbf{L}$ denote the Cholesky root of $\mathbf{V}$ — that is, $\mathbf{V} = \mathbf{L}\mathbf{L}'$ with $\mathbf{L}$ lower triangular. For an AR(m) model, $\mathbf{L}^{-1}$ is a band diagonal matrix with m anomalous rows at the beginning and the autoregressive parameters along the remaining rows. Thus, if there are no missing values, after the first $m-1$ observations the data are transformed as

$$z_t = x_t + \hat{\varphi}_1 x_{t-1} + \ldots + \hat{\varphi}_m x_{t-m}$$

The transformation is carried out using a Kalman filter, and the lower triangular matrix $\mathbf{L}$ is never directly computed. The Kalman filter algorithm, as it applies here, is described in Harvey and Phillips (1979) and Jones (1980). Although $\mathbf{L}$ is not computed explicitly, for ease of presentation the remaining discussion is in terms of $\mathbf{L}$. If there are missing values, then the submatrix of $\mathbf{L}$ consisting of the rows and columns with nonmissing values is used to generate the transformations.

Gauss-Newton Algorithms

The ULS and ML estimates employ a Gauss-Newton algorithm to minimize the sum of squares and maximize the log likelihood, respectively. The relevant optimization is performed simultaneously for both the regression and AR parameters. The OLS estimates of β and the Yule-Walker estimates of $\boldsymbol{\varphi}$ are used as starting values for these methods.

The Gauss-Newton algorithm requires the derivatives of $\mathbf{e}$ or $|\mathbf{L}|^{1/N}\mathbf{e}$ with respect to the parameters. The derivatives with respect to the parameter vector β are

$$\frac{\partial \mathbf{e}}{\partial \beta'} = -\mathbf{L}^{-1}\mathbf{X}$$

$$\frac{\partial |\mathbf{L}|^{1/N}\mathbf{e}}{\partial \boldsymbol{\beta}'} = -|\mathbf{L}|^{1/N}\mathbf{L}^{-1}\mathbf{X}$$

These derivatives are computed by the transformation described previously. The derivatives with respect to $\boldsymbol{\varphi}$ are computed by differentiating the Kalman filter recurrences and the equations for the initial conditions.

Variance Estimates and Standard Errors

For the Yule-Walker method, the estimate of the error variance, s^2, is the error sum of squares from the last application of GLS, divided by the error degrees of freedom (number of observations N minus the number of free parameters).

The variance-covariance matrix for the components of $\mathbf{b}$ is taken as $s^2(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1}$ for the Yule-Walker method. For the ULS and ML methods, the variance-covariance matrix of the parameter estimates is computed as $s^2(\mathbf{J}'\mathbf{J})^{-1}$. For the ULS method, $\mathbf{J}$ is the matrix of derivatives of $\mathbf{e}$ with respect to the parameters. For the ML method, $\mathbf{J}$ is the matrix of derivatives of $|\mathbf{L}|^{1/N}\mathbf{e}$ divided by $|\mathbf{L}|^{1/N}$. The estimate of the variance-covariance matrix of $\mathbf{b}$ assuming that $\boldsymbol{\varphi}$ is known is $s^2(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1}$. For OLS model, the estimate of the variance-covariance matrix is $s^2(\mathbf{X}'\mathbf{X})^{-1}$.

Park and Mitchell (1980) investigated the small sample performance of the standard error estimates obtained from some of these methods. In particular, simulating an AR(1) model for the noise term, they found that the standard errors calculated using GLS with an estimated autoregressive parameter underestimated the true standard errors. These estimates of standard errors are the ones calculated by PROC AUTOREG with the Yule-Walker method.

The estimates of the standard errors calculated with the ULS or ML method take into account the joint estimation of the AR and the regression parameters and may give more accurate standard-error values than the YW method. At the same values of the autoregressive parameters, the ULS and ML standard errors will always be larger than those computed from Yule-Walker. However, simulations of the models used by Park and Mitchell (1980) suggest that the ULS and ML standard error estimates can also be underestimates. Caution is advised, especially when the estimated autocorrelation is high and the sample size is small.

High autocorrelation in the residuals is a symptom of lack of fit. An autoregressive error model should not be used as a nostrum for models that simply do not fit. It is often the case that time series variables tend to move as a random walk. This means that an AR(1) process with a parameter near one absorbs a great deal of the variation. See Example 9.3, which fits a linear trend to a sine wave.

For ULS or ML estimation, the joint variance-covariance matrix of all the regression and autoregression parameters is computed. For the Yule-Walker method, the variance-covariance matrix is computed only for the regression parameters.

Lagged Dependent Variables

The Yule-Walker estimation method is not directly appropriate for estimating models that include lagged dependent variables among the regressors. Therefore, the maximum likelihood method is the default when the LAGDEP or LAGDEP= option is specified in the MODEL statement. However, when lagged dependent variables are used, the maximum likelihood estimator is not exact maximum likelihood but is conditional on the first few values of the dependent variable.

Alternative Autocorrelation Correction Methods

Autocorrelation correction in regression analysis has a long history, and various approaches have been suggested. Moreover, the same method may be referred to by different names.

Pioneering work in the field was done by Cochrane and Orcutt (1949). The *Cochrane-Orcutt method* refers to a more primitive version of the Yule-Walker method that drops the first observation. The Cochrane-Orcutt method is like the Yule-Walker method for first-order autoregression, except that the Yule-Walker method retains information from the first observation. The iterative Cochrane-Orcutt method is also in use.

The Yule-Walker method used by PROC AUTOREG is also known by other names. Harvey (1981) refers to the Yule-Walker method as the *two-step full transform method*. The Yule-Walker method can be considered as generalized least squares using the OLS residuals to estimate the covariances across observations, and Judge et al. (1985) use the term *estimated generalized least squares* (EGLS) for this method. For a first-order AR process, the Yule-Walker estimates are often termed *Prais-Winsten estimates* (Prais and Winsten 1954). There are variations to these methods that use different estimators of the autocorrelations or the autoregressive parameters.

The unconditional least squares (ULS) method, which minimizes the error sum of squares for all observations, is referred to as the nonlinear least squares (NLS) method by Spitzer (1979).

The *Hildreth-Lu* method (Hildreth and Lu 1960) uses nonlinear least squares to jointly estimate the parameters with an AR(1) model, but it omits the first transformed residual from the sum of squares. Thus, the Hildreth-Lu method is a more primitive version of the ULS method supported by PROC AUTOREG in the same way Cochrane-Orcutt is a more primitive version of Yule-Walker.

The maximum likelihood method is also widely cited in the literature. Although the maximum likelihood method is well defined, some early literature refers to estimators that are called maximum likelihood but are not full unconditional maximum likelihood estimates. The AUTOREG procedure produces full unconditional maximum likelihood estimates.

Harvey (1981) and Judge et al. (1985) summarize the literature on various estimators for the autoregressive error model. Although asymptotically efficient, the various methods have different small sample properties. Several Monte Carlo experiments have been conducted, although usually for the AR(1) model.

Harvey and McAvinchey (1978) found that for a one-variable model, when the independent variable is trending, methods similar to Cochrane-Orcutt are inefficient in estimating the structural parameter. This is not surprising since a pure trend model is well modeled by an autoregressive process with a parameter close to 1.

Harvey and McAvinchey (1978) also made the following conclusions:

- The Yule-Walker method appears to be about as efficient as the maximum likelihood method. Although Spitzer (1979) recommended ML and NLS, the Yule-Walker method (labeled Prais-Winsten) did as well or better in estimating the structural parameter in Spitzer's Monte Carlo study (table A2 in their article) when the autoregressive parameter was not too large. Maximum likelihood tends to do better when the autoregressive parameter is large.
- For small samples, it is important to use a full transformation (Yule-Walker) rather than the Cochrane-Orcutt method, which loses the first observation. This was also demonstrated by Maeshiro (1976), Chipman (1979), and Park and Mitchell (1980).
- For large samples (Harvey and McAvinchey used 100), losing the first few observations does not make much difference.

GARCH Models

Consider the series y_t, which follows the GARCH process. The conditional distribution of the series Y for time t is written

$$y_t | \Psi_{t-1} \sim \mathrm{N}(0, h_t)$$

where Ψ_{t-1} denotes all available information at time $t-1$. The conditional variance h_t is

$$h_t = \omega + \sum_{i=1}^{q} \alpha_i y_{t-i}^2 + \sum_{j=1}^{p} \gamma_j h_{t-j}$$

where

$$p \geq 0, q > 0$$

$$\omega > 0, \alpha_i \geq 0, \gamma_j \geq 0$$

The GARCH(p,q) model reduces to the ARCH(q) process when $p = 0$. At least one of the ARCH parameters must be nonzero ($q > 0$). The GARCH regression model can be written

$$y_t = \mathbf{x}_t'\beta + \epsilon_t$$

$$\epsilon_t = \sqrt{h_t}e_t$$

$$h_t = \omega + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2 + \sum_{j=1}^{p} \gamma_j h_{t-j}$$

where $e_t \sim \text{IN}(0,1)$.

In addition, you can consider the model with disturbances following an autoregressive process and with the GARCH errors. The AR(m)-GARCH(p,q) regression model is denoted

$$y_t = \mathbf{x}_t'\beta + \nu_t$$

$$\nu_t = \epsilon_t - \varphi_1 \nu_{t-1} - \ldots - \varphi_m \nu_{t-m}$$

$$\epsilon_t = \sqrt{h_t}e_t$$

$$h_t = \omega + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2 + \sum_{j=1}^{p} \gamma_j h_{t-j}$$

GARCH Estimation with Nelson-Cao Inequality Constraints

The GARCH(p,q) model is written in ARCH(∞) form as

$$\begin{aligned} h_t &= \left(1 - \sum_{j=1}^{p} \gamma_j B^j\right)^{-1} \left[\omega + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2\right] \\ &= \omega^* + \sum_{i=1}^{\infty} \phi_i \epsilon_{t-i}^2 \end{aligned}$$

where B is a backshift operator. Therefore, $h_t \geq 0$ if $\omega^* \geq 0$ and $\phi_i \geq 0, \ \forall i$. Assume that the roots of the following polynomial equation are inside the unit circle:

$$\sum_{j=0}^{p} -\gamma_j Z^{p-j}$$

where $\gamma_0 = -1$ and Z is a complex scalar. $-\sum_{j=0}^{p} \gamma_j Z^{p-j}$ and $\sum_{i=1}^{q} \alpha_i Z^{q-i}$ do not share common factors. Under these conditions, $|\omega^*| < \infty$, $|\phi_i| < \infty$, and these coefficients of the ARCH(∞) process are well defined.

Define $n = \max(p, q)$. The coefficient ϕ_i is written

$$\begin{aligned}
\phi_0 &= \alpha_1 \\
\phi_1 &= \gamma_1\phi_0 + \alpha_2 \\
&\cdots \\
\phi_{n-1} &= \gamma_1\phi_{n-2} + \gamma_2\phi_{n-3} + \cdots + \gamma_{n-1}\phi_0 + \alpha_n \\
\phi_k &= \gamma_1\phi_{k-1} + \gamma_2\phi_{k-2} + \cdots + \gamma_n\phi_{k-n} \text{ for } k \geq n
\end{aligned}$$

where $\alpha_i = 0$ for $i > q$ and $\gamma_j = 0$ for $j > p$.

Nelson and Cao (1992) proposed the finite inequality constraints for GARCH$(1, q)$ and GARCH$(2, q)$ cases. However, it is not straightforward to derive the finite inequality constraints for the general GARCH(p, q) model.

For the GARCH$(1, q)$ model, the nonlinear inequality constraints are

$$\begin{aligned}
\omega &\geq 0 \\
\gamma_1 &\geq 0 \\
\phi_k &\geq 0 \text{ for } k = 0, 1, \cdots, q-1
\end{aligned}$$

For the GARCH$(2, q)$ model, the nonlinear inequality constraints are

$$\begin{aligned}
\Delta_i &\in R \text{ for } i = 1, 2 \\
\omega^* &\geq 0 \\
\Delta_1 &> 0 \\
\sum_{j=0}^{q-1} \Delta_1^{-j}\alpha_{j+1} &> 0 \\
\phi_k &\geq 0 \text{ for } k = 0, 1, \cdots, q
\end{aligned}$$

where Δ_1 and Δ_2 are the roots of $(Z^2 - \gamma_1 Z - \gamma_2)$.

For the GARCH(p, q) model with $p > 2$, only $\max(q-1, p) + 1$ nonlinear inequality constraints ($\phi_k \geq 0$ for $k = 0$ to $\max(q-1, p)$) are imposed, together with the in-sample positivity constraints of the conditional variance h_t.

Using the HETERO Statement with GARCH Models

The HETERO statement can be combined with the GARCH= option in the MODEL statement to include input variables in the GARCH conditional variance model. For example, the GARCH$(1, 1)$ variance model with two dummy input variables D1 and D2 is

$$\begin{aligned}
\epsilon_t &= \sqrt{h_t}e_t \\
h_t &= \omega + \alpha_1\epsilon_{t-1}^2 + \gamma_1 h_{t-1} + \eta_1 D1_t + \eta_2 D2_t
\end{aligned}$$

The following statements estimate this GARCH model:

```
proc autoreg data=one;
   model y = x z / garch=(p=1,q=1);
   hetero d1 d2;
run;
```

The parameters for the variables D1 and D2 can be constrained using the COEF= option. For example, the constraints $\eta_1 = \eta_2 = 1$ are imposed by the following statements:

```
proc autoreg data=one;
   model y = x z / garch=(p=1,q=1);
   hetero d1 d2 / coef=unit;
run;
```

IGARCH and Stationary GARCH Model

The condition $\sum_{i=1}^{q} \alpha_i + \sum_{j=1}^{p} \gamma_j < 1$ implies that the GARCH process is weakly stationary since the mean, variance, and autocovariance are finite and constant over time. When the GARCH process is stationary, the unconditional variance of ϵ_t is computed as

$$\mathbf{V}(\epsilon_t) = \frac{\omega}{(1 - \sum_{i=1}^{q} \alpha_i - \sum_{j=1}^{p} \gamma_j)}$$

where $\epsilon_t = \sqrt{h_t} e_t$ and h_t is the GARCH(p, q) conditional variance.

Sometimes the multistep forecasts of the variance do not approach the unconditional variance when the model is integrated in variance; that is, $\sum_{i=1}^{q} \alpha_i + \sum_{j=1}^{p} \gamma_j = 1$.

The unconditional variance for the IGARCH model does not exist. However, it is interesting that the IGARCH model can be strongly stationary even though it is not weakly stationary. Refer to Nelson (1990) for details.

EGARCH Model

The EGARCH model was proposed by Nelson (1991). Nelson and Cao (1992) argue that the nonnegativity constraints in the linear GARCH model are too restrictive. The GARCH model imposes the nonnegative constraints on the parameters, α_i and γ_j, while there are no restrictions on these parameters in the EGARCH model. In the EGARCH model, the conditional variance, h_t, is an asymmetric function of lagged disturbances ϵ_{t-i}:

$$\ln(h_t) = \omega + \sum_{i=1}^{q} \alpha_i g(z_{t-i}) + \sum_{j=1}^{p} \gamma_j \ln(h_{t-j})$$

where

$$g(z_t) = \theta z_t + \gamma[|z_t| - E|z_t|]$$

$$z_t = \epsilon_t / \sqrt{h_t}$$

The coefficient of the second term in $g(z_t)$ is set to be 1 (γ=1) in our formulation. Note that $E|z_t| = (2/\pi)^{1/2}$ if $z_t \sim \mathrm{N}(0, 1)$. The properties of the EGARCH model are summarized as follows:

- The function $g(z_t)$ is linear in z_t with slope coefficient $\theta + 1$ if z_t is positive while $g(z_t)$ is linear in z_t with slope coefficient $\theta - 1$ if z_t is negative.

- Suppose that $\theta = 0$. Large innovations increase the conditional variance if $|z_t| - E|z_t| > 0$ and decrease the conditional variance if $|z_t| - E|z_t| < 0$.
- Suppose that $\theta < 1$. The innovation in variance, $g(z_t)$, is positive if the innovations z_t are less than $(2/\pi)^{1/2}/(\theta - 1)$. Therefore, the negative innovations in returns, ϵ_t, cause the innovation to the conditional variance to be positive if θ is much less than 1.

QGARCH, TGARCH, and PGARCH Models

As shown in many empirical studies, positive and negative innovations have different impacts on future volatility. There is a long list of variations of GARCH models that consider the asymmetricity. Three typical variations are the quadratic GARCH (QGARCH) model (Engle and Ng 1993), the threshold GARCH (TGARCH) model (Glosten, Jaganathan, and Runkle 1993; Zakoian 1994), and the power GARCH (PGARCH) model (Ding, Granger, and Engle 1993). For more details about the asymmetric GARCH models, see Engle and Ng (1993).

In the QGARCH model, the lagged errors' centers are shifted from zero to some constant values:

$$h_t = \omega + \sum_{i=1}^{q} \alpha_i (\epsilon_{t-i} - \psi_i)^2 + \sum_{j=1}^{p} \gamma_j h_{t-j}$$

In the TGARCH model, there is an extra slope coefficient for each lagged squared error,

$$h_t = \omega + \sum_{i=1}^{q} (\alpha_i + 1_{\epsilon_{t-i}<0}\psi_i)\epsilon_{t-i}^2 + \sum_{j=1}^{p} \gamma_j h_{t-j}$$

where the indicator function $1_{\epsilon_t<0}$ is one if $\epsilon_t < 0$; otherwise, zero.

The PGARCH model not only considers the asymmetric effect, but also provides another way to model the long memory property in the volatility,

$$h_t^{\lambda} = \omega + \sum_{i=1}^{q} \alpha_i (|\epsilon_{t-i}| - \psi_i \epsilon_{t-i})^{2\lambda} + \sum_{j=1}^{p} \gamma_j h_{t-j}^{\lambda}$$

where $\lambda > 0$ and $|\psi_i| \leq 1, i = 1, ..., q$.

Note that the implemented TGARCH model is also well known as GJR-GARCH (Glosten, Jaganathan, and Runkle 1993), which is similar to the threshold GARCH model proposed by Zakoian (1994) but not exactly same. In Zakoian's model, the conditional standard deviation is a linear function of the past values of the white noise. Zakoian's version can be regarded as a special case of PGARCH model when $\lambda = 1/2$.

GARCH-in-Mean

The GARCH-M model has the added regressor that is the conditional standard deviation:

$$y_t = \mathbf{x}_t'\beta + \delta\sqrt{h_t} + \epsilon_t$$

$$\epsilon_t = \sqrt{h_t}e_t$$

where h_t follows the ARCH or GARCH process.

Maximum Likelihood Estimation

The family of GARCH models are estimated using the maximum likelihood method. The log-likelihood function is computed from the product of all conditional densities of the prediction errors.

When e_t is assumed to have a standard normal distribution ($e_t \sim \mathrm{N}(0,1)$), the log-likelihood function is given by

$$l = \sum_{t=1}^{N} \frac{1}{2}\left[-\ln(2\pi) - \ln(h_t) - \frac{\epsilon_t^2}{h_t}\right]$$

where $\epsilon_t = y_t - \mathbf{x}_t'\beta$ and h_t is the conditional variance. When the GARCH(p,q)-M model is estimated, $\epsilon_t = y_t - \mathbf{x}_t'\beta - \delta\sqrt{h_t}$. When there are no regressors, the residuals ϵ_t are denoted as y_t or $y_t - \delta\sqrt{h_t}$.

If e_t has the standardized Student's t distribution, the log-likelihood function for the conditional t distribution is

$$\ell = \sum_{t=1}^{N}\left[\ln\left(\Gamma\left(\frac{\nu+1}{2}\right)\right) - \ln\left(\Gamma\left(\frac{\nu}{2}\right)\right) - \frac{1}{2}\ln((\nu-2)\pi h_t)\right.$$

$$\left. -\frac{1}{2}(\nu+1)\ln\left(1 + \frac{\epsilon_t^2}{h_t(\nu-2)}\right)\right]$$

where $\Gamma(\cdot)$ is the gamma function and ν is the degree of freedom ($\nu > 2$). Under the conditional t distribution, the additional parameter $1/\nu$ is estimated. The log-likelihood function for the conditional t distribution converges to the log-likelihood function of the conditional normal GARCH model as $1/\nu \to 0$.

The likelihood function is maximized via either the dual quasi-Newton or the trust region algorithm. The default is the dual quasi-Newton algorithm. The starting values for the regression parameters β are obtained from the OLS estimates. When there are autoregressive parameters in the model, the initial values are obtained from the Yule-Walker estimates. The starting value 1.0^{-6} is used for the GARCH process parameters.

The variance-covariance matrix is computed using the Hessian matrix. The dual quasi-Newton method approximates the Hessian matrix while the quasi-Newton method gets an approximation of the inverse of Hessian. The trust region method uses the Hessian matrix obtained using numerical differentiation. When there are active constraints, that is, $\mathbf{q}(\theta) = \mathbf{0}$, the variance-covariance matrix is given by

$$\mathbf{V}(\hat{\theta}) = \mathbf{H}^{-1}[\mathbf{I} - \mathbf{Q}'(\mathbf{Q}\mathbf{H}^{-1}\mathbf{Q}')^{-1}\mathbf{Q}\mathbf{H}^{-1}]$$

where $\mathbf{H} = -\partial^2 l/\partial\theta\partial\theta'$ and $\mathbf{Q} = \partial\mathbf{q}(\theta)/\partial\theta'$. Therefore, the variance-covariance matrix without active constraints reduces to $\mathbf{V}(\hat{\theta}) = \mathbf{H}^{-1}$.

Heteroscedasticity- and Autocorrelation-Consistent Covariance Matrix Estimator

The heteroscedasticity-consistent covariance matrix estimator (HCCME), also known as the sandwich (or robust or empirical) covariance matrix estimator, has been popular in recent years because it gives the consistent estimation of the covariance matrix of the parameter estimates even when the heteroscedasticity structure might be unknown or misspecified. White (1980) proposes the concept of HCCME, known as HC0.

However, the small-sample performance of HC0 is not good in some cases. Davidson and MacKinnon (1993) introduce more improvements to HC0, namely HC1, HC2 and HC3, with the degrees-of-freedom or leverage adjustment. Cribari-Neto (2004) proposes HC4 for cases that have points of high leverage.

HCCME can be expressed in the following general "sandwich" form:

$$\Sigma = B^{-1}MB^{-1}$$

where B, which stands for "bread," is the Hessian matrix and M, which stands for "meat," is the outer product of gradient (OPG) with or without adjustment. For HC0, M is the OPG without adjustment; that is,

$$M_{\mathrm{HC0}} = \sum_{t=1}^{T} g_t g_t'$$

where T is the sample size and g_t is the gradient vector of tth observation. For HC1, M is the OPG with the degrees-of-freedom correction; that is,

$$M_{\mathrm{HC1}} = \frac{T}{T-k}\sum_{t=1}^{T} g_t g_t'$$

where k is the number of parameters. For HC2, HC3, and HC4, the adjustment is related to leverage, namely,

$$M_{\mathrm{HC2}} = \sum_{t=1}^{T} \frac{g_t g_t'}{1-h_{tt}} \qquad M_{\mathrm{HC3}} = \sum_{t=1}^{T} \frac{g_t g_t'}{(1-h_{tt})^2} \qquad M_{\mathrm{HC4}} = \sum_{t=1}^{T} \frac{g_t g_t'}{(1-h_{tt})^{\min(4,Th_{tt}/k)}}$$

The leverage h_{tt} is defined as $h_{tt} \equiv j_t'(\sum_{t=1}^{T} j_t j_t')^{-1} j_t$, where j_t is defined as follows:

- For an OLS model, j_t is the tth observed regressors in column vector form.
- For an AR error model, j_t is the derivative vector of the tth residual with respect to the parameters.
- For a GARCH or heteroscedasticity model, j_t is the gradient of the tth observation (that is, g_t).

The heteroscedasticity- and autocorrelation-consistent (HAC) covariance matrix estimator can also be expressed in "sandwich" form:

$$\Sigma = B^{-1}MB^{-1}$$

where B is still the Hessian matrix, but M is the kernel estimator in the following form:

$$M_{\mathrm{HAC}} = a\left(\sum_{t=1}^{T} g_t g_t' + \sum_{j=1}^{T-1} k\left(\frac{j}{b}\right)\sum_{t=1}^{T-j}\left(g_t g_{t+j}' + g_{t+j} g_t'\right)\right)$$

where T is the sample size, g_t is the gradient vector of tth observation, $k(.)$ is the real-valued kernel function, b is the bandwidth parameter, and a is the adjustment factor of small-sample degrees of freedom (that is, $a = 1$ if ADJUSTDF option is not specified and otherwise $a = T/(T-k)$, where k is the number of parameters). The types of kernel functions are listed in Table 9.2.

Table 9.2 Kernel Functions

Kernel Name	Equation
Bartlett	$k(x) = \begin{cases} 1-\lvert x\rvert & \lvert x\rvert \le 1 \\ 0 & \text{otherwise} \end{cases}$
Parzen	$k(x) = \begin{cases} 1-6x^2+6\lvert x\rvert^3 & 0 \le \lvert x\rvert \le 1/2 \\ 2(1-\lvert x\rvert)^3 & 1/2 \le \lvert x\rvert \le 1 \\ 0 & \text{otherwise} \end{cases}$
Quadratic spectral	$k(x) = \frac{25}{12\pi^2 x^2}\left(\frac{\sin(6\pi x/5)}{6\pi x/5} - \cos(6\pi x/5)\right)$
Truncated	$k(x) = \begin{cases} 1 & \lvert x\rvert \le 1 \\ 0 & \text{otherwise} \end{cases}$
Tukey-Hanning	$k(x) = \begin{cases} (1+\cos(\pi x))/2 & \lvert x\rvert \le 1 \\ 0 & \text{otherwise} \end{cases}$

When you specify BANDWIDTH=ANDREWS91, according to Andrews (1991) the bandwidth parameter is estimated as shown in Table 9.3.

Table 9.3 Bandwidth Parameter Estimation

Kernel Name	Bandwidth Parameter
Bartlett	$b = 1.1447(\alpha(1)T)^{1/3}$
Parzen	$b = 2.6614(\alpha(2)T)^{1/5}$
Quadratic spectral	$b = 1.3221(\alpha(2)T)^{1/5}$
Truncated	$b = 0.6611(\alpha(2)T)^{1/5}$
Tukey-Hanning	$b = 1.7462(\alpha(2)T)^{1/5}$

Let $\{g_{at}\}$ denote each series in $\{g_t\}$, and let (ρ_a, σ_a^2) denote the corresponding estimates of the autoregressive and innovation variance parameters of the AR(1) model on $\{g_{at}\}$, $a = 1, ..., k$, where the AR(1) model is parameterized as $g_{at} = \rho g_{at-1} + \epsilon_{at}$ with $Var(\epsilon_{at}) = \sigma_a^2$. The factors $\alpha(1)$ and $\alpha(2)$ are estimated with the following formulas:

$$\alpha(1) = \frac{\sum_{a=1}^{k} \frac{4\rho_a^2\sigma_a^4}{(1-\rho_a)^6(1+\rho_a)^2}}{\sum_{a=1}^{k} \frac{\sigma_a^4}{(1-\rho_a)^4}} \qquad \alpha(2) = \frac{\sum_{a=1}^{k} \frac{4\rho_a^2\sigma_a^4}{(1-\rho_a)^8}}{\sum_{a=1}^{k} \frac{\sigma_a^4}{(1-\rho_a)^4}}$$

When you specify BANDWIDTH=NEWEYWEST94, according to Newey and West (1994) the bandwidth parameter is estimated as shown in Table 9.4.

Table 9.4 Bandwidth Parameter Estimation

Kernel Name	Bandwidth Parameter
Bartlett	$b = 1.1447(\{s_1/s_0\}^2 T)^{1/3}$
Parzen	$b = 2.6614(\{s_1/s_0\}^2 T)^{1/5}$
Quadratic spectral	$b = 1.3221(\{s_1/s_0\}^2 T)^{1/5}$
Truncated	$b = 0.6611(\{s_1/s_0\}^2 T)^{1/5}$
Tukey-Hanning	$b = 1.7462(\{s_1/s_0\}^2 T)^{1/5}$

The factors s_1 and s_0 are estimated with the following formulas:

$$s_1 = 2\sum_{j=1}^{n} j\sigma_j \qquad\qquad s_0 = \sigma_0 + 2\sum_{j=1}^{n} \sigma_j$$

where n is the lag selection parameter and is determined by kernels, as listed in Table 9.5.

Table 9.5 Lag Selection Parameter Estimation

Kernel Name	Lag Selection Parameter
Bartlett	$n = c(T/100)^{2/9}$
Parzen	$n = c(T/100)^{4/25}$
Quadratic spectral	$n = c(T/100)^{2/25}$
Truncated	$n = c(T/100)^{1/5}$
Tukey-Hanning	$n = c(T/100)^{1/5}$

The factor c in Table 9.5 is specified by the C= option; by default it is 12.

The factor σ_j is estimated with the equation

$$\sigma_j = T^{-1}\sum_{t=j+1}^{T}\left(\sum_{a=i}^{k} g_{at}\sum_{a=i}^{k} g_{at-j}\right), j = 0, ..., n$$

where i is 1 if the NOINT option in the MODEL statement is specified (otherwise, it is 2), and g_{at} is the same as in the Andrews method.

If you specify BANDWIDTH=SAMPLESIZE, the bandwidth parameter is estimated with the equation

$$b = \begin{cases} \lfloor \gamma T^r + c \rfloor & \text{if BANDWIDTH=SAMPLESIZE(INT) option is specified} \\ \gamma T^r + c & \text{otherwise} \end{cases}$$

where T is the sample size; $\lfloor x \rfloor$ is the largest integer less than or equal to x; and γ, r, and c are values specified by the BANDWIDTH=SAMPLESIZE(GAMMA=, RATE=, CONSTANT=) options, respectively.

If you specify the PREWHITENING option, g_t is prewhitened by the VAR(1) model,

$$g_t = Ag_{t-1} + w_t$$

Then M is calculated by

$$M_{\mathrm{HAC}} = a(I-A)^{-1}\left(\sum_{t=1}^{T} w_t w_t' + \sum_{j=1}^{T-1} k\left(\frac{j}{b}\right)\sum_{t=1}^{T-j}\left(w_t w_{t+j}' + w_{t+j} w_t'\right)\right)\left((I-A)^{-1}\right)'$$

The bandwidth calculation is also based on the prewhitened series w_t.

Goodness-of-Fit Measures and Information Criteria

This section discusses various goodness-of-fit statistics produced by the AUTOREG procedure.

Total R-Square Statistic

The total R-Square statistic (Total Rsq) is computed as

$$\mathrm{R}^2_{\mathrm{tot}} = 1 - \frac{\mathrm{SSE}}{\mathrm{SST}}$$

where SST is the sum of squares for the original response variable corrected for the mean and SSE is the final error sum of squares. The Total Rsq is a measure of how well the next value can be predicted using the structural part of the model and the past values of the residuals. If the NOINT option is specified, SST is the uncorrected sum of squares.

Transformed Regression R-Square Statistic

The transformed regression R-Square statistic is computed as

$$\mathrm{R}^2_{\mathrm{tr}} = 1 - \frac{\mathrm{TSSE}}{\mathrm{TSST}}$$

where TSST is the total sum of squares of the transformed response variable corrected for the transformed intercept, and TSSE is the error sum of squares for this transformed regression problem. If the NOINT option is requested, no correction for the transformed intercept is made. The transformed regression R-Square statistic is a measure of the fit of the structural part of the model after transforming for the autocorrelation and is the R-Square for the transformed regression.

Mean Absolute Error and Mean Absolute Percentage Error

The mean absolute error (MAE) is computed as

$$\mathrm{MAE} = \frac{1}{T}\sum_{t=1}^{T}|e_t|$$

where e_t are the estimated model residuals and T is the number of observations.

The mean absolute percentage error (MAPE) is computed as

$$\text{MAPE} = \frac{1}{T'} \sum_{t=1}^{T} \delta_{y_t \neq 0} \frac{|e_t|}{|y_t|}$$

where e_t are the estimated model residuals, y_t are the original response variable observations, $\delta_{y_t \neq 0} = 1$ if $y_t \neq 0$, $\delta_{y_t \neq 0} |e_t / y_t| = 0$ if $y_t = 0$, and T' is the number of nonzero original response variable observations.

Calculation of Recursive Residuals and CUSUM Statistics

The recursive residuals w_t are computed as

$$w_t = \frac{e_t}{\sqrt{v_t}}$$

$$e_t = y_t - x_t' \beta^{(t)}$$

$$\beta^{(t)} = \left[\sum_{i=1}^{t-1} \mathbf{x}_i \mathbf{x}_i' \right]^{-1} \left(\sum_{i=1}^{t-1} \mathbf{x}_i y_i \right)$$

$$v_t = 1 + \mathbf{x}'_t \left[\sum_{i=1}^{t-1} \mathbf{x}_i \mathbf{x}_i' \right]^{-1} \mathbf{x}_t$$

Note that the first $\beta^{(t)}$ can be computed for $t = p + 1$, where p is the number of regression coefficients. As a result, first p recursive residuals are not defined. Note also that the forecast error variance of e_t is the scalar multiple of v_t such that $V(e_t) = \sigma^2 v_t$.

The CUSUM and CUSUMSQ statistics are computed using the preceding recursive residuals.

$$\text{CUSUM}_t = \sum_{i=k+1}^{t} \frac{w_i}{\sigma_w}$$

$$\text{CUSUMSQ}_t = \frac{\sum_{i=k+1}^{t} w_i^2}{\sum_{i=k+1}^{T} w_i^2}$$

where w_i are the recursive residuals,

$$\sigma_w = \sqrt{\frac{\sum_{i=k+1}^{T} (w_i - \hat{w})^2}{(T - k - 1)}}$$

$$\hat{w} = \frac{1}{T - k} \sum_{i=k+1}^{T} w_i$$

and k is the number of regressors.

The CUSUM statistics can be used to test for misspecification of the model. The upper and lower critical values for CUSUM_t are

$$\pm a\left[\sqrt{T-k}+2\frac{(t-k)}{(T-k)^{\frac{1}{2}}}\right]$$

where $a = 1.143$ for a significance level 0.01, 0.948 for 0.05, and 0.850 for 0.10. These critical values are output by the CUSUMLB= and CUSUMUB= options for the significance level specified by the ALPHACSM= option.

The upper and lower critical values of CUSUMSQ_t are given by

$$\pm a+\frac{(t-k)}{T-k}$$

where the value of a is obtained from the table by Durbin (1969) if the $\frac{1}{2}(T-k)-1 \leq 60$. Edgerton and Wells (1994) provided the method of obtaining the value of a for large samples.

These critical values are output by the CUSUMSQLB= and CUSUMSQUB= options for the significance level specified by the ALPHACSM= option.

Information Criteria AIC, AICC, SBC, and HQC

Akaike's information criterion (AIC), the corrected Akaike's information criterion (AICC), Schwarz's Bayesian information criterion (SBC), and the Hannan-Quinn information criterion (HQC), are computed as follows:

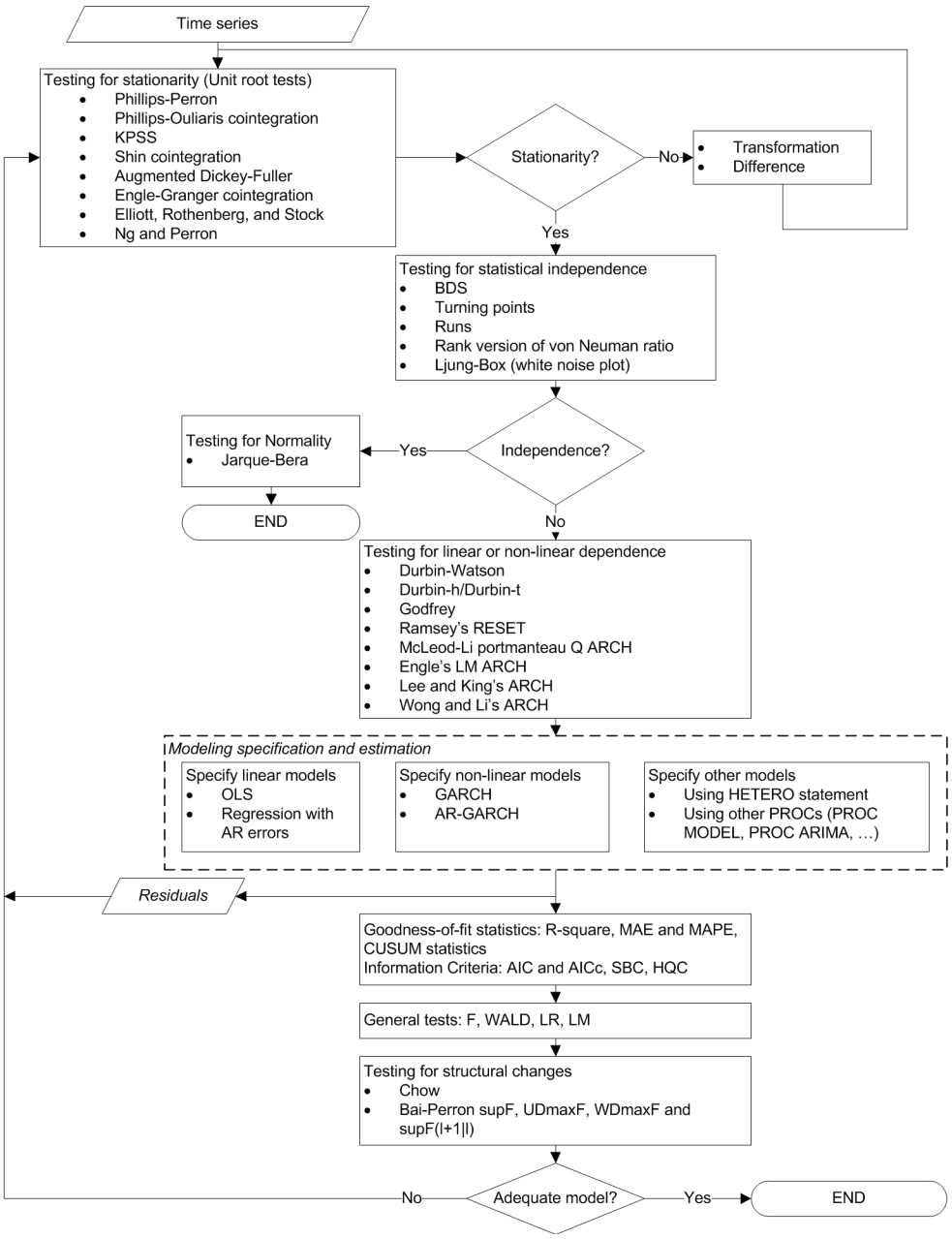
$$\begin{aligned}
\text{AIC} &= -2\ln(L)+2k \\
\text{AICC} &= \text{AIC}+2\frac{k(k+1)}{N-k-1} \\
\text{SBC} &= -2\ln(L)+\ln(N)k \\
\text{HQC} &= -2\ln(L)+2\ln(\ln(N))k
\end{aligned}$$

In these formulas, L is the value of the likelihood function evaluated at the parameter estimates, N is the number of observations, and k is the number of estimated parameters. Refer to Judge et al. (1985), Hurvich and Tsai (1989), Schwarz (1978) and Hannan and Quinn (1979) for additional details.

Testing

The modeling process consists of four stages: identification, specification, estimation, and diagnostic checking (Cromwell, Labys, and Terraza 1994). The AUTOREG procedure supports tens of statistical tests for identification and diagnostic checking. Figure 9.15 illustrates how to incorporate these statistical tests into the modeling process.

Figure 9.15 Statistical Tests in the AUTOREG Procedure



Testing for Stationarity

Most of the theories of time series require stationarity; therefore, it is critical to determine whether a time series is stationary. Two nonstationary time series are fractionally integrated time series and autoregressive series with random coefficients. However, more often some time series are nonstationary due to an upward trend over time. The trend can be captured by either of the following two models.

- The *difference stationary* process

$$(1 - L)y_t = \delta + \psi(L)\epsilon_t$$

where L is the lag operator, $\psi(1) \neq 0$, and ϵ_t is a white noise sequence with mean zero and variance σ^2. Hamilton (1994) also refers to this model the *unit root* process.

- The *trend stationary* process

$$y_t = \alpha + \delta t + \psi(L)\epsilon_t$$

When a process has a unit root, it is said to be integrated of order one or I(1). An I(1) process is stationary after differencing once. The trend stationary process and difference stationary process require different treatment to transform the process into stationary one for analysis. Therefore, it is important to distinguish the two processes. Bhargava (1986) nested the two processes into the following general model

$$y_t = \gamma_0 + \gamma_1 t + \alpha(y_{t-1} - \gamma_0 - \gamma_1(t-1)) + \psi(L)\epsilon_t$$

However, a difficulty is that the right-hand side is nonlinear in the parameters. Therefore, it is convenient to use a different parametrization

$$y_t = \beta_0 + \beta_1 t + \alpha y_{t-1} + \psi(L)\epsilon_t$$

The test of null hypothesis that $\alpha = 1$ against the one-sided alternative of $\alpha < 1$ is called a *unit root test*.

Dickey-Fuller unit root tests are based on regression models similar to the previous model

$$y_t = \beta_0 + \beta_1 t + \alpha y_{t-1} + \epsilon_t$$

where ϵ_t is assumed to be white noise. The t statistic of the coefficient α does not follow the normal distribution asymptotically. Instead, its distribution can be derived using the functional central limit theorem. Three types of regression models including the preceding one are considered by the Dickey-Fuller test. The deterministic terms that are included in the other two types of regressions are either null or constant only.

An assumption in the Dickey-Fuller unit root test is that it requires the errors in the autoregressive model to be white noise, which is often not true. There are two popular ways to account for general serial correlation between the errors. One is the augmented Dickey-Fuller (ADF) test, which uses the lagged difference in the regression model. This was originally proposed by Dickey and Fuller (1979) and later studied by Said and Dickey (1984) and Phillips and Perron (1988). Another method is proposed by Phillips and Perron (1988); it is called Phillips-Perron (PP) test. The tests adopt the original Dickey-Fuller regression with intercept, but modify the test statistics to take account of the serial correlation and heteroscedasticity. It is called nonparametric because no specific form of the serial correlation of the errors is assumed.

A problem of the augmented Dickey-Fuller and Phillips-Perron unit root tests is that they are subject to size distortion and low power. It is reported in Schwert (1989) that the size distortion is significant when the series contains a large moving average (MA) parameter. DeJong et al. (1992) find that the ADF has power around one third and PP test has power less than 0.1 against the trend stationary alternative, in some common settings. Among some more recent unit root tests that improve upon the size distortion and the low power are the tests described by Elliott, Rothenberg, and Stock (1996) and Ng and Perron (2001). These tests involve a step of detrending before constructing the test statistics and are demonstrated to perform better than the traditional ADF and PP tests.

Most testing procedures specify the unit root processes as the null hypothesis. Tests of the null hypothesis of stationarity have also been studied, among which Kwiatkowski et al. (1992) is very popular.

Economic theories often dictate that a group of economic time series are linked together by some long-run equilibrium relationship. Statistically, this phenomenon can be modeled by *cointegration*. When several

nonstationary processes $\mathbf{z}_t = (z_{1t}, \cdots, z_{kt})'$ are cointegrated, there exists a $(k \times 1)$ cointegrating vector $\mathbf{c}$ such that $\mathbf{c}'\mathbf{z}_t$ is stationary and $\mathbf{c}$ is a nonzero vector. One way to test the relationship of cointegration is the *residual based cointegration test*, which assumes the regression model

$$y_t = \beta_1 + \mathbf{x}_t'\beta + u_t$$

where $y_t = z_{1t}$, $\mathbf{x}_t = (z_{2t}, \cdots, z_{kt})'$, and $\beta = (\beta_2, \cdots, \beta_k)'$. The OLS residuals from the regression model are used to test for the null hypothesis of no cointegration. Engle and Granger (1987) suggest using ADF on the residuals while Phillips and Ouliaris (1990) study the tests using PP and other related test statistics.

Augmented Dickey-Fuller Unit Root and Engle-Granger Cointegration Testing

Common unit root tests have the null hypothesis that there is an autoregressive unit root $H_0 : \alpha = 1$, and the alternative is $H_a : |\alpha| < 1$, where α is the autoregressive coefficient of the time series

$$y_t = \alpha y_{t-1} + \epsilon_t$$

This is referred to as the zero mean model. The standard Dickey-Fuller (DF) test assumes that errors ϵ_t are white noise. There are two other types of regression models that include a constant or a time trend as follows:

$$y_t = \mu + \alpha y_{t-1} + \epsilon_t$$
$$y_t = \mu + \beta t + \alpha y_{t-1} + \epsilon_t$$

These two models are referred to as the constant mean model and the trend model, respectively. The constant mean model includes a constant mean μ of the time series. However, the interpretation of μ depends on the stationarity in the following sense: the mean in the stationary case when $\alpha < 1$ is the trend in the integrated case when $\alpha = 1$. Therefore, the null hypothesis should be the joint hypothesis that $\alpha = 1$ and $\mu = 0$. However for the unit root tests, the test statistics are concerned with the null hypothesis of $\alpha = 1$. The joint null hypothesis is not commonly used. This issue is address in Bhargava (1986) with a different nesting model.

There are two types of test statistics. The conventional t ratio is

$$DF_\tau = \frac{\hat{\alpha} - 1}{sd(\hat{\alpha})}$$

and the second test statistic, called ρ-test, is

$$T(\hat{\alpha} - 1)$$

For the zero mean model, the asymptotic distributions of the Dickey-Fuller test statistics are

$$T(\hat{\alpha} - 1) \Rightarrow \left(\int_0^1 W(r) dW(r) \right) \left(\int_0^1 W(r)^2 dr \right)^{-1}$$
$$DF_\tau \Rightarrow \left(\int_0^1 W(r) dW(r) \right) \left(\int_0^1 W(r)^2 dr \right)^{-1/2}$$

For the constant mean model, the asymptotic distributions are

$$T(\hat{\alpha} - 1) \Rightarrow \left([W(1)^2 - 1]/2 - W(1) \int_0^1 W(r) dr \right) \left(\int_0^1 W(r)^2 dr - \left(\int_0^1 W(r) dr \right)^2 \right)^{-1}$$
$$DF_\tau \Rightarrow \left([W(1)^2 - 1]/2 - W(1) \int_0^1 W(r) dr \right) \left(\int_0^1 W(r)^2 dr - \left(\int_0^1 W(r) dr \right)^2 \right)^{-1/2}$$

For the trend model, the asymptotic distributions are

$$T(\hat{\alpha}-1) \Rightarrow \left[W(r)dW + 12\left(\int_0^1 rW(r)dr - \frac{1}{2}\int_0^1 W(r)dr\right)\left(\int_0^1 W(r)dr - \frac{1}{2}W(1)\right) - W(1)\int_0^1 W(r)dr \right] D^{-1}$$

$$DF_\tau \Rightarrow \left[W(r)dW + 12\left(\int_0^1 rW(r)dr - \frac{1}{2}\int_0^1 W(r)dr\right)\left(\int_0^1 W(r)dr - \frac{1}{2}W(1)\right) - W(1)\int_0^1 W(r)dr \right] D^{1/2}$$

where

$$D = \int_0^1 W(r)^2 dr - 12\left(\int_0^1 r(W(r)dr\right)^2 + 12\int_0^1 W(r)dr\int_0^1 rW(r)dr - 4\left(\int_0^1 W(r)dr\right)^2$$

One problem of the Dickey-Fuller and similar tests that employ three types of regressions is the difficulty in the specification of the deterministic trends. Campbell and Perron (1991) claimed that "the proper handling of deterministic trends is a vital prerequisite for dealing with unit roots". However the "proper handling" is not obvious since the distribution theory of the relevant statistics about the deterministic trends is not available. Hayashi (2000) suggests to using the constant mean model when you think there is no trend, and using the trend model when you think otherwise. However no formal procedure is provided.

The null hypothesis of the Dickey-Fuller test is a random walk, possibly with drift. The differenced process is not serially correlated under the null of I(1). There is a great need for the generalization of this specification. The augmented Dickey-Fuller (ADF) test, originally proposed in Dickey and Fuller (1979), adjusts for the serial correlation in the time series by adding lagged first differences to the autoregressive model,

$$\Delta y_t = \mu + \delta t + \alpha y_{t-1} + \sum_{j=1}^{p} \alpha_j \Delta y_{t-j} + \epsilon_t$$

where the deterministic terms δt and μ can be absent for the models without drift or linear trend. As previously, there are two types of test statistics. One is the OLS t value

$$\frac{\hat{\alpha}-1}{sd(\hat{\alpha})}$$

and the other is given by

$$\frac{T(\hat{\alpha}-1)}{1-\hat{\alpha}_1-\ldots-\hat{\alpha}_p}$$

The asymptotic distributions of the test statistics are the same as those of the standard Dickey-Fuller test statistics.

Nonstationary multivariate time series can be tested for cointegration, which means that a linear combination of these time series is stationary. Formally, denote the series by $\mathbf{z}_t = (z_{1t}, \cdots, z_{kt})'$. The null hypothesis of cointegration is that there exists a vector $\mathbf{c}$ such that $\mathbf{c}'\mathbf{z}_t$ is stationary. Residual-based cointegration tests were studied in Engle and Granger (1987) and Phillips and Ouliaris (1990). The latter are described in the next subsection. The first step regression is

$$y_t = \mathbf{x}_t'\beta + u_t$$

where $y_t = z_{1t}, \mathbf{x}_t = (z_{2t}, \cdots, z_{kt})'$, and $\beta = (\beta_2, \cdots, \beta_k)'$. This regression can also include an intercept or an intercept with a linear trend. The residuals are used to test for the existence of an autoregressive unit root. Engle and Granger (1987) proposed augmented Dickey-Fuller type regression without an intercept on the residuals to test the unit root. When the first step OLS does not include an intercept, the asymptotic distribution of the ADF test statistic DF_τ is given by

$$
\begin{aligned}
DF_\tau &\Longrightarrow \int_0^1 \frac{Q(r)}{(\int_0^1 Q^2)^{1/2}} dS \\
Q(r) &= W_1(r) - \int_0^1 W_1 W_2' \left(\int_0^1 W_2 W_2' \right)^{-1} W_2(r) \\
S(r) &= \frac{Q(r)}{(\kappa'\kappa)^{1/2}} \\
\kappa' &= \left(1, -\int_0^1 W_1 W_2' \left(\int_0^1 W_2 W_2' \right)^{-1} \right)
\end{aligned}
$$

where $W(r)$ is a k vector standard Brownian motion and

$$W(r) = \Big(W_1(r), W_2(r) \Big)$$

is a partition such that $W_1(r)$ is a scalar and $W_2(r)$ is $k-1$ dimensional. The asymptotic distributions of the test statistics in the other two cases have the same form as the preceding formula. If the first step regression includes an intercept, then $W(r)$ is replaced by the demeaned Brownian motion $\overline{W}(r) = W(r) - \int_0^1 W(r)dr$. If the first step regression includes a time trend, then $W(r)$ is replaced by the detrended Brownian motion. The critical values of the asymptotic distributions are tabulated in Phillips and Ouliaris (1990) and MacKinnon (1991).

The residual based cointegration tests have a major shortcoming. Different choices of the dependent variable in the first step OLS might produce contradictory results. This can be explained theoretically. If the dependent variable is in the cointegration relationship, then the test is consistent against the alternative that there is cointegration. On the other hand, if the dependent variable is not in the cointegration system, the OLS residual $y_t - \mathbf{x}_t'\beta$ do not converge to a stationary process. Changing the dependent variable is more likely to produce conflicting results in finite samples.

Phillips-Perron Unit Root and Cointegration Testing

Besides the ADF test, there is another popular unit root test that is valid under general serial correlation and heteroscedasticity, developed by Phillips (1987) and Phillips and Perron (1988). The tests are constructed using the AR(1) type regressions, unlike ADF tests, with corrected estimation of the long run variance of Δy_t. In the case without intercept, consider the driftless random walk process

$$y_t = y_{t-1} + u_t$$

where the disturbances might be serially correlated with possible heteroscedasticity. Phillips and Perron (1988) proposed the unit root test of the OLS regression model,

$$y_t = \rho y_{t-1} + u_t$$

Denote the OLS residual by $\hat{u}_t$. The asymptotic variance of $\frac{1}{T}\sum_{t=1}^{T}\hat{u}_t^2$ can be estimated by using the truncation lag l.

$$\hat{\lambda} = \sum_{j=0}^{l} \kappa_j [1 - j/(l+1)]\hat{\gamma}_j$$

where $\kappa_0 = 1, \kappa_j = 2$ for $j > 0$, and $\hat{\gamma}_j = \frac{1}{T}\sum_{t=j+1}^{T}\hat{u}_t\hat{u}_{t-j}$. This is a consistent estimator suggested by Newey and West (1987).

The variance of u_t can be estimated by $s^2 = \frac{1}{T-k}\sum_{t=1}^{T}\hat{u}_t^2$. Let $\hat{\sigma}^2$ be the variance estimate of the OLS estimator $\hat{\rho}$. Then the Phillips-Perron $\hat{Z}_\rho$ test (zero mean case) is written

$$\hat{Z}_\rho = T(\hat{\rho} - 1) - \frac{1}{2}T^2\hat{\sigma}^2(\hat{\lambda} - \hat{\gamma}_0)/s^2$$

The $\hat{Z}_\rho$ statistic is just the ordinary Dickey-Fuller $\hat{Z}_\alpha$ statistic with a correction term that accounts for the serial correlation. The correction term goes to zero asymptotically if there is no serial correlation.

Note that $\mathrm{P}(\hat{\rho} < 1) \approx 0.68$ as $T \to \infty$, which shows that the limiting distribution is skewed to the left.

Let τ_ρ be the τ statistic for $\hat{\rho}$. The Phillips-Perron $\hat{Z}_t$ (defined here as $\hat{Z}_\tau$) test is written

$$\hat{Z}_\tau = (\hat{\gamma}_0/\hat{\lambda})^{1/2} t_{\hat{\rho}} - \frac{1}{2}T\hat{\sigma}(\hat{\lambda} - \hat{\gamma}_0)/(s\hat{\lambda}^{1/2})$$

To incorporate a constant intercept, the regression model $y_t = \mu + \rho y_{t-1} + u_t$ is used (single mean case) and null hypothesis the series is a driftless random walk with nonzero unconditional mean. To incorporate a time trend, we used the regression model $y_t = \mu + \delta t + \rho y_{t-1} + u_t$ and under the null the series is a random walk with drift.

The limiting distributions of the test statistics for the zero mean case are

$$\hat{Z}_\rho \Rightarrow \frac{\frac{1}{2}\{B(1)^2 - 1\}}{\int_0^1 [B(s)]^2 ds}$$

$$\hat{Z}_\tau \Rightarrow \frac{\frac{1}{2}\{[B(1)]^2 - 1\}}{\{\int_0^1 [B(x)]^2 dx\}^{1/2}}$$

where $B(\cdot)$ is a standard Brownian motion.

The limiting distributions of the test statistics for the intercept case are

$$\hat{Z}_\rho \Rightarrow \frac{\frac{1}{2}\{[B(1)]^2 - 1\} - B(1)\int_0^1 B(x)dx}{\int_0^1 [B(x)]^2 dx - \left[\int_0^1 B(x)dx\right]^2}$$

$$\hat{Z}_\tau \Rightarrow \frac{\frac{1}{2}\{[B(1)]^2 - 1\} - B(1)\int_0^1 B(x)dx}{\{\int_0^1 [B(x)]^2 dx - \left[\int_0^1 B(x)dx\right]^2\}^{1/2}}$$

Finally, The limiting distributions of the test statistics for the trend case are can be derived as

$$\begin{bmatrix} 0 & c & 0 \end{bmatrix} V^{-1} \begin{bmatrix} B(1) \\ \left(B(1)^2 - 1\right)/2 \\ B(1) - \int_0^1 B(x)dx \end{bmatrix}$$

where $c = 1$ for $\hat{Z}_\rho$ and $c = \frac{1}{\sqrt{Q}}$ for $\hat{Z}_\tau$,

$$V = \begin{bmatrix} 1 & \int_0^1 B(x)dx & 1/2 \\ \int_0^1 B(x)dx & \int_0^1 B(x)^2dx & \int_0^1 xB(x)dx \\ 1/2 & \int_0^1 xB(x)dx & 1/3 \end{bmatrix}$$

$$Q = \begin{bmatrix} 0 & c & 0 \end{bmatrix} V^{-1} \begin{bmatrix} 0 & c & 0 \end{bmatrix}^T$$

The finite sample performance of the PP test is not satisfactory (see Hayashi (2000)).

When several variables $\mathbf{z}_t = (z_{1t}, \cdots, z_{kt})'$ are cointegrated, there exists a $(k \times 1)$ cointegrating vector $\mathbf{c}$ such that $\mathbf{c}'\mathbf{z}_t$ is stationary and $\mathbf{c}$ is a nonzero vector. The residual based cointegration test assumes the following regression model:

$$y_t = \beta_1 + \mathbf{x}_t'\beta + u_t$$

where $y_t = z_{1t}$, $\mathbf{x}_t = (z_{2t}, \cdots, z_{kt})'$, and $\beta = (\beta_2, \cdots, \beta_k)'$. You can estimate the consistent cointegrating vector by using OLS if all variables are difference stationary — that is, I(1). The estimated cointegrating vector is $\hat{\mathbf{c}} = (1, -\hat{\beta}_2, \cdots, -\hat{\beta}_k)'$. The Phillips-Ouliaris test is computed using the OLS residuals from the preceding regression model, and it uses the PP unit root tests $\hat{Z}_\rho$ and $\hat{Z}_\tau$ developed in Phillips (1987), although in Phillips and Ouliaris (1990) the asymptotic distributions of some other leading unit root tests are also derived. The null hypothesis is no cointegration.

You need to refer to the tables by Phillips and Ouliaris (1990) to obtain the p-value of the cointegration test. Before you apply the cointegration test, you may want to perform the unit root test for each variable (see the option STATIONARITY=).

As in the Engle-Granger cointegration tests, the Phillips-Ouliaris test can give conflicting results for different choices of the regressand. There are other cointegration tests that are invariant to the order of the variables, including Johansen (1988), Johansen (1991), Stock and Watson (1988).

ERS and Ng-Perron Unit Root Tests

As mentioned earlier, ADF and PP both suffer severe size distortion and low power. There is a class of newer tests that improve both size and power. These are sometimes called efficient unit root tests, and among them tests by Elliott, Rothenberg, and Stock (1996) and Ng and Perron (2001) are prominent.

Elliott, Rothenberg, and Stock (1996) consider the data generating process

$$y_t = \beta' z_t + u_t$$
$$u_t = \alpha u_{t-1} + v_t, t = 1, \ldots, T$$

where $\{z_t\}$ is either $\{1\}$ or $\{(1, t)\}$ and $\{v_t\}$ is an unobserved stationary zero-mean process with positive spectral density at zero frequency. The null hypothesis is $H_0 : \alpha = 1$, and the alternative is $H_a : |\alpha| < 1$. The key idea of Elliott, Rothenberg, and Stock (1996) is to study the asymptotic power and asymptotic power envelope of some new tests. Asymptotic power is defined with a sequence of local alternatives. For a fixed alternative hypothesis, the power of a test usually goes to one when sample size goes to infinity; however, this says nothing about the finite sample performance. On the other hand, when the data generating process under the alternative moves closer to the null hypothesis as the sample size increases, the power does not necessarily converge to one. The local-to-unity alternatives in ERS are

$$\alpha = 1 + \frac{c}{T}$$

and the power against the local alternatives has a limit as T goes to infinity, which is called asymptotic power. This value is strictly between 0 and 1. Asymptotic power indicates the adequacy of a test to distinguish small deviations from the null hypothesis.

Define

$$y_\alpha = (y_1, (1-\alpha L)y_2, \ldots, (1-\alpha L)y_T)$$
$$z_\alpha = (z_1, (1-\alpha L)z_2, \ldots, (1-\alpha L)z_T)$$

Let $S(\alpha)$ be the sum of squared residuals from a least squares regression of y_α on z_α. Then the *point optimal test* against the local alternative $\bar{\alpha} = 1 + \bar{c}/T$ has the form

$$P_T^{GLS} = \frac{S(\bar{\alpha}) - \bar{\alpha}S(1)}{\hat{\omega}^2}$$

where $\hat{\omega}^2$ is an estimator for $\omega^2 = \sum_{k=-\infty}^{\infty} E v_t v_{t-k}$. The autoregressive (AR) estimator is used for $\hat{\omega}^2$ (Elliott, Rothenberg, and Stock 1996, equations (13) and (14)):

$$\hat{\omega}^2 = \frac{\hat{\sigma}_\eta^2}{(1 - \sum_{i=1}^{p} \hat{a}_i)^2}$$

where $\hat{\sigma}_\eta^2$ and $\hat{a}_i$ are OLS estimates from the regression

$$\Delta y_t = a_0 y_{t-1} + \sum_{i=1}^{p} a_i \Delta y_{t-i} + a_{p+1} + \eta_t$$

where p is selected according to the Schwarz Bayesian information criterion. The test rejects the null when P_T is small. The asymptotic power function for the point optimal test that is constructed with $\bar{c}$ under local alternatives with c is denoted by $\pi(c, \bar{c})$. Then the power envelope is $\pi(c, c)$ because the test formed with $\bar{c}$ is the most powerful against the alternative $c = \bar{c}$. In other words, the asymptotic function $\pi(c, \bar{c})$ is always below the power envelope $\pi(c)$ except that at one point, $c = \bar{c}$, they are tangent. Elliott, Rothenberg, and Stock (1996) show that choosing some specific values for $\bar{c}$ can cause the asymptotic power function $\pi(c, \bar{c})$ of the point optimal test to be very close to the power envelope. The optimal $\bar{c}$ is -7 when $z_t = 1$, and -13.5 when $z_t = (1, t)'$. This choice of $\bar{c}$ corresponds to the tangent point where $\pi = 0.5$. This is also true of the DF-GLS test.

Elliott, Rothenberg, and Stock (1996) also propose the *DF-GLS test*, given by the t statistic for testing $\psi_0 = 0$ in the regression

$$\Delta y_t^d = \psi_0 y_{t-1}^d + \sum_{j=1}^{p} \psi_j \Delta y_{t-j}^d + \epsilon_{tp}$$

where y_t^d is obtained in a first step detrending

$$y_t^d = y_t - \hat{\beta}_{\bar{\alpha}}' z_t$$

and $\hat{\beta}_{\bar{\alpha}}$ is least squares regression coefficient of y_α on z_α. Regarding the lag length selection, Elliott, Rothenberg, and Stock (1996) favor the Schwarz Bayesian information criterion. The optimal selection of the lag length p and the estimation of ω^2 is further discussed in Ng and Perron (2001). The lag length is selected from the interval $[0, p_{max}]$ for some fixed p_{max} by using the modified Akaike's information criterion,

$$\mathrm{MAIC}(p) = \log(\hat{\sigma}_p^2) + \frac{2(\tau_T(p) + p)}{T - p_{max}}$$

where $\tau_T(p) = (\hat{\sigma}_p^2)^{-1}\hat{\psi}_0^2 \sum_{t=p_{max}+1}^{T-1}(y_t^d)^2$ and $\hat{\sigma}_p^2 = (T - p_{max} - 1)^{-1}\sum_{t=p_{max}+1}^{T-1}\hat{\epsilon}_{tp}^2$. For fixed lag length p, an estimate of ω^2 is given by

$$\hat{\omega}^2 = \frac{(T-1-p)^{-1}\sum_{t=p+2}^{T}\hat{\epsilon}_{tp}^2}{\left(1-\sum_{j=1}^{p}\hat{\psi}_j\right)^2}$$

DF-GLS is indeed a superior unit root test, according to Stock (1994), Schwert (1989), and Elliott, Rothenberg, and Stock (1996). In terms of the size of the test, DF-GLS is almost as good as the ADF t test DF_τ and better than the PP $\hat{\mathrm{Z}}_\rho$ and $\hat{\mathrm{Z}}_\tau$ test. In addition, the power of the DF-GLS test is greater than that of both the ADF t test and the ρ-test.

Ng and Perron (2001) also apply GLS detrending to obtain the following M-tests:

$$MZ_\alpha = ((T-1)^{-1}(y_T^d)^2 - \hat{\omega}^2)\left(2(T-1)^{-2}\sum_{t=1}^{T-1}(y_t^d)^2\right)^{-1}$$

$$MSB = \left(\frac{\sum_{t=1}^{T-1}(y_t^d)^2}{(T-1)^2\hat{\omega}^2}\right)^{1/2}$$

$$MZ_t = MZ_\alpha \times MSB$$

The first one is a modified version of the Phillips-Perron Z_ρ test,

$$MZ_\rho = \mathrm{Z}_\rho + \frac{T}{2}(\hat{\alpha}-1)^2$$

where the detrended data $\{y_t^d\}$ is used. The second is a modified Bhargava (1986) R_1 test statistic. The third can be perceived as a modified Phillips-Perron Z_τ statistic because of the relationship $\mathrm{Z}_\tau = MSB \times \mathrm{Z}_\rho$.

The modified point optimal tests that use the GLS detrended data are

$$MP_T^{GLS} = \frac{\bar{c}^2(T-1)^{-2}\sum_{t=1}^{T-1}(y_t^d)^2 - \bar{c}(T-1)^{-1}(y_T^d)^2}{\hat{\omega}^2} \quad \text{for } z_t = 1$$

$$MP_T^{GLS} = \frac{\bar{c}^2(T-1)^{-2}\sum_{t=1}^{T-1}(y_t^d)^2 + (1-\bar{c})(T-1)^{-1}(y_T^d)^2}{\hat{\omega}^2} \quad \text{for } z_t = (1,t)$$

The DF-GLS test and the MZ_t test have the same limiting distribution:

$$\text{DF-GLS} \approx MZ_t \Rightarrow 0.5\frac{(J_c(1)^2-1)}{\left(\int_0^1 J_c(r)^2dr\right)^{1/2}} \quad \text{for } z_t = 1$$

$$\text{DF-GLS} \approx MZ_t \Rightarrow 0.5\frac{(V_{c,\bar{c}}(1)^2-1)}{\left(\int_0^1 V_{c,\bar{c}}(r)^2dr\right)^{1/2}} \quad \text{for } z_t = (1,t)$$

The point optimal test and the modified point optimal test have the same limiting distribution:

$$P_T^{GLS} \approx MP_T^{GLS} \Rightarrow \bar{c}^2\int_0^1 J_c(r)^2dr - \bar{c}J_c(1)^2 \quad \text{for } z_t = 1$$

$$P_T^{GLS} \approx MP_T^{GLS} \Rightarrow \bar{c}^2\int_0^1 V_{c,\bar{c}}(r)^2dr + (1-\bar{c})V_{c,\bar{c}}(1)^2 \quad \text{for } z_t = (1,t)$$

where $W(r)$ is a standard Brownian motion and $J_c(r)$ is an Ornstein-Uhlenbeck process defined by $dJ_c(r) = cJ_c(r)dr + dW(r)$ with $J_c(0) = 0$, $V_{c,\bar{c}}(r) = J_c(r) - r\left[\lambda J_c(1) + 3(1-\lambda)\int_0^1 sJ_c(s)ds\right]$, and $\lambda = (1-\bar{c})/(1-\bar{c}+\bar{c}^2/3)$.

Overall, the M-tests have the smallest size distortion, with the ADF t test having the next smallest. The ADF ρ-test, $\hat{\mathrm{Z}}_\rho$, and $\hat{\mathrm{Z}}_\tau$ have the largest size distortion. In addition, the power of the DF-GLS and M-tests is greater than that of the ADF t test and ρ-test. The ADF $\hat{\mathrm{Z}}_\rho$ has more severe size distortion than the ADF $\hat{\mathrm{Z}}_\tau$, but it has more power for a fixed lag length.

Kwiatkowski, Phillips, Schmidt, and Shin (KPSS) Unit Root Test and Shin Cointegration Test

There are fewer tests available for the null hypothesis of trend stationarity I(0). The main reason is the difficulty of theoretical development. The KPSS test was introduced in Kwiatkowski et al. (1992) to test the null hypothesis that an observable series is stationary around a deterministic trend. For consistency, the notation used here differs from the notation in the original paper. The setup of the problem is as follows: it is assumed that the series is expressed as the sum of the deterministic trend, random walk r_t, and stationary error u_t; that is,

$$y_t = \mu + \delta t + r_t + u_t$$
$$r_t = r_{t-1} + e_t$$

where $e_t \sim$iid $(0, \sigma_e^2)$, and an intercept μ (in the original paper, the authors use r_0 instead of μ; here we assume $r_0 = 0$.) The null hypothesis of trend stationarity is specified by $H_0 : \sigma_e^2 = 0$, while the null of level stationarity is the same as above with the model restriction $\delta = 0$. Under the alternative that $\sigma_e^2 \neq 0$, there is a random walk component in the observed series y_t.

Under stronger assumptions of normality and iid of u_t and e_t, a one-sided LM test of the null that there is no random walk ($e_t = 0, \forall t$) can be constructed as follows:

$$\widehat{LM} = \frac{1}{T^2} \sum_{t=1}^{T} \frac{S_t^2}{s^2(l)}$$
$$s^2(l) = \frac{1}{T} \sum_{t=1}^{T} \hat{u}_t^2 + \frac{2}{T} \sum_{s=1}^{l} w(s,l) \sum_{t=s+1}^{T} \hat{u}_t \hat{u}_{t-s}$$
$$S_t = \sum_{\tau=1}^{t} \hat{u}_\tau$$

Under the null hypothesis, $\hat{u}_t$ can be estimated by ordinary least squares regression of y_t on an intercept and the time trend. Following the original work of Kwiatkowski et al. (1992), under the null ($\sigma_e^2 = 0$), the $\widehat{LM}$ statistic converges asymptotically to three different distributions depending on whether the model is trend-stationary, level-stationary ($\delta = 0$), or zero-mean stationary ($\delta = 0$, $\mu = 0$). The trend-stationary model is denoted by subscript τ and the level-stationary model is denoted by subscript μ. The case when there is no trend and zero intercept is denoted as 0. The last case, although rarely used in practice, is considered in Hobijn, Franses, and Ooms (2004):

$$y_t = u_t : \quad \widehat{LM}_0 \xrightarrow{D} \int_0^1 B^2(r)dr$$
$$y_t = \mu + u_t : \quad \widehat{LM}_\mu \xrightarrow{D} \int_0^1 V^2(r)dr$$
$$y_t = \mu + \delta t + u_t : \quad \widehat{LM}_\tau \xrightarrow{D} \int_0^1 V_2^2(r)dr$$

with

$$V(r) = B(r) - rB(1)$$
$$V_2(r) = B(r) + (2r - 3r^2)B(1) + (-6r + 6r^2) \int_0^1 B(s)ds$$

where $B(r)$ is a Brownian motion (Wiener process) and $\xrightarrow{D}$ is convergence in distribution. $V(r)$ is a standard Brownian bridge, and $V_2(r)$ is a second-level Brownian bridge.

Using the notation of Kwiatkowski et al. (1992), the $\widehat{LM}$ statistic is named as $\hat{\eta}$. This test depends on the computational method used to compute the long-run variance $s(l)$; that is, the window width l and the kernel type $w(\cdot,\cdot)$. You can specify the kernel used in the test by using the KERNEL option:

- Newey-West/Bartlett (KERNEL=NW | BART) (this is the default)

$$w(s,l) = 1 - \frac{s}{l+1}$$

- quadratic spectral (KERNEL=QS)

$$w(s,l) = \tilde{w}\left(\frac{s}{l}\right) = \tilde{w}(x) = \frac{25}{12\pi^2 x^2}\left(\frac{\sin(6\pi x/5)}{6\pi x/5} - \cos\left(\frac{6}{5}\pi x\right)\right)$$

You can specify the number of lags, l, in three different ways:

- Schwert (SCHW = c) (default for NW, c=12)

$$l = \max\left\{1, \text{floor}\left[c\left(\frac{T}{100}\right)^{1/4}\right]\right\}$$

- manual (LAG = l)
- automatic selection (AUTO) (default for QS), from Hobijn, Franses, and Ooms (2004). The number of lags, l, is calculated as in the following table:

KERNEL=NW	**KERNEL=QS**
$l = \min(T, \text{floor}(\hat{\gamma} T^{1/3}))$	$l = \min(T, \text{floor}(\hat{\gamma} T^{1/5}))$
$\hat{\gamma} = 1.1447\left\{\left(\frac{\hat{s}^{(1)}}{\hat{s}^{(0)}}\right)^2\right\}^{1/3}$	$\hat{\gamma} = 1.3221\left\{\left(\frac{\hat{s}^{(2)}}{\hat{s}^{(0)}}\right)^2\right\}^{1/5}$
$\hat{s}^{(j)} = \delta_{0,j}\hat{\gamma}_0 + 2\sum_{i=1}^{n} i^j \hat{\gamma}_i$	$\hat{s}^{(j)} = \delta_{0,j}\hat{\gamma}_0 + 2\sum_{i=1}^{n} i^j \hat{\gamma}_i$
$n = \text{floor}(T^{2/9})$	$n = \text{floor}(T^{2/25})$

where T is the number of observations, $\delta_{0,j} = 1$ if $j = 0$ and 0 otherwise, and $\hat{\gamma}_i = \frac{1}{T}\sum_{t=1}^{T-i} u_t u_{t+i}$.

Simulation evidence shows that the KPSS has size distortion in finite samples. For an example, see Caner and Kilian (2001). The power is reduced when the sample size is large; this can be derived theoretically (see Breitung (1995)). Another problem of the KPSS test is that the power depends on the truncation lag used in the Newey-West estimator of the long-run variance $s^2(l)$.

Shin (1994) extends the KPSS test to incorporate the regressors to be a cointegration test. The cointegrating regression becomes

$$y_t = \mu + \delta t + X_t'\beta + r_t + u_t$$
$$r_t = r_{t-1} + e_t$$

where y_t and X_t are scalar and m-vector $I(1)$ variables. There are still three cases of cointegrating regressions: without intercept and trend, with intercept only, and with intercept and trend. The null hypothesis of the cointegration test is the same as that for the KPSS test, $H_0 : \sigma_e^2 = 0$. The test statistics for cointegration in the three cases of cointegrating regressions are exactly the same as those in the KPSS test; these test statistics are then ignored here. Under the null hypothesis, the statistics converge asymptotically to three different distributions:

$$
\begin{aligned}
y_t = X_t'\beta + u_t : \quad & \widehat{LM}_0 \xrightarrow{D} \int_0^1 Q_1^2(r)dr \\
y_t = \mu + X_t'\beta + u_t : \quad & \widehat{LM}_\mu \xrightarrow{D} \int_0^1 Q_2^2(r)dr \\
y_t = \mu + \delta t + X_t'\beta + u_t : \quad & \widehat{LM}_\tau \xrightarrow{D} \int_0^1 Q_3^2(r)dr
\end{aligned}
$$

with

$$
\begin{aligned}
Q_1(r) &= B(r) - \left(\int_0^r \mathbf{B_m}(x)dx\right)\left(\int_0^1 \mathbf{B_m}(x)\mathbf{B_m}'(x)dx\right)^{-1}\left(\int_0^1 \mathbf{B_m}(x)dB(x)\right) \\
Q_2(r) &= V(r) - \left(\int_0^r \bar{\mathbf{B}}_\mathbf{m}(x)dx\right)\left(\int_0^1 \bar{\mathbf{B}}_\mathbf{m}(x)\bar{\mathbf{B}}_\mathbf{m}'(x)dx\right)^{-1}\left(\int_0^1 \bar{\mathbf{B}}_\mathbf{m}(x)dB(x)\right) \\
Q_3(r) &= V_2(r) - \left(\int_0^r \mathbf{B}_\mathbf{m}^*(x)dx\right)\left(\int_0^1 \mathbf{B}_\mathbf{m}^*(x)\mathbf{B}_\mathbf{m}^{*\prime}(x)dx\right)^{-1}\left(\int_0^1 \mathbf{B}_\mathbf{m}^*(x)dB(x)\right)
\end{aligned}
$$

where $B(.)$ and $\mathbf{B_m}(.)$ are independent scalar and m-vector standard Brownian motion, and $\xrightarrow{D}$ is convergence in distribution. $V(r)$ is a standard Brownian bridge, $V_2(r)$ is a Brownian bridge of a second-level, $\bar{\mathbf{B}}_\mathbf{m}(r) = \mathbf{B_m}(r) - \int_0^1 \mathbf{B_m}(x)dx$ is an m-vector standard demeaned Brownian motion, and $\mathbf{B}_\mathbf{m}^*(r) = \mathbf{B_m}(r) + (6r - 4)\int_0^1 \mathbf{B_m}(x)dx + (-12r + 6)\int_0^1 x\mathbf{B_m}(x)dx$ is an m-vector standard demeaned and detrended Brownian motion.

The p-values that are reported for the KPSS test and Shin test are calculated via a Monte Carlo simulation of the limiting distributions, using a sample size of 2,000 and 1,000,000 replications.

Testing for Statistical Independence

Independence tests are widely used in model selection, residual analysis, and model diagnostics because models are usually based on the assumption of independently distributed errors. If a given time series (for example, a series of residuals) is independent, then no deterministic model is necessary for this completely random process; otherwise, there must exist some relationship in the series to be addressed. In the following section, four independence tests are introduced: the BDS test, the runs test, the turning point test, and the rank version of von Neumann ratio test.

BDS Test

Brock, Dechert, and Scheinkman (1987) propose a test (BDS test) of independence based on the correlation dimension. Brock et al. (1996) show that the first-order asymptotic distribution of the test statistic is independent of the estimation error provided that the parameters of the model under test can be estimated $\sqrt{n}$-consistently. Hence, the BDS test can be used as a model selection tool and as a specification test.

Given the sample size T, the embedding dimension m, and the value of the radius r, the BDS statistic is

$$S_{\mathrm{BDS}}(T,m,r) = \sqrt{T-m+1}\frac{c_{m,m,T}(r) - c^m_{1,m,T}(r)}{\sigma_{m,T}(r)}$$

where

$$c_{m,n,N}(r) = \frac{2}{(N-n+1)(N-n)} \sum_{s=n}^{N} \sum_{t=s+1}^{N} \prod_{j=0}^{m-1} I_r(z_{s-j}, z_{t-j})$$

$$I_r(z_s, z_t) = \begin{cases} 1 & \text{if } |z_s - z_t| < r \\ 0 & \text{otherwise} \end{cases}$$

$$\sigma^2_{m,T}(r) = 4\left(k^m + 2\sum_{j=1}^{m-1} k^{m-j}c^{2j} + (m-1)^2 c^{2m} - m^2 k c^{2m-2}\right)$$

$$c = c_{1,1,T}(r)$$

$$k = k_T(r) = \frac{6}{T(T-1)(T-2)} \sum_{t=1}^{T} \sum_{s=t+1}^{T} \sum_{l=s+1}^{T} h_r(z_t, z_s, z_l)$$

$$h_r(z_t, z_s, z_l) = \frac{1}{3}\left(I_r(z_t, z_s) I_r(z_s, z_l) + I_r(z_t, z_l) I_r(z_l, z_s) + I_r(z_s, z_t) I_r(z_t, z_l)\right)$$

The statistic has a standard normal distribution if the sample size is large enough. For small sample size, the distribution can be approximately obtained through simulation. Kanzler (1999) has a comprehensive discussion on the implementation and empirical performance of BDS test.

Runs Test and Turning Point Test

The runs test and turning point test are two widely used tests for independence (Cromwell, Labys, and Terraza 1994).

The runs test needs several steps. First, convert the original time series into the sequence of signs, $\{+ + - - \ldots + - - -\}$, that is, map $\{z_t\}$ into $\{sign(z_t - z_M)\}$ where z_M is the sample mean of z_t and $sign(x)$ is "+" if x is nonnegative and "−" if x is negative. Second, count the number of runs, R, in the sequence. A run of a sequence is a maximal non-empty segment of the sequence that consists of adjacent equal elements. For example, the following sequence contains $R = 8$ runs:

$$\underbrace{+++}_{1}\underbrace{---}_{1}\underbrace{++}_{1}\underbrace{--}_{1}\underbrace{+}_{1}\underbrace{-}_{1}\underbrace{+++++}_{1}\underbrace{--}_{1}$$

Third, count the number of pluses and minuses in the sequence and denote them as N_+ and N_-, respectively. In the preceding example sequence, $N_+ = 11$ and $N_- = 8$. Note that the sample size $T = N_+ + N_-$. Finally, compute the statistic of runs test,

$$S_{\mathrm{runs}} = \frac{R - \mu}{\sigma}$$

where

$$\mu = \frac{2N_+N_-}{T} + 1$$

$$\sigma^2 = \frac{(\mu - 1)(\mu - 2)}{T - 1}$$

The statistic of the turning point test is defined as follows:

$$S_{\mathrm{TP}} = \frac{\sum_{t=2}^{T-1} TP_t - 2(T-2)/3}{\sqrt{(16T - 29)/90}}$$

where the indicator function of the turning point TP_t is 1 if $z_t > z_{t\pm1}$ or $z_t < z_{t\pm1}$ (that is, both the previous and next values are greater or less than the current value); otherwise, 0.

The statistics of both the runs test and the turning point test have the standard normal distribution under the null hypothesis of independence.

Rank Version of the von Neumann Ratio Test

Because the runs test completely ignores the magnitudes of the observations, Bartels (1982) proposes a rank version of the von Neumann Ratio test for independence,

$$S_{\mathrm{RVN}} = \frac{\sqrt{T}}{2}\left(\frac{\sum_{t=1}^{T-1}(R_{t+1} - R_t)^2}{(T(T^2-1)/12)} - 2\right)$$

where R_t is the rank of tth observation in the sequence of T observations. For large samples, the statistic follows the standard normal distribution under the null hypothesis of independence. For small samples of size between 11 and 100, the critical values that have been simulated would be more precise. For samples of size less than or equal to 10, the exact CDF of the statistic is available. Hence, the VNRRANK=(PVALUE=SIM) option is recommended for small samples whose size is no more than 100, although it might take longer to obtain the p-value than if you use the VNRRANK=(PVALUE=DIST) option.

Testing for Normality

Based on skewness and kurtosis, Jarque and Bera (1980) calculated the test statistic

$$T_N = \left[\frac{N}{6}b_1^2 + \frac{N}{24}(b_2 - 3)^2\right]$$

where

$$b_1 = \frac{\sqrt{N}\sum_{t=1}^{N}\hat{u}_t^3}{\left(\sum_{t=1}^{N}\hat{u}_t^2\right)^{\frac{3}{2}}}$$

$$b_2 = \frac{N\sum_{t=1}^{N}\hat{u}_t^4}{\left(\sum_{t=1}^{N}\hat{u}_t^2\right)^{2}}$$

The $\chi^2(2)$ distribution gives an approximation to the normality test T_N.

When the GARCH model is estimated, the normality test is obtained using the standardized residuals $\hat{u}_t = \hat{\epsilon}_t/\sqrt{h_t}$. The normality test can be used to detect misspecification of the family of ARCH models.

Testing for Linear Dependence

Generalized Durbin-Watson Tests

Consider the following linear regression model:

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\nu}$$

where $\mathbf{X}$ is an $N \times k$ data matrix, β is a $k \times 1$ coefficient vector, and $\boldsymbol{\nu}$ is a $N \times 1$ disturbance vector. The error term $\boldsymbol{\nu}$ is assumed to be generated by the jth-order autoregressive process $\nu_t = \epsilon_t - \varphi_j \nu_{t-j}$ where $|\varphi_j| < 1$, ϵ_t is a sequence of independent normal error terms with mean 0 and variance σ^2. Usually, the Durbin-Watson statistic is used to test the null hypothesis $H_0 : \varphi_1 = 0$ against $H_1 : -\varphi_1 > 0$. Vinod (1973) generalized the Durbin-Watson statistic:

$$d_j = \frac{\sum_{t=j+1}^{N} (\hat{\nu}_t - \hat{\nu}_{t-j})^2}{\sum_{t=1}^{N} \hat{\nu}_t^2}$$

where $\hat{\boldsymbol{\nu}}$ are OLS residuals. Using the matrix notation,

$$d_j = \frac{\mathbf{Y}'\mathbf{M}\mathbf{A}_j'\mathbf{A}_j\mathbf{M}\mathbf{Y}}{\mathbf{Y}'\mathbf{M}\mathbf{Y}}$$

where $\mathbf{M} = \mathbf{I}_N - \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$ and $\mathbf{A}_j$ is a $(N-j) \times N$ matrix:

$$\mathbf{A}_j = \begin{bmatrix} -1 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 0 & \cdots & 0 & 1 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & -1 & 0 & \cdots & 0 & 1 \end{bmatrix}$$

and there are $j-1$ zeros between -1 and 1 in each row of matrix $\mathbf{A}_j$.

The QR factorization of the design matrix $\mathbf{X}$ yields a $N \times N$ orthogonal matrix $\mathbf{Q}$:

$$\mathbf{X} = \mathbf{Q}\mathbf{R}$$

where R is an $N \times k$ upper triangular matrix. There exists an $N \times (N-k)$ submatrix of $\mathbf{Q}$ such that $\mathbf{Q}_1\mathbf{Q}_1' = \mathbf{M}$ and $\mathbf{Q}_1'\mathbf{Q}_1 = \mathbf{I}_{N-k}$. Consequently, the generalized Durbin-Watson statistic is stated as a ratio of two quadratic forms:

$$d_j = \frac{\sum_{l=1}^{n} \lambda_{jl} {\xi_l}^2}{\sum_{l=1}^{n} \xi_l^2}$$

where $\lambda_{j1} \ldots \lambda_{jn}$ are upper n eigenvalues of $\mathbf{M}\mathbf{A'}_j\mathbf{A}_j\mathbf{M}$ and ξ_l is a standard normal variate, and $n = \min(N-k, N-j)$. These eigenvalues are obtained by a singular value decomposition of $\mathbf{Q'}_1\mathbf{A'}_j$ (Golub and Van Loan 1989; Savin and White 1978).

The marginal probability (or p-value) for d_j given c_0 is

$$\text{Prob}\left(\frac{\sum_{l=1}^{n} \lambda_{jl}\xi_l^2}{\sum_{l=1}^{n} \xi_l^2} < c_0\right) = \text{Prob}(q_j < 0)$$

where

$$q_j = \sum_{l=1}^{n} (\lambda_{jl} - c_0)\xi_l^2$$

When the null hypothesis $H_0 : \varphi_j = 0$ holds, the quadratic form q_j has the characteristic function

$$\phi_j(t) = \prod_{l=1}^{n} (1 - 2(\lambda_{jl} - c_0)it)^{-1/2}$$

The distribution function is uniquely determined by this characteristic function:

$$F(x) = \frac{1}{2} + \frac{1}{2\pi} \int_0^\infty \frac{e^{itx}\phi_j(-t) - e^{-itx}\phi_j(t)}{it} dt$$

For example, to test $H_0 : \varphi_4 = 0$ given $\varphi_1 = \varphi_2 = \varphi_3 = 0$ against $H_1 : -\varphi_4 > 0$, the marginal probability (p-value) can be used:

$$F(0) = \frac{1}{2} + \frac{1}{2\pi} \int_0^\infty \frac{(\phi_4(-t) - \phi_4(t))}{it} dt$$

where

$$\phi_4(t) = \prod_{l=1}^{n} (1 - 2(\lambda_{4l} - \hat{d}_4)it)^{-1/2}$$

and $\hat{d}_4$ is the calculated value of the fourth-order Durbin-Watson statistic.

In the Durbin-Watson test, the marginal probability indicates positive autocorrelation ($-\varphi_j > 0$) if it is less than the level of significance (α), while you can conclude that a negative autocorrelation ($-\varphi_j < 0$) exists if the marginal probability based on the computed Durbin-Watson statistic is greater than $1 - \alpha$. Wallis (1972) presented tables for bounds tests of fourth-order autocorrelation, and Vinod (1973) has given tables for a 5% significance level for orders two to four. Using the AUTOREG procedure, you can calculate the exact p-values for the general order of Durbin-Watson test statistics. Tests for the absence of autocorrelation of order p can be performed sequentially; at the jth step, test $H_0 : \varphi_j = 0$ given $\varphi_1 = \ldots = \varphi_{j-1} = 0$ against $\varphi_j \neq 0$. However, the size of the sequential test is not known.

The Durbin-Watson statistic is computed from the OLS residuals, while that of the autoregressive error model uses residuals that are the difference between the predicted values and the actual values. When you use the Durbin-Watson test from the residuals of the autoregressive error model, you must be aware that this test is only an approximation. See "Autoregressive Error Model" on page 387 earlier in this chapter. If there are missing values, the Durbin-Watson statistic is computed using all the nonmissing values and ignoring the gaps caused by missing residuals. This does not affect the significance level of the resulting test, although the power of the test against certain alternatives may be adversely affected. Savin and White (1978) have examined the use of the Durbin-Watson statistic with missing values.

The Durbin-Watson probability calculations have been enhanced to compute the p-value of the generalized Durbin-Watson statistic for large sample sizes. Previously, the Durbin-Watson probabilities were only calculated for small sample sizes.

Consider the following linear regression model:

$$\mathbf{Y} = \mathbf{X}\beta + \mathbf{u}$$

$$u_t + \varphi_j u_{t-j} = \epsilon_t, \qquad t = 1, \ldots, N$$

where $\mathbf{X}$ is an $N \times k$ data matrix, β is a $k \times 1$ coefficient vector, $\mathbf{u}$ is a $N \times 1$ disturbance vector, and ϵ_t is a sequence of independent normal error terms with mean 0 and variance σ^2.

The generalized Durbin-Watson statistic is written as

$$\mathrm{DW}_j = \frac{\hat{\mathbf{u}}'\mathbf{A}_j'\mathbf{A}_j\hat{\mathbf{u}}}{\hat{\mathbf{u}}'\hat{\mathbf{u}}}$$

where $\hat{\mathbf{u}}$ is a vector of OLS residuals and $\mathbf{A}_j$ is a $(T-j) \times T$ matrix. The generalized Durbin-Watson statistic DW_j can be rewritten as

$$\mathrm{DW}_j = \frac{\mathbf{Y}'\mathbf{M}\mathbf{A}_j'\mathbf{A}_j\mathbf{M}\mathbf{Y}}{\mathbf{Y}'\mathbf{M}\mathbf{Y}} = \frac{\eta'(\mathbf{Q}_1'\mathbf{A}_j'\mathbf{A}_j\mathbf{Q}_1)\eta}{\eta'\eta}$$

where $\mathbf{Q}_1'\mathbf{Q}_1 = \mathbf{I}_{T-k}$, $\mathbf{Q}_1'\mathbf{X} = 0$, and $\eta = \mathbf{Q}_1'\mathbf{u}$.

The marginal probability for the Durbin-Watson statistic is

$$\Pr(\mathrm{DW}_j < c) = \Pr(h < 0)$$

where $h = \eta'(\mathbf{Q}_1'\mathbf{A}_j'\mathbf{A}_j\mathbf{Q}_1 - c\mathbf{I})\eta$.

The p-value or the marginal probability for the generalized Durbin-Watson statistic is computed by numerical inversion of the characteristic function $\phi(u)$ of the quadratic form $h = \eta'(\mathbf{Q}_1'\mathbf{A}_j'\mathbf{A}_j\mathbf{Q}_1 - c\mathbf{I})\eta$. The trapezoidal rule approximation to the marginal probability $\Pr(h < 0)$ is

$$\Pr(h < 0) = \frac{1}{2} - \sum_{k=0}^{K} \frac{\mathrm{Im}\left[\phi((k+\frac{1}{2})\Delta)\right]}{\pi(k+\frac{1}{2})} + \mathrm{E}_I(\Delta) + \mathrm{E}_T(K)$$

where $\mathrm{Im}\left[\phi(\cdot)\right]$ is the imaginary part of the characteristic function, $\mathrm{E}_I(\Delta)$ and $\mathrm{E}_T(K)$ are integration and truncation errors, respectively. Refer to Davies (1973) for numerical inversion of the characteristic function.

Ansley, Kohn, and Shively (1992) proposed a numerically efficient algorithm that requires O(N) operations for evaluation of the characteristic function $\phi(u)$. The characteristic function is denoted as

$$\begin{aligned}\phi(u) &= \left|\mathbf{I} - 2iu(\mathbf{Q}_1'\mathbf{A}_j'\mathbf{A}_j\mathbf{Q}_1 - c\mathbf{I}_{N-k})\right|^{-1/2} \\ &= |\mathbf{V}|^{-1/2}\left|\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right|^{-1/2}\left|\mathbf{X}'\mathbf{X}\right|^{1/2}\end{aligned}$$

where $\mathbf{V} = (1+2iuc)\mathbf{I} - 2iu\mathbf{A}_j'\mathbf{A}_j$ and $i = \sqrt{-1}$. By applying the Cholesky decomposition to the complex matrix $\mathbf{V}$, you can obtain the lower triangular matrix $\mathbf{G}$ that satisfies $\mathbf{V} = \mathbf{G}\mathbf{G}'$. Therefore, the characteristic function can be evaluated in O(N) operations by using the following formula:

$$\phi(u) = |\mathbf{G}|^{-1}\left|\mathbf{X}^{*\prime}\mathbf{X}^*\right|^{-1/2}\left|\mathbf{X}'\mathbf{X}\right|^{1/2}$$

where $\mathbf{X}^* = \mathbf{G}^{-1}\mathbf{X}$. Refer to Ansley, Kohn, and Shively (1992) for more information on evaluation of the characteristic function.

Tests for Serial Correlation with Lagged Dependent Variables

When regressors contain lagged dependent variables, the Durbin-Watson statistic (d_1) for the first-order autocorrelation is biased toward 2 and has reduced power. Wallis (1972) shows that the bias in the Durbin-Watson statistic (d_4) for the fourth-order autocorrelation is smaller than the bias in d_1 in the presence of a first-order lagged dependent variable. Durbin (1970) proposes two alternative statistics (Durbin h and t) that are asymptotically equivalent. The h statistic is written as

$$h = \hat{\rho}\sqrt{N/(1 - N\hat{V})}$$

where $\hat{\rho} = \sum_{t=2}^{N} \hat{v}_t \hat{v}_{t-1} / \sum_{t=1}^{N} \hat{v}_t^2$ and $\hat{V}$ is the least squares variance estimate for the coefficient of the lagged dependent variable. Durbin's t test consists of regressing the OLS residuals $\hat{v}_t$ on explanatory variables and $\hat{v}_{t-1}$ and testing the significance of the estimate for coefficient of $\hat{v}_{t-1}$.

Inder (1984) shows that the Durbin-Watson test for the absence of first-order autocorrelation is generally more powerful than the h test in finite samples. Refer to Inder (1986) and King and Wu (1991) for the Durbin-Watson test in the presence of lagged dependent variables.

Godfrey LM test

The GODFREY= option in the MODEL statement produces the Godfrey Lagrange multiplier test for serially correlated residuals for each equation (Godfrey 1978b, a). r is the maximum autoregressive order, and specifies that Godfrey's tests be computed for lags 1 through r. The default number of lags is four.

Testing for Nonlinear Dependence: Ramsey's Reset Test

Ramsey's reset test is a misspecification test associated with the functional form of models to check whether power transforms need to be added to a model. The original linear model, henceforth called the restricted model, is

$$y_t = \mathbf{x_t}\beta + u_t$$

To test for misspecification in the functional form, the unrestricted model is

$$y_t = \mathbf{x_t}\beta + \sum_{j=2}^{p} \phi_j \hat{y}_t^j + u_t$$

where $\hat{y}_t$ is the predicted value from the linear model and p is the power of $\hat{y}_t$ in the unrestricted model equation starting from 2. The number of higher-ordered terms to be chosen depends on the discretion of the analyst. The RESET option produces test results for $p = 2$, 3, and 4.

The reset test is an F statistic for testing $H_0 : \phi_j = 0$, for all $j = 2, \ldots, p$, against $H_1 : \phi_j \neq 0$ for at least one $j = 2, \ldots, p$ in the unrestricted model and is computed as follows:

$$F_{(p-1,n-k-p+1)} = \frac{(SSE_R - SSE_U)/(p-1)}{SSE_U/(n-k-p+1)}$$

where SSE_R is the sum of squared errors due to the restricted model, SSE_U is the sum of squared errors due to the unrestricted model, n is the total number of observations, and k is the number of parameters in the original linear model.

Ramsey's test can be viewed as a linearity test that checks whether any nonlinear transformation of the specified independent variables has been omitted, but it need not help in identifying a new relevant variable other than those already specified in the current model.

Testing for Nonlinear Dependence: Heteroscedasticity Tests

Portmanteau Q Test

For nonlinear time series models, the portmanteau test statistic based on squared residuals is used to test for independence of the series (McLeod and Li 1983):

$$Q(q) = N(N+2)\sum_{i=1}^{q}\frac{r(i;\hat{\nu}_t^2)}{(N-i)}$$

where

$$r(i;\hat{\nu}_t^2) = \frac{\sum_{t=i+1}^{N}(\hat{\nu}_t^2-\hat{\sigma}^2)(\hat{\nu}_{t-i}^2-\hat{\sigma}^2)}{\sum_{t=1}^{N}(\hat{\nu}_t^2-\hat{\sigma}^2)^2}$$

$$\hat{\sigma}^2 = \frac{1}{N}\sum_{t=1}^{N}\hat{\nu}_t^2$$

This Q statistic is used to test the nonlinear effects (for example, GARCH effects) present in the residuals. The GARCH(p,q) process can be considered as an ARMA$(\max(p,q), p)$ process. See the section "Predicting the Conditional Variance" on page 429. Therefore, the Q statistic calculated from the squared residuals can be used to identify the order of the GARCH process.

Engle's Lagrange Multiplier Test for ARCH Disturbances

Engle (1982) proposed a Lagrange multiplier test for ARCH disturbances. The test statistic is asymptotically equivalent to the test used by Breusch and Pagan (1979). Engle's Lagrange multiplier test for the qth order ARCH process is written

$$LM(q) = \frac{N\mathbf{W}'\mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'\mathbf{W}}{\mathbf{W}'\mathbf{W}}$$

where

$$\mathbf{W} = \left(\frac{\hat{\nu}_1^2}{\hat{\sigma}^2}-1,\ldots,\frac{\hat{\nu}_N^2}{\hat{\sigma}^2}-1\right)'$$

and

$$\mathbf{Z} = \begin{bmatrix} 1 & \hat{\nu}_0^2 & \cdots & \hat{\nu}_{-q+1}^2 \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \hat{\nu}_{N-1}^2 & \cdots & \hat{\nu}_{N-q}^2 \end{bmatrix}$$

The presample values ($\nu_0^2,\ldots,\nu_{-q+1}^2$) have been set to 0. Note that the LM(q) tests might have different finite-sample properties depending on the presample values, though they are asymptotically equivalent regardless of the presample values.

Lee and King's Test for ARCH Disturbances

Engle's Lagrange multiplier test for ARCH disturbances is a two-sided test; that is, it ignores the inequality constraints for the coefficients in ARCH models. Lee and King (1993) propose a one-sided test and prove that the test is locally most mean powerful. Let $\varepsilon_t, t = 1, ..., T$, denote the residuals to be tested. Lee and King's test checks

$$H_0 : \alpha_i = 0, i = 1, ..., q$$

$$H_1 : \alpha_i > 0, i = 1, ..., q$$

where $\alpha_i, i = 1, ..., q$, are in the following ARCH(q) model:

$$\varepsilon_t = \sqrt{h_t} e_t, e_t \; iid(0, 1)$$

$$h_t = \alpha_0 + \sum_{i=1}^{q} \alpha_i \varepsilon_{t-i}^2$$

The statistic is written as

$$S = \frac{\sum_{t=q+1}^{T} (\frac{\varepsilon_t^2}{h_0} - 1) \sum_{i=1}^{q} \varepsilon_{t-i}^2}{\left[2 \sum_{t=q+1}^{T} (\sum_{i=1}^{q} \varepsilon_{t-i}^2)^2 - \frac{2(\sum_{t=q+1}^{T} \sum_{i=1}^{q} \varepsilon_{t-i}^2)^2}{T-q} \right]^{1/2}}$$

Wong and Li's Test for ARCH Disturbances

Wong and Li (1995) propose a rank portmanteau statistic to minimize the effect of the existence of outliers in the test for ARCH disturbances. They first rank the squared residuals; that is, $R_t = rank(\varepsilon_t^2)$. Then they calculate the rank portmanteau statistic

$$Q_R = \sum_{i=1}^{q} \frac{(r_i - \mu_i)^2}{\sigma_i^2}$$

where r_i, μ_i, and σ_i^2 are defined as follows:

$$r_i = \frac{\sum_{t=i+1}^{T} (R_t - (T+1)/2)(R_{t-i} - (T+1)/2)}{T(T^2-1)/12}$$

$$\mu_i = -\frac{T-i}{T(T-1)}$$

$$\sigma_i^2 = \frac{5T^4 - (5i+9)T^3 + 9(i-2)T^2 + 2i(5i+8)T + 16i^2}{5(T-1)^2 T^2 (T+1)}$$

The Q, Engle's LM, Lee and King's, and Wong and Li's statistics are computed from the OLS residuals, or residuals if the NLAG= option is specified, assuming that disturbances are white noise. The Q, Engle's LM, and Wong and Li's statistics have an approximate $\chi_{(q)}^2$ distribution under the white-noise null hypothesis, while the Lee and King's statistic has a standard normal distribution under the white-noise null hypothesis.

Testing for Structural Change

Chow Test

Consider the linear regression model

$$\mathbf{y} = \mathbf{X}\beta + \mathbf{u}$$

where the parameter vector β contains k elements.

Split the observations for this model into two subsets at the break point specified by the CHOW= option, so that

$$\begin{aligned} \mathbf{y} &= (\mathbf{y'}_1, \mathbf{y'}_2)' \\ \mathbf{X} &= (\mathbf{X'}_1, \mathbf{X'}_2)' \\ \mathbf{u} &= (\mathbf{u'}_1, \mathbf{u'}_2)' \end{aligned}$$

Now consider the two linear regressions for the two subsets of the data modeled separately,

$$\mathbf{y}_1 = \mathbf{X}_1\beta_1 + \mathbf{u}_1$$

$$\mathbf{y}_2 = \mathbf{X}_2\beta_2 + \mathbf{u}_2$$

where the number of observations from the first set is n_1 and the number of observations from the second set is n_2.

The Chow test statistic is used to test the null hypothesis $H_0 : \beta_1 = \beta_2$ conditional on the same error variance $V(\mathbf{u}_1) = V(\mathbf{u}_2)$. The Chow test is computed using three sums of square errors:

$$\mathrm{F}_{chow} = \frac{(\hat{\mathbf{u}}'\hat{\mathbf{u}} - \hat{\mathbf{u}}'_1\hat{\mathbf{u}}_1 - \hat{\mathbf{u}}'_2\hat{\mathbf{u}}_2)/k}{(\hat{\mathbf{u}}'_1\hat{\mathbf{u}}_1 + \hat{\mathbf{u}}'_2\hat{\mathbf{u}}_2)/(n_1 + n_2 - 2k)}$$

where $\hat{\mathbf{u}}$ is the regression residual vector from the full set model, $\hat{\mathbf{u}}_1$ is the regression residual vector from the first set model, and $\hat{\mathbf{u}}_2$ is the regression residual vector from the second set model. Under the null hypothesis, the Chow test statistic has an F distribution with k and $(n_1 + n_2 - 2k)$ degrees of freedom, where k is the number of elements in β.

Chow (1960) suggested another test statistic that tests the hypothesis that the mean of prediction errors is 0. The predictive Chow test can also be used when $n_2 < k$.

The PCHOW= option computes the predictive Chow test statistic

$$\mathrm{F}_{pchow} = \frac{(\hat{\mathbf{u}}'\hat{\mathbf{u}} - \hat{\mathbf{u}}'_1\hat{\mathbf{u}}_1)/n_2}{\hat{\mathbf{u}}'_1\hat{\mathbf{u}}_1/(n_1 - k)}$$

The predictive Chow test has an F distribution with n_2 and $(n_1 - k)$ degrees of freedom.

Bai and Perron's Multiple Structural Change Tests

Bai and Perron (1998) propose several kinds of multiple structural change tests: (1) the test of no break versus a fixed number of breaks ($supF$ test), (2) the equal and unequal weighted versions of double maximum tests of no break versus an unknown number of breaks given some upper bound ($UDmaxF$ test and $WDmaxF$ test), and (3) the test of l versus $l + 1$ breaks ($supF_{l+1|l}$ test). Bai and Perron (2003a, b, 2006) also show how to implement these tests, the commonly used critical values, and the simulation analysis on these tests.

Consider the following partial structural change model with m breaks ($m + 1$ regimes):

$$y_t = x_t'\beta + z_t'\delta_j + u_t, \qquad t = T_{j-1} + 1, ..., T_j, \, j = 1, ..., m$$

Here, y_t is the dependent variable observed at time t, $x_t(p \times 1)$ is a vector of covariates with coefficients β unchanged over time, and $z_t(q \times 1)$ is a vector of covariates with coefficients δ_j at regime j, $j = 1, ..., m$. If $p = 0$ (that is, there are no x regressors), the regression model becomes the pure structural change model. The indices $(T_1, ..., T_m)$ (that is, the break dates or break points) are unknown, and the convenient notation $T_0 = 0$ and $T_{m+1} = T$ applies. For any given m-partition $(T_1, ..., T_m)$, the associated least squares estimates of β and δ_j, $j = 1, ..., m$, are obtained by minimizing the sum of squared residuals (SSR),

$$S_T(T_1, ..., T_m) = \sum_{i=1}^{m+1} \sum_{t=T_{i-1}+1}^{T_i} (y_t - x_t'\beta - z_t'\delta_i)^2$$

Let $\hat{S}_T(T_1, ..., T_m)$ denote the minimized SSR for a given $(T_1, ..., T_m)$. The estimated break dates $(\hat{T}_1, ..., \hat{T}_m)$ are such that

$$(\hat{T}_1, ..., \hat{T}_m) = \arg \min_{T_1,...,T_m} \hat{S}_T(T_1, ..., T_m)$$

where the minimization is taken over all partitions $(T_1, ..., T_m)$ such that $T_i - T_{i-1} \geq T\epsilon$. Bai and Perron (2003a) propose an efficient algorithm, based on the principle of dynamic programming, to estimate the preceding model.

In the case that the data are nontrending, as stated in Bai and Perron (1998), the limiting distribution of the break dates is as follows:

$$\frac{(\Delta_i' Q_i \Delta_i)^2}{(\Delta_i' \Omega_i \Delta_i)}(\hat{T}_i - T_i^0) \Rightarrow \arg\max_s V^{(i)}(s), \qquad i = 1, ..., m$$

where

$$V^{(i)}(s) = \begin{cases} W_1^{(i)}(-s) - |s|/2 & \text{if } s \leq 0 \\ \sqrt{\eta_i}(\phi_{i,2}/\phi_{i,1}) W_2^{(i)}(s) - \eta_i |s|/2 & \text{if } s > 0 \end{cases}$$

and

$$\begin{aligned} \Delta T_i^0 &= T_i^0 - T_{i-1}^0 \\ \Delta_i &= \delta_{i+1}^0 - \delta_i^0 \\ Q_i &= \lim (\Delta T_i^0)^{-1} \sum_{t=T_{i-1}^0+1}^{T_i^0} E(z_t z_t') \\ \Omega_i &= \lim (\Delta T_i^0)^{-1} \sum_{r=T_{i-1}^0+1}^{T_i^0} \sum_{t=T_{i-1}^0+1}^{T_i^0} E(z_r z_t' u_r u_t) \\ \eta_i &= \Delta_i' Q_{i+1} \Delta_i / \Delta_i' Q_i \Delta_i \\ \phi_{i,1}^2 &= \Delta_i' \Omega_i \Delta_i / \Delta_i' Q_i \Delta_i \\ \phi_{i,2}^2 &= \Delta_i' \Omega_{i+1} \Delta_i / \Delta_i' Q_{i+1} \Delta_i \end{aligned}$$

Also, $W_1^{(i)}(s)$ and $W_2^{(i)}(s)$ are independent standard Weiner processes that are defined on $[0, \infty)$, starting at the origin when $s = 0$; these processes are also independent across i. The cumulative distribution function of $\arg\max_s V^{(i)}(s)$ is shown in Bai (1997). Hence, with the estimates of Δ_i, Q_i, and Ω_i, the relevant critical values for confidence interval of break dates T_i can be calculated. The estimate of Δ_i is $\hat{\delta}_{i+1} - \hat{\delta}_i$. The estimate of Q_i is either

$$\hat{Q}_i = (\Delta \hat{T}_i)^{-1} \sum_{t=\hat{T}_{i-1}^0+1}^{\hat{T}_i^0} z_t z_t'$$

if the regressors are assumed to have heterogeneous distributions across regimes (that is, the HQ option is specified), or

$$\hat{Q}_i = \hat{Q} = (T)^{-1} \sum_{t=1}^{T} z_t z_t'$$

if the regressors are assumed to have identical distributions across regimes (that is, the HQ option is not specified). The estimate of Ω_i can also be constructed with data over regime i only or the whole sample, depending on whether the vectors $z_t \hat{u}_t$ are heterogeneously distributed across regimes (that is, the HO option is specified). If the HAC option is specified, $\hat{\Omega}_i$ is estimated through the heteroscedasticity- and autocorrelation-consistent (HAC) covariance matrix estimator applied to vectors $z_t \hat{u}_t$.

The $supF$ test of no structural break ($m = 0$) versus the alternative hypothesis that there are a fixed number, $m = k$, of breaks is defined as follows:

$$supF(k) = \frac{1}{T}\left(\frac{T-(k+1)q-p}{kq}\right)(R\hat{\theta})'(R\hat{V}(\hat{\theta})R')^{-1}(R\hat{\theta})$$

where

$$R_{(kq)\times(p+(k+1)q)} = \begin{pmatrix} 0_{q\times p} & I_q & -I_q & 0 & 0 & \cdots & 0 \\ 0_{q\times p} & 0 & I_q & -I_q & 0 & \cdots & 0 \\ \vdots & \cdots & \ddots & \ddots & \ddots & \ddots & \cdots \\ 0_{q\times p} & 0 & \cdots & \cdots & 0 & I_q & -I_q \end{pmatrix}$$

and I_q is the $q \times q$ identity matrix; $\hat{\theta}$ is the coefficient vector $(\hat{\beta}'\hat{\delta}_1'...\hat{\delta}_{k+1})'$, which together with the break dates $(\hat{T}_1...\hat{T}_k)$ minimizes the global sum of squared residuals; and $\hat{V}(\hat{\theta})$ is an estimate of the variance-covariance matrix of $\hat{\theta}$, which could be estimated by using the HAC estimator or another way, depending on how the HAC, HR, and HE options are specified. The output $supF$ test statistics are scaled by q, the number of regressors, to be consistent with the limiting distribution; Bai and Perron (2003b, 2006) take the same action.

There are two versions of double maximum tests of no break against an unknown number of breaks given some upper bound M: the $UDmaxF$ test:

$$UDmaxF(M) = \max_{1\le m\le M} supF(m)$$

and the $WDmaxF$ test:

$$WDmaxF(M,\alpha) = \max_{1\le m\le M} \frac{c_\alpha(1)}{c_\alpha(m)} supF(m)$$

where α is the significance level and $c_\alpha(m)$ is the critical value of $supF(m)$ test given the significance level α. Four kinds of $WDmaxF$ tests that correspond to $\alpha = 0.100, 0.050, 0.025$, and 0.010 are implemented.

The $supF(l+1|l)$ test of l versus $l+1$ breaks is calculated in two ways that are asymptotically the same. In the first calculation, the method amounts to the application of $(l+1)$ tests of the null hypothesis of no structural change versus the alternative hypothesis of a single change. The test is applied to each segment that contains the observations $\hat{T}_{i-1}$ to $\hat{T}_i$ $(i = 1, ..., l+1)$. The $supF(l+1|l)$ test statistics are the maximum of these $(l+1)$ $supF$ test statistics. In the second calculation, for the given l breaks $(\hat{T}_1, ..., \hat{T}_l)$, the new break $\hat{T}^{(N)}$ is to minimize the global SSR:

$$\hat{T}^{(N)} = \arg\min_{T^{(N)}} SSR(\hat{T}_1, ..., \hat{T}_l; T^{(N)})$$

Then,

$$supF(l+1|l) = (T-(l+1)q-p)\frac{SSR(\hat{T}_1, ..., \hat{T}_l) - SSR(\hat{T}_1, ..., \hat{T}_l; \hat{T}^{(N)})}{SSR(\hat{T}_1, ..., \hat{T}_l)}$$

The p-value of each test is based on the simulation of the limiting distribution of that test.

Predicted Values

The AUTOREG procedure can produce two kinds of predicted values for the response series and corresponding residuals and confidence limits. The residuals in both cases are computed as the actual value minus the predicted value. In addition, when GARCH models are estimated, the AUTOREG procedure can output predictions of the conditional error variance.

Predicting the Unconditional Mean

The first type of predicted value is obtained from only the structural part of the model, $\mathbf{x}_t'\mathbf{b}$. These are useful in predicting values of new response time series, which are assumed to be described by the same model as the current response time series. The predicted values, residuals, and upper and lower confidence limits for the structural predictions are requested by specifying the PREDICTEDM=, RESIDUALM=, UCLM=, or LCLM= option in the OUTPUT statement. The ALPHACLM= option controls the confidence level for UCLM= and LCLM=. These confidence limits are for estimation of the mean of the dependent variable, $\mathbf{x}_t'\mathbf{b}$, where $\mathbf{x}_t$ is the column vector of independent variables at observation t.

The predicted values are computed as

$$\hat{y}_t = \mathbf{x}_t'\mathbf{b}$$

and the upper and lower confidence limits as

$$\hat{u}_t = \hat{y}_t + t_{\alpha/2}\mathrm{v}$$

$$\hat{l}_t = \hat{y}_t - t_{\alpha/2}\mathrm{v}$$

where v^2 is an estimate of the variance of $\hat{y}_t$ and $t_{\alpha/2}$ is the upper $\alpha/2$ percentage point of the t distribution.

$$\mathrm{Prob}(T > t_{\alpha/2}) = \alpha/2$$

where T is an observation from a t distribution with q degrees of freedom. The value of α can be set with the ALPHACLM= option. The degrees of freedom parameter, q, is taken to be the number of observations minus the number of free parameters in the final model. For the YW estimation method, the value of v is calculated as

$$\mathrm{v} = \sqrt{s^2 \mathbf{x}_t' (\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1} \mathbf{x}_t}$$

where s^2 is the error sum of squares divided by q. For the ULS and ML methods, it is calculated as

$$\mathrm{v} = \sqrt{s^2 \mathbf{x}_t' \mathbf{W} \mathbf{x}_t}$$

where $\mathbf{W}$ is the $k \times k$ submatrix of $(\mathbf{J}'\mathbf{J})^{-1}$ that corresponds to the regression parameters. For details, see the section "Computational Methods" on page 389 earlier in this chapter.

Predicting Future Series Realizations

The other predicted values use both the structural part of the model and the predicted values of the error process. These conditional mean values are useful in predicting future values of the current response time series. The predicted values, residuals, and upper and lower confidence limits for future observations conditional on past values are requested by the PREDICTED=, RESIDUAL=, UCL=, or LCL= option in the OUTPUT statement. The ALPHACLI= option controls the confidence level for UCL= and LCL=. These confidence limits are for the predicted value,

$$\tilde{y}_t = \mathbf{x}_t' \mathbf{b} + v_{t|t-1}$$

where $\mathbf{x}_t$ is the vector of independent variables if all independent variables at time t are nonmissing, and $v_{t|t-1}$ is the minimum variance linear predictor of the error term, which is defined in the following recursive way given the autoregressive model, AR(m) model, for v_t:

$$v_{s|t} = \begin{cases} -\sum_{i=1}^{m} \hat{\varphi}_i v_{s-i|t} & s > t \text{ or observation } s \text{ is missing} \\ y_s - \mathbf{x}_s' \mathbf{b} & 0 < s \leq t \text{ and observation } s \text{ is nonmissing} \\ 0 & s \leq 0 \end{cases}$$

where $\hat{\varphi}_i, i = 1, \ldots, m$, are the estimated AR parameters. Observation s is considered to be missing if the dependent variable or at least one independent variable is missing. If some of the independent variables at time t are missing, the predicted $\tilde{y}_t$ is also missing. With the same definition of $v_{s|t}$, the prediction method can be easily extended to the multistep forecast of $\tilde{y}_{t+d}, d > 0$:

$$\tilde{y}_{t+d} = \mathbf{x}_{t+d}' \mathbf{b} + v_{t+d|t-1}$$

The prediction method is implemented through the Kalman filter.

If $\tilde{y}_t$ is not missing, the upper and lower confidence limits are computed as

$$\tilde{u}_t = \tilde{y}_t + t_{\alpha/2} \mathrm{v}$$

$$\tilde{l}_t = \tilde{y}_t - t_{\alpha/2} \mathrm{v}$$

where v, in this case, is computed as

$$\mathrm{v} = \sqrt{\mathbf{z}_t' \mathbf{V}_\beta \mathbf{z}_t + s^2 r}$$

where $\mathbf{V}_\beta$ is the variance-covariance matrix of the estimation of regression parameter β; $\mathbf{z}_t$ is defined as

$$\mathbf{z}_t = \mathbf{x}_t + \sum_{i=1}^{m} \hat{\varphi}_i \mathbf{x}_{t-i|t-1}$$

and $\mathbf{x}_{s|t}$ is defined in a similar way as $\nu_{s|t}$:

$$\mathbf{x}_{s|t} = \begin{cases} -\sum_{i=1}^{m} \hat{\varphi}_i \mathbf{x}_{s-i|t} & s > t \text{ or observation } s \text{ is missing} \\ \mathbf{x}_s & 0 < s \leq t \text{ and observation } s \text{ is nonmissing} \\ 0 & s \leq 0 \end{cases}$$

The formula for computing the prediction variance v is deducted based on Baillie (1979).

The value $s^2 r$ is the estimate of the conditional prediction error variance. At the start of the series, and after missing values, r is usually greater than 1. For the computational details of r, see the section "Predicting the Conditional Variance" on page 429. The plot of residuals and confidence limits in Example 9.4 illustrates this behavior.

Except to adjust the degrees of freedom for the error sum of squares, the preceding formulas do not account for the fact that the autoregressive parameters are estimated. In particular, the confidence limits are likely to be somewhat too narrow. In large samples, this is probably not an important effect, but it might be appreciable in small samples. For some discussion of this problem for AR(1) models, see Harvey (1981) .

At the beginning of the series (the first m observations, where m is the value of the NLAG= option) and after missing values, these residuals do not match the residuals obtained by using OLS on the transformed variables. This is because, in these cases, the predicted noise values must be based on less than a complete set of past noise values and, thus, have larger variance. The GLS transformation for these observations includes a scale factor in addition to a linear combination of past values. Put another way, the $\mathbf{L}^{-1}$ matrix defined in the section "Computational Methods" on page 389 has the value 1 along the diagonal, except for the first m observations and after missing values.

Predicting the Conditional Variance

The GARCH process can be written as

$$\epsilon_t^2 = \omega + \sum_{i=1}^{n} (\alpha_i + \gamma_i)\epsilon_{t-i}^2 - \sum_{j=1}^{p} \gamma_j \eta_{t-j} + \eta_t$$

where $\eta_t = \epsilon_t^2 - h_t$ and $n = \max(p, q)$. This representation shows that the squared residual ϵ_t^2 follows an ARMA(n, p) process. Then for any $d > 0$, the conditional expectations are as follows:

$$\mathbf{E}(\epsilon_{t+d}^2|\Psi_t) = \omega + \sum_{i=1}^{n} (\alpha_i + \gamma_i)\mathbf{E}(\epsilon_{t+d-i}^2|\Psi_t) - \sum_{j=1}^{p} \gamma_j \mathbf{E}(\eta_{t+d-j}|\Psi_t)$$

The d-step-ahead prediction error, $\xi_{t+d} = y_{t+d} - y_{t+d|t}$, has the conditional variance

$$\mathbf{V}(\xi_{t+d}|\Psi_t) = \sum_{j=0}^{d-1} g_j^2 \sigma_{t+d-j|t}^2$$

where

$$\sigma^2_{t+d-j|t} = \mathbf{E}(\epsilon^2_{t+d-j}|\Psi_t)$$

Coefficients in the conditional d-step prediction error variance are calculated recursively using the formula

$$g_j = -\varphi_1 g_{j-1} - \ldots - \varphi_m g_{j-m}$$

where $g_0 = 1$ and $g_j = 0$ if $j < 0$; $\varphi_1, \ldots, \varphi_m$ are autoregressive parameters. Since the parameters are not known, the conditional variance is computed using the estimated autoregressive parameters. The d-step-ahead prediction error variance is simplified when there are no autoregressive terms:

$$\mathbf{V}(\xi_{t+d}|\Psi_t) = \sigma^2_{t+d|t}$$

Therefore, the one-step-ahead prediction error variance is equivalent to the conditional error variance defined in the GARCH process:

$$h_t = \mathbf{E}(\epsilon^2_t|\Psi_{t-1}) = \sigma^2_{t|t-1}$$

The multistep forecast of conditional error variance of the EGARCH, QGARCH, TGARCH, PGARCH, and GARCH-M models cannot be calculated using the preceding formula for the GARCH model. The following formulas are recursively implemented to obtain the multistep forecast of conditional error variance of these models:

- for the EGARCH(p, q) model:

$$\ln(\sigma^2_{t+d|t}) = \omega + \sum_{i=d}^{q} \alpha_i g(z_{t+d-i}) + \sum_{j=1}^{d-1} \gamma_j \ln(\sigma^2_{t+d-j|t}) + \sum_{j=d}^{p} \gamma_j \ln(h_{t+d-j})$$

  where

$$g(z_t) = \theta z_t + |z_t| - E|z_t|$$

$$z_t = \epsilon_t / \sqrt{h_t}$$

- for the QGARCH(p, q) model:

$$\begin{aligned}\sigma^2_{t+d|t} = \omega &+ \sum_{i=1}^{d-1} \alpha_i (\sigma^2_{t+d-i|t} + \psi_i^2) + \sum_{i=d}^{q} \alpha_i (\epsilon_{t+d-i} - \psi_i)^2 \\ &+ \sum_{j=1}^{d-1} \gamma_j \sigma^2_{t+d-j|t} + \sum_{j=d}^{p} \gamma_j h_{t+d-j}\end{aligned}$$

- for the TGARCH(p, q) model:

$$\begin{aligned}\sigma^2_{t+d|t} = \omega &+ \sum_{i=1}^{d-1} (\alpha_i + \psi_i/2)\sigma^2_{t+d-i|t} + \sum_{i=d}^{q} (\alpha_i + 1_{\epsilon_{t+d-i}<0}\psi_i)\epsilon^2_{t+d-i} \\ &+ \sum_{j=1}^{d-1} \gamma_j \sigma^2_{t+d-j|t} + \sum_{j=d}^{p} \gamma_j h_{t+d-j}\end{aligned}$$

- for the PGARCH(p, q) model:

$$(\sigma^2_{t+d|t})^\lambda = \omega + \sum_{i=1}^{d-1} \alpha_i((1+\psi_i)^{2\lambda} + (1-\psi_i)^{2\lambda})(\sigma^2_{t+d-i|t})^\lambda/2 + \sum_{i=d}^{q} \alpha_i(|\epsilon_{t+d-i}| - \psi_i\epsilon_{t+d-i})^{2\lambda} + \sum_{j=1}^{d-1} \gamma_j(\sigma^2_{t+d-j|t})^\lambda + \sum_{j=d}^{p} \gamma_j h^\lambda_{t+d-j}$$

- for the GARCH-M model: ignoring the mean effect and directly using the formula of the corresponding GARCH model.

If the conditional error variance is homoscedastic, the conditional prediction error variance is identical to the unconditional prediction error variance

$$\mathbf{V}(\xi_{t+d}|\Psi_t) = \mathbf{V}(\xi_{t+d}) = \sigma^2 \sum_{j=0}^{d-1} g_j^2$$

since $\sigma^2_{t+d-j|t} = \sigma^2$. You can compute $s^2 r$ (which is the second term of the variance for the predicted value $\tilde{y}_t$ explained in the section "Predicting Future Series Realizations" on page 428) by using the formula $\sigma^2 \sum_{j=0}^{d-1} g_j^2$, and r is estimated from $\sum_{j=0}^{d-1} g_j^2$ by using the estimated autoregressive parameters.

Consider the following conditional prediction error variance:

$$\mathbf{V}(\xi_{t+d}|\Psi_t) = \sigma^2 \sum_{j=0}^{d-1} g_j^2 + \sum_{j=0}^{d-1} g_j^2(\sigma^2_{t+d-j|t} - \sigma^2)$$

The second term in the preceding equation can be interpreted as the noise from using the homoscedastic conditional variance when the errors follow the GARCH process. However, it is expected that if the GARCH process is covariance stationary, the difference between the conditional prediction error variance and the unconditional prediction error variance disappears as the forecast horizon d increases.

OUT= Data Set

The output SAS data set produced by the OUTPUT statement contains all the variables in the input data set and the new variables specified by the OUTPUT statement options. See the section "OUTPUT Statement" on page 382 earlier in this chapter for information on the output variables that can be created. The output data set contains one observation for each observation in the input data set.

OUTEST= Data Set

The OUTEST= data set contains all the variables used in any MODEL statement. Each regressor variable contains the estimate for the corresponding regression parameter in the corresponding model. In addition, the OUTEST= data set contains the following variables:

_A_*i*	the *i*th order autoregressive parameter estimate. There are *m* such variables _A_1 through _A_*m*, where *m* is the value of the NLAG= option.
_AH_*i*	the *i*th order ARCH parameter estimate, if the GARCH= option is specified. There are *q* such variables _AH_1 through _AH_*q*, where *q* is the value of the Q= option. The variable _AH_0 contains the estimate of ω.
_AHP_*i*	the estimate of the ψ_i parameter in the PGARCH model, if a PGARCH model is specified. There are *q* such variables _AHP_1 through _AHP_*q*, where *q* is the value of the Q= option.
_AHQ_*i*	the estimate of the ψ_i parameter in the QGARCH model, if a QGARCH model is specified. There are *q* such variables _AHQ_1 through _AHQ_*q*, where *q* is the value of the Q= option.
_AHT_*i*	the estimate of the ψ_i parameter in the TGARCH model, if a TGARCH model is specified. There are *q* such variables _AHT_1 through _AHT_*q*, where *q* is the value of the Q= option.
DELTA	the estimated mean parameter for the GARCH-M model if a GARCH-in-mean model is specified
DEPVAR	the name of the dependent variable
_GH_*i*	the *i*th order GARCH parameter estimate, if the GARCH= option is specified. There are *p* such variables _GH_1 through _GH_*p*, where *p* is the value of the P= option.
_HET_*i*	the *i*th heteroscedasticity model parameter specified by the HETERO statement
INTERCEPT	the intercept estimate. INTERCEPT contains a missing value for models for which the NOINT option is specified.
METHOD	the estimation method that is specified in the METHOD= option
MODEL	the label of the MODEL statement if one is given, or blank otherwise
MSE	the value of the mean square error for the model
NAME	the name of the row of covariance matrix for the parameter estimate, if the COVOUT option is specified
LAMBDA	the estimate of the power parameter λ in the PGARCH model, if a PGARCH model is specified.
LIKLHD	the log-likelihood value of the GARCH model
SSE	the value of the error sum of squares
START	the estimated start-up value for the conditional variance when GARCH= (STARTUP=ESTIMATE) option is specified
STATUS	This variable indicates the optimization status. _STATUS_ = 0 indicates that there were no errors during the optimization and the algorithm converged. _STATUS_ = 1 indicates that the optimization could not improve the function value and means that the results should be interpreted with caution. _STATUS_ = 2 indicates that the optimization failed due to the number of iterations exceeding either the maximum default or the specified number of iterations or the number of function calls allowed. _STATUS_ = 3 indicates that an error occurred during the optimization process. For example, this error message is obtained when a function or its derivatives cannot be calculated at the initial values or

during the iteration process, when an optimization step is outside of the feasible region or when active constraints are linearly dependent.

STDERR standard error of the parameter estimate, if the COVOUT option is specified.

TDFI the estimate of the inverted degrees of freedom for Student's t distribution, if DIST=T is specified.

THETA the estimate of the θ parameter in the EGARCH model, if an EGARCH model is specified.

TYPE PARM for observations containing parameter estimates, or COV for observations containing covariance matrix elements.

The OUTEST= data set contains one observation for each MODEL statement giving the parameter estimates for that model. If the COVOUT option is specified, the OUTEST= data set includes additional observations for each MODEL statement giving the rows of the covariance of parameter estimates matrix. For covariance observations, the value of the _TYPE_ variable is COV, and the _NAME_ variable identifies the parameter associated with that row of the covariance matrix.

Printed Output

The AUTOREG procedure prints the following items:

1. the name of the dependent variable
2. the ordinary least squares estimates
3. Estimates of autocorrelations, which include the estimates of the autocovariances, the autocorrelations, and (if there is sufficient space) a graph of the autocorrelation at each LAG
4. if the PARTIAL option is specified, the partial autocorrelations
5. the preliminary MSE, which results from solving the Yule-Walker equations. This is an estimate of the final MSE.
6. the estimates of the autoregressive parameters (Coefficient), their standard errors (Standard Error), and the ratio of estimate to standard error (t Value)
7. the statistics of fit for the final model. These include the error sum of squares (SSE), the degrees of freedom for error (DFE), the mean square error (MSE), the mean absolute error (MAE), the mean absolute percentage error (MAPE), the root mean square error (Root MSE), the Schwarz information criterion (SBC), the Hannan-Quinn information criterion (HQC), the Akaike information criterion (AIC), the corrected Akaike information criterion (AICC), the Durbin-Watson statistic (Durbin-Watson), the transformed regression R^2 (Transformed Regress R-Square), and the total R^2 (Total R-Square). For GARCH models, the following additional items are printed:
   - the value of the log-likelihood function (Log Likelihood)
   - the number of observations that are used in estimation (Observations)
   - the unconditional variance (Uncond Var)
   - the normality test statistic and its p-value (Normality Test and Pr > ChiSq)

8. the parameter estimates for the structural model (Estimate), a standard error estimate (Standard Error), the ratio of estimate to standard error (t Value), and an approximation to the significance probability for the parameter being 0 (Approx Pr > |t|)

9. If the NLAG= option is specified with METHOD=ULS or METHOD=ML, the regression parameter estimates are printed again, assuming that the autoregressive parameter estimates are known. In this case, the Standard Error and related statistics for the regression estimates will, in general, be different from the case when they are estimated. Note that from a standpoint of estimation, Yule-Walker and iterated Yule-Walker methods (NLAG= with METHOD=YW, ITYW) generate only one table, assuming AR parameters are given.

10. If you specify the NORMAL option, the Bera-Jarque normality test statistics are printed. If you specify the LAGDEP option, Durbin's *h* or Durbin's *t* is printed.

ODS Table Names

PROC AUTOREG assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in the Table 9.6.

Table 9.6 ODS Tables Produced in PROC AUTOREG

ODS Table Name	Description	Option
ODS Tables Created by the MODEL Statement		
ClassLevels	Class Levels	default
FitSummary	Summary of regression	default
SummaryDepVarCen	Summary of regression (centered dependent var)	CENTER
SummaryNoIntercept	Summary of regression (no intercept)	NOINT
YWIterSSE	Yule-Walker iteration sum of squared error	METHOD=ITYW
PreMSE	Preliminary MSE	NLAG=
Dependent	Dependent variable	default
DependenceEquations	Linear dependence equation	
ARCHTest	Tests for ARCH disturbances based on OLS residuals	ARCHTEST=
ARCHTestAR	Tests for ARCH disturbances based on residuals	ARCHTEST= (with NLAG=)
BDSTest	BDS test for independence	BDS<=()>
RunsTest	Runs test for independence	RUNS<=()>
TurningPointTest	Turning point test for independence	TP<=()>

Table 9.6 *continued*

ODS Table Name	Description	Option
VNRRankTest	Rank version of von Neumann ratio test for independence	VNRRANK<=()>
FitSummarySCBP	Fit summary of Bai and Perron's multiple structural change models	BP=
BreakDatesSCBP	Break dates of Bai and Perron's multiple structural change models	BP=
SupFSCBP	supF tests of Bai and Perron's multiple structural change models	BP=
UDmaxFSCBP	UDmaxF test of Bai and Perron's multiple structural change models	BP=
WDmaxFSCBP	WDmaxF tests of Bai and Perron's multiple structural change models	BP=
SeqFSCBP	supF(l+1|l) tests of Bai and Perron's multiple structural change models	BP=
ParameterEstimatesSCBP	Parameter estimates of Bai and Perron's multiple structural change models	BP=
ChowTest	Chow test and predictive Chow test	CHOW= PCHOW=
Godfrey	Godfrey's serial correlation test	GODFREY<=>
PhilPerron	Phillips-Perron unit root test	STATIONARITY= (PHILIPS<=()>) (no regressor)
PhilOul	Phillips-Ouliaris cointegration test	STATIONARITY= (PHILIPS<=()>) (has regressor)
ADF	Augmented Dickey-Fuller unit root test	STATIONARITY= (ADF<=()>) (no regressor)
EngleGranger	Engle-Granger cointegration test	STATIONARITY= (ADF<=()>) (has regressor)
ERS	ERS unit root test	STATIONARITY= (ERS<=()>)
NgPerron	Ng-Perron Unit root tests	STATIONARITY= (NP=<()>)
KPSS	Kwiatkowski, Phillips, Schmidt, and Shin (KPSS) test or Shin cointegration test	STATIONARITY= (KPSS<=()>)
ResetTest	Ramsey's RESET test	RESET

Table 9.6 *continued*

ODS Table Name	Description	Option
ARParameterEstimates	Estimates of autoregressive parameters	NLAG=
CorrGraph	Estimates of autocorrelations	NLAG=
BackStep	Backward elimination of autoregressive terms	BACKSTEP
ExpAutocorr	Expected autocorrelations	NLAG=
IterHistory	Iteration history	ITPRINT
ParameterEstimates	Parameter estimates	default
ParameterEstimatesGivenAR	Parameter estimates assuming AR parameters are given	NLAG=, METHOD= ULS | ML
PartialAutoCorr	Partial autocorrelation	PARTIAL
CovB	Covariance of parameter estimates	COVB
CorrB	Correlation of parameter estimates	CORRB
CholeskyFactor	Cholesky root of gamma	ALL
Coefficients	Coefficients for first NLAG observations	COEF
GammaInverse	Gamma inverse	GINV
ConvergenceStatus	Convergence status table	default
MiscStat	Durbin *t* or Durbin *h*, Bera-Jarque normality test	LAGDEP=; NORMAL
DWTest	Durbin-Watson statistics	DW=
ODS Tables Created by the RESTRICT Statement		
Restrict	Restriction table	default
ODS Tables Created by the TEST Statement		
FTest	*F* test	default, TYPE=ALL
WaldTest	Wald test	TYPE=WALD|ALL
LMTest	LM test	TYPE=LM|ALL (only supported with GARCH= option)
LRTest	LR test	TYPE=LR|ALL (only supported with GARCH= option)

ODS Graphics

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*).

Before you create graphs, ODS Graphics must be enabled (for example, with the ODS GRAPHICS ON statement). For more information about enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" in that chapter.

The overall appearance of graphs is controlled by ODS styles. Styles and other aspects of using ODS Graphics are discussed in the section "A Primer on ODS Statistical Graphics" in that chapter.

This section describes the use of ODS for creating graphics with the AUTOREG procedure.

To request these graphs, you must specify the ODS GRAPHICS statement. By default, only the residual, predicted versus actual, and autocorrelation of residuals plots are produced. If, in addition to the ODS GRAPHICS statement, you also specify the ALL option in either the PROC AUTOREG statement or MODEL statement, all plots are created. For HETERO, GARCH, and AR models studentized residuals are replaced by standardized residuals. For the autoregressive models, the conditional variance of the residuals is computed as described in the section "Predicting Future Series Realizations" on page 428. For the GARCH and HETERO models, residuals are assumed to have h_t conditional variance invoked by the HT= option of the OUTPUT statement. For all these cases, the Cook's D plot is not produced.

ODS Graph Names

PROC AUTOREG assigns a name to each graph it creates using ODS. You can use these names to reference the graphs when using ODS. The names are listed in Table 9.7.

Table 9.7 ODS Graphics Produced in PROC AUTOREG

ODS Table Name	Description	Plots= Option
DiagnosticsPanel	All applicable plots	
ACFPlot	Autocorrelation of residuals	ACF
FitPlot	Predicted versus actual plot	FITPLOT, default
CooksD	Cook's D plot	COOKSD (no NLAG=)
IACFPlot	Inverse autocorrelation of residuals	IACF
QQPlot	Q-Q plot of residuals	QQ
PACFPlot	Partial autocorrelation of residuals	PACF
ResidualHistogram	Histogram of the residuals	RESIDUALHISTOGRAM or RESIDHISTOGRAM
ResidualPlot	Residual plot	RESIDUAL or RES, default
StudentResidualPlot	Studentized residual plot	STUDENTRESIDUAL (no NLAG=, GARCH=, or HETERO)
StandardResidualPlot	Standardized residual plot	STANDARDRESIDUAL
WhiteNoiseLogProbPlot	Tests for white noise residuals	WHITENOISE

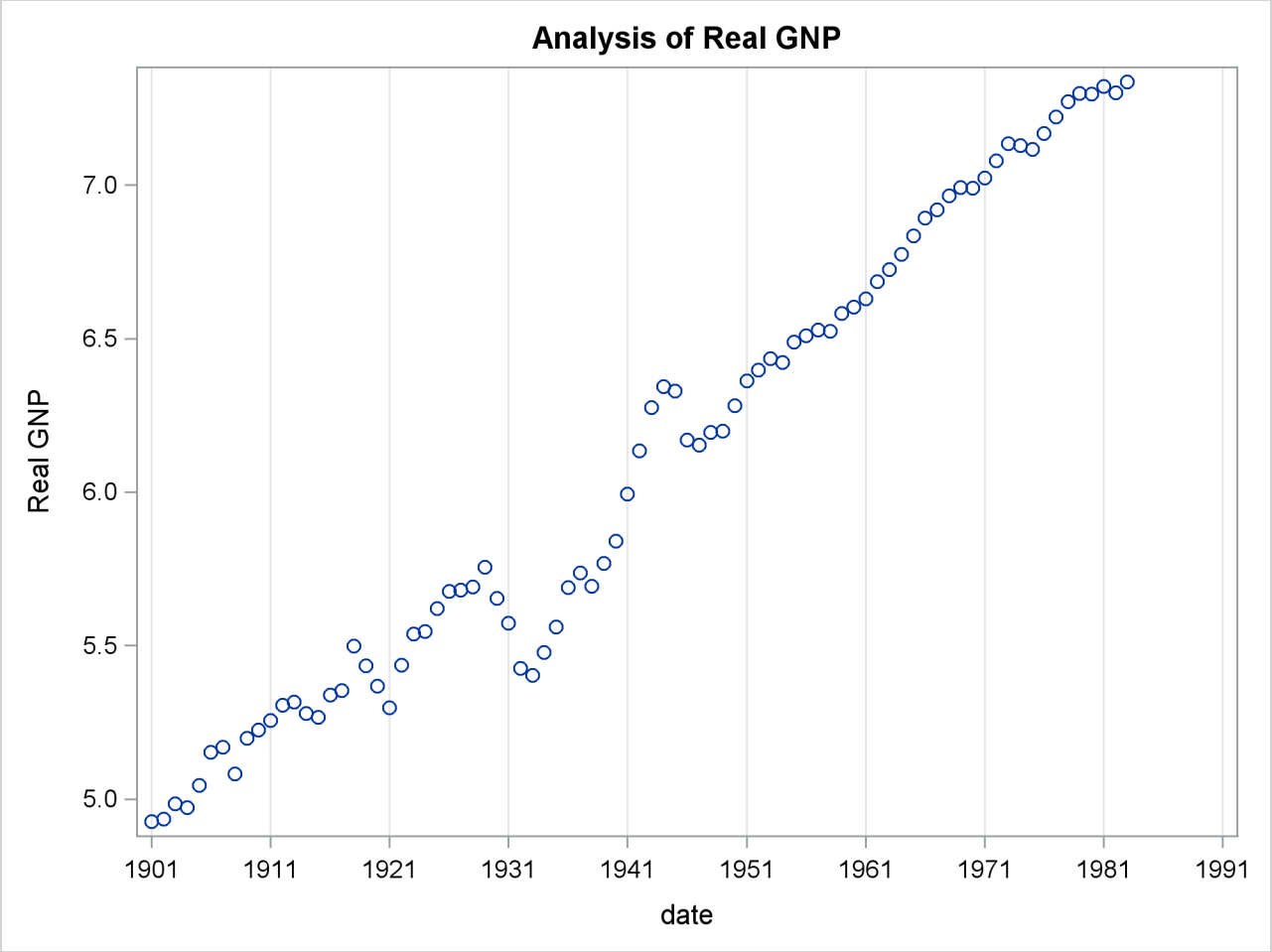
Examples: AUTOREG Procedure

Example 9.1: Analysis of Real Output Series

In this example, the annual real output series is analyzed over the period 1901 to 1983 (Balke and Gordon 1986, pp. 581–583). With the following DATA step, the original data are transformed using the natural logarithm, and the differenced series DY is created for further analysis. The log of real output is plotted in Output 9.1.1.

```
title 'Analysis of Real GNP';
data gnp;
   date = intnx( 'year', '01jan1901'd, _n_-1 );
   format date year4.;
   input x @@;
   y  = log(x);
   dy = dif(y);
   t  = _n_;
   label y  = 'Real GNP'
         dy = 'First Difference of Y'
         t  = 'Time Trend';
datalines;
  137.87  139.13  146.10  144.21  155.04  172.97  175.61  161.22

   ... more lines ...


proc sgplot data=gnp noautolegend;
   scatter x=date y=y;
   xaxis grid values=('01jan1901'd '01jan1911'd '01jan1921'd '01jan1931'd
             '01jan1941'd '01jan1951'd '01jan1961'd '01jan1971'd
             '01jan1981'd '01jan1991'd);
run;
```

Output 9.1.1 Real Output Series: 1901 – 1983



The (linear) trend-stationary process is estimated using the following form:

$$y_t = \beta_0 + \beta_1 t + \nu_t$$

where

$$\nu_t = \epsilon_t - \varphi_1 \nu_{t-1} - \varphi_2 \nu_{t-2}$$

$$\epsilon_t \sim \mathrm{IN}(0, \sigma_\epsilon)$$

The preceding trend-stationary model assumes that uncertainty over future horizons is bounded since the error term, ν_t, has a finite variance. The maximum likelihood AR estimates from the statements that follow are shown in Output 9.1.2:

```
proc autoreg data=gnp;
   model y = t / nlag=2 method=ml;
run;
```

Output 9.1.2 Estimating the Linear Trend Model

Analysis of Real GNP

The AUTOREG Procedure

Maximum Likelihood Estimates			
SSE	0.23954331	**DFE**	79
MSE	0.00303	**Root MSE**	0.05507
SBC	-230.39355	**AIC**	-240.06891
MAE	0.04016596	**AICC**	-239.55609
MAPE	0.69458594	**HQC**	-236.18189
Log Likelihood	124.034454	**Transformed Regression R-Square**	0.8645
Durbin-Watson	1.9935	**Total R-Square**	0.9947
		Observations	83

Parameter Estimates						
Variable	**DF**	**Estimate**	**Standard Error**	**t Value**	**Approx Pr > |t|**	**Variable Label**
Intercept	1	4.8206	0.0661	72.88	<.0001	
t	1	0.0302	0.001346	22.45	<.0001	Time Trend
AR1	1	-1.2041	0.1040	-11.58	<.0001	
AR2	1	0.3748	0.1039	3.61	0.0005	

Autoregressive parameters assumed given						
Variable	**DF**	**Estimate**	**Standard Error**	**t Value**	**Approx Pr > |t|**	**Variable Label**
Intercept	1	4.8206	0.0661	72.88	<.0001	
t	1	0.0302	0.001346	22.45	<.0001	Time Trend

Nelson and Plosser (1982) failed to reject the hypothesis that macroeconomic time series are nonstationary and have no tendency to return to a trend line. In this context, the simple random walk process can be used as an alternative process:

$$y_t = \beta_0 + y_{t-1} + v_t$$

where $v_t = \epsilon_t$ and $y_0 = 0$. In general, the difference-stationary process is written as

$$\phi(L)(1-L)y_t = \beta_0\phi(1) + \theta(L)\epsilon_t$$

where L is the lag operator. You can observe that the class of a difference-stationary process should have at least one unit root in the AR polynomial $\phi(L)(1-L)$.

The Dickey-Fuller procedure is used to test the null hypothesis that the series has a unit root in the AR polynomial. Consider the following equation for the augmented Dickey-Fuller test:

$$\Delta y_t = \beta_0 + \delta t + \beta_1 y_{t-1} + \sum_{i=1}^{m} \gamma_i \Delta y_{t-i} + \epsilon_t$$

where $\Delta = 1 - L$. The test statistic τ_τ is the usual t ratio for the parameter estimate $\hat{\beta}_1$, but the τ_τ does not follow a t distribution.

The following code performs the augmented Dickey-Fuller test with $m = 3$ and we are interesting in the test results in the linear time trend case since the previous plot reveals there is a linear trend.

```
proc autoreg data = gnp;
   model y = / stationarity =(adf =3);
run;
```

The augmented Dickey-Fuller test indicates that the output series may have a difference-stationary process. The statistic Tau with linear time trend has a value of -2.6190 and its p-value is 0.2732. The statistic Rho has a p-value of 0.0817 which also indicates the null of unit root is accepted at the 5% level. (See Output 9.1.3.)

Output 9.1.3 Augmented Dickey-Fuller Test Results

Analysis of Real GNP

The AUTOREG Procedure

Augmented Dickey-Fuller Unit Root Tests							
Type	**Lags**	**Rho**	**Pr < Rho**	**Tau**	**Pr < Tau**	**F**	**Pr > F**
Zero Mean	**3**	0.3827	0.7732	3.3342	0.9997		
Single Mean	**3**	-0.1674	0.9465	-0.2046	0.9326	5.7521	0.0211
Trend	**3**	-18.0246	0.0817	-2.6190	0.2732	3.4472	0.4957

The AR(1) model for the differenced series DY is estimated using the maximum likelihood method for the period 1902 to 1983. The difference-stationary process is written

$$\Delta y_t = \beta_0 + \nu_t$$

$$\nu_t = \epsilon_t - \varphi_1 \nu_{t-1}$$

The estimated value of φ_1 is -0.297 and that of β_0 is 0.0293. All estimated values are statistically significant. The PROC step follows:

```
proc autoreg data=gnp;
   model dy = / nlag=1 method=ml;
run;
```

The printed output produced by the PROC step is shown in Output 9.1.4.

Output 9.1.4 Estimating the Differenced Series with AR(1) Error

Analysis of Real GNP

The AUTOREG Procedure

Maximum Likelihood Estimates			
SSE	0.27107673	**DFE**	80
MSE	0.00339	**Root MSE**	0.05821
SBC	-226.77848	**AIC**	-231.59192
MAE	0.04333026	**AICC**	-231.44002
MAPE	153.637587	**HQC**	-229.65939
Log Likelihood	117.795958	**Transformed Regression R-Square**	0.0000
Durbin-Watson	1.9268	**Total R-Square**	0.0900
		Observations	82

Output 9.1.4 *continued*

Parameter Estimates

Variable	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	0.0293	0.009093	3.22	0.0018
AR1	1	-0.2967	0.1067	-2.78	0.0067

Autoregressive parameters assumed given

Variable	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	0.0293	0.009093	3.22	0.0018

Example 9.2: Comparing Estimates and Models

In this example, the Grunfeld series are estimated using different estimation methods. Refer to Maddala (1977) for details of the Grunfeld investment data set. For comparison, the Yule-Walker method, ULS method, and maximum likelihood method estimates are shown. With the DWPROB option, the *p*-value of the Durbin-Watson statistic is printed. The Durbin-Watson test indicates the positive autocorrelation of the regression residuals. The DATA and PROC steps follow:

```
title 'Grunfeld''s Investment Models Fit with Autoregressive Errors';
data grunfeld;
   input year gei gef gec;
   label gei = 'Gross investment GE'
         gec = 'Lagged Capital Stock GE'
         gef = 'Lagged Value of GE shares';
datalines;
1935     33.1      1170.6     97.8

   ... more lines ...


proc autoreg data=grunfeld;
   model gei = gef gec / nlag=1 dwprob;
   model gei = gef gec / nlag=1 method=uls;
   model gei = gef gec / nlag=1 method=ml;
run;
```

The printed output produced by each of the MODEL statements is shown in Output 9.2.1 through Output 9.2.4.

Output 9.2.1 OLS Analysis of Residuals

Grunfeld's Investment Models Fit with Autoregressive Errors

The AUTOREG Procedure

Dependent Variable	gei
	Gross investment GE

Output 9.2.1 *continued*

Ordinary Least Squares Estimates			
SSE	13216.5878	**DFE**	17
MSE	777.44634	**Root MSE**	27.88272
SBC	195.614652	**AIC**	192.627455
MAE	19.9433255	**AICC**	194.127455
MAPE	23.2047973	**HQC**	193.210587
Durbin-Watson	1.0721	**Total R-Square**	0.7053

Parameter Estimates

Variable	DF	Estimate	Standard Error	t Value	Approx Pr > |t|	Variable Label
Intercept	1	-9.9563	31.3742	-0.32	0.7548	
gef	1	0.0266	0.0156	1.71	0.1063	Lagged Value of GE shares
gec	1	0.1517	0.0257	5.90	<.0001	Lagged Capital Stock GE

Estimates of Autocorrelations

Lag	Covariance	Correlation	−1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1
0	660.8	1.000000	|********************|
1	304.6	0.460867	|********* |

Preliminary MSE	520.5

Output 9.2.2 Regression Results Using Default Yule-Walker Method

Estimates of Autoregressive Parameters

Lag	Coefficient	Standard Error	t Value
1	-0.460867	0.221867	-2.08

Yule-Walker Estimates			
SSE	10238.2951	**DFE**	16
MSE	639.89344	**Root MSE**	25.29612
SBC	193.742396	**AIC**	189.759467
MAE	18.0715195	**AICC**	192.426133
MAPE	21.0772644	**HQC**	190.536976
Durbin-Watson	1.3321	**Transformed Regression R-Square**	0.5717
		Total R-Square	0.7717

Parameter Estimates

Variable	DF	Estimate	Standard Error	t Value	Approx Pr > |t|	Variable Label
Intercept	1	-18.2318	33.2511	-0.55	0.5911	
gef	1	0.0332	0.0158	2.10	0.0523	Lagged Value of GE shares
gec	1	0.1392	0.0383	3.63	0.0022	Lagged Capital Stock GE

Output 9.2.3 Regression Results Using Unconditional Least Squares Method

Estimates of Autoregressive Parameters

Lag	Coefficient	Standard Error	t Value
1	-0.460867	0.221867	-2.08

Algorithm converged.

Unconditional Least Squares Estimates

SSE	10220.8455	**DFE**	16
MSE	638.80284	**Root MSE**	25.27455
SBC	193.756692	**AIC**	189.773763
MAE	18.1317764	**AICC**	192.44043
MAPE	21.149176	**HQC**	190.551273
Durbin-Watson	1.3523	**Transformed Regression R-Square**	0.5511
		Total R-Square	0.7721

Parameter Estimates

Variable	DF	Estimate	Standard Error	t Value	Approx Pr > |t|	Variable Label
Intercept	1	-18.6582	34.8101	-0.54	0.5993	
gef	1	0.0339	0.0179	1.89	0.0769	Lagged Value of GE shares
gec	1	0.1369	0.0449	3.05	0.0076	Lagged Capital Stock GE
AR1	1	-0.4996	0.2592	-1.93	0.0718	

Autoregressive parameters assumed given

Variable	DF	Estimate	Standard Error	t Value	Approx Pr > |t|	Variable Label
Intercept	1	-18.6582	33.7567	-0.55	0.5881	
gef	1	0.0339	0.0159	2.13	0.0486	Lagged Value of GE shares
gec	1	0.1369	0.0404	3.39	0.0037	Lagged Capital Stock GE

Output 9.2.4 Regression Results Using Maximum Likelihood Method

Estimates of Autoregressive Parameters

Lag	Coefficient	Standard Error	t Value
1	-0.460867	0.221867	-2.08

Algorithm converged.

Maximum Likelihood Estimates

SSE	10229.2303	**DFE**	16
MSE	639.32689	**Root MSE**	25.28491
SBC	193.738877	**AIC**	189.755947
MAE	18.0892426	**AICC**	192.422614
MAPE	21.0978407	**HQC**	190.533457
Log Likelihood	-90.877974	**Transformed Regression R-Square**	0.5656
Durbin-Watson	1.3385	**Total R-Square**	0.7719
		Observations	20

Parameter Estimates

Variable	DF	Estimate	Standard Error	t Value	Approx Pr > |t|	Variable Label
Intercept	1	-18.3751	34.5941	-0.53	0.6026	
gef	1	0.0334	0.0179	1.87	0.0799	Lagged Value of GE shares
gec	1	0.1385	0.0428	3.23	0.0052	Lagged Capital Stock GE
AR1	1	-0.4728	0.2582	-1.83	0.0858	

Autoregressive parameters assumed given

Variable	DF	Estimate	Standard Error	t Value	Approx Pr > |t|	Variable Label
Intercept	1	-18.3751	33.3931	-0.55	0.5897	
gef	1	0.0334	0.0158	2.11	0.0512	Lagged Value of GE shares
gec	1	0.1385	0.0389	3.56	0.0026	Lagged Capital Stock GE

Example 9.3: Lack-of-Fit Study

Many time series exhibit high positive autocorrelation, having the smooth appearance of a random walk. This behavior can be explained by the partial adjustment and adaptive expectation hypotheses.

Short-term forecasting applications often use autoregressive models because these models absorb the behavior of this kind of data. In the case of a first-order AR process where the autoregressive parameter is exactly 1 (a *random walk*), the best prediction of the future is the immediate past.

PROC AUTOREG can often greatly improve the fit of models, not only by adding additional parameters but also by capturing the random walk tendencies. Thus, PROC AUTOREG can be expected to provide good short-term forecast predictions.

However, good forecasts do not necessarily mean that your structural model contributes anything worthwhile to the fit. In the following example, random noise is fit to part of a sine wave. Notice that the structural model does not fit at all, but the autoregressive process does quite well and is very nearly a first difference (AR(1) = −.976). The DATA step, PROC AUTOREG step, and PROC SGPLOT step follow:

```
title1 'Lack of Fit Study';
title2 'Fitting White Noise Plus Autoregressive Errors to a Sine Wave';

data a;
   pi=3.14159;
   do time = 1 to 75;
      if time > 75 then y = .;
      else y = sin( pi * ( time / 50 ) );
      x = ranuni( 1234567 );
      output;
   end;
run;

proc autoreg data=a plots;
   model y = x / nlag=1;
   output out=b p=pred pm=xbeta;
run;

proc sgplot data=b;
   scatter y=y x=time / markerattrs=(color=black);
   series y=pred x=time / lineattrs=(color=blue);
   series y=xbeta x=time / lineattrs=(color=red);
run;
```

The printed output produced by PROC AUTOREG is shown in Output 9.3.1 and Output 9.3.2. Plots of observed and predicted values are shown in Output 9.3.3 and Output 9.3.4. Note: the plot Output 9.3.3 can be viewed in the Autoreg.Model.FitDiagnosticPlots category by selecting **View►Results**.

Output 9.3.1 Results of OLS Analysis: No Autoregressive Model Fit

Lack of Fit Study
Fitting White Noise Plus Autoregressive Errors to a Sine Wave

The AUTOREG Procedure

Dependent Variable	y

Ordinary Least Squares Estimates			
SSE	34.8061005	**DFE**	73
MSE	0.47680	**Root MSE**	0.69050
SBC	163.898598	**AIC**	159.263622
MAE	0.59112447	**AICC**	159.430289
MAPE	117894.045	**HQC**	161.114317
Durbin-Watson	0.0057	**Total R-Square**	0.0008

Parameter Estimates

Variable	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	0.2383	0.1584	1.50	0.1367
x	1	-0.0665	0.2771	-0.24	0.8109

Estimates of Autocorrelations

Lag	Covariance	Correlation	-1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1
0	0.4641	1.000000	|**********|
1	0.4531	0.976386	|**********|

Preliminary MSE	0.0217

Output 9.3.2 Regression Results with AR(1) Error Correction

Estimates of Autoregressive Parameters

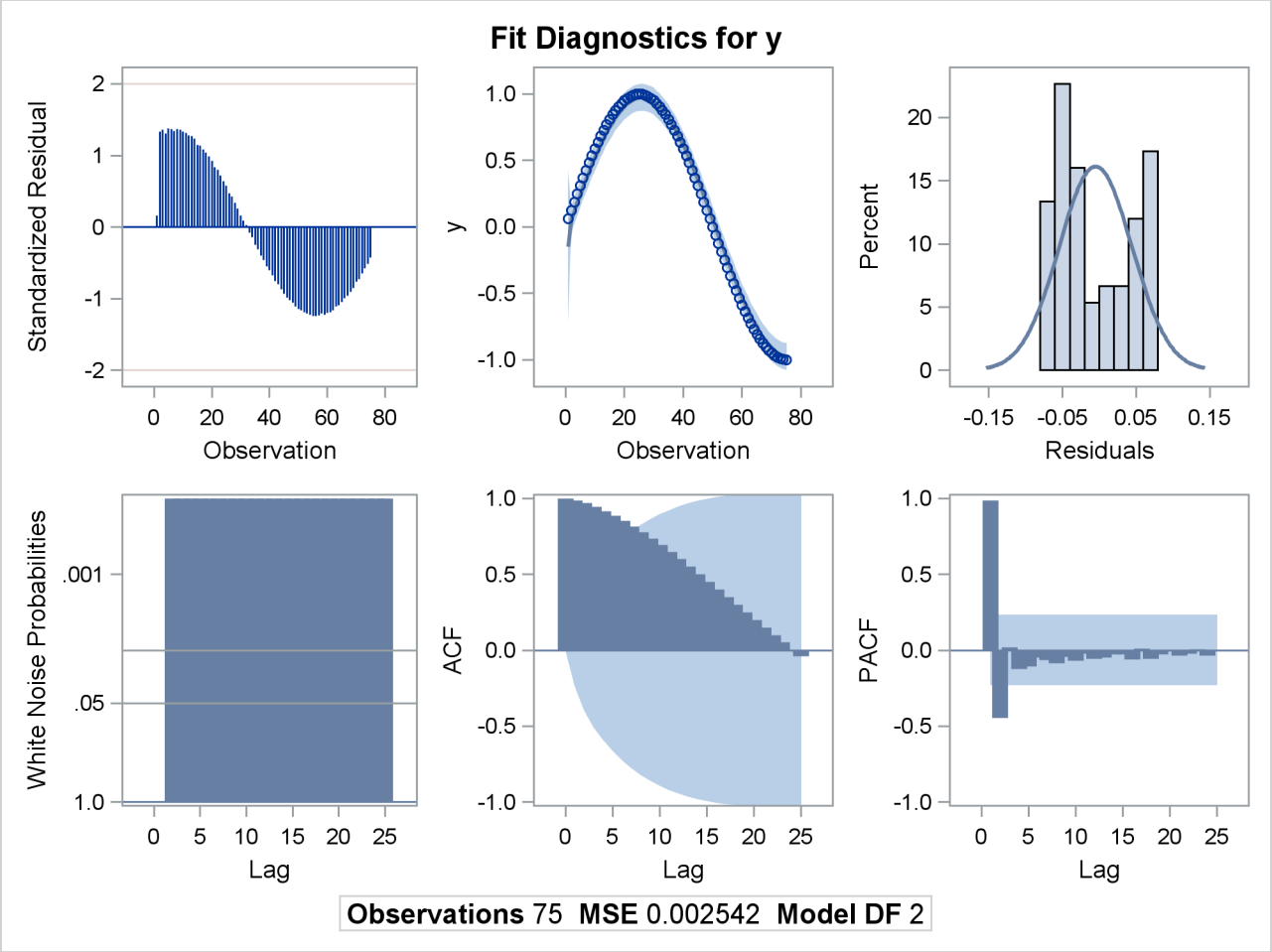
Lag	Coefficient	Standard Error	t Value
1	-0.976386	0.025460	-38.35

Yule-Walker Estimates			
SSE	0.18304264	**DFE**	72
MSE	0.00254	**Root MSE**	0.05042
SBC	-222.30643	**AIC**	-229.2589
MAE	0.04551667	**AICC**	-228.92087
MAPE	29145.3526	**HQC**	-226.48285
Durbin-Watson	0.0942	**Transformed Regression R-Square**	0.0001
		Total R-Square	0.9947

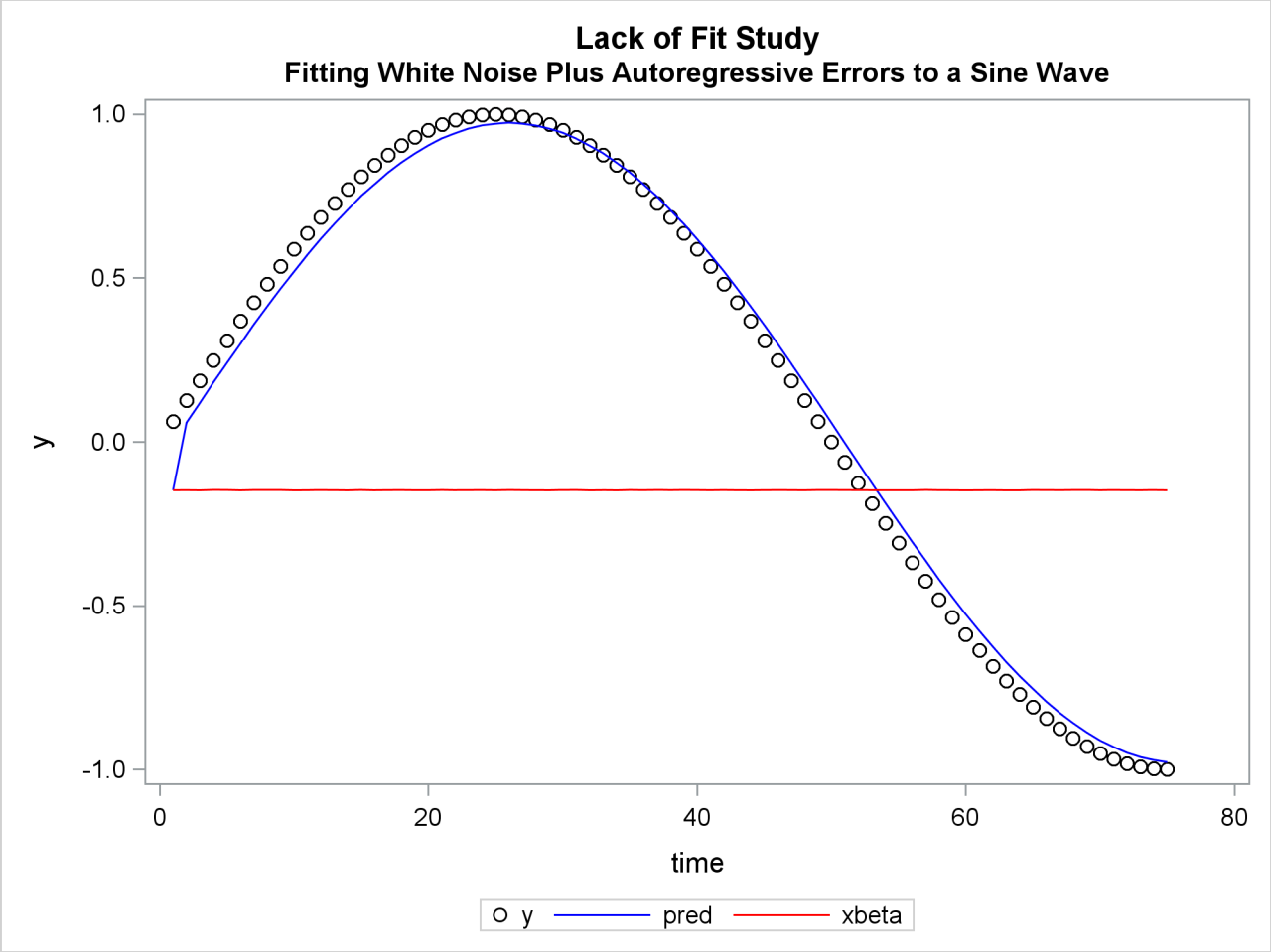
Parameter Estimates

Variable	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	-0.1473	0.1702	-0.87	0.3898
x	1	-0.001219	0.0141	-0.09	0.9315

Output 9.3.3 Diagnostics Plots



Output 9.3.4 Plot of Autoregressive Prediction



Example 9.4: Missing Values

In this example, a pure autoregressive error model with no regressors is used to generate 50 values of a time series. Approximately 15% of the values are randomly chosen and set to missing. The following statements generate the data:

```
title  'Simulated Time Series with Roots:';
title2 ' (X-1.25)(X**4-1.25)';
title3 'With 15% Missing Values';
data ar;
   do i=1 to 550;
      e = rannor(12345);
      n = sum( e, .8*n1, .8*n4, -.64*n5 );  /* ar process  */
      y = n;
      if ranuni(12345) > .85 then y = .;    /* 15% missing */
      n5=n4; n4=n3; n3=n2; n2=n1; n1=n;     /* set lags    */
      if i>500 then output;
   end;
run;
```

The model is estimated using maximum likelihood, and the residuals are plotted with 99% confidence limits. The PARTIAL option prints the partial autocorrelations. The following statements fit the model:

```
proc autoreg data=ar partial;
   model y = / nlag=(1 4 5) method=ml;
   output out=a predicted=p residual=r ucl=u lcl=l alphacli=.01;
run;
```

The printed output produced by the AUTOREG procedure is shown in Output 9.4.1 and Output 9.4.2. Note: the plot Output 9.4.2 can be viewed in the Autoreg.Model.FitDiagnosticPlots category by selecting **View►Results**.

Output 9.4.1 Autocorrelation-Corrected Regression Results

Simulated Time Series with Roots:
(X-1.25)(X4-1.25)**
With 15% Missing Values

The AUTOREG Procedure

Dependent Variable	y

Ordinary Least Squares Estimates			
SSE	182.972379	**DFE**	40
MSE	4.57431	**Root MSE**	2.13876
SBC	181.39282	**AIC**	179.679248
MAE	1.80469152	**AICC**	179.781813
MAPE	270.104379	**HQC**	180.303237
Durbin-Watson	1.3962	**Total R-Square**	0.0000

Output 9.4.1 *continued*

Parameter Estimates

Variable	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	-2.2387	0.3340	-6.70	<.0001

Estimates of Autocorrelations

Lag	Covariance	Correlation	−1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1
0	4.4627	1.000000	||********************|
1	1.4241	0.319109	||******|
2	1.6505	0.369829	||*******|
3	0.6808	0.152551	||***|
4	2.9167	0.653556	||*************|
5	-0.3816	-0.085519	|*||

Partial Autocorrelations

1	0.319109
4	0.619288
5	-0.821179

Preliminary MSE 0.7609

Estimates of Autoregressive Parameters

Lag	Coefficient	Standard Error	t Value
1	-0.733182	0.089966	-8.15
4	-0.803754	0.071849	-11.19
5	0.821179	0.093818	8.75

Expected Autocorrelations

Lag	Autocorr
0	1.0000
1	0.4204
2	0.2480
3	0.3160
4	0.6903
5	0.0228

Algorithm converged.

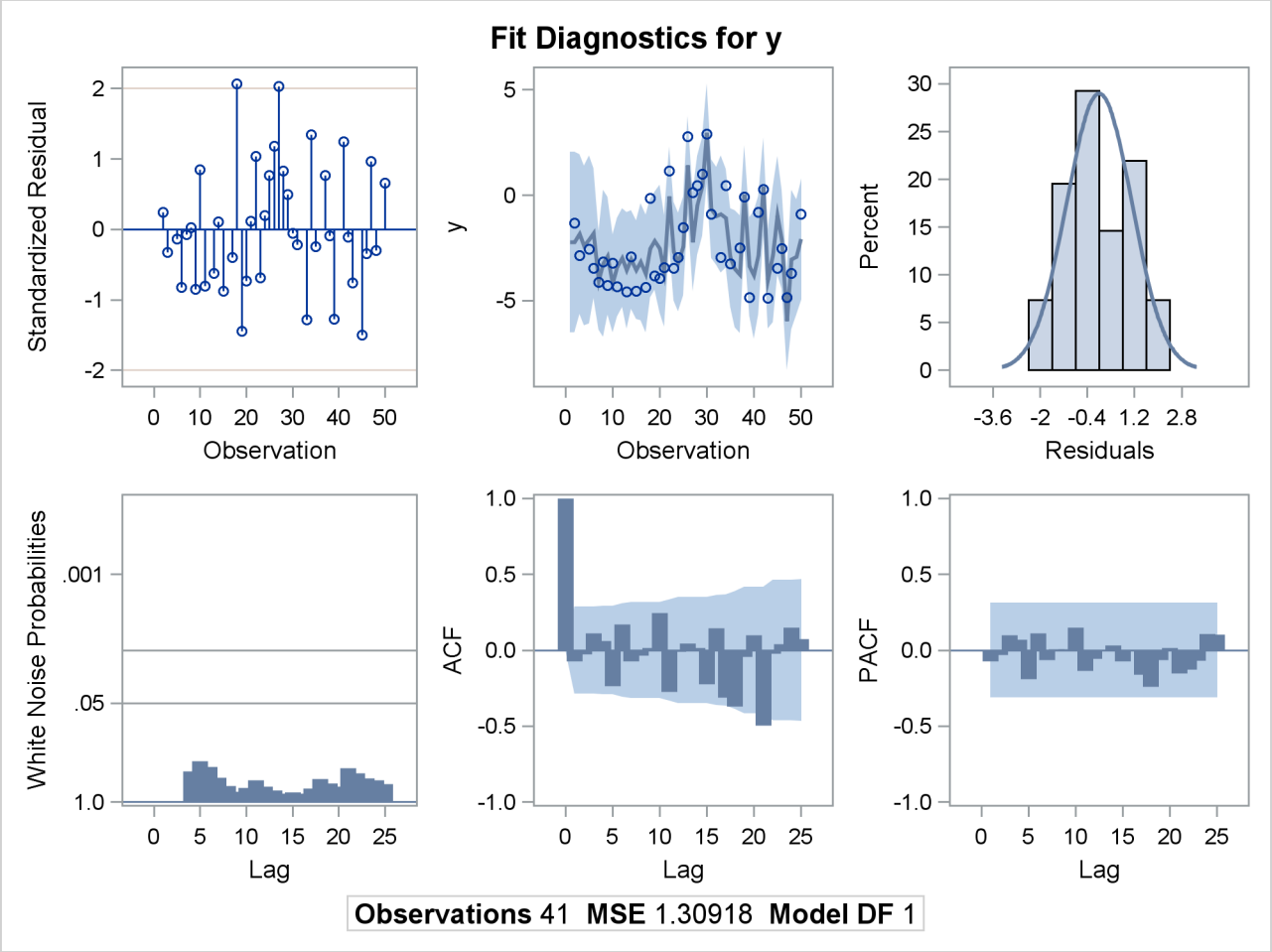
Maximum Likelihood Estimates			
SSE	48.4396756	**DFE**	37
MSE	1.30918	**Root MSE**	1.14419
SBC	146.879013	**AIC**	140.024725
MAE	0.88786192	**AICC**	141.135836
MAPE	141.377721	**HQC**	142.520679
Log Likelihood	-66.012362	**Transformed Regression R-Square**	0.0000
Durbin-Watson	2.9457	**Total R-Square**	0.7353
		Observations	41

Parameter Estimates					
Variable	**DF**	**Estimate**	**Standard Error**	**t Value**	**Approx Pr > |t|**
Intercept	1	-2.2370	0.5239	-4.27	0.0001
AR1	1	-0.6201	0.1129	-5.49	<.0001
AR4	1	-0.7237	0.0914	-7.92	<.0001
AR5	1	0.6550	0.1202	5.45	<.0001

Expected Autocorrelations	
Lag	**Autocorr**
0	1.0000
1	0.4204
2	0.2423
3	0.2958
4	0.6318
5	0.0411

Autoregressive parameters assumed given					
Variable	**DF**	**Estimate**	**Standard Error**	**t Value**	**Approx Pr > |t|**
Intercept	1	-2.2370	0.5225	-4.28	0.0001

Output 9.4.2 Diagnostic Plots

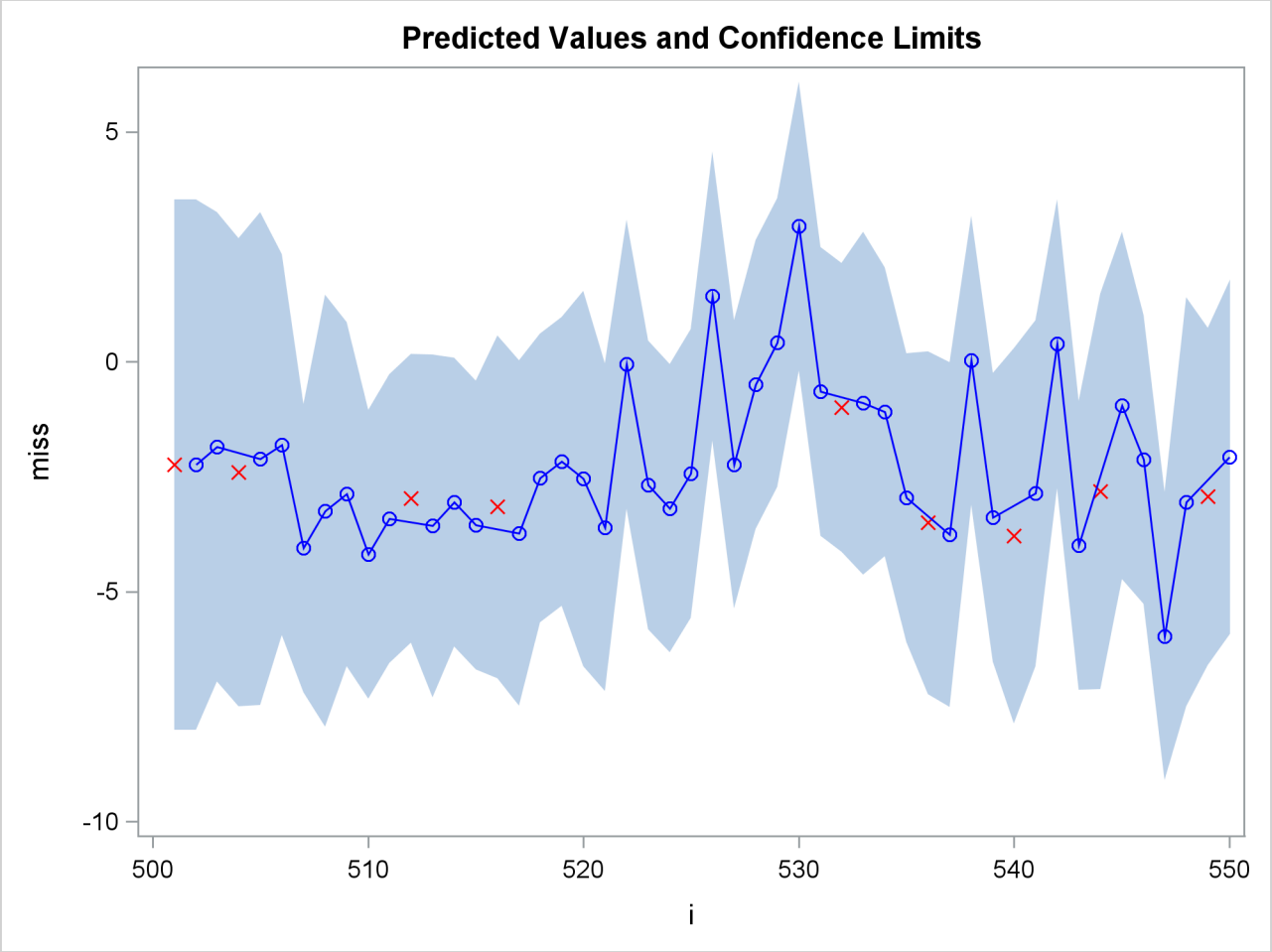


The following statements plot the residuals and confidence limits:

```
data reshape1;
   set a;
   miss = .;
   if r=. then do;
      miss = p;
      p = .;
   end;
run;

title 'Predicted Values and Confidence Limits';

proc sgplot data=reshape1 NOAUTOLEGEND;
   band x=i upper=u lower=l;
   scatter y=miss x=i/ MARKERATTRS =(symbol=x color=red);
   series y=p x=i/markers MARKERATTRS =(color=blue) lineattrs=(color=blue);
run;
```

The plot of the predicted values and the upper and lower confidence limits is shown in Output 9.4.3. Note that the confidence interval is wider at the beginning of the series (when there are no past noise values to use in the forecast equation) and after missing values where, again, there is an incomplete set of past residuals.

Output 9.4.3 Plot of Predicted Values and Confidence Interval



Example 9.5: Money Demand Model

This example estimates the log-log money demand equation by using the maximum likelihood method. The money demand model contains four explanatory variables. The lagged nominal money stock M1 is divided by the current price level GDF to calculate a new variable M1CP since the money stock is assumed to follow the partial adjustment process. The variable M1CP is then used to estimate the coefficient of adjustment. All variables are transformed using the natural logarithm with a DATA step. Refer to Balke and Gordon (1986) for a data description.

The first eight observations are printed using the PRINT procedure and are shown in Output 9.5.1. Note that the first observation of the variables M1CP and INFR are missing. Therefore, the money demand equation is estimated for the period 1968:2 to 1983:4 since PROC AUTOREG ignores the first missing observation. The DATA step that follows generates the transformed variables.

```
data money;
   date = intnx( 'qtr', '01jan1968'd, _n_-1 );
   format date yyqc6.;
   input m1 gnp gdf ycb @@;
   m = log( 100 * m1 / gdf );
   m1cp = log( 100 * lag(m1) / gdf );
   y = log( gnp );
   intr = log( ycb );
   infr = 100 * log( gdf / lag(gdf) );
   label m    = 'Real Money Stock (M1)'
         m1cp = 'Lagged M1/Current GDF'
         y    = 'Real GNP'
         intr = 'Yield on Corporate Bonds'
         infr = 'Rate of Prices Changes';
datalines;
  187.15 1036.22   81.18  6.84

   ... more lines ...
```

Output 9.5.1 Money Demand Data Series – First 8 Observations

Predicted Values and Confidence Limits

Obs	date	m1	gnp	gdf	ycb	m	m1cp	y	intr	infr
1	1968:1	187.15	1036.22	81.18	6.84	5.44041	.	6.94333	1.92279	.
2	1968:2	190.63	1056.02	82.12	6.97	5.44732	5.42890	6.96226	1.94162	1.15127
3	1968:3	194.30	1068.72	82.80	6.98	5.45815	5.43908	6.97422	1.94305	0.82465
4	1968:4	198.55	1071.28	84.04	6.84	5.46492	5.44328	6.97661	1.92279	1.48648
5	1969:1	201.73	1084.15	84.97	7.32	5.46980	5.45391	6.98855	1.99061	1.10054
6	1969:2	203.18	1088.73	86.10	7.54	5.46375	5.45659	6.99277	2.02022	1.32112
7	1969:3	204.18	1091.90	87.49	7.70	5.45265	5.44774	6.99567	2.04122	1.60151
8	1969:4	206.10	1085.53	88.62	8.22	5.44917	5.43981	6.98982	2.10657	1.28331

The money demand equation is first estimated using OLS. The DW=4 option produces generalized Durbin-Watson statistics up to the fourth order. Their exact marginal probabilities (p-values) are also calculated with the DWPROB option. The Durbin-Watson test indicates positive first-order autocorrelation at, say, the 10% confidence level. You can use the Durbin-Watson table, which is available only for 1% and 5% significance points. The relevant upper (d_U) and lower (d_L) bounds are $d_U = 1.731$ and $d_L = 1.471$, respectively, at 5% significance level. However, the bounds test is inconvenient, since sometimes you may get the statistic in the inconclusive region while the interval between the upper and lower bounds becomes smaller with the increasing sample size. The PROC step follows:

```
title 'Partial Adjustment Money Demand Equation';
title2 'Quarterly Data - 1968:2 to 1983:4';

proc autoreg data=money outest=est covout;
   model m = m1cp y intr infr / dw=4 dwprob;
run;
```

Output 9.5.2 OLS Estimation of the Partial Adjustment Money Demand Equation

Partial Adjustment Money Demand Equation
Quarterly Data - 1968:2 to 1983:4

The AUTOREG Procedure

Dependent Variable	m
	Real Money Stock (M1)

Ordinary Least Squares Estimates			
SSE	0.00271902	DFE	58
MSE	0.0000469	Root MSE	0.00685
SBC	-433.68709	AIC	-444.40276
MAE	0.00483389	AICC	-443.35013
MAPE	0.08888324	HQC	-440.18824
		Total R-Square	0.9546

Durbin-Watson Statistics

Order	DW	Pr < DW	Pr > DW
1	1.7355	0.0607	0.9393
2	2.1058	0.5519	0.4481
3	2.0286	0.5002	0.4998
4	2.2835	0.8880	0.1120

NOTE: Pr<DW is the p-value for testing positive autocorrelation, and Pr>DW is the p-value for testing negative autocorrelation.

Parameter Estimates

Variable	DF	Estimate	Standard Error	t Value	Approx Pr > |t|	Variable Label
Intercept	1	0.3084	0.2359	1.31	0.1963	
m1cp	1	0.8952	0.0439	20.38	<.0001	Lagged M1/Current GDF
y	1	0.0476	0.0122	3.89	0.0003	Real GNP
intr	1	-0.0238	0.007933	-3.00	0.0040	Yield on Corporate Bonds
infr	1	-0.005646	0.001584	-3.56	0.0007	Rate of Prices Changes

The autoregressive model is estimated using the maximum likelihood method. Though the Durbin-Watson test statistic is calculated after correcting the autocorrelation, it should be used with care since the test based on this statistic is not justified theoretically. The PROC step follows:

```
proc autoreg data=money;
   model m = m1cp y intr infr / nlag=1 method=ml maxit=50;
   output out=a p=p pm=pm r=r rm=rm ucl=ucl lcl=lcl
               uclm=uclm lclm=lclm;
run;

proc print data=a(obs=8);
   var p pm r rm ucl lcl uclm lclm;
run;
```

A difference is shown between the OLS estimates in Output 9.5.2 and the AR(1)-ML estimates in Output 9.5.3. The estimated autocorrelation coefficient is significantly negative (-0.88345). Note that the negative coefficient of AR(1) should be interpreted as a positive autocorrelation.

Two predicted values are produced: predicted values computed for the structural model and predicted values computed for the full model. The full model includes both the structural and error-process parts. The predicted values and residuals are stored in the output data set A, as are the upper and lower 95% confidence limits for the predicted values. Part of the data set A is shown in Output 9.5.4. The first observation is missing since the explanatory variables, M1CP and INFR, are missing for the corresponding observation.

Output 9.5.3 Estimated Partial Adjustment Money Demand Equation

Partial Adjustment Money Demand Equation
Quarterly Data - 1968:2 to 1983:4

The AUTOREG Procedure

Estimates of Autoregressive Parameters

Lag	Coefficient	Standard Error	t Value
1	-0.126273	0.131393	-0.96

Algorithm converged.

Maximum Likelihood Estimates

SSE	0.00226719	**DFE**	57
MSE	0.0000398	**Root MSE**	0.00631
SBC	-439.47665	**AIC**	-452.33545
MAE	0.00506044	**AICC**	-450.83545
MAPE	0.09302277	**HQC**	-447.27802
Log Likelihood	232.167727	**Transformed Regression R-Square**	0.6954
Durbin-Watson	2.1778	**Total R-Square**	0.9621
		Observations	63

Parameter Estimates

Variable	DF	Estimate	Standard Error	t Value	Approx Pr > |t|	Variable Label
Intercept	1	2.4121	0.4880	4.94	<.0001	
m1cp	1	0.4086	0.0908	4.50	<.0001	Lagged M1/Current GDF
y	1	0.1509	0.0411	3.67	0.0005	Real GNP
intr	1	-0.1101	0.0159	-6.92	<.0001	Yield on Corporate Bonds
infr	1	-0.006348	0.001834	-3.46	0.0010	Rate of Prices Changes
AR1	1	-0.8835	0.0686	-12.89	<.0001	

Autoregressive parameters assumed given

Variable	DF	Estimate	Standard Error	t Value	Approx Pr > |t|	Variable Label
Intercept	1	2.4121	0.4685	5.15	<.0001	
m1cp	1	0.4086	0.0840	4.87	<.0001	Lagged M1/Current GDF
y	1	0.1509	0.0402	3.75	0.0004	Real GNP
intr	1	-0.1101	0.0155	-7.08	<.0001	Yield on Corporate Bonds
infr	1	-0.006348	0.001828	-3.47	0.0010	Rate of Prices Changes

Output 9.5.4 Partial List of the Predicted Values

Partial Adjustment Money Demand Equation
Quarterly Data - 1968:2 to 1983:4

Obs	p	pm	r	rm	ucl	lcl	uclm	lclm
1	.	.	.	.	.	.	.	.
2	5.45962	5.45962	-.005763043	-0.012301	5.49319	5.42606	5.47962	5.43962
3	5.45663	5.46750	0.001511258	-0.009356	5.46954	5.44373	5.48700	5.44800
4	5.45934	5.46761	0.005574104	-0.002691	5.47243	5.44626	5.48723	5.44799
5	5.46636	5.46874	0.003442075	0.001064	5.47944	5.45328	5.48757	5.44991
6	5.46675	5.46581	-.002994443	-0.002054	5.47959	5.45390	5.48444	5.44718
7	5.45672	5.45854	-.004074196	-0.005889	5.46956	5.44388	5.47667	5.44040
8	5.44404	5.44924	0.005136019	-0.000066	5.45704	5.43103	5.46726	5.43122

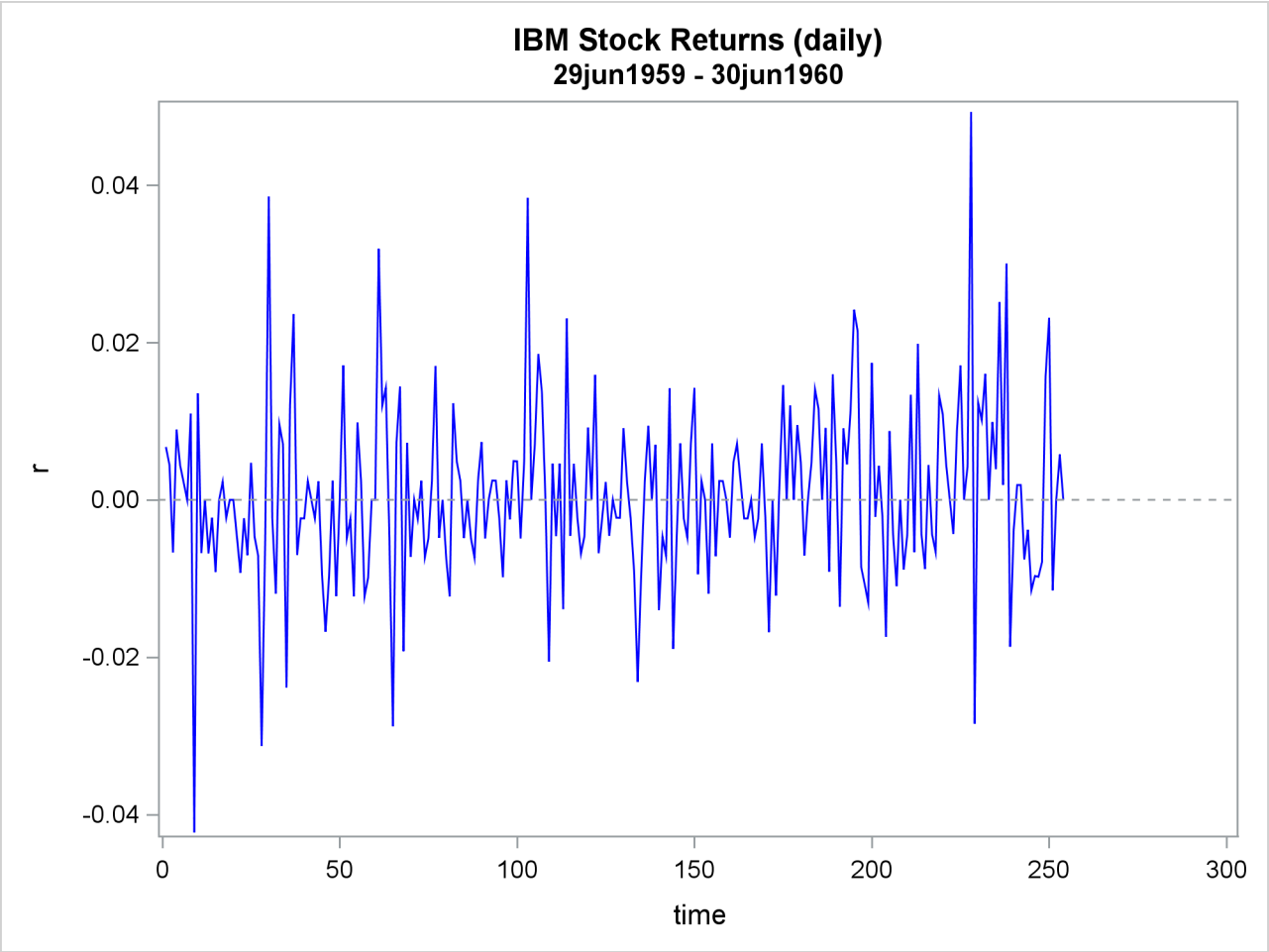
Example 9.6: Estimation of ARCH(2) Process

Stock returns show a tendency for small changes to be followed by small changes while large changes are followed by large changes. The plot of daily price changes of IBM common stock (Box and Jenkins 1976, p. 527) is shown in Output 9.6.1. The time series look serially uncorrelated, but the plot makes us skeptical of their independence.

With the following DATA step, the stock (capital) returns are computed from the closing prices. To forecast the conditional variance, an additional 46 observations with missing values are generated.

```
title 'IBM Stock Returns (daily)';
title2 '29jun1959 - 30jun1960';

data ibm;
   infile datalines eof=last;
   input x @@;
   r = dif( log( x ) );
   time = _n_-1;
   output;
   return;
last:
   do i = 1 to 46;
      r = .;
      time + 1;
      output;
   end;
   return;
datalines;
445 448 450 447 451 453 454 454 459 440 446 443 443 440

   ... more lines ...


proc sgplot data=ibm;
   series y=r x=time/lineattrs=(color=blue);
         refline 0/ axis = y LINEATTRS = (pattern=ShortDash);
run;
```

Output 9.6.1 IBM Stock Returns: Daily



The simple ARCH(2) model is estimated using the AUTOREG procedure. The MODEL statement option GARCH=(Q=2) specifies the ARCH(2) model. The OUTPUT statement with the CEV= option produces the conditional variances V. The conditional variance and its forecast are calculated using parameter estimates:

$$h_t = \hat{\omega} + \hat{\alpha}_1 \epsilon_{t-1}^2 + \hat{\alpha}_2 \epsilon_{t-2}^2$$

$$\mathbf{E}(\epsilon_{t+d}^2 | \Psi_t) = \hat{\omega} + \sum_{i=1}^{2} \hat{\alpha}_i \mathbf{E}(\epsilon_{t+d-i}^2 | \Psi_t)$$

where $d > 1$. This model can be estimated as follows:

```
proc autoreg data=ibm maxit=50;
   model r = / noint garch=(q=2);
   output out=a cev=v;
run;
```

The parameter estimates for ω, α_1, and α_2 are 0.00011, 0.04136, and 0.06976, respectively. The normality test indicates that the conditional normal distribution may not fully explain the leptokurtosis in the stock returns (Bollerslev 1987).

The ARCH model estimates are shown in Output 9.6.2, and conditional variances are also shown in Output 9.6.3. The code that generates Output 9.6.3 is shown below.

```
data b; set a;
   length type $ 8.;
   if r ^= . then do;
      type = 'ESTIMATE'; output; end;
   else do;
      type = 'FORECAST'; output; end;
run;
proc sgplot data=b;
   series x=time y=v/group=type;
   refline 254/ axis = x LINEATTRS = (pattern=ShortDash);
run;
```

Output 9.6.2 ARCH(2) Estimation Results

IBM Stock Returns (daily)
29jun1959 - 30jun1960

The AUTOREG Procedure

Dependent Variable	r

Ordinary Least Squares Estimates			
SSE	0.03214307	**DFE**	254
MSE	0.0001265	**Root MSE**	0.01125
SBC	-1558.802	**AIC**	-1558.802
MAE	0.00814086	**AICC**	-1558.802
MAPE	100.378566	**HQC**	-1558.802
Durbin-Watson	2.1377	**Total R-Square**	0.0000

NOTE: No intercept term is used.
R-squares are redefined.

Algorithm converged.

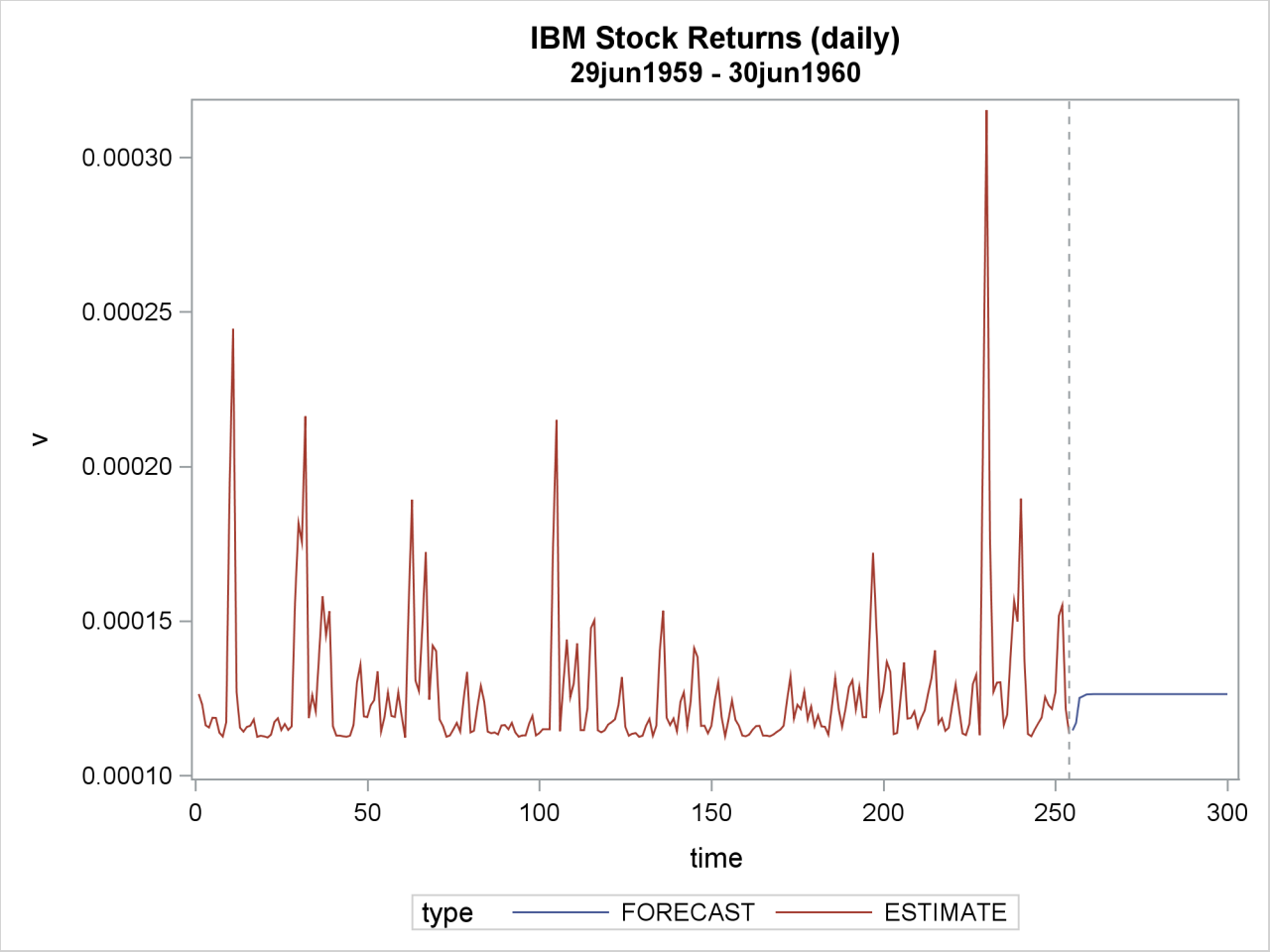
GARCH Estimates			
SSE	0.03214307	**Observations**	254
MSE	0.0001265	**Uncond Var**	0.00012632
Log Likelihood	781.017441	**Total R-Square**	0.0000
SBC	-1545.4229	**AIC**	-1556.0349
MAE	0.00805675	**AICC**	-1555.9389
MAPE	100	**HQC**	-1551.7658
		Normality Test	105.8587
		Pr > ChiSq	<.0001

NOTE: No intercept term is used.
R-squares are redefined.

Output 9.6.2 *continued*

Parameter Estimates					
Variable	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
ARCH0	1	0.000112	7.6059E-6	14.76	<.0001
ARCH1	1	0.0414	0.0514	0.81	0.4208
ARCH2	1	0.0698	0.0434	1.61	0.1082

Output 9.6.3 Conditional Variance for IBM Stock Prices



Example 9.7: Estimation of GARCH-Type Models

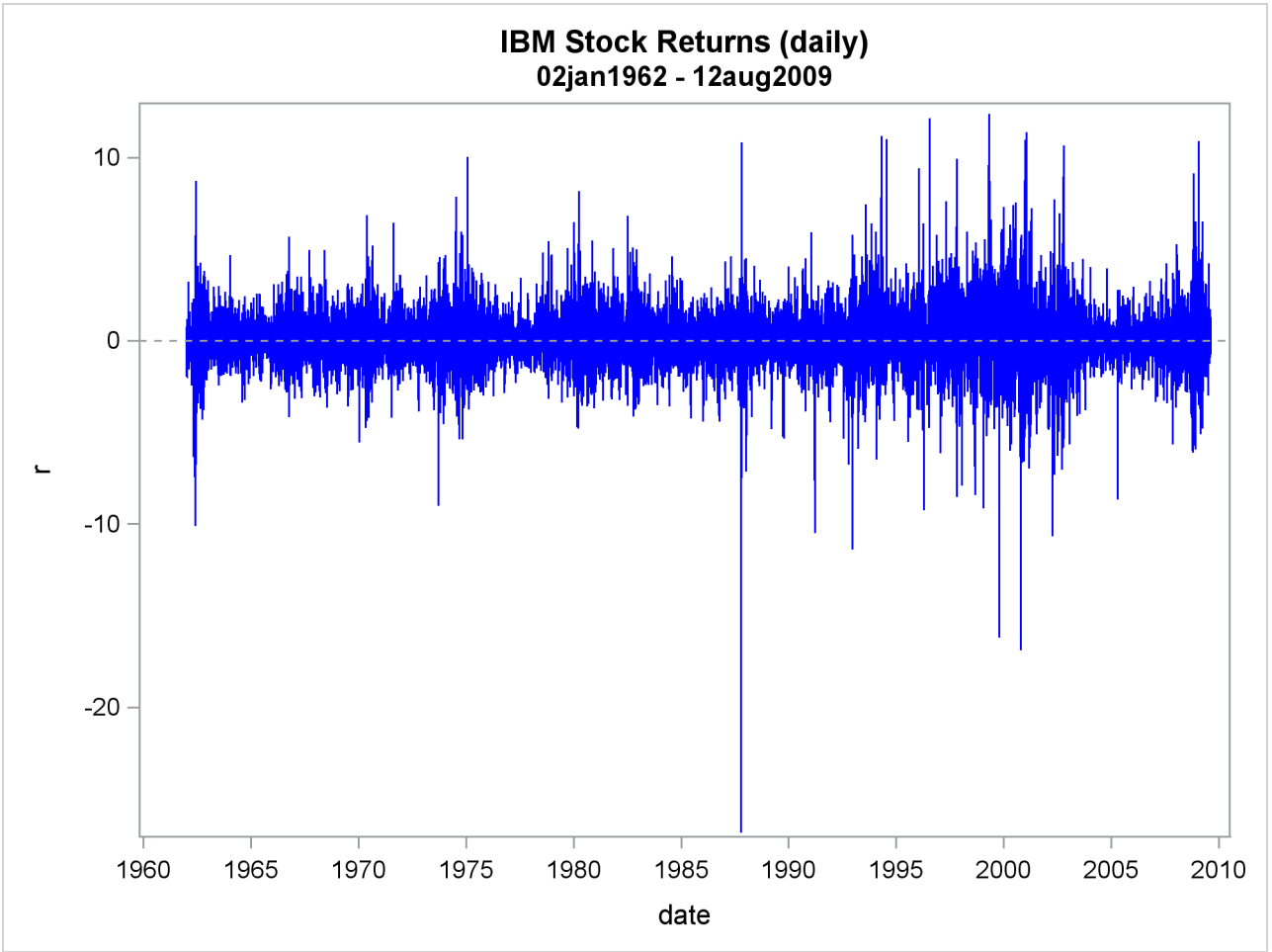
This example extends Example 9.6 to include more volatility models and to perform model selection and diagnostics.

Following is the data of daily IBM stock prices for the long period from 1962 to 2009.

```
data ibm_long;
   infile datalines;
   format date MMDDYY10.;
   input date:MMDDYY10. price_ibm;
   r = 100*dif( log( price_ibm ) );
datalines;
01/02/1962 2.68
01/03/1962 2.7
01/04/1962 2.67
01/05/1962 2.62
01/08/1962 2.57

   ... more lines ...

08/12/2009 119.29
;
```

The time series of IBM returns is depicted graphically in Output 9.7.1.

Output 9.7.1 IBM Stock Returns: Daily



The following statements perform estimation of different kinds of GARCH-type models. First, ODS listing output that contains fit summary tables for each single model is captured by using an ODS OUTPUT statement with the appropriate ODS table name assigned to a new SAS data set. Along with these new data sets, another one that contains parameter estimates is created by using the OUTEST= option in AUTOREG statement.

```
/* Capturing ODS tables into SAS data sets */
ods output Autoreg.ar_1.FinalModel.FitSummary
           =fitsum_ar_1;
ods output Autoreg.arch_2.FinalModel.Results.FitSummary
           =fitsum_arch_2;
ods output Autoreg.garch_1_1.FinalModel.Results.FitSummary
           =fitsum_garch_1_1;
ods output Autoreg.st_garch_1_1.FinalModel.Results.FitSummary
           =fitsum_st_garch_1_1;
ods output Autoreg.ar_1_garch_1_1.FinalModel.Results.FitSummary
           =fitsum_ar_1_garch_1_1;
ods output Autoreg.igarch_1_1.FinalModel.Results.FitSummary
           =fitsum_igarch_1_1;
ods output Autoreg.garchm_1_1.FinalModel.Results.FitSummary
           =fitsum_garchm_1_1;
ods output Autoreg.egarch_1_1.FinalModel.Results.FitSummary
           =fitsum_egarch_1_1;
ods output Autoreg.qgarch_1_1.FinalModel.Results.FitSummary
           =fitsum_qgarch_1_1;
ods output Autoreg.tgarch_1_1.FinalModel.Results.FitSummary
           =fitsum_tgarch_1_1;
ods output Autoreg.pgarch_1_1.FinalModel.Results.FitSummary
           =fitsum_pgarch_1_1;

/* Estimating multiple GARCH-type models */
title "GARCH family";
proc autoreg data=ibm_long outest=garch_family;
   ar_1 :           model r = / noint nlag=1 method=ml;
   arch_2 :         model r = / noint garch=(q=2);
   garch_1_1 :      model r = / noint garch=(p=1,q=1);
   st_garch_1_1 :   model r = / noint garch=(p=1,q=1,type=stationary);
   ar_1_garch_1_1 : model r = / noint nlag=1 garch=(p=1,q=1);
   igarch_1_1 :     model r = / noint garch=(p=1,q=1,type=integ,noint);
   egarch_1_1 :     model r = / noint garch=(p=1,q=1,type=egarch);
   garchm_1_1 :     model r = / noint garch=(p=1,q=1,mean=log);
   qgarch_1_1 :     model r = / noint garch=(p=1,q=1,type=qgarch);
   tgarch_1_1 :     model r = / noint garch=(p=1,q=1,type=tgarch);
   pgarch_1_1 :     model r = / noint garch=(p=1,q=1,type=pgarch);
run;
```

The following statements print partial contents of the data set GARCH_FAMILY. The columns of interest are explicitly specified in the VAR statement.

```
/* Printing summary table of parameter estimates */
title "Parameter Estimates for Different Models";
proc print data=garch_family;
   var _MODEL_ _A_1 _AH_0 _AH_1 _AH_2
       _GH_1  _AHQ_1 _AHT_1 _AHP_1 _THETA_ _LAMBDA_ _DELTA_;
run;
```

These statements produce the results shown in Output 9.7.2.

Output 9.7.2 GARCH-Family Estimation Results

Parameter Estimates for Different Models

Obs	_MODEL_	_A_1	_AH_0	_AH_1	_AH_2	_GH_1	_AHQ_1	_AHT_1	_AHP_1
1	ar_1	0.017112	.	.	.	.	.	.	.
2	arch_2	.	1.60288	0.23235	0.21407	.	.	.	.
3	garch_1_1	.	0.02730	0.06984	.	0.92294	.	.	.
4	st_garch_1_1	.	0.02831	0.06913	.	0.92260	.	.	.
5	ar_1_garch_1_1	-0.005995	0.02734	0.06994	.	0.92282	.	.	.
6	igarch_1_1	.	.	0.00000	.	1.00000	.	.	.
7	egarch_1_1	.	0.01541	0.12882	.	0.98914	.	.	.
8	garchm_1_1	.	0.02897	0.07139	.	0.92079	.	.	.
9	qgarch_1_1	.	0.00120	0.05792	.	0.93458	0.66461	.	.
10	tgarch_1_1	.	0.02706	0.02966	.	0.92765	.	0.074815	.
11	pgarch_1_1	.	0.01623	0.06724	.	0.93952	.	.	0.43445

Obs	_THETA_	_LAMBDA_	_DELTA_
1	.	.	.
2	.	.	.
3	.	.	.
4	.	.	.
5	.	.	.
6	.	.	.
7	-0.41706	.	.
8	.	.	0.094773
9	.	.	.
10	.	.	.
11	.	0.53625	.

The table shown in Output 9.7.2 is convenient for reporting the estimation result of multiple models and their comparison.

The following statements merge multiple tables that contain fit statistics for each estimated model, leaving only columns of interest, and rename them.

```
/* Merging ODS output tables and extracting AIC and SBC measures */
data sbc_aic;
   set fitsum_arch_2 fitsum_garch_1_1 fitsum_st_garch_1_1
       fitsum_ar_1 fitsum_ar_1_garch_1_1 fitsum_igarch_1_1
       fitsum_egarch_1_1 fitsum_garchm_1_1
       fitsum_tgarch_1_1 fitsum_pgarch_1_1 fitsum_qgarch_1_1;
   keep Model SBC AIC;
   if Label1="SBC" then do; SBC=input(cValue1,BEST12.4); end;
   if Label2="SBC" then do; SBC=input(cValue2,BEST12.4); end;
   if Label1="AIC" then do; AIC=input(cValue1,BEST12.4); end;
   if Label2="AIC" then do; AIC=input(cValue2,BEST12.4); end;
   if not (SBC=.) then output;
run;
```

Next, sort the models by one of the criteria, for example, by AIC:

```
/* Sorting data by AIC criterion */
proc sort data=sbc_aic;
   by AIC;
run;
```

Finally, print the sorted data set:

```
title "Selection Criteria for Different Models";
proc print data=sbc_aic;
   format _NUMERIC_ BEST12.4;
run;
```

The result is given in Output 9.7.3.

Output 9.7.3 GARCH-Family Model Selection on the Basis of AIC and SBC

Selection Criteria for Different Models

Obs	Model	SBC	AIC
1	pgarch_1_1	42907.7292	42870.7722
2	egarch_1_1	42905.9616	42876.3959
3	tgarch_1_1	42995.4893	42965.9236
4	qgarch_1_1	43023.106	42993.5404
5	garchm_1_1	43158.4139	43128.8483
6	garch_1_1	43176.5074	43154.3332
7	ar_1_garch_1_1	43185.5226	43155.957
8	st_garch_1_1	43178.2497	43156.0755
9	arch_2	44605.4332	44583.259
10	ar_1	45922.0721	45914.6807
11	igarch_1_1	45925.5828	45918.1914

According to the smaller-is-better rule for the information criteria, the PGARCH(1,1) model is the leader by AIC while the EGARCH(1,1) is the model of choice according to SBC.

Next, check whether the power GARCH model is misspecified, especially, if dependence exists in the standardized residuals that correspond to the assumed independently and identically distributed (iid) disturbance. The following statements reestimate the power GARCH model and use the BDS test to check the independence of the standardized residuals.

```
proc autoreg data=ibm_long;
   model r = / noint garch=(p=1,q=1,type=pgarch) BDS=(Z=SR,D=2.0);
run;
```

The partial results listing of the preceding statements is given in Output 9.7.4.

Output 9.7.4 Diagnostic Checking of the PGARCH(1,1) Model

Selection Criteria for Different Models

The AUTOREG Procedure

BDS Test for Independence			
Distance	Embedding Dimension	BDS	Pr > |BDS|
2.0000	2	2.9691	0.0030
	3	3.3810	0.0007
	4	3.1299	0.0017
	5	3.3805	0.0007
	6	3.3368	0.0008
	7	3.1888	0.0014
	8	2.9576	0.0031
	9	2.7386	0.0062
	10	2.5553	0.0106
	11	2.3510	0.0187
	12	2.1520	0.0314
	13	1.9373	0.0527
	14	1.7210	0.0852
	15	1.4919	0.1357
	16	1.2569	0.2088
	17	1.0647	0.2870
	18	0.9635	0.3353
	19	0.8678	0.3855
	20	0.7660	0.4437

The results in Output 9.7.4 indicate that when embedded size is greater than 9, you fail to reject the null hypothesis of independence at 1% significance level, which is a good indicator that the PGARCH model is not misspecified.

Example 9.8: Illustration of ODS Graphics

This example illustrates the use of ODS GRAPHICS. This is a continuation of the section "Forecasting Autoregressive Error Models" on page 341.

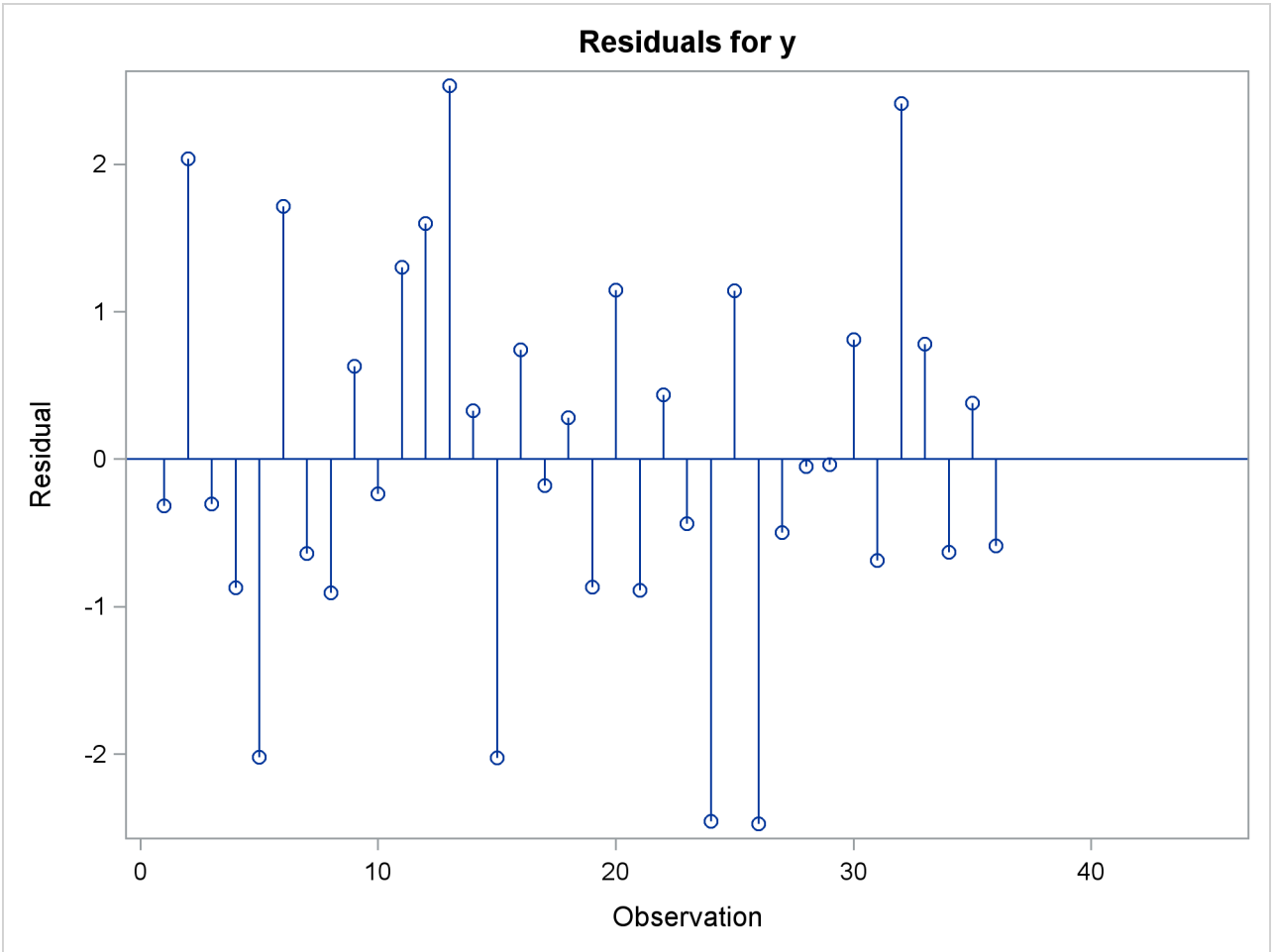
These graphical displays are requested by specifying the ODS GRAPHICS statement. For information about the graphs available in the AUTOREG procedure, see the section "ODS Graphics" on page 437.

The following statements show how to generate ODS GRAPHICS plots with the AUTOREG procedure. In this case, all plots are requested using the ALL option in the PROC AUTOREG statement, in addition to the ODS GRAPHICS statement. The plots are displayed in Output 9.8.1 through Output 9.8.8. Note: these plots can be viewed in the Autoreg.Model.FitDiagnosticPlots category by selecting **View►Results**.

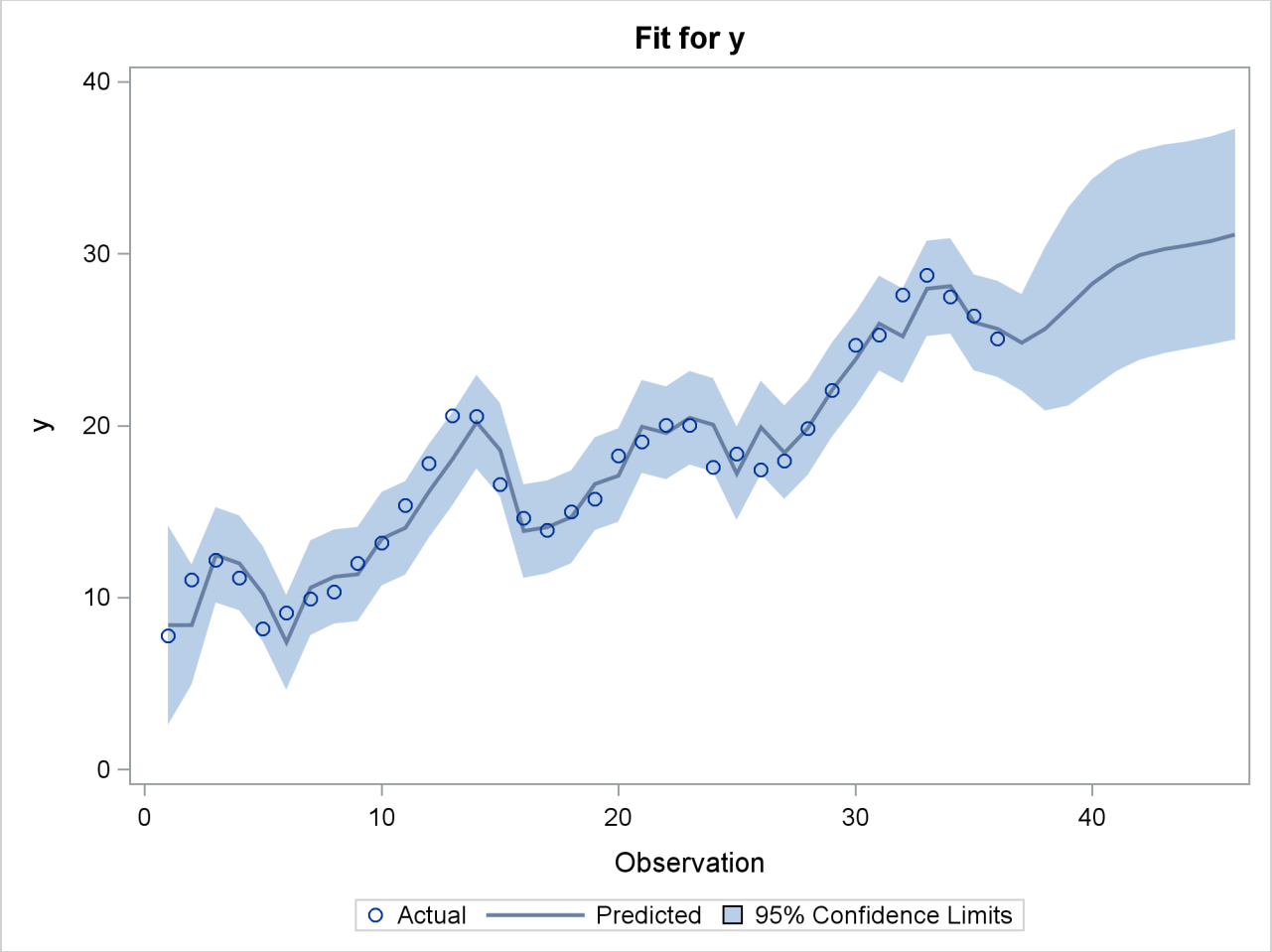
```
data a;
   ul = 0; ull = 0;
   do time = -10 to 36;
      u = + 1.3 * ul - .5 * ull + 2*rannor(12346);
```

```
      y = 10 + .5 * time + u;
      if time > 0 then output;
      ull = ul; ul = u;
   end;
run;

data b;
   y = .;
   do time = 37 to 46; output; end;
run;

data b;
   merge a b;
   by time;
run;

proc autoreg data=b all plots(unpack);
   model y = time / nlag=2 method=ml;
   output out=p p=yhat  pm=ytrend
               lcl=lcl ucl=ucl;
run;
```

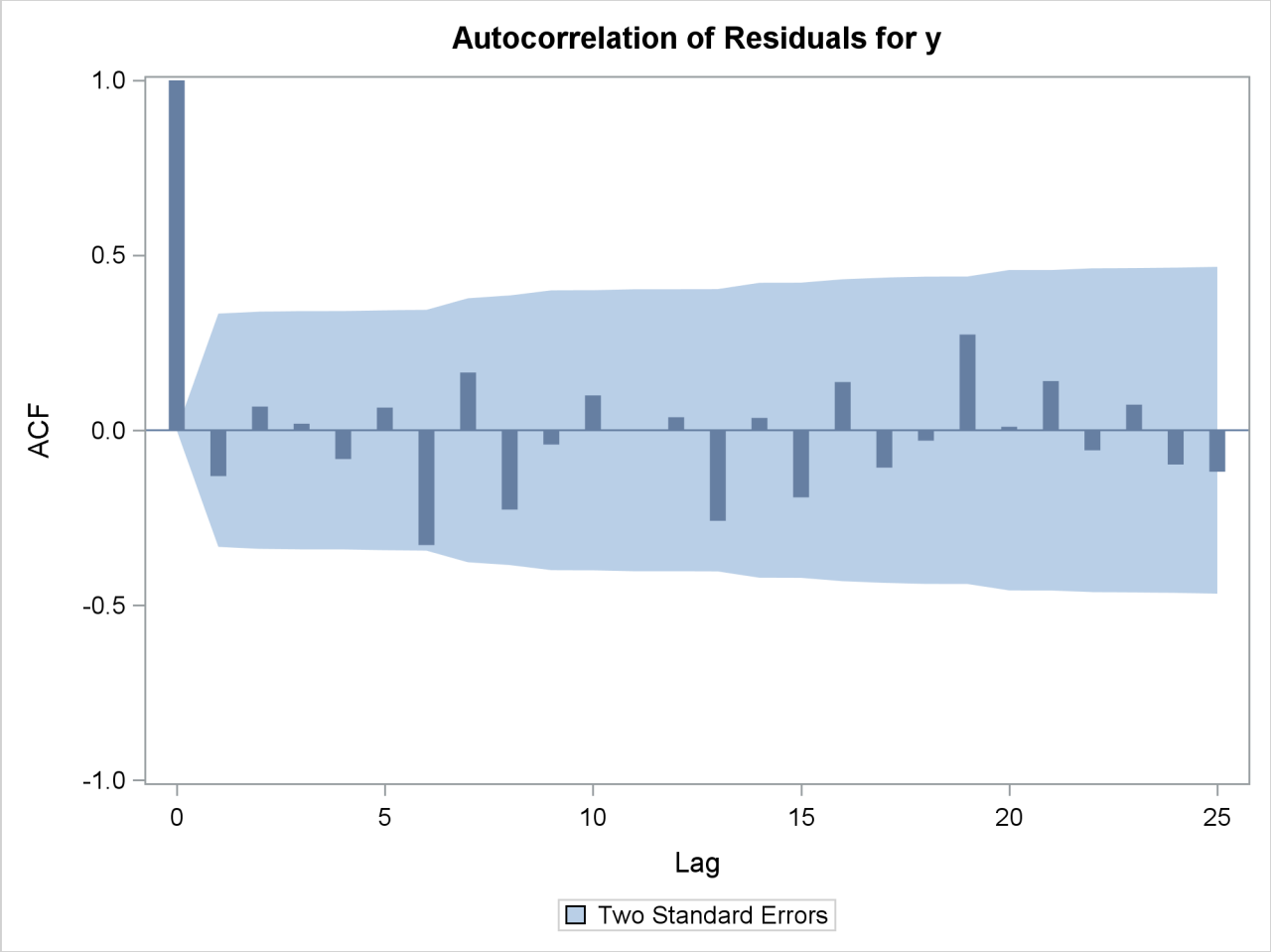
Output 9.8.1 Residuals Plot



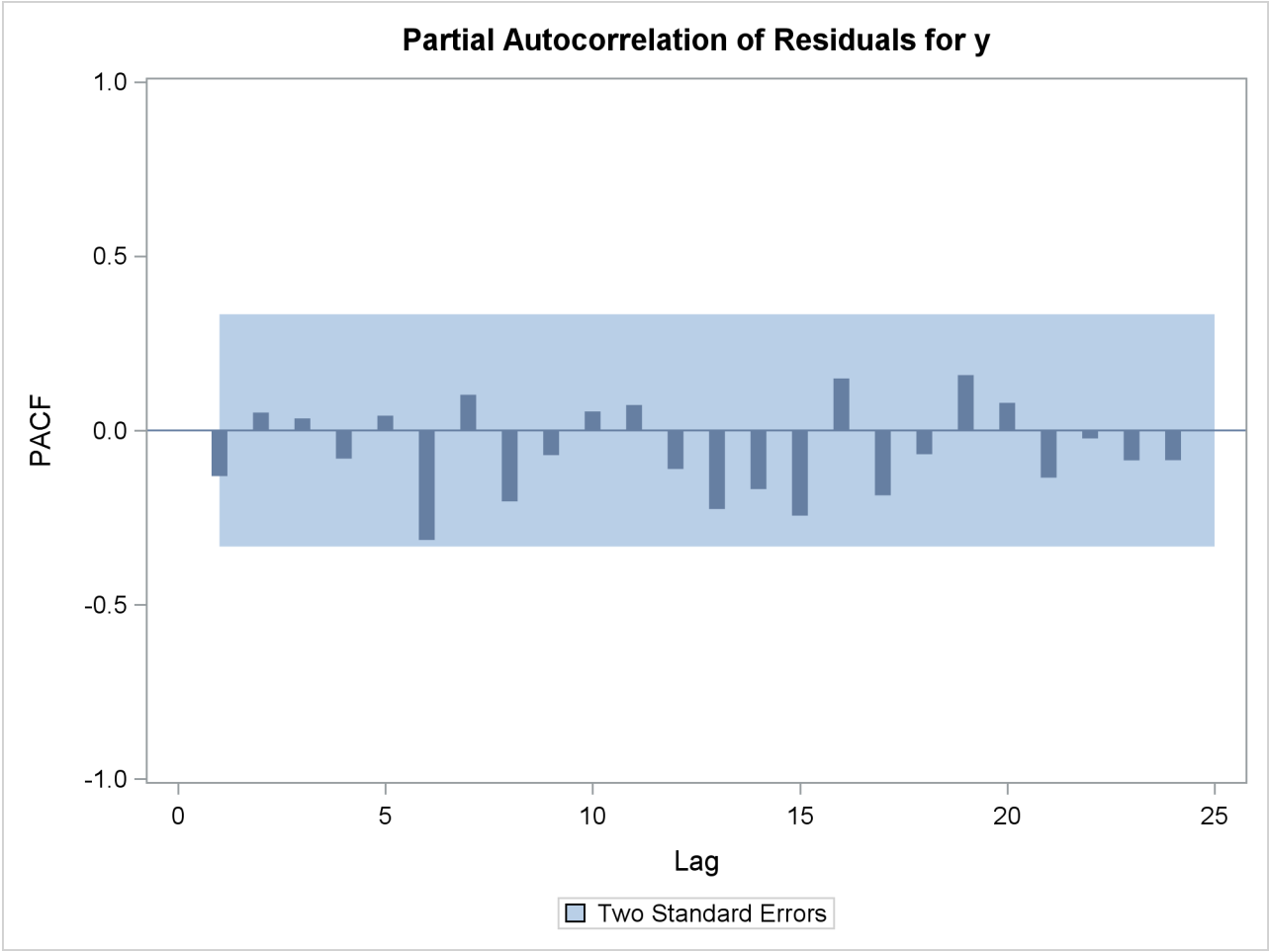
Output 9.8.2 Predicted versus Actual Plot



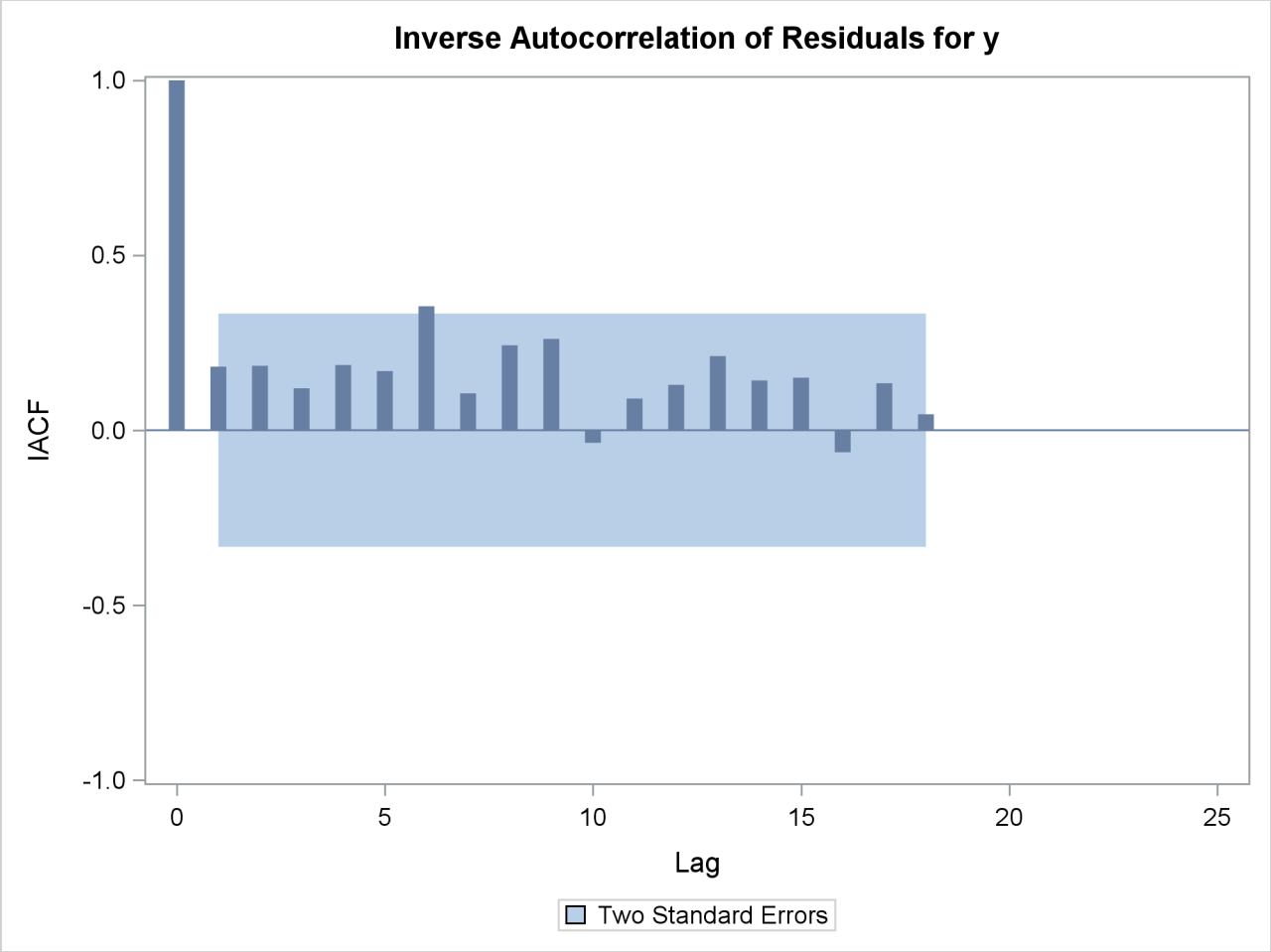
Output 9.8.3 Autocorrelation of Residuals Plot



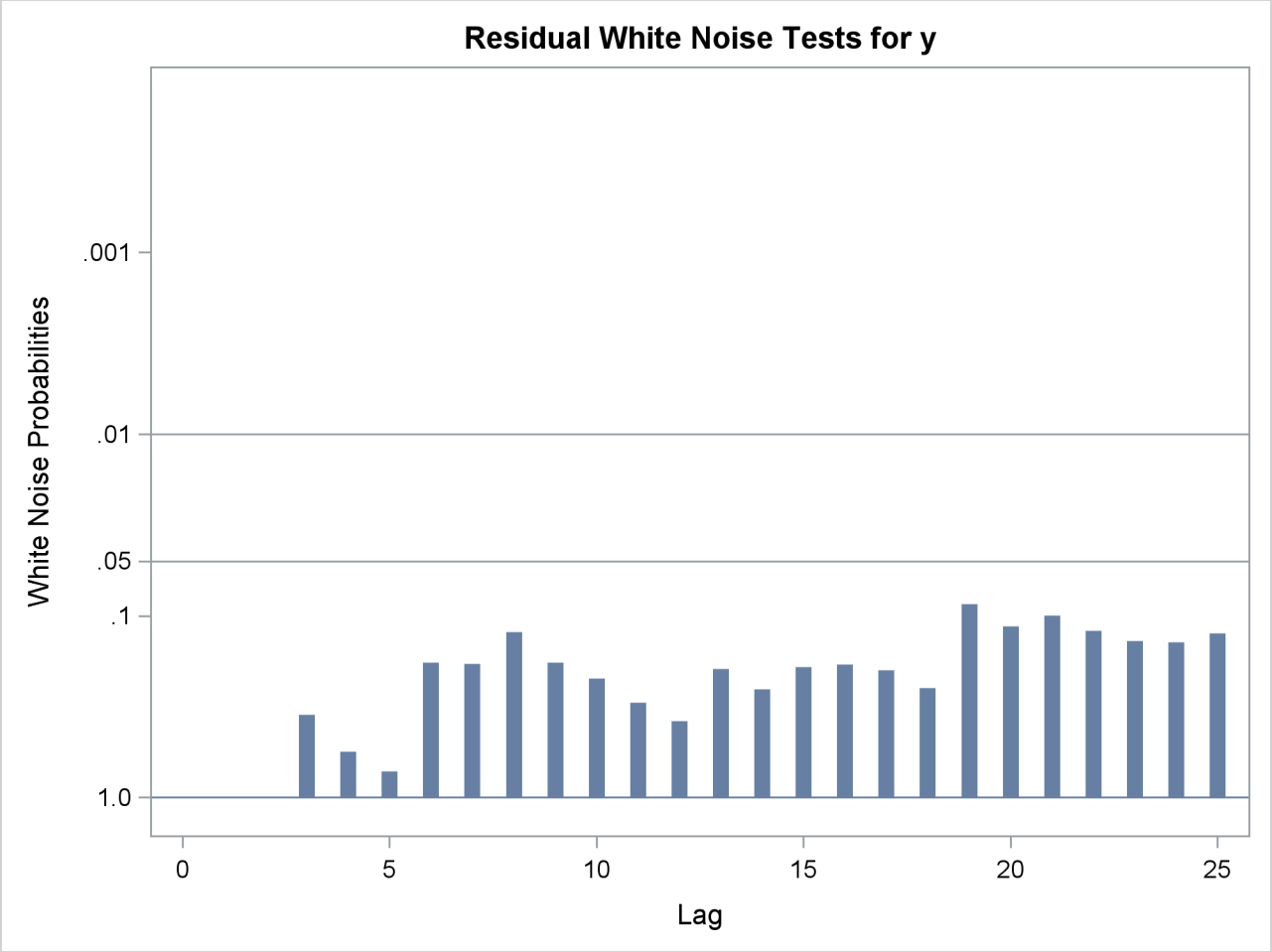
Output 9.8.4 Partial Autocorrelation of Residuals Plot



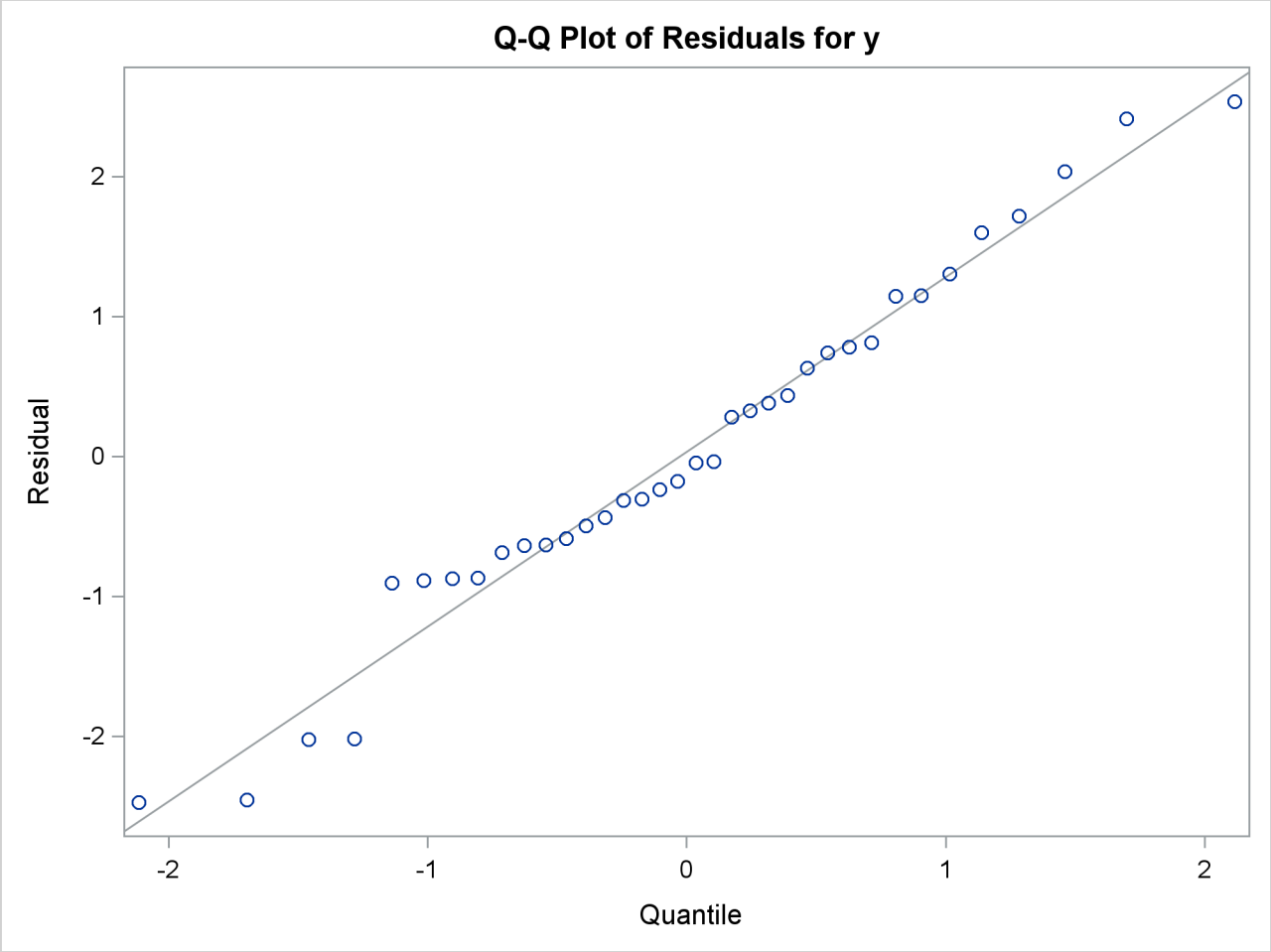
Output 9.8.5 Inverse Autocorrelation of Residuals Plot



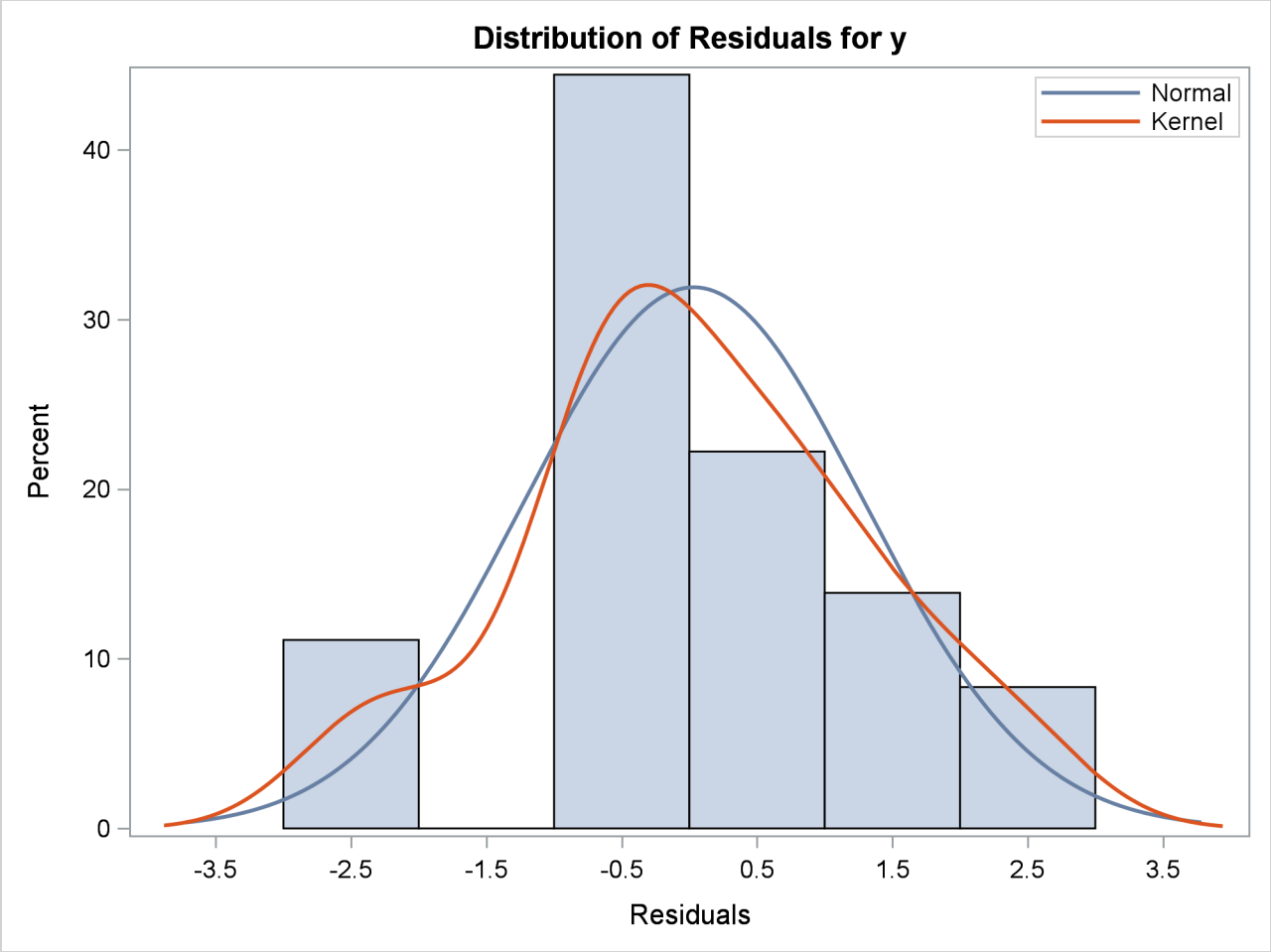
Output 9.8.6 Tests for White Noise Residuals Plot



Output 9.8.7 Q-Q Plot of Residuals



Output 9.8.8 Histogram of Residuals



References

Anderson, T. W., and Mentz, R. P. (1980). "On the Structure of the Likelihood Function of Autoregressive and Moving Average Models." *Journal of Time Series* 1:83–94.

Andrews, D. W. K. (1991). "Heteroscedasticity and Autocorrelation Consistent Covariance Matrix Estimation." *Econometrica* 59:817–858.

Ansley, C. F., Kohn, R., and Shively, T. S. (1992). "Computing p-Values for the Generalized Durbin-Watson and Other Invariant Test Statistics." *Journal of Econometrics* 54:277–300.

Bai, J. (1997). "Estimation of a Change Point in Multiple Regression Models." *Review of Economics and Statistics* 79:551–563.

Bai, J., and Perron, P. (1998). "Estimating and Testing Linear Models with Multiple Structural Changes." *Econometrica* 66:47–78. http://www.jstor.org/stable/2998540.

Bai, J., and Perron, P. (2003a). "Computation and Analysis of Multiple Structural Change Models." *Journal of Applied Econometrics* 18:1–22. doi:10.1002/jae.659.

Bai, J., and Perron, P. (2003b). "Critical Values for Multiple Structural Change Tests." *Econometrics Journal* 6:72–78. doi:10.1111/1368-423X.00102.

Bai, J., and Perron, P. (2006). "Multiple Structural Change Models: A Simulation Analysis." In *Econometric Theory and Practice: Frontiers of Analysis and Applied Research*, edited by D. Corbae, S. N. Durlauf, and B. E. Hansen, 212–237. Cambridge: Cambridge University Press.

Baillie, R. T. (1979). "The Asymptotic Mean Squared Error of Multistep Prediction from the Regression Model with Autoregressive Errors." *Journal of the American Statistical Association* 74:175–184.

Baillie, R. T., and Bollerslev, T. (1992). "Prediction in Dynamic Models with Time-Dependent Conditional Variances." *Journal of Econometrics* 52:91–113.

Balke, N. S., and Gordon, R. J. (1986). "Historical Data." In *The American Business Cycle*, edited by R. J. Gordon, 781–850. Chicago: University of Chicago Press.

Bartels, R. (1982). "The Rank Version of von Neumann's Ratio Test for Randomness." *Journal of the American Statistical Association* 77:40–46.

Beach, C. M., and MacKinnon, J. G. (1978). "A Maximum Likelihood Procedure for Regression with Autocorrelated Errors." *Econometrica* 46:51–58.

Bhargava, A. (1986). "On the Theory of Testing for Unit Roots in Observed Time Series." *Review of Economic Studies* 53:369–384.

Bollerslev, T. (1986). "Generalized Autoregressive Conditional Heteroskedasticity." *Journal of Econometrics* 31:307–327.

Bollerslev, T. (1987). "A Conditionally Heteroskedastic Time Series Model for Speculative Prices and Rates of Return." *Review of Economics and Statistics* 69:542–547.

Box, G. E. P., and Jenkins, G. M. (1976). *Time Series Analysis: Forecasting and Control*. Rev. ed. San Francisco: Holden-Day.

Breitung, J. (1995). "Modified Stationarity Tests with Improved Power in Small Samples." *Statistical Papers* 36:77–95.

Breusch, T. S., and Pagan, A. R. (1979). "A Simple Test for Heteroscedasticity and Random Coefficient Variation." *Econometrica* 47:1287–1294.

Brock, W. A., Dechert, W. D., and Scheinkman, J. A. (1987). "A Test for Independence Based on the Correlation Dimension." Departments of Economics, University of Wisconsin–Madison, University of Houston, and University of Chicago.

Brock, W. A., Scheinkman, J. A., Dechert, W. D., and LeBaron, B. (1996). "A Test for Independence Based on the Correlation Dimension." *Econometric Reviews* 15:197–235.

Campbell, J. Y., and Perron, P. (1991). "Pitfalls and Opportunities: What Macroeconomists Should Know about Unit Roots." In *NBER Macroeconomics Annual*, edited by O. Blanchard, and S. Fisher, 141–201. Cambridge, MA: MIT Press.

Caner, M., and Kilian, L. (2001). "Size Distortions of Tests of the Null Hypothesis of Stationarity: Evidence and Implications for the PPP Debate." *Journal of International Money and Finance* 20:639–657.

Chipman, J. S. (1979). "Efficiency of Least Squares Estimation of Linear Trend When Residuals Are Autocorrelated." *Econometrica* 47:115–128.

Chow, G. (1960). "Tests of Equality between Sets of Coefficients in Two Linear Regressions." *Econometrica* 28:531–534.

Cochrane, D., and Orcutt, G. H. (1949). "Application of Least Squares Regression to Relationships Containing Autocorrelated Error Terms." *Journal of the American Statistical Association* 44:32–61.

Cribari-Neto, F. (2004). "Asymptotic Inference under Heteroskedasticity of Unknown Form." *Computational Statistics and Data Analysis* 45:215–233.

Cromwell, J. B., Labys, W. C., and Terraza, M. (1994). *Univariate Tests for Time Series Models*. Thousand Oaks, CA: Sage Publications.

Davidson, R., and MacKinnon, J. G. (1993). *Estimation and Inference in Econometrics*. New York: Oxford University Press.

Davies, R. B. (1973). "Numerical Inversion of a Characteristic Function." *Biometrika* 60:415–417.

DeJong, D. N., Nankervis, J. C., Savin, N. E., and Whiteman, C. H. (1992). "The Power Problems of Unit Root Rest in Time Series with Autoregressive Errors." *Journal of Econometrics* 53:323–343.

Dickey, D. A., and Fuller, W. A. (1979). "Distribution of the Estimators for Autoregressive Time Series with a Unit Root." *Journal of the American Statistical Association* 74:427–431.

Ding, Z., Granger, C. W. J., and Engle, R. F. (1993). "A Long Memory Property of Stock Market Returns and a New Model." *Journal of Empirical Finance* 1:83–106.

Duffie, D. (1989). *Futures Markets*. Englewood Cliffs, NJ: Prentice-Hall.

Durbin, J. (1969). "Tests for Serial Correlation in Regression Analysis Based on the Periodogram of Least-Squares Residuals." *Biometrika* 56:1–15.

Durbin, J. (1970). "Testing for Serial Correlation in Least-Squares Regression When Some of the Regressors Are Lagged Dependent Variables." *Econometrica* 38:410–421.

Edgerton, D., and Wells, C. (1994). "Critical Values for the CUSUMSQ Statistic in Medium and Large Sized Samples." *Oxford Bulletin of Economics and Statistics* 56:355–365.

Elliott, G., Rothenberg, T. J., and Stock, J. H. (1996). "Efficient Tests for an Autoregressive Unit Root." *Econometrica* 64:813–836.

Engle, R. F. (1982). "Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation." *Econometrica* 50:987–1007.

Engle, R. F., and Bollerslev, T. (1986). "Modelling the Persistence of Conditional Variances." *Econometric Reviews* 5:1–50.

Engle, R. F., and Granger, C. W. J. (1987). "Co-integration and Error Correction: Representation, Estimation, and Testing." *Econometrica* 55:251–276.

Engle, R. F., Lilien, D. M., and Robins, R. P. (1987). "Estimating Time Varying Risk in the Term Structure: The ARCH-M Model." *Econometrica* 55:391–407.

Engle, R. F., and Ng, V. K. (1993). "Measuring and Testing the Impact of News on Volatility." *Journal of Finance* 48:1749–1778.

Engle, R. F., and Yoo, B. S. (1987). "Forecasting and Testing in Co-integrated Systems." *Journal of Econometrics* 35:143–159.

Fuller, W. A. (1976). *Introduction to Statistical Time Series*. New York: John Wiley & Sons.

Gallant, A. R., and Goebel, J. J. (1976). "Nonlinear Regression with Autoregressive Errors." *Journal of the American Statistical Association* 71:961–967.

Glosten, L., Jaganathan, R., and Runkle, D. (1993). "Relationship between the Expected Value and Volatility of the Nominal Excess Returns on Stocks." *Journal of Finance* 48:1779–1802.

Godfrey, L. G. (1978a). "Testing against General Autoregressive and Moving Average Error Models When the Regressors Include Lagged Dependent Variables." *Econometrica* 46:1293–1301.

Godfrey, L. G. (1978b). "Testing for Higher Order Serial Correlation in Regression Equations When the Regressors Include Lagged Dependent Variables." *Econometrica* 46:1303–1310.

Godfrey, L. G. (1988). *Misspecification Tests in Econometrics*. Cambridge: Cambridge University Press.

Golub, G. H., and Van Loan, C. F. (1989). *Matrix Computations*. 2nd ed. Baltimore: Johns Hopkins University Press.

Greene, W. H. (1993). *Econometric Analysis*. 2nd ed. New York: Macmillan.

Hamilton, J. D. (1994). *Time Series Analysis*. Princeton, NJ: Princeton University Press.

Hannan, E. J., and Quinn, B. G. (1979). "The Determination of the Order of an Autoregression." *Journal of the Royal Statistical Society, Series B* 41:190–195.

Harvey, A. C. (1981). *The Econometric Analysis of Time Series*. New York: John Wiley & Sons.

Harvey, A. C. (1990). *The Econometric Analysis of Time Series*. 2nd ed. Cambridge, MA: MIT Press.

Harvey, A. C., and McAvinchey, I. D. (1978). "The Small Sample Efficiency of Two-Step Estimators in Regression Models with Autoregressive Disturbances." Discussion Paper No. 78-10, University of British Columbia.

Harvey, A. C., and Phillips, G. D. A. (1979). "Maximum Likelihood Estimation of Regression Models with Autoregressive–Moving Average Disturbances." *Biometrika* 66:49–58.

Hayashi, F. (2000). *Econometrics*. Princeton, NJ: Princeton University Press.

Hildreth, C., and Lu, J. Y. (1960). *Demand Relations with Autocorrelated Disturbances*. Technical Report 276, Michigan State University Agricultural Experiment Station.

Hobijn, B., Franses, P. H., and Ooms, M. (2004). "Generalization of the KPSS-Test for Stationarity." *Statistica Neerlandica* 58:483–502.

Hurvich, C. M., and Tsai, C.-L. (1989). "Regression and Time Series Model Selection in Small Samples." *Biometrika* 76:297–307.

Inder, B. A. (1984). "Finite-Sample Power of Tests for Autocorrelation in Models Containing Lagged Dependent Variables." *Economics Letters* 14:179–185.

Inder, B. A. (1986). "An Approximation to the Null Distribution of the Durbin-Watson Statistic in Models Containing Lagged Dependent Variables." *Econometric Theory* 2:413–428.

Jarque, C. M., and Bera, A. K. (1980). "Efficient Tests for Normality, Homoskedasticity, and Serial Independence of Regression Residuals." *Economics Letters* 6:255–259.

Johansen, S. (1988). "Statistical Analysis of Cointegration Vectors." *Journal of Economic Dynamics and Control* 12:231–254.

Johansen, S. (1991). "Estimation and Hypothesis Testing of Cointegration Vectors in Gaussian Vector Autoregressive Models." *Econometrica* 59:1551–1580.

Johnston, J. (1972). *Econometric Methods*. 2nd ed. New York: McGraw-Hill.

Jones, R. H. (1980). "Maximum Likelihood Fitting of ARMA Models to Time Series with Missing Observations." *Technometrics* 22:389–396.

Judge, G. G., Griffiths, W. E., Hill, R. C., Lütkepohl, H., and Lee, T.-C. (1985). *The Theory and Practice of Econometrics*. 2nd ed. New York: John Wiley & Sons.

Kanzler, L. (1999). "Very Fast and Correctly Sized Estimation of the BDS Statistic." Working paper, Department of Economics, Christ Church, University of Oxford.

King, M. L., and Wu, P. X. (1991). "Small-Disturbance Asymptotics and the Durbin-Watson and Related Tests in the Dynamic Regression Model." *Journal of Econometrics* 47:145–152.

Kwiatkowski, D., Phillips, P. C. B., Schmidt, P., and Shin, Y. (1992). "Testing the Null Hypothesis of Stationarity against the Alternative of a Unit Root." *Journal of Econometrics* 54:159–178.

Lee, J. H., and King, M. L. (1993). "A Locally Most Mean Powerful Based Score Test for ARCH and GARCH Regression Disturbances." *Journal of Business and Economic Statistics* 11:17–27.

L'Esperance, W. L., Chall, D., and Taylor, D. (1976). "An Algorithm for Determining the Distribution Function of the Durbin-Watson Test Statistic." *Econometrica* 44:1325–1326.

MacKinnon, J. G. (1991). "Critical Values for Cointegration Tests." In *Long-Run Economic Relationships: Readings in Cointegration*, edited by R. F. Engle, and C. W. J. Granger, 267–276. Oxford: Oxford University Press.

Maddala, G. S. (1977). *Econometrics*. New York: McGraw-Hill.

Maeshiro, A. (1976). "Autoregressive Transformation, Trended Independent Variables and Autocorrelated Disturbance Terms." *Review of Economics and Statistics* 58:497–500.

McLeod, A. I., and Li, W. K. (1983). "Diagnostic Checking ARMA Time Series Models Using Squared-Residual Autocorrelations." *Journal of Time Series Analysis* 4:269–273.

Nelson, C. R., and Plosser, C. I. (1982). "Trends and Random Walks in Macroeconomic Time Series: Some Evidence and Implications." *Journal of Monetary Economics* 10:139–162.

Nelson, D. B. (1990). "Stationarity and Persistence in the GARCH(1,1) Model." *Econometric Theory* 6:318–334.

Nelson, D. B. (1991). "Conditional Heteroskedasticity in Asset Returns: A New Approach." *Econometrica* 59:347–370.

Nelson, D. B., and Cao, C. Q. (1992). "Inequality Constraints in the Univariate GARCH Model." *Journal of Business and Economic Statistics* 10:229–235.

Newey, W. K., and West, D. W. (1987). "A Simple, Positive Semi-definite, Heteroskedasticity and Autocorrelation Consistent Covariance Matrix." *Econometrica* 55:703–708.

Newey, W. K., and West, D. W. (1994). "Automatic Lag Selection in Covariance Matrix Estimation." *Review of Economic Studies* 61:631–653.

Ng, S., and Perron, P. (2001). "Lag Length Selection and the Construction of Unit Root Tests with Good Size and Power." *Econometrica* 69:1519–1554.

Park, R. E., and Mitchell, B. M. (1980). "Estimating the Autocorrelated Error Model with Trended Data." *Journal of Econometrics* 13:185–201.

Phillips, P. C. B. (1987). "Time Series Regression with a Unit Root." *Econometrica* 55:277–301.

Phillips, P. C. B., and Ouliaris, S. (1990). "Asymptotic Properties of Residual Based Tests for Cointegration." *Econometrica* 58:165–193.

Phillips, P. C. B., and Perron, P. (1988). "Testing for a Unit Root in Time Series Regression." *Biometrika* 75:335–346.

Prais, S. J., and Winsten, C. B. (1954). "Trend Estimators and Serial Correlation." Cowles Commission Discussion Paper No. 383.

Ramsey, J. B. (1969). "Tests for Specification Errors in Classical Linear Least Squares Regression Analysis." *Journal of the Royal Statistical Society, Series B* 31:350–371.

Said, S. E., and Dickey, D. A. (1984). "Testing for Unit Roots in ARMA Models of Unknown Order." *Biometrika* 71:599–607.

Savin, N. E., and White, K. J. (1978). "Testing for Autocorrelation with Missing Observations." *Econometrica* 46:59–67.

Schwarz, G. (1978). "Estimating the Dimension of a Model." *Annals of Statistics* 6:461–464.

Schwert, G. (1989). "Tests for Unit Roots: A Monte Carlo Investigation." *Journal of Business and Economic Statistics* 7:147–159.

Shin, Y. (1994). "A Residual-Based Test of the Null of Cointegration against the Alternative of No Cointegration." *Econometric Theory* 10:91–115. doi:10.1017/S0266466600008240.

Shively, T. S. (1990). "Fast Evaluation of the Distribution of the Durbin-Watson and Other Invariant Test Statistics in Time Series Regression." *Journal of the American Statistical Association* 85:676–685.

Spitzer, J. J. (1979). "Small-Sample Properties of Nonlinear Least Squares and Maximum Likelihood Estimators in the Context of Autocorrelated Errors." *Journal of the American Statistical Association* 74:41–47.

Stock, J. H. (1994). "Unit Roots, Structural Breaks, and Trends." In *Handbook of Econometrics*, edited by R. F. Engle, and D. L. McFadden, 2739–2841. Amsterdam: North-Holland.

Stock, J. H., and Watson, M. W. (1988). "Testing for Common Trends." *Journal of the American Statistical Association* 83:1097–1107.

Stock, J. H., and Watson, M. W. (2002). *Introduction to Econometrics*. 3rd ed. Reading, MA: Addison-Wesley.

Theil, H. (1971). *Principles of Econometrics*. New York: John Wiley & Sons.

Vinod, H. D. (1973). "Generalization of the Durbin-Watson Statistic for Higher Order Autoregressive Process." *Communications in Statistics* 2:115–144.

Wallis, K. F. (1972). "Testing for Fourth Order Autocorrelation in Quarterly Regression Equations." *Econometrica* 40:617–636.

White, H. (1980). "A Heteroskedasticity-Consistent Covariance Matrix Estimator and a Direct Test for Heteroskedasticity." *Econometrica* 48:817–838.

White, H. (1982). "Maximum Likelihood Estimation of Misspecified Models." *Econometrica* 50:1–25.

White, K. J. (1992). "The Durbin-Watson Test for Autocorrelation in Nonlinear Models." *Review of Economics and Statistics* 74:370–373.

Wong, H., and Li, W. K. (1995). "Portmanteau Test for Conditional Heteroscedasticity, Using Ranks of Squared Residuals." *Journal of Applied Statistics* 22:121–134.

Zakoian, J. M. (1994). "Threshold Heteroscedastic Models." *Journal of Economic Dynamics and Control* 18:931–955.

Chapter 10
The COMPUTAB Procedure

Contents

Overview: COMPUTAB Procedure **482**
Getting Started: COMPUTAB Procedure **482**
Producing a Simple Report 482
Using PROC COMPUTAB 484
Defining Report Layout 484
Adding Computed Rows and Columns 485
Enhancing the Report 486
Syntax: COMPUTAB Procedure **487**
Functional Summary 488
PROC COMPUTAB Statement 489
COLUMNS Statement 491
ROWS Statement 493
CELL Statement 494
INIT Statement 495
Programming Statements 496
BY Statement 496
SUMBY Statement 497
NOTRANS Option 498
Details: COMPUTAB Procedure **498**
Program Flow Example 498
Order of Calculations 501
Column Selection 503
Controlling Execution within Row and Column Blocks 504
Program Flow 505
Direct Access to Table Cells 506
Reserved Words 507
Missing Values 507
OUT= Data Set 507
Examples: COMPUTAB Procedure **508**
Example 10.1: Using Programming Statements 508
Example 10.2: Enhancing a Report 510
Example 10.3: Comparison of Actual and Budget 516
Example 10.4: Consolidations 518
Example 10.5: Creating an Output Data Set 524
Example 10.6: A What-If Market Analysis 526
Example 10.7: Cash Flows 530

Overview: COMPUTAB Procedure

The COMPUTAB (computing and tabular reporting) procedure produces tabular reports generated using a programmable data table.

The COMPUTAB procedure is especially useful when you need both the power of a programmable spreadsheet and a report generation system, but you want to set up a program to run in a batch mode and generate routine reports.

With PROC COMPUTAB, you can select a subset of observations from the input data set, define the format of a table, operate on its row and column values, and create new columns and rows. Access to individual table values is available when needed.

The COMPUTAB procedure can tailor reports to almost any desired specification and provide consolidation reports over summarization variables. The generated report values can be stored in an output data set. PROC COMPUTAB is especially useful in creating tabular reports such as income statements, balance sheets, and other row and column reports.

Getting Started: COMPUTAB Procedure

The following example shows the different types of reports that can be generated by PROC COMPUTAB.

Suppose a company has monthly expense data on three of its divisions and wants to produce the year-to-date expense report shown in Figure 10.1. This section starts out with the default report produced by the COMPUTAB procedure and modifies it until the desired report is achieved.

Figure 10.1 Year-to-Date Expense Report

Year to Date Expenses

	Division A	Division B	Division C	All Divisions
Travel Expenses within U.S.	18700	211000	12800	$242,500
Advertising	18500	176000	34500	$229,000
Permanent Staff Salaries	186000	1270000	201000	$1,657,000
Benefits Including Insurance	3900	11100	17500	$32,500
Total	227100	1668100	265800	$2,161,000

Producing a Simple Report

Without any specifications, the COMPUTAB procedure transposes and prints the input data set. The variables in the input data set become rows in the report, and the observations in the input data set become columns. The variable names are used as the row titles. The column headings default to COL1 through COLn. For example, the following input data set contains the monthly expenses reported by different divisions of the company:

```
data report;
   length compdiv $ 1;
   input compdiv $ date:date7. salary travel insure advrtise;
   format date date7.;
   label travel   = 'Travel Expenses within U.S.'
         advrtise = 'Advertising'
         salary   = 'Permanent Staff Salaries'
         insure   = 'Benefits Including Insurance';
datalines;
A 31JAN1989 95000  10500  2000 6500
B 31JAN1989 668000 112000 5600 90000
C 31JAN1989 105000 6800   9000 18500
A 28FEB1989 91000  8200   1900 12000
B 28FEB1989 602000 99000  5500 86000
C 28FEB1989 96000  6000   8500 16000
;
```

You can get a listing of the data set by using the PRINT procedure, as follows:

```
title 'Listing of Monthly Divisional Expense Data';
proc print data=report;
run;
```

Figure 10.2 Listing of Data Set by PROC PRINT

Listing of Monthly Divisional Expense Data

Obs	compdiv	date	salary	travel	insure	advrtise
1	A	31JAN89	95000	10500	2000	6500
2	B	31JAN89	668000	112000	5600	90000
3	C	31JAN89	105000	6800	9000	18500
4	A	28FEB89	91000	8200	1900	12000
5	B	28FEB89	602000	99000	5500	86000
6	C	28FEB89	96000	6000	8500	16000

To get a simple, transposed report of the same data set, use the following PROC COMPUTAB statement:

```
title 'Monthly Divisional Expense Report';
proc computab data=report;
run;
```

Figure 10.3 Listing of Data Set by PROC COMPUTAB

Monthly Divisional Expense Report

	COL1	COL2	COL3	COL4	COL5	COL6
compdiv	A	B	C	A	B	C
date	31JAN89	31JAN89	31JAN89	28FEB89	28FEB89	28FEB89
salary	95000.00	668000.00	105000.00	91000.00	602000.00	96000.00
travel	10500.00	112000.00	6800.00	8200.00	99000.00	6000.00
insure	2000.00	5600.00	9000.00	1900.00	5500.00	8500.00
advrtise	6500.00	90000.00	18500.00	12000.00	86000.00	16000.00

Using PROC COMPUTAB

The COMPUTAB procedure is best understood by examining the following features:

- definition of the report layout with ROWS and COLUMNS statements
- input block
- row blocks
- column blocks

PROC COMPUTAB builds a table according to the specifications in the ROWS and COLUMNS statements. Row names and column names define the rows and columns of the table. Options in the ROWS and COLUMNS statements control titles, spacing, and formatting.

The input block places input observations into the appropriate columns of the report. It consists of programming statements used to select observations to be included in the report, to determine the column into which the observation should be placed, and to calculate row and column values that are not in the input data set.

Row blocks and column blocks perform operations on the values of rows and columns of the report after the input block has executed. Row blocks are a block of programming statements labeled ROW*xxxxx*: that create or modify row values; column blocks are a block of programming statements labeled COL*xxxxx*: that create or modify column values. Row and column blocks can make multiple passes through the report for final calculations.

For most reports, these features are sufficient. More complicated applications might require knowledge of the program data vector and the COMPUTAB data table. These topics are discussed in the section "Details: COMPUTAB Procedure" on page 498.

Defining Report Layout

ROWS and COLUMNS statements define the rows and columns of the report. The order of row and column names in these statements determines the order of rows and columns in the report. Additional ROWS and COLUMNS statements can be used to specify row and column formatting options.

The following statements select and order the variables from the input data set and produce the report in Figure 10.4:

```
proc computab data=report;
   rows travel advrtise salary;
run;
```

Figure 10.4 Report Produced Using a ROWS Statement

Monthly Divisional Expense Report

	COL1	COL2	COL3	COL4	COL5	COL6
TRAVEL	10500.00	112000.00	6800.00	8200.00	99000.00	6000.00
ADVRTISE	6500.00	90000.00	18500.00	12000.00	86000.00	16000.00
SALARY	95000.00	668000.00	105000.00	91000.00	602000.00	96000.00

When a COLUMNS statement is not specified, each observation becomes a new column. If you use a COLUMNS statement, you must specify to which column each observation belongs by using program statements for column selection. When more than one observation is selected for the same column, values are summed.

The following statements produce Figure 10.5:

```
proc computab data= report;
   rows travel advrtise salary insure;
   columns a b c;
   *----select column for company division,
        based on value of compdiv----*;
   a = compdiv = 'A';
   b = compdiv = 'B';
   c = compdiv = 'C';
run;
```

The statement A=COMPDIV='A'; illustrates the use of logical operators as a selection technique. If COMPDIV='A', then the current observation is added to the A column. See *SAS Language: Reference, Version 6, First Edition* for more information about logical operators.

Figure 10.5 Report Produced Using ROWS and COLUMNS Statements

Monthly Divisional Expense Report

	A	B	C
TRAVEL	18700.00	211000.00	12800.00
ADVRTISE	18500.00	176000.00	34500.00
SALARY	186000.00	1270000.0	201000.00
INSURE	3900.00	11100.00	17500.00

Adding Computed Rows and Columns

In addition to the variables and observations in the input data set, you can create additional rows or columns by using SAS programming statements in PROC COMPUTAB. You can do the following:

- modify input data and select columns in the input block
- create or modify columns in column blocks

- create or modify rows in row blocks

The following statements add one computed row (SUM) and one computed column (TOTAL) to the report in Figure 10.5. In the input block the logical operators indicate the observations that correspond to each column of the report. After the input block reads in the values from the input data set, the column block creates the column variable TOTAL by summing the columns A, B, and C. The additional row variable, SUM, is calculated as the sum of the other rows. The result is shown in Figure 10.6.

```
proc computab data= report;
   rows travel advrtise salary insure sum;
   columns a b c total;
   a = compdiv = 'A';
   b = compdiv = 'B';
   c = compdiv = 'C';
   colblk: total = a + b + c;
   rowblk: sum   = travel + advrtise + salary + insure;
run;
```

Figure 10.6 Report Produced Using Row and Column Blocks

Monthly Divisional Expense Report

	A	B	C	TOTAL
TRAVEL	18700.00	211000.00	12800.00	242500.00
ADVRTISE	18500.00	176000.00	34500.00	229000.00
SALARY	186000.00	1270000.0	201000.00	1657000.0
INSURE	3900.00	11100.00	17500.00	32500.00
SUM	227100.00	1668100.0	265800.00	2161000.0

Enhancing the Report

To enhance the appearance of the final report, you can use the following:

- TITLE and LABEL statements
- column headings
- row titles
- row and column spacing control
- overlining and underlining
- formats

The following example enhances the report in the previous example. The enhanced report is shown in Figure 10.7.

The TITLE statement assigns the report title. The column headings in Figure 10.7 (Division A, Division B, and Division C) are assigned in the first COLUMNS statement by “Division” _name_ specification. The

second COLUMNS statement assigns the column heading (“All” “Divisions”), sets the spacing (+4), and formats the values in the TOTAL column.

Similarly, the first ROWS statement uses previously assigned variable labels for row labels by specifying the _LABEL_ option. The DUL option in the second ROWS statement double-underlines the INSURE row. The third ROWS statement assigns the row label TOTAL to the SUM row.

```
title 'Year to Date Expenses';

proc computab cwidth=8 cdec=0;

   columns a  b  c / 'Division' _name_;
   columns total / 'All' 'Divisions' +4 f=dollar10.0;

   rows travel advrtise salary insure / _label_;
   rows insure / dul;
   rows sum / 'Total';

   a = compdiv = 'A';
   b = compdiv = 'B';
   c = compdiv = 'C';

   colblk: total = a + b + c;
   rowblk: sum   = travel + advrtise + salary + insure;
run;
```

Figure 10.7 Report Produced by PROC COMPUTAB Using Enhancements

Year to Date Expenses

	Division A	Division B	Division C	All Divisions
Travel Expenses within U.S.	18700	211000	12800	$242,500
Advertising	18500	176000	34500	$229,000
Permanent Staff Salaries	186000	1270000	201000	$1,657,000
Benefits Including Insurance	3900	11100	17500	$32,500
Total	227100	1668100	265800	$2,161,000

Syntax: COMPUTAB Procedure

The following statements are used with the COMPUTAB procedure:

PROC COMPUTAB *options* ;
BY *variables* ;
COLUMNS *names / options* ;
ROWS *names / options* ;
CELL *names / FORMAT= format* ;
INIT *anchor-name locator-name values locator-name values* ;
programming *statements* ;
SUMBY *variables* ;

The PROC COMPUTAB statement is the only required statement. The COLUMNS, ROWS, and CELL statements define the COMPUTAB table. The INIT statement initializes the COMPUTAB table values. Programming statements process COMPUTAB table values. The BY and SUMBY statements provide BY-group processing and consolidation (roll up) tables.

Functional Summary

Table 10.1 summarizes the COMPUTAB procedure statements and options.

Table 10.1 COMPUTAB Functional Summary

Description	Statement	Option
Statements		
specify BY-group processing	BY	
specify the format for printing a particular cell	CELL	
define columns of the report	COLUMNS	
initialize values in the COMPUTAB data table	INIT	
define rows of the report	ROWS	
produce consolidation tables	SUMBY	
Data Set Options		
specify the input data set	COMPUTAB	DATA=
specify an output data set	COMPUTAB	OUT=
Input Options		
specify a value to use when testing for 0	COMPUTAB	FUZZ=
initialize the data table to missing	COMPUTAB	INITMISS
prevent the transposition of the input data set	COMPUTAB	NOTRANS
Printing Control Options		
suppress printing of the listed columns	COLUMNS	NOPRINT
suppress all printed output	COMPUTAB	NOPRINT
suppress printing of the listed rows	ROWS	NOPRINT
suppress columns with all 0 or missing values	COLUMNS	NOZERO
suppress rows with all 0 or missing values	ROWS	NOZERO
list option values	COMPUTAB	OPTIONS
overprint titles, values, overlining, and underlining associated with listed rows	ROWS	OVERPRINT
print only consolidation tables	COMPUTAB	SUMONLY
Report Formatting Options		
specify number of decimal places to print	COMPUTAB	CDEC=
specify number of spaces between columns	COMPUTAB	CSPACE=
specify column width for the report	COMPUTAB	CWIDTH=
overline the listed rows with double lines	ROWS	DOL
underline the listed rows with double lines	ROWS	DUL

Description	Statement	Option
specify a format for printing the cell values	CELL	FORMAT=
specify a format for printing column values	COLUMNS	FORMAT=
specify a format for printing the row values	ROWS	FORMAT=
left-align the column headings	COLUMNS	LJC
left-justify character rows in each column	ROWS	LJC
specify indentation from the margin	ROWS	+n
suppress printing of row titles on later pages	COMPUTAB	NORTR
overline the listed rows with a single line	ROWS	OL
start a new page before printing the listed rows	ROWS	_PAGE_
specify number of spaces before row titles	COMPUTAB	RTS=
print a blank row	ROWS	SKIP
underline the listed rows with a single line	ROWS	UL
specify text to print if column is 0 or missing	COLUMNS	ZERO=
specify text to print if row is 0 or missing	ROWS	ZERO=
Row and Column Type Options		
specify that columns contain character data	COLUMNS	CHAR
specify that rows contain character data	ROWS	CHAR
Options for Column Headings		
specify literal column headings	COLUMNS	‘column heading’
use variable labels in column headings	COLUMNS	_LABEL_
specify a master title centered over columns	COLUMNS	MTITLE=
use column names in column headings	COLUMNS	_NAME_
Options for Row Titling		
use labels in row titles	ROWS	_LABEL_
use row names in row titles	ROWS	_NAME_
specify literal row titles	ROWS	‘row title’

PROC COMPUTAB Statement

PROC COMPUTAB *options* **;**

The following options can be used in the PROC COMPUTAB statement.

Input Options

DATA=*SAS-data-set*
names the SAS data set that contains the input data. If this option is not specified, the last created data set is used. If you are not reading a data set, use DATA=_NULL_.

FUZZ=*value*
specifies the criterion to use when testing for 0. If a number is within the FUZZ= value of 0, the number is set to 0.

INITMISS
initializes the COMPUTAB data table to missing rather than to 0. The COMPUTAB data table is discussed further in the section "Details: COMPUTAB Procedure" on page 498.

NOTRANSPOSE
NOTRANS
prevents the transposition of the input data set in building the COMPUTAB report tables. The NOTRANS option causes input data set variables to appear among the columns of the report rather than among the rows.

Report Formatting Options

The formatting options specify default values. Many of the formatting options can be modified for specific columns in COLUMNS statements and for rows in ROWS statements.

CDEC=*d*
specifies the default number of decimal places for printing. The default is CDEC=2. See the FORMAT= option in the sections on COLUMN, ROWS, and CELL statements later in this chapter.

CSPACE=*n*
specifies the default number of spaces to insert between columns. The value of the CSPACE= option is used as the default value for the +*n* option in the COLUMNS statement. The default is CSPACE=2.

CWIDTH=*w*
specifies a default column width for the report. The default is CWIDTH=9. The width must be in the range of 1–32.

NORTR
suppresses the printing of row titles on each page. The NORTR (no row-title repeat) option is useful to suppress row titles when report pages are to be joined together in a larger report.

RTS=*n*
specifies the default number of spaces to be inserted before row titles when row titles appear after the first printed column. The default row-title spacing is RTS=2.

Output Options

NOPRINT
suppresses all printed output. Use the NOPRINT option with the OUT= option to produce an output data set but no printed reports.

OPTIONS
lists PROC COMPUTAB option values. The option values appear on a separate page preceding the procedure's normal output.

OUT=*SAS-data-set*
names the SAS data set to contain the output data. See the section "Details: COMPUTAB Procedure" on page 498 for a description of the structure of the output data set.

SUMONLY
suppresses printing of detailed reports. When the SUMONLY option is used, PROC COMPUTAB generates and prints only consolidation tables as specified in the SUMBY statement.

COLUMNS Statement

COLUMNS *column-list / options* **;**

COLUMNS statements define the columns of the report. The COLUMNS statement can be abbreviated COLUMN, COLS, or COL.

The specified column names must be valid SAS names. Abbreviated lists, as described in *SAS Language: Reference*, can also be used.

You can use as many COLUMNS statements as you need. A COLUMNS statement can describe more than one column, and one column of the report can be described with several different COLUMNS statements. The order of the columns on the report is determined by the order of appearance of column names in COLUMNS statements. The first occurrence of the name determines where in the sequence of columns a particular column is located.

The following options can be used in the COLUMNS statement.

Option for Column Type

CHAR
indicates that the columns contain character data.

Options for Column Headings

You can specify as many lines of column headings as needed. If no options are specified, the column names from the COLUMNS statement are used as column headings. Any or all of the following options can be used in a column heading:

"column heading"
specifies that the characters enclosed in quotes are to be used in the column heading for the variable or variables listed in the COLUMNS statement. Each quoted string appears on a separate line of the heading.

LABEL
uses labels, if provided, in the heading for the column or columns listed in the COLUMNS statement. If a label has not been provided, the name of the column is used. See *SAS Language: Reference* for information about the LABEL statement.

MTITLE=*"text"*
specifies that the string of characters enclosed in quotes is a master title to be centered over all the columns listed in the COLUMNS statement. The list of columns must be consecutive. Special characters ("+", "*", "=", and so forth) placed on either side of the text expand to fill the space. The MTITLE= option can be abbreviated M=.

NAME
uses column names in column headings for the columns listed in the COLUMNS statement. This option allows headings (*"text"*) and names to be combined in a heading.

Options for Column Print Control

+n
inserts *n* spaces before each column listed in the COLUMNS statement. The default spacing is given by the CSPACE= option in the PROC COMPUTAB statement.

NOPRINT
suppresses printing of columns listed in the COLUMNS statement. This option enables you to create columns to be used for intermediate calculations without having those columns printed.

NOZERO
suppresses printing of columns when all the values in a column are 0 or missing. Numbers within the FUZZ= value of 0 are treated as 0.

PAGE
starts a new page of the report before printing each of the columns in the list that follows.

TITLES
prints row titles before each column in the list. The _TITLES_ option can be abbreviated as _TITLE_.

Options for Column Formatting

Column formats override row formats for particular table cells only when the input data set is not transposed (when the NOTRANS option is specified).

FORMAT=*format*
specifies a format for printing the values of the columns listed in the COLUMNS statement. The FORMAT= option can be abbreviated F=.

LJC
left-justifies the column headings for the columns listed. By default, columns are right-justified. When the LJC (left-justify character) option is used, any character row values in the column are also left-justified rather than right-justified.

ZERO=*"text"*
substitutes *"text"* when the value in the column is 0 or missing.

ROWS Statement

ROWS *row-list / options* **;**

ROWS statements define the rows of the report. The ROWS statement can be abbreviated ROW.

The specified row names must be valid SAS names. Abbreviated lists, as described in *SAS Language: Reference*, can also be used.

You can use as many ROWS statements as you need. A ROWS statement can describe more than one row, and one row of the report can be described with several different ROWS statements. The order of the rows in the report is determined by the order of appearance of row names in ROWS statements. The first occurrence of the name determines where the row is located.

The following options can be used in the ROWS statement.

Option for Row Type

CHAR
indicates that the rows contain character data.

Options for Row Titling

You can specify as many lines of row titles as needed. If no options are specified, the names from the ROWS statement are used as row titles. Any or all of the following options can be used in a row title:

LABEL
uses labels as row titles for the row or rows listed in the ROWS statement. If a label is not provided, the name of the row is substituted. See *SAS Language: Reference* for more information about the LABEL statement.

NAME
uses row names in row titles for the row or rows listed in the ROWS statement.

"row title"
specifies that the string of characters enclosed in quotes is to be used in the row title for the row or rows listed in the ROWS statement. Each quoted string appears on a separate line of the heading.

Options for Row Print Control

+n
indents *n* spaces from the margin for the rows in the ROWS statement.

DOL
overlines the rows listed in the ROWS statement with double lines. Overlines are printed on the line before any row titles or data for the row.

DUL
underlines the rows listed in the ROWS statement with double lines. Underlines are printed on the line after the data for the row. A row can have both an underline and an overline option.

NOPRINT
suppresses printing of the rows listed in the ROWS statement. This option enables you to create rows to be used for intermediate calculations without having those rows printed.

NOZERO
suppresses the printing of a row when all the values are 0 or missing.

OL
overlines the rows listed in the ROWS statement with a single line. Overlines are printed on the line before any row titles or data for the row.

OVERPRINT
overprints titles, values, overlining, and underlining associated with rows listed in the ROWS statement. The OVERPRINT option can be abbreviated OVP. This option is valid only when the system option OVP is in effect. See *SAS Language: Reference* for more information about the OVP option.

PAGE
starts a new page of the report before printing these rows.

SKIP
prints a blank line after the data lines for these rows.

UL
underlines the rows listed in the ROWS statement with a single line. Underlines are printed on the line after the data for the row. A row can have both an underline and an overline option.

Options for Row Formatting

Row formatting options take precedence over column-formatting options when the input data set is transposed. Row print width can never be wider than column width. Character values are truncated on the right.

FORMAT= *format*
specifies a format for printing the values of the rows listed in the ROWS statement. The FORMAT= option can be abbreviated as F=.

LJC
left-justifies character rows in each column.

ZERO=*"text"*
substitutes *text* when the value in the row is 0 or missing.

CELL Statement

CELL *cell_names / FORMAT= format* **;**

The CELL statement specifies the format for printing a particular cell in the COMPUTAB data table. Cell variable names are compound SAS names of the form *name1.name2*, where *name1* is the name of a row variable and *name2* is the name of a column variable. Formats specified with the FORMAT= option in CELL statements override formats specified in ROWS and COLUMNS statements.

INIT Statement

INIT *anchor-name [locator-name] values [locator-name values]* **;**

The INIT statement initializes values in the COMPUTAB data table at the beginning of each execution of the procedure and at the beginning of each BY group if a BY statement is present.

The INIT statement in the COMPUTAB procedure is similar in function to the RETAIN statement in the DATA step, which initializes values in the program data vector. The INIT statement can be used at any point after the variable to which it refers has been defined in COLUMNS or ROWS statements. Each INIT statement initializes one row or column. Any number of INIT statements can be used.

The first term after the keyword INIT, *anchor-name*, anchors initialization to a row or column. If *anchor-name* is a row name, then all *locator-name* values in the statement are columns of that row. If *anchor-name* is a column name, then all *locator-name* values in the statement are rows of that column.

The following terms appear in the INIT statement:

anchor-name names the row or column in which values are to be initialized. This term is required.

locator-name identifies the starting column in the row (or starting row in the column) into which values are to be placed. For example, in a table with a row SALES and a column for each month of the year, the following statement initializes values for columns JAN, FEB, and JUN:

```
init sales jan 500 feb 600 jun 800;
```

If you do not specify *locator-name* values, the first value is placed into the first row or column, the second value into the second row or column, and so on. For example, the following statement assigns 500 to column JAN, 600 to FEB, and 450 to MAR:

```
init sales 500 600 450;
```

+n specifies the number of columns in a row (or rows in a column) that are to be skipped when initializing values. For example, the following statement assigns 500 to JAN and 900 to JUL:

```
init sales jan 500 +5 900;
```

n*value assigns *value* to *n* columns in the row (or rows in the column). For example, both of the following statements assign 500 to columns JAN through JUN and 1000 to JUL through DEC:

```
init sales jan 6*500 jul 6*1000;
```

```
init sales 6*500 6*1000;
```

Programming Statements

You can use most SAS programming statements the same way you use them in the DATA step. Also, all DATA step functions can be used in the COMPUTAB procedure.

Lines written by the PUT statement are not integrated with the COMPUTAB report. PUT statement output is written to the SAS log.

The automatic variable _N_ can be used; its value is the number of observations read or the number read in the current BY group, if a BY statement is used. FIRST.*variable* and LAST.*variable* references cannot be used.

The following statements are also available in PROC COMPUTAB:

```
ABORT                    FORMAT
ARRAY                    GOTO
ATTRIB                   IF-THEN/ELSE
assignment statement     LABEL
CALL                     LINK
DELETE                   PUT
DO                       RETAIN
iterative DO             SELECT
DO UNTIL                 STOP
DO WHILE                 sum statement
END                      TITLE
FOOTNOTE
```

The programming statements can be assigned labels ROW*xxxxx*: or COL*xxxxx*: to indicate the start of a row and column block, respectively. Statements in a row block create or change values in all the columns in the specified rows. Similarly, statements in a column block create or change values in all the rows in the specified columns.

There is an implied RETURN statement before each new row or column block. Thus, the flow of execution does not leave the current row (column) block before the block repeats for all columns (rows.) Row and column variables and nonretained variables are initialized prior to each execution of the block.

The next COL*xxxxx*: label, ROW*xxxxx*: label, or the end of the PROC COMPUTAB step signals the end of a row (column) block. Column blocks and row blocks can be mixed in any order. In some cases, performing calculations in different orders can lead to different results.

See the sections "Program Flow Example" on page 498, "Order of Calculations" on page 501, and "Controlling Execution within Row and Column Blocks" on page 504 for more information.

BY Statement

BY *variables* **;**

A BY statement can be used with PROC COMPUTAB to obtain separate reports for observations in groups defined by the BY variables. At the beginning of each BY group, before PROC COMPUTAB reads any observations, all table values are set to 0 unless the INITMISS option or an INIT statement is specified.

SUMBY Statement

SUMBY *variables* **;**

The SUMBY statement produces consolidation tables for variables whose names are in the SUMBY list. Only one SUMBY statement can be used.

To use a SUMBY statement, you must use a BY statement. The SUMBY and BY variables must be in the same relative order in both statements. For example:

```
by a b c;
sumby a b;
```

This SUMBY statement produces tables that consolidate over values of C within levels of B and over values of B within levels of A. Suppose A has values 1, 2; B has values 1, 2; and C has values 1, 2, 3. Table 10.2 indicates the consolidation tables produced by the SUMBY statement.

Table 10.2 Consolidation Tables Produced by the SUMBY Statement

SUMBY Consolidations	Consolidated BY Groups		
A=1, B=1	C=1	C=2	C=3
A=1, B=2	C=1	C=2	C=3
A=1	B=1, C=1 B=2, C=1	B=1, C=2 B=2, C=2	B=1, C=3 B=2, C=3
A=2, B=1	C=1	C=2	C=3
A=2, B=2	C=1	C=2	C=3
A=2	B=1, C=1 B=2, C=1	B=1, C=2 B=2, C=2	B=1, C=3 B=2, C=3

Two consolidation tables for B are produced for each value of A. The first table consolidates the three tables produced for the values of C while B is 1; the second table consolidates the three tables produced for C while B is 2.

Tables are similarly produced for values of A. Nested consolidation tables are produced for B (as described previously) for each value of A. Thus, this SUMBY statement produces a total of six consolidation tables in addition to the tables produced for each BY group.

To produce a table that consolidates the entire data set (the equivalent of using PROC COMPUTAB with neither BY nor SUMBY statements), use the special name _TOTAL_ as the first entry in the SUMBY variable list. For example,

```
sumby _total_ a b;
```

PROC COMPUTAB then produces consolidation tables for SUMBY variables as well as a consolidation table for all observations.

To produce only consolidation tables, use the SUMONLY option in the PROC COMPUTAB statement.

NOTRANS Option

The NOTRANS option in the PROC COMPUTAB statement prevents the transposition of the input data set. NOTRANS affects the input block, the precedence of row and column options, and the structure of the output data set if the OUT= option is specified.

When the input data set is transposed, input variables are among the rows of the COMPUTAB report, and observations compose columns. The reverse is true if the data set is not transposed; therefore, the input block must select rows to receive data values, and input variables are among the columns.

Variables from the input data set dominate the format specification and data type. When the input data set is transposed, input variables are among the rows of the report, and row options take precedence over column options. When the input data set is not transposed, input variables are among the columns, and column options take precedence over row options.

Variables for the output data set are taken from the dimension (row or column) that contains variables from the input data set. When the input data set is transposed, this dimension is the row dimension; otherwise, the output variables come from the column dimension.

Details: COMPUTAB Procedure

Program Flow Example

This example shows how the COMPUTAB procedure processes observations in the program working storage and the COMPUTAB data table (CDT).

Assume you have three years of figures for sales and cost of goods sold (CGS), and you want to determine total sales and cost of goods sold and calculate gross profit and the profit margin.

```
data example;
   input year sales cgs;
datalines;
1988    83      52
1989   106      85
1990   120     114
;

proc computab data=example;

   columns c88 c89 c90 total;
   rows sales cgs gprofit pctmarg;

   /* calculate gross profit */
   gprofit = sales - cgs;

   /* select a column */
   c88 = year = 1988;
   c89 = year = 1989;
```

```
    c90 = year = 1990;

    /* calculate row totals for sales */
    /* and cost of goods sold */
    col: total = c88 + c89 + c90;

    /* calculate profit margin */
    row: pctmarg = gprofit / cgs * 100;
run;
```

Table 10.3 shows the CDT before any observation is read in. All the columns and rows are defined with the values initialized to 0.

Table 10.3 CDT before Any Input

	C88	C89	C90	TOTAL
SALES	0	0	0	0
CGS	0	0	0	0
GPROFIT	0	0	0	0
PCTMARG	0	0	0	0

When the first input is read in (year=1988, sales=83, and cgs=52), the input block puts the values for SALES and CGS in the C88 column since year=1988. Also the value for the gross profit for that year (GPROFIT) is calculated as indicated in the following statements:

```
gprofit = sales-cgs;
c88 = year = 1988;
c89 = year = 1989;
c90 = year = 1990;
```

Table 10.4 shows the CDT after the first observation is input.

Table 10.4 CDT after First Observation Input (C88=1)

	C88	C89	C90	TOTAL
SALES	83	0	0	0
CGS	52	0	0	0
GPROFIT	31	0	0	0
PCTMARG	0	0	0	0

Similarly, the second observation (year=1989, sales=106, cgs=85) is put in the second column, and the GPROFIT is calculated to be 21. The third observation (year=1990, sales=120, cgs=114) is put in the third column, and the GPROFIT is calculated to be 6. Table 10.5 shows the CDT after all observations are input.

Table 10.5 CDT after All Observations Input

	C88	C89	C90	TOTAL
SALES	83	106	120	0
CGS	52	85	114	0
GPROFIT	31	21	6	0
PCTMARG	0	0	0	0

After the input block is executed for each observation in the input data set, the first row or column block is processed. In this case, the column block is

```
col: total = c88 + c89 + c90;
```

The column block executes for each row, calculating the TOTAL column for each row. Table 10.6 shows the CDT after the column block has executed for the first row (total=83 + 106 + 120). The total sales for the three years is 309.

Table 10.6 CDT after Column Block Executed for First Row

	C88	C89	C90	TOTAL
SALES	83	106	120	309
CGS	52	85	114	0
GPROFIT	31	21	6	0
PCTMARG	0	0	0	0

Table 10.7 shows the CDT after the column block has executed for all rows and the values for total cost of goods sold and total gross profit have been calculated.

Table 10.7 CDT after Column Block Executed for All Rows

	C88	C89	C90	TOTAL
SALES	83	106	120	309
CGS	52	85	114	251
GPROFIT	31	21	6	58
PCTMARG	0	0	0	0

After the column block has been executed for all rows, the next block is processed. The row block is

```
row: pctmarg = gprofit / cgs * 100;
```

The row block executes for each column, calculating the PCTMARG for each year and the total (TOTAL column) for three years. Table 10.8 shows the CDT after the row block has executed for all columns.

Table 10.8 CDT after Row Block Executed for All Columns

	C88	C89	C90	TOTAL
SALES	83	106	120	309
CGS	52	85	114	251
GPROFIT	31	21	6	58
PCTMARG	59.62	24.71	5.26	23.11

Order of Calculations

The COMPUTAB procedure provides alternative programming methods for performing most calculations. New column and row values are formed by adding values from the input data set, directly or with modification, into existing columns or rows. New columns can be formed in the input block or in column blocks. New rows can be formed in the input block or in row blocks.

This example illustrates the different ways to collect totals. Table 10.9 is the total sales report for two products, SALES1 and SALES2, during the years 1988–1990. The values for SALES1 and SALES2 in columns C88, C89, and C90 come from the input data set.

Table 10.9 Total Sales Report

	C88	C89	C90	SALESTOT
SALES1	15	45	80	140
SALES2	30	40	50	120
YRTOT	45	85	130	260

The new column SALESTOT, which is the total sales for each product over three years, can be computed in several different ways:

- in the input block by selecting SALESTOT for each observation:

```
salestot = 1;
```

- in a column block:

```
coltot: salestot = c88 + c89 + c90;
```

In a similar fashion, the new row YRTOT, which is the total sales for each year, can be formed as follows:

- in the input block:

```
yrtot = sales1 + sales2;
```

- in a row block:

```
rowtot: yrtot = sales1 + sales2;
```

Performing some calculations in PROC COMPUTAB in different orders can yield different results, because many operations are not commutative. Be sure to perform calculations in the proper sequence. It might take several column and row blocks to produce the desired report values.

Notice that in the previous example, the grand total for all rows and columns is 260 and is the same whether it is calculated from row subtotals or column subtotals. It makes no difference in this case whether you compute the row block or the column block first.

However, consider the following example where a new column and a new row are formed:

Table 10.10 Report Sensitive to Order of Calculations

	STORE1	STORE2	STORE3	MAX
PRODUCT1	12	13	27	27
PRODUCT2	11	15	14	15
TOTAL	23	28	41	?

The new column MAX contains the maximum value in each row, and the new row TOTAL contains the column totals. MAX is calculated in a column block:

```
col: max = max(store1,store2,store3);
```

TOTAL is calculated in a row block:

```
row: total = product1 + product2;
```

Notice that either of two values, 41 or 42, is possible for the element in column MAX and row TOTAL. If the row block is first, the value is the maximum of the column totals (41). If the column block is first, the value is the sum of the MAX values (42). Whether to compute a column block before a row block can be a critical decision.

Column Selection

The following discussion assumes that the NOTRANS option has not been specified. When NOTRANS is specified, this section applies to rows rather than columns.

If a COLUMNS statement appears in PROC COMPUTAB, a target column must be selected for the incoming observation. If there is no COLUMNS statement, a new column is added for each observation. When a COLUMNS statement is present and the selection criteria fail to designate a column, the current observation is ignored. Faulty column selection can result in columns or entire tables of 0s (or missing values if the INITMISS option is specified).

During execution of the input block, when an observation is read, its values are copied into row variables in the program data vector (PDV).

To select columns, use either the column variable names themselves or the special variable _COL_. Use the column names by setting a column variable equal to some nonzero value. The example in the section "Getting Started: COMPUTAB Procedure" on page 482 uses the logical expression COMPDIV= *value*, and the result is assigned to the corresponding column variable.

```
a = compdiv = 'A';
b = compdiv = 'B';
c = compdiv = 'C';
```

IF statements can also be used to select columns. The following statements are equivalent to the preceding example:

```
if      compdiv = 'A' then a = 1;
else if compdiv = 'B' then b = 1;
else if compdiv = 'C' then c = 1;
```

At the end of the input block for each observation, PROC COMPUTAB multiplies numeric input values by any nonzero selector values and adds the result to selected columns. Character values simply overwrite the contents already in the table. If more than one column is selected, the values are added to each of the selected columns.

Use the _COL_ variable to select a column by assigning the column number to it. The COMPUTAB procedure automatically initializes column variables and sets the _COL_ variable to 0 at the start of each execution of the input block. At the end of the input block for each observation, PROC COMPUTAB examines the value of _COL_. If the value is nonzero and within range, the row variable values are added to the CDT cells of the _COL_th column, for example,

```
data rept;
   input div sales cgs;
datalines;
2   106    85
3   120   114
1    83    52
;

proc computab data=rept;
   row div sales cgs;
   columns div1 div2 div3;
   _col_ = div;
run;
```

The code in this example places the first observation (DIV=2) in column 2 (DIV2), the second observation (DIV=3) in column 3 (DIV3), and the third observation (DIV=1) in column 1 (DIV1).

Controlling Execution within Row and Column Blocks

Row names, column names, and the special variables _ROW_ and _COL_ can be used to limit the execution of programming statements to selected rows or columns. A row block operates on all columns of the table for a specified row unless restricted in some way. Likewise, a column block operates on all rows for a specified column. Use column names or _COL_ in a row block to execute programming statements conditionally; use row names or _ROW_ in a column block.

For example, consider a simple column block that consists of only one statement:

```
col: total = qtr1 + qtr2 + qtr3 + qtr4;
```

This column block assigns a value to each row in the TOTAL column. As each row participates in the execution of a column block, the following changes occur:

- Its row variable in the program data vector is set to 1.
- The value of _ROW_ is the number of the participating row.
- The value from each column of the row is copied from the COMPUTAB data table to the program data vector.

To avoid calculating TOTAL on particular rows, use row names or _ROW_. For example,

```
col: if sales|cost then total = qtr1 + qtr2 + qtr3 + qtr4;
```

or

```
col: if _row_ < 3  then total = qtr1 + qtr2 + qtr3 + qtr4;
```

Row and column blocks can appear in any order, and rows and columns can be selected in each block.

Program Flow

This section describes in detail the different steps in PROC COMPUTAB execution.

Step 1: Define Report Organization and Set Up the COMPUTAB Data Table

Before the COMPUTAB procedure reads in data or executes programming statements, the columns list from the COLUMNS statements and the rows list from the ROWS statements are used to set up a matrix of all columns and rows in the report. This matrix is called the COMPUTAB data table (CDT). When you define columns and rows of the CDT, the COMPUTAB procedure also sets up corresponding variables in working storage called the program data vector (PDV) for programming statements. Data values reside in the CDT but are copied into the program data vector as they are needed for calculations.

Step 2: Select Input Data with Input Block Programming Statements

The input block copies input observations into rows or columns of the CDT. By default, observations go to columns; if the data set is not transposed (the NOTRANS option is specified), observations go to rows of the report table. The input block consists of all executable statements before any ROW*xxxxx*: or COL*xxxxx*: statement label. Use programming statements to perform calculations and select a given observation to be added into the report.

Input Block

The input block is executed once for each observation in the input data set. If there is no input data set, the input block is not executed. The program logic of the input block is as follows:

1. Determine which variables, row or column, are selector variables and which are data variables. Selector variables determine which rows or columns receive values at the end of the block. Data variables contain the values that the selected rows or columns receive. By default, column variables are selector variables and row variables are data variables. If the input data set is not transposed (the NOTRANS option is specified), the roles are reversed.

2. Initialize nonretained program variables (including selector variables) to 0 (or missing if the INITMISS option is specified). Selector variables are temporarily associated with a numeric data item supplied by the procedure. Using these variables to control row and column selection does not affect any other data values.

3. Transfer data from an observation in the data set to data variables in the PDV.

4. Execute the programming statements in the input block by using values from the PDV and storing results in the PDV.

5. Transfer data values from the PDV into the appropriate columns of the CDT. If a selector variable for a row or column has a nonmissing and nonzero value, multiply each PDV value for variables used in the report by the selector variable and add the results to the selected row or column of the CDT.

Step 3: Calculate Final Values by Using Column Blocks and Row Blocks

Column Blocks

A column block is executed once for each row of the CDT. The program logic of a column block is as follows:

1. Indicate the current row by setting the corresponding row variable in the PDV to 1 and the other row variables to missing. Assign the current row number to the special variable _ROW_.
2. Move values from the current row of the CDT to the respective column variables in the PDV.
3. Execute programming statements in the column block by using the column values in the PDV. Here new columns can be calculated and old ones adjusted.
4. Move the values back from the PDV to the current row of the CDT.

Row Blocks

A row block is executed once for each column of the CDT. The program logic of a row block is as follows:

1. Indicate the current column by setting the corresponding column variable in the PDV to 1 and the other column variables to missing. Assign the current column number to the special variable _COL_.
2. Move values from the current column of the CDT to the respective row variables in the PDV.
3. Execute programming statements in the row block by using the row values in the PDV. Here new rows can be calculated and old ones adjusted.
4. Move the values back from the PDV to the current column of the CDT.

See the section "Controlling Execution within Row and Column Blocks" on page 504.

Any number of column blocks and row blocks can be used. Each can include any number of programming statements.

The values of row variables and column variables are determined by the order in which different row-block and column-block programming statements are processed. These values can be modified throughout the COMPUTAB procedure, and final values are printed in the report.

Direct Access to Table Cells

You can insert or retrieve numeric values from specific table cells by using the special reserved name TABLE with row and column subscripts. References to the TABLE have the form

```
TABLE[ row-index, column-index ]
```

where *row-index* and *column-index* can be numbers, character literals, numeric variables, character variables, or expressions that produce a number or a name. If an index is numeric, it must be within range; if it is character, it must name a row or column.

References to TABLE elements can appear on either side of an equal sign in an assignment statement and can be used in a SAS expression.

Reserved Words

Certain words are reserved for special use by the COMPUTAB procedure, and using these words as variable names can lead to syntax errors or warnings. They are:

- COLUMN
- COLUMNS
- COL
- COLS
- _COL_
- ROW
- ROWS
- _ROW_
- INIT
- _N_
- TABLE

Missing Values

Missing values for variables in programming statements are treated in the same way that missing values are treated in the DATA step; that is, missing values used in expressions propagate missing values to the result. See *SAS Language: Reference* for more information about missing values.

Missing values in the input data are treated as follows in the COMPUTAB report table. At the end of the input block, either one or more rows or one or more columns can have been selected to receive values from the program data vector (PDV). Numeric data values from variables in the PDV are added into selected report table rows or columns. If a PDV value is missing, the values already in the selected rows or columns for that variable are unchanged by the current observation. Other values from the current observation are added to table values as usual.

OUT= Data Set

The output data set contains the following variables:

- BY variables
- a numeric variable _TYPE_

- a character variable _NAME_
- the column variables from the COMPUTAB data table

The BY variables contain values for the current BY group. For observations in the output data set from consolidation tables, the consolidated BY variables have missing values.

The special variable _TYPE_ is a numeric variable that can have one of three values: 1, 2, or 3. _TYPE_= 1 indicates observations from the normal report table produced for each BY group; _TYPE_= 2 indicates observations from the _TOTAL_ consolidation table; _TYPE_= 3 indicates observations from other consolidation tables. _TYPE_= 2 and _TYPE_= 3 observations have one or more BY variables with missing values.

The special variable _NAME_ is a character variable of length 8 that contains the row or column name associated with the observation from the report table. If the input data set is transposed, _NAME_ contains column names; otherwise, _NAME_ contains row names.

If the input data set is transposed, the remaining variables in the output data set are row variables from the report table. They are column variables if the input data set is not transposed.

Examples: COMPUTAB Procedure

Example 10.1: Using Programming Statements

This example illustrates two ways of operating on the same input variables and producing the same tabular report. To simplify the example, no report enhancements are shown.

The manager of a hotel chain wants a report that shows the number of bookings at its hotels in each of four cities, the total number of bookings in the current quarter, and the percentage of the total coming from each location for each quarter of the year. Input observations contain the following variables: REPTDATE (report date), LA (number of bookings in Los Angeles), ATL (number of bookings in Atlanta), CH (number of bookings in Chicago), and NY (number of bookings in New York).

The following DATA step creates the SAS data set BOOKINGS:

```
data bookings;
   input reptdate date9. la atl ch ny;
datalines;
01JAN1989 100 110 120 130
01FEB1989 140 150 160 170
01MAR1989 180 190 200 210
01APR1989 220 230 240 250
01MAY1989 260 270 280 290
01JUN1989 300 310 320 330
01JUL1989 340 350 360 370
01AUG1989 380 390 400 410
01SEP1989 420 430 440 450
01OCT1989 460 470 480 490
01NOV1989 500 510 520 530
01DEC1989 540 550 560 570
;
```

The following PROC COMPUTAB statements select columns by setting _COL_ to an appropriate value. The PCT1, PCT2, PCT3, and PCT4 columns represent the percentage contributed by each city to the total for the quarter. These statements produce Output 10.1.1.

```
proc computab data=bookings cspace=1 cwidth=6;

   columns qtr1 pct1 qtr2 pct2 qtr3 pct3 qtr4 pct4;
   columns qtr1-qtr4 / format=6.;
   columns pct1-pct4 / format=6.2;
   rows la atl ch ny total;

   /* column selection */
   _col_ = qtr( reptdate ) * 2 - 1;

   /* copy qtr column values temporarily into pct columns */
   colcopy:
      pct1 = qtr1;
      pct2 = qtr2;
      pct3 = qtr3;
      pct4 = qtr4;

   /* calculate total row for all columns */
   /* calculate percentages for all rows in pct columns only  */
   rowcalc:
      total = la + atl + ch + ny;
      if mod( _col_, 2 ) = 0 then do;
         la  = la  / total * 100;
         atl = atl / total * 100;
         ch  = ch  / total * 100;
         ny  = ny  / total * 100;
         total = 100;
         end;
run;
```

Output 10.1.1 Quarterly Report of Hotel Bookings

Year to Date Expenses

	QTR1	PCT1	QTR2	PCT2	QTR3	PCT3	QTR4	PCT4
LA	420	22.58	780	23.64	1140	24.05	1500	24.27
ATL	450	24.19	810	24.55	1170	24.68	1530	24.76
CH	480	25.81	840	25.45	1200	25.32	1560	25.24
NY	510	27.42	870	26.36	1230	25.95	1590	25.73
TOTAL	1860	100.00	3300	100.00	4740	100.00	6180	100.00

Using the same input data, the next set of statements shows the usefulness of arrays in allowing PROC COMPUTAB to work in two directions at once. Arrays in larger programs can both reduce the amount of program source code and simplify otherwise complex methods of referring to rows and columns. The same report as in Output 10.1.1 is produced.

```
proc computab data=bookings cspace=1 cwidth=6;

   columns qtr1 pct1 qtr2 pct2 qtr3 pct3 qtr4 pct4;
   columns qtr1-qtr4 / format=6.;
   columns pct1-pct4 / format=6.2;
   rows la atl ch ny total;

   array pct[4] pct1-pct4;
   array qt[4] qtr1-qtr4;
   array rowlist[5] la atl ch ny total;

   /* column selection */
   _col_ = qtr(reptdate) * 2 - 1;

   /* copy qtr column values temporarily into pct columns */
   colcopy:
      do i = 1 to 4;
         pct[i] = qt[i];
         end;

   /* calculate total row for all columns */
   /* calculate percentages for all rows in pct columns only */

   rowcalc:
      total = la + atl + ch + ny;
      if mod(_col_,2) = 0 then
         do i = 1 to 5;
            rowlist[i] = rowlist[i] / total * 100;
         end;
run;
```

Example 10.2: Enhancing a Report

The following example shows how a report can be enhanced from a simple listing to a complex report. The simplest COMPUTAB report is a transposed listing of the data in the SAS data set INCOMREP shown in Output 10.2.1. To produce this output, nothing is specified except the PROC COMPUTAB statement and a TITLE statement.

```
data incomrep;
   length type $ 8;
   input type :$8. date :monyy7.
         sales retdis tcos selling randd
         general admin deprec other taxes;
   format date monyy7.;
datalines;
BUDGET JAN1989 4600 300 2200 480 110 500 210 14 -8 510
BUDGET FEB1989 4700 330 2300 500 110 500 200 14  0 480
BUDGET MAR1989 4800 360 2600 500 120 600 250 15  2 520
ACTUAL JAN1989 4900 505 2100 430 130 410 200 14 -8 500
ACTUAL FEB1989 5100 480 2400 510 110 390 230 15  2 490
;

title 'Computab Report without Any Specifications';
proc computab data=incomrep;
run;
```

Output 10.2.1 Simple Report

Computab Report without Any Specifications

	COL1	COL2	COL3	COL4	COL5
type	BUDGET	BUDGET	BUDGET	ACTUAL	ACTUAL
date	JAN1989	FEB1989	MAR1989	JAN1989	FEB1989
sales	4600.00	4700.00	4800.00	4900.00	5100.00
retdis	300.00	330.00	360.00	505.00	480.00
tcos	2200.00	2300.00	2600.00	2100.00	2400.00
selling	480.00	500.00	500.00	430.00	510.00
randd	110.00	110.00	120.00	130.00	110.00
general	500.00	500.00	600.00	410.00	390.00
admin	210.00	200.00	250.00	200.00	230.00
deprec	14.00	14.00	15.00	14.00	15.00
other	-8.00	0.00	2.00	-8.00	2.00
taxes	510.00	480.00	520.00	500.00	490.00

To exclude the budgeted values from your report, select columns for ACTUAL observations only. To remove unwanted variables, specify the variables you want in a ROWS statement.

```
title 'Column Selection by Month';

proc computab data=incomrep;
   rows sales--other;
   columns jana feba mara;
   mnth = month(date);
   if type = 'ACTUAL';
      jana = mnth = 1;
      feba = mnth = 2;
      mara = mnth = 3;
run;
```

The report is shown in Output 10.2.2.

Output 10.2.2 Report That Uses Column Selection Techniques

Column Selection by Month

	JANA	FEBA	MARA
sales	4900.00	5100.00	0.00
retdis	505.00	480.00	0.00
tcos	2100.00	2400.00	0.00
selling	430.00	510.00	0.00
randd	130.00	110.00	0.00
general	410.00	390.00	0.00
admin	200.00	230.00	0.00
deprec	14.00	15.00	0.00
other	-8.00	2.00	0.00

To complete the report, compute new rows from existing rows. This is done in a row block (although it can also be done in the input block). Add a new column (QTR1) that accumulates all the actual data. The NOZERO option suppresses the zero column for March. The output produced by these statements is shown in Output 10.2.3.

```
proc computab data=incomrep;

   /* add a new column to be selected */
   /* qtr1 column will be selected several times */
   columns actual1-actual3 qtr1 / nozero;
   array collist[3] actual1-actual3;
   rows sales retdis netsales tcos grosspft selling randd general
        admin deprec operexp operinc other taxblinc taxes netincom;

   if type='ACTUAL';
   i = month(date);
   if i <= 3 then qtr1 = 1;
   collist[i]=1;

   rowcalc:
      if sales = . then return;
      netsales = sales - retdis;
      grosspft = netsales - tcos;
      operexp  = selling + randd + general + admin + deprec;
      operinc  = grosspft - operexp;
      taxblinc = operinc + other;
      netincom = taxblinc - taxes;
run;
```

Output 10.2.3 Report That Uses Techniques to Compute New Rows

Column Selection by Month

	ACTUAL1	ACTUAL2	QTR1
SALES	4900.00	5100.00	10000.00
RETDIS	505.00	480.00	985.00
NETSALES	4395.00	4620.00	9015.00
TCOS	2100.00	2400.00	4500.00
GROSSPFT	2295.00	2220.00	4515.00
SELLING	430.00	510.00	940.00
RANDD	130.00	110.00	240.00
GENERAL	410.00	390.00	800.00
ADMIN	200.00	230.00	430.00
DEPREC	14.00	15.00	29.00
OPEREXP	1184.00	1255.00	2439.00
OPERINC	1111.00	965.00	2076.00
OTHER	-8.00	2.00	-6.00
TAXBLINC	1103.00	967.00	2070.00
TAXES	500.00	490.00	990.00
NETINCOM	603.00	477.00	1080.00

Now that you have all the numbers calculated, add specifications to improve the report's appearance. Specify titles, row and column labels, and formats. The report produced by these statements is shown in Output 10.2.4.

```
/* now get the report to look the way you want it */
title  'Pro Forma Income Statement';
title2 'XYZ Computer Services, Inc.';
title3 'Period to Date Actual';
title4 'Amounts in Thousands';

proc computab data=incomrep;

   columns actual1-actual3 qtr1 /
            nozero f=comma7. +3 ' ';
   array collist[3] actual1-actual3;
   columns actual1 / 'Jan';
   columns actual2 / 'Feb';
   columns actual3 / 'Mar';
   columns qtr1 / 'Total' 'Qtr 1';
   rows sales    / ' '
                   'Gross Sales ';
   rows retdis   / 'Less Returns & Discounts';
   rows netsales / 'Net Sales'          +3 ol;
   rows tcos     / ' '
                   'Total Cost of Sales';
   rows grosspft / ' '
                   'Gross Profit';
   rows selling  / ' '
                   'Operating Expenses:'
                   '   Selling';
   rows randd    / '   R & D';
   rows general  / +3;
```

```
   rows admin    / '   Administrative';
   rows deprec   / '   Depreciation'       ul;
   rows operexp  / ' '                     skip;
   rows operinc  / 'Operating Income';
   rows other    / 'Other Income/-Expense' ul;
   rows taxblinc / 'Taxable Income';
   rows taxes    / 'Income Taxes'          ul;
   rows netincom / '   Net Income'         dul;

   if type = 'ACTUAL';
   i = month( date );
   collist[i] = 1;

   colcalc:
      qtr1 = actual1 + actual2 + actual3;

   rowcalc:
      if sales = . then return;
      netsales = sales - retdis;
      grosspft = netsales - tcos;
      operexp = selling + randd + general + admin + deprec;
      operinc = grosspft - operexp;
      taxblinc = operinc + other;
      netincom = taxblinc - taxes;
run;
```

Output 10.2.4 Specifying Titles, Row and Column Labels, and Formats

Pro Forma Income Statement
XYZ Computer Services, Inc.
Period to Date Actual
Amounts in Thousands

	Jan	Feb	Total Qtr 1
Gross Sales	4,900	5,100	10,000
Less Returns & Discounts	505	480	985
Net Sales	4,395	4,620	9,015
Total Cost of Sales	2,100	2,400	4,500
Gross Profit	2,295	2,220	4,515
Operating Expenses:			
Selling	430	510	940
R & D	130	110	240
GENERAL	410	390	800
Administrative	200	230	430
Depreciation	14	15	29
	1,184	1,255	2,439
Operating Income	1,111	965	2,076
Other Income/-Expense	-8	2	-6
Taxable Income	1,103	967	2,070
Income Taxes	500	490	990
Net Income	603	477	1,080

Example 10.3: Comparison of Actual and Budget

This example shows a more complex report that compares the actual data with the budgeted values. The same input data as in the previous example is used.

The report produced by these statements is shown in Output 10.3.1. The report shows the values for the current month and the year-to-date totals for budgeted amounts, actual amounts, and the actuals as a percentage of the budgeted amounts. The data have the values for January and February. Therefore, the CURMO variable (current month) in the RETAIN statement is set to 2. The values for the observations where the month of the year is 2 (February) are accumulated for the current month values. The year-to-date values are accumulated from those observations where the month of the year is less than or equal to 2 (January and February).

```
data incomrep;
   length type $ 8;
   input type :$8. date :monyy7.
         sales retdis tcos selling randd
         general admin deprec other taxes;
   format date monyy7.;
datalines;
BUDGET JAN1989 4600 300 2200 480 110 500 210 14 -8 510
BUDGET FEB1989 4700 330 2300 500 110 500 200 14  0 480
BUDGET MAR1989 4800 360 2600 500 120 600 250 15  2 520
ACTUAL JAN1989 4900 505 2100 430 130 410 200 14 -8 500
ACTUAL FEB1989 5100 480 2400 510 110 390 230 15  2 490
;

title  'Pro Forma Income Statement';
title2 'XYZ Computer Services, Inc.';
title3 'Budget Analysis';
title4 'Amounts in Thousands';

options linesize=96;
proc computab data=incomrep;

   columns cmbud cmact cmpct ytdbud ytdact ytdpct /
           zero=' ';
   columns cmbud--cmpct / mtitle='- Current Month: February -';
   columns ytdbud--ytdpct / mtitle='- Year To Date -';
   columns cmbud ytdbud / 'Budget' f=comma6.;
   columns cmact ytdact / 'Actual' f=comma6.;
   columns cmpct ytdpct / '%  ' f=7.2;
   columns cmbud--ytdpct / '-';
   columns ytdbud / _titles_;
   retain curmo 2; /* current month: February */
   rows sales    / ' '
                   'Gross Sales';
   rows retdis   / 'Less Returns & Discounts';
   rows netsales / 'Net Sales'            +3 ol;
   rows tcos     / ' '
                   'Total Cost of Sales';
   rows grosspft / ' '
                   'Gross Profit'         +3;
```

```
   rows selling  / ' '
                   'Operating Expenses:'
                   '   Selling';
   rows randd    / '   R & D';
   rows general  / +3;
   rows admin    / '   Administrative';
   rows deprec   / '   Depreciation'       ul;
   rows operexp  / ' ';
   rows operinc  / 'Operating Income'      ol;
   rows other    / 'Other Income/-Expense' ul;
   rows taxblinc / 'Taxable Income';
   rows taxes    / 'Income Taxes'          ul;
   rows netincom / '   Net Income'         dul;

   cmbud = type = 'BUDGET' & month(date) = curmo;
   cmact = type = 'ACTUAL' & month(date) = curmo;
   ytdbud = type = 'BUDGET' & month(date) <= curmo;
   ytdact = type = 'ACTUAL' & month(date) <= curmo;

   rowcalc:
      if cmpct | ytdpct then return;
      netsales = sales - retdis;
      grosspft = netsales - tcos;
      operexp  = selling + randd + general + admin + deprec;
      operinc  = grosspft - operexp;
      taxblinc = operinc + other;
      netincom = taxblinc - taxes;

   colpct:
      if cmbud  & cmact  then cmpct  = 100 * cmact  / cmbud;
      if ytdbud & ytdact then ytdpct = 100 * ytdact / ytdbud;
run;
```

Output 10.3.1 Report That Uses Specifications to Tailor Output

Pro Forma Income Statement
XYZ Computer Services, Inc.
Budget Analysis
Amounts in Thousands

--- Current Month: February ---				-------- Year To Date ---------		
Budget	Actual	%		Budget	Actual	%
4,700	5,100	108.51	Gross Sales	9,300	10,000	107.53
330	480	145.45	Less Returns & Discounts	630	985	156.35
4,370	4,620	105.72	Net Sales	8,670	9,015	103.98
2,300	2,400	104.35	Total Cost of Sales	4,500	4,500	100.00
2,070	2,220	107.25	Gross Profit	4,170	4,515	108.27
			Operating Expenses:			
500	510	102.00	Selling	980	940	95.92
110	110	100.00	R & D	220	240	109.09
500	390	78.00	GENERAL	1,000	800	80.00
200	230	115.00	Administrative	410	430	104.88
14	15	107.14	Depreciation	28	29	103.57
1,324	1,255	94.79		2,638	2,439	92.46
746	965	129.36	Operating Income	1,532	2,076	135.51
	2		Other Income/-Expense	-8	-6	75.00
746	967	129.62	Taxable Income	1,524	2,070	135.83
480	490	102.08	Income Taxes	990	990	100.00
266	477	179.32	Net Income	534	1,080	202.25

Example 10.4: Consolidations

This example consolidates product tables by region and region tables by corporate division. Output 10.4.1 shows the North Central and Northeast regional summaries for the Equipment division for the first quarter. Output 10.4.2 shows the profit summary for the Equipment division. Similar tables for the Publishing division are produced but not shown here.

```
data product;
   input pcode div region month sold revenue recd cost;
datalines;
1 1 1 1 56 5600 29 2465
1 1 1 2 13 1300 30 2550
1 1 1 3 17 1700 65 5525
2 1 1 1  2  240 50 4900
2 1 1 2 82 9840 17 1666
1      1       1       1      37      3700     75    6375

   ... more lines ...


proc format;
   value divfmt 1='Equipment'
                2='Publishing';
   value regfmt 1='North Central'
                2='Northeast'
                3='South'
                4='West';
run;

proc sort data=product;
   by div region pcode;
run;

title1 '     XYZ Development Corporation      ';
title2 ' Corporate Headquarters: New York, NY ';
title3 '           Profit Summary             ';
title4 '                                      ';

options linesize=96;
proc computab data=product sumonly;
   by div region pcode;
   sumby _total_ div region;

   format div    divfmt.;
   format region regfmt.;
   label  div = 'DIVISION';

   /* specify order of columns and column titles */
   columns jan feb mar qtr1 /
                    mtitle='- first quarter -' ' '  nozero;
   columns apr may jun qtr2 /
                    mtitle='- second quarter -' ' ' nozero;
   columns jul aug sep qtr3 /
                    mtitle='- third quarter -' ' '  nozero;
   columns oct nov dec qtr4 /
                    mtitle='- fourth quarter -' ' ' nozero;
   column  jan  / ' ' 'January' '=';
   column  feb  / ' ' 'February' '=';
   column  mar  / ' ' 'March' '=';
   column  qtr1 / 'Quarter' 'Summary' '=';
```

```
   column  apr  / ' ' 'April' '=' _page_;
   column  may  / ' ' 'May' '=';
   column  jun  / ' ' 'June' '=';
   column  qtr2 / 'Quarter' 'Summary' '=';

   column  jul  / ' ' 'July' '=' _page_;
   column  aug  / ' ' 'August' '=';
   column  sep  / ' ' 'September' '=';
   column  qtr3 / 'Quarter' 'Summary' '=';

   column  oct  / ' ' 'October' '=' _page_;
   column  nov  / ' ' 'November' '=';
   column  dec  / ' ' 'December' '=';
   column  qtr4 / 'Quarter' 'Summary' '=';

   /* specify order of rows and row titles */
   row     sold    / ' ' 'Number Sold' f=8.;
   row     revenue / ' ' 'Sales Revenue';
   row     recd    / ' ' 'Number Received' f=8.;
   row     cost    / ' ' 'Cost of' 'Items Received';
   row     profit  / ' ' 'Profit' 'Within Period' ol;
   row     pctmarg / ' ' 'Profit Margin' dul;

   /* select column for appropriate month */
   _col_ = month + ceil( month / 3 ) - 1;

   /* calculate quarterly summary columns */
   colcalc:
      qtr1 = jan + feb + mar;
      qtr2 = apr + may + jun;
      qtr3 = jul + aug + sep;
      qtr4 = oct + nov + dec;

   /* calculate profit rows */
    rowcalc:
       profit = revenue - cost;
       if cost > 0 then pctmarg = profit / cost * 100;
run;
```

Output 10.4.1 Summary by Regions for the Equipment Division

XYZ Development Corporation
Corporate Headquarters: New York, NY
Profit Summary

DIVISION=Equipment region=Northeast pcode=1

SUMMARY TABLE: DIVISION=Equipment region=North Central

	first quarter			
	January	February	March	Quarter Summary
Number Sold	198	223	119	540
Sales Revenue	22090.00	26830.00	14020.00	62940.00
Number Received	255	217	210	682
Cost of Items Received	24368.00	20104.00	19405.00	63877.00
Profit Within Period	-2278.00	6726.00	-5385.00	-937.00
Profit Margin	-9.35	33.46	-27.75	-1.47

Output 10.4.1 *continued*

XYZ Development Corporation
Corporate Headquarters: New York, NY
Profit Summary

DIVISION=Publishing region=North Central pcode=4

----------SUMMARY TABLE: DIVISION=Equipment region=Northeast----------

	first quarter			
	January	February	March	Quarter Summary
Number Sold	82	180	183	445
Sales Revenue	9860.00	21330.00	21060.00	52250.00
Number Received	162	67	124	353
Cost of Items Received	16374.00	6325.00	12333.00	35032.00
Profit Within Period	-6514.00	15005.00	8727.00	17218.00
Profit Margin	-39.78	237.23	70.76	49.15

Output 10.4.2 Profit Summary for the Equipment Division

XYZ Development Corporation
Corporate Headquarters: New York, NY
Profit Summary

DIVISION=Publishing region=North Central pcode=4

SUMMARY TABLE: DIVISION=Equipment

	first quarter			
	January	February	March	Quarter Summary
Number Sold	280	403	302	985
Sales Revenue	31950.00	48160.00	35080.00	115190.00
Number Received	417	284	334	1035
Cost of Items Received	40742.00	26429.00	31738.00	98909.00
Profit Within Period	-8792.00	21731.00	3342.00	16281.00
Profit Margin	-21.58	82.22	10.53	16.46

Output 10.4.3 shows the consolidation report of profit summary over both divisions and regions.

Output 10.4.3 Profit Summary

XYZ Development Corporation
Corporate Headquarters: New York, NY
Profit Summary

SUMMARY TABLE: TOTALS

	first quarter			
	January	February	March	Quarter Summary
Number Sold	590	683	627	1900
Sales Revenue	41790.00	55910.00	44800.00	142500.00
Number Received	656	673	734	2063
Cost of Items Received	46360.00	35359.00	40124.00	121843.00
Profit Within Period	-4570.00	20551.00	4676.00	20657.00
Profit Margin	-9.86	58.12	11.65	16.95

Example 10.5: Creating an Output Data Set

This example uses data and reports similar to those in Example 10.3 to illustrate the creation of an output data set.

```
data product;
   input pcode div region month sold revenue recd cost;
datalines;
1 1 1 1 56 5600 29 2465
1 1 1 2 13 1300 30 2550
1 1 1 3 17 1700 65 5525
2 1 1 1  2  240 50 4900
2 1 1 2 82 9840 17 1666
1      1       1       1      37      3700     75    6375

   ... more lines ...
```

```
proc sort data=product out=sorted;
   by div region;
run;

/* create data set, profit */
proc computab data=sorted notrans out=profit noprint;
   by div region;
   sumby div;

   /* specify order of rows and row titles */
   row     jan feb mar qtr1;
   row     apr may jun qtr2;
   row     jul aug sep qtr3;
   row     oct nov dec qtr4;

   /* specify order of columns and column titles */
   columns sold revenue recd cost profit pctmarg;

   /* select row for appropriate month */
   _row_ = month + ceil( month / 3 ) - 1;

   /* calculate quarterly summary rows */
   rowcalc:
      qtr1 = jan + feb + mar;
      qtr2 = apr + may + jun;
      qtr3 = jul + aug + sep;
      qtr4 = oct + nov + dec;

   /* calculate profit columns */
   colcalc:
      profit = revenue - cost;
      if cost > 0 then pctmarg = profit / cost * 100;
run;

/* make a partial listing of the output data set */
options linesize=96;
proc print data=profit(obs=10) noobs;
run;
```

Because the NOTRANS option is specified, column names become variables in the data set. REGION has missing values in the output data set for observations associated with consolidation tables. The output data set PROFIT, in conjunction with the option NOPRINT, illustrates how you can use the computational features of PROC COMPUTAB for creating additional rows and columns as in a spreadsheet without producing a report. Output 10.5.1 shows a partial listing of the output data set PROFIT.

Output 10.5.1 Partial Listing of the PROFIT Data Set

XYZ Development Corporation
Corporate Headquarters: New York, NY
Profit Summary

div	region	_TYPE_	_NAME_	sold	revenue	recd	cost	PROFIT	PCTMARG
1	1	1	JAN	198	22090	255	24368	-2278	-9.348
1	1	1	FEB	223	26830	217	20104	6726	33.456
1	1	1	MAR	119	14020	210	19405	-5385	-27.751
1	1	1	QTR1	540	62940	682	63877	-937	-1.467
1	1	1	APR	82	9860	162	16374	-6514	-39.783
1	1	1	MAY	180	21330	67	6325	15005	237.233
1	1	1	JUN	183	21060	124	12333	8727	70.761
1	1	1	QTR2	445	52250	353	35032	17218	49.149
1	1	1	JUL	194	23210	99	10310	12900	125.121
1	1	1	AUG	153	17890	164	16704	1186	7.100

Example 10.6: A What-If Market Analysis

PROC COMPUTAB can be used with other SAS/ETS procedures and with macros to implement commonly needed decision support tools for financial and marketing analysis.

The following input data set reads quarterly sales figures:

```
data market;
   input date :yyq6. units @@;
datalines;
1980Q1  3608.9  1980Q2  5638.4  1980Q3  6017.9  1980Q4  4929.6

   ... more lines ...
```

The following statements illustrate how PROC FORECAST makes a total market forecast for the next four quarters:

```
/* forecast the total number of units to be */
/* sold in the next four quarters */
proc forecast out=outcome trend=2
              interval=qtr lead=4;
   id date;
   var units;
run;
```

The macros WHATIF and SHOW build a report table and provide the flexibility of examining alternate what-if situations. The row and column calculations of PROC COMPUTAB compute the income statement. With macros stored in a macro library, the only statements required with PROC COMPUTAB are macro invocations and TITLE statements.

```
/* set up rows and columns of report and initialize */
/* market share and program constants */
%macro whatif(mktshr=,price=,ucost=,taxrate=,numshar=,overhead=);

   columns mar / ' ' 'March';
   columns jun / ' ' 'June';
   columns sep / ' ' 'September';
   columns dec / ' ' 'December';
   columns total / 'Calculated' 'Total';
   rows mktshr / 'Market Share'          f=5.2;
   rows tunits / 'Market Forecast';
   rows units  / 'Items Sold';
   rows sales  / 'Sales';
   rows cost   / 'Cost of Goods';
   rows ovhd   / 'Overhead';
   rows gprof  / 'Gross Profit';
   rows tax    / 'Tax';
   rows pat    / 'Profit After Tax';
   rows earn   / 'Earnings per Share';

   rows mktshr--earn / skip;
   rows sales--earn  /  f=dollar12.2;
   rows tunits units /  f=comma12.2;

   /* initialize market share values */
   init mktshr &mktshr;

   /* define constants */
   retain price &price ucost &ucost taxrate &taxrate
          numshar &numshar;

   /* retain overhead and sales from previous quarter */
   retain prevovhd &overhead prevsale;
   %mend whatif;

/* perform calculations and print the specified rows */
%macro show(rows);

   /* initialize list of row names */
   %let row1  = mktshr;
   %let row2  = tunits;
   %let row3  = units;
   %let row4  = sales;
   %let row5  = cost;
   %let row6  = ovhd;
   %let row7  = gprof;
   %let row8  = tax;
   %let row9  = pat;
   %let row10 = earn;

   /* find parameter row names in list and eliminate */
   /* them from the list of noprint rows */
   %let n = 1;
```

```
%let word = %scan(&rows,&n);
%do %while(&word NE );
   %let i = 1;
   %let row11 = &word;
   %do %while(&&row&i NE &word);
      %let i = %eval(&i+1);
      %end;
   %if &i<11 %then %let row&i = ;
   %let n = %eval(&n+1);
   %let word = %scan(&rows,&n);
%end;

rows &row1 &row2 &row3 &row4 &row5 &row6 &row7
     &row8 &row9 &row10 dummy / noprint;

/* select column using lead values from proc forecast */
mar = _lead_ = 1;
jun = _lead_ = 2;
sep = _lead_ = 3;
dec = _lead_ = 4;

rowreln:;
   /* inter-relationships */
   share  = round( mktshr, 0.01 );
   tunits = units;
   units  = share * tunits;
   sales  = units * price;
   cost   = units * ucost;

   /* calculate overhead */
   if mar then prevsale = sales;
   if sales > prevsale
      then ovhd = prevovhd + .05 * ( sales - prevsale );
      else ovhd = prevovhd;
   prevovhd = ovhd;
   prevsale = sales;
   gprof = sales - cost - ovhd;
   tax  = gprof * taxrate;
   pat  = gprof - tax;
   earn = pat / numshar;

coltot:;
   if mktshr
      then total = ( mar + jun + sep + dec ) / 4;
      else total = mar + jun + sep + dec;
%mend show;
run;
```

The following PROC COMPUTAB statements use the PROC FORECAST output data set with invocations of the macros defined previously to perform a what-if analysis of the predicted income statement. The report is shown in Output 10.6.1.

```
title1 'Fleet Footwear, Inc.';
title2 'Marketing Analysis Income Statement';
title3 'Based on Forecasted Unit Sales';
title4 'All Values Shown';

options linesize=96;

proc computab data=outcome cwidth=12;

   %whatif(mktshr=.02 .07 .15 .25,price=38.00,
           ucost=20.00,taxrate=.48,numshar=15000,overhead=5000);

   %show(mktshr tunits units sales cost ovhd gprof tax pat earn);
run;
```

Output 10.6.1 PROC COMPUTAB Report That Uses Macro Invocations

Fleet Footwear, Inc.
Marketing Analysis Income Statement
Based on Forecasted Unit Sales
All Values Shown

	March	June	September	December	Calculated Total
Market Share	0.02	0.07	0.15	0.25	0.12
Market Forecast	23,663.94	24,169.61	24,675.27	25,180.93	97,689.75
Items Sold	473.28	1,691.87	3,701.29	6,295.23	12,161.67
Sales	$17,984.60	$64,291.15	$140,649.03	$239,218.83	$462,143.61
Cost of Goods	$9,465.58	$33,837.45	$74,025.80	$125,904.65	$243,233.48
Overhead	$5,000.00	$7,315.33	$11,133.22	$16,061.71	$39,510.26
Gross Profit	$3,519.02	$23,138.38	$55,490.00	$97,252.47	$179,399.87
Tax	$1,689.13	$11,106.42	$26,635.20	$46,681.19	$86,111.94
Profit After Tax	$1,829.89	$12,031.96	$28,854.80	$50,571.28	$93,287.93
Earnings per Share	$0.12	$0.80	$1.92	$3.37	$6.22

The following statements produce a similar report for different values of market share and unit costs. The report in Output 10.6.2 displays the values for the market share, market forecast, sales, after-tax profit, and earnings per share.

```
title3 'Revised';
title4 'Selected Values Shown';

options linesize=96;

proc computab data=outcome cwidth=12;
   %whatif(mktshr=.01 .06 .12 .20,price=38.00,
           ucost=23.00,taxrate=.48,numshar=15000,overhead=5000);
   %show(mktshr tunits sales pat earn);
run;
```

Output 10.6.2 Report That Uses Macro Invocations for Selected Values

Fleet Footwear, Inc.
Marketing Analysis Income Statement
Revised
Selected Values Shown

	March	June	September	December	Calculated Total
Market Share	0.01	0.06	0.12	0.20	0.10
Market Forecast	23,663.94	24,169.61	24,675.27	25,180.93	97,689.75
Sales	$8,992.30	$55,106.70	$112,519.22	$191,375.06	$367,993.28
Profit After Tax	$-754.21	$7,512.40	$17,804.35	$31,940.30	$56,502.84
Earnings per Share	$-0.05	$0.50	$1.19	$2.13	$3.77

Example 10.7: Cash Flows

The COMPUTAB procedure can be used to model cash flows from one time period to the next. The RETAIN statement is useful for enabling a row or column to contribute one of its values to its successor. Financial functions such as IRR (internal rate of return) and NPV (net present value) can be used on PROC COMPUTAB table values to provide a more comprehensive report. The following statements produce Output 10.7.1:

```
data cashflow;
   input date date9. netinc depr borrow invest tax div adv ;
datalines;
30MAR1982 65 42 32 126 43 51 41
30JUN1982 68 47 32 144 45 54 46
30SEP1982 70 49 30 148 46 55 47
30DEC1982 73 49 30 148 48 55 47
;

title1 'Blue Sky Endeavors';
title2 'Financial Summary';
title4 '(Dollar Figures in Thousands)';

proc computab data=cashflow;

   cols qtr1 qtr2 qtr3 qtr4 / 'Quarter' f=7.1;
   col  qtr1 / 'One';
   col  qtr2 / 'Two';
   col  qtr3 / 'Three';
   col  qtr4 / 'Four';
   row  begcash / 'Beginning Cash';
   row  netinc  / 'Income' '   Net income';
   row  depr    / 'Depreciation';
   row  borrow;
   row  subtot1 / 'Subtotal';
   row  invest  / 'Expenditures' '   Investment';
   row  tax     / 'Taxes';
   row  div     / 'Dividend';
   row  adv     / 'Advertising';
   row  subtot2 / 'Subtotal';
   row  cashflow/  skip;
   row  irret   / 'Internal Rate' 'of Return' zero=' ';
   rows depr borrow subtot1 tax div adv subtot2 / +3;

   retain cashin -5;
   _col_ = qtr( date );

   rowblock:
      subtot1 = netinc + depr + borrow;
      subtot2 = tax + div + adv;
      begcash = cashin;
      cashflow = begcash + subtot1 - subtot2;
      irret = cashflow;
      cashin = cashflow;

   colblock:
      if begcash then cashin = qtr1;
      if irret then do;
         temp = irr( 4, cashin, qtr1, qtr2, qtr3, qtr4 );
         qtr1 = temp;
         qtr2 = 0; qtr3 = 0; qtr4 = 0;
         end;
run;
```

Output 10.7.1 Report That Uses a RETAIN Statement and the IRR Financial Function

Blue Sky Endeavors
Financial Summary

(Dollar Figures in Thousands)

	Quarter One	Quarter Two	Quarter Three	Quarter Four
Beginning Cash	-5.0	-1.0	1.0	2.0
Income				
Net income	65.0	68.0	70.0	73.0
Depreciation	42.0	47.0	49.0	49.0
BORROW	32.0	32.0	30.0	30.0
Subtotal	139.0	147.0	149.0	152.0
Expenditures				
Investment	126.0	144.0	148.0	148.0
Taxes	43.0	45.0	46.0	48.0
Dividend	51.0	54.0	55.0	55.0
Advertising	41.0	46.0	47.0	47.0
Subtotal	135.0	145.0	148.0	150.0
CASHFLOW	-1.0	1.0	2.0	4.0
Internal Rate of Return	20.9			

Chapter 11
The COPULA Procedure

Contents

Overview: COPULA Procedure **534**
Getting Started: COPULA Procedure **534**
Syntax: COPULA Procedure **538**
- Functional Summary 539
- PROC COPULA Statement 540
- BOUNDS Statement 540
- BY Statement 541
- DEFINE Statement 541
- FIT Statement 543
- SIMULATE Statement 545
- VAR Statement 546

Details: COPULA Procedure **547**
- Sklar's Theorem 547
- Dependence Measures 547
- Normal Copula 548
- Student's *t* copula 549
- Archimedean Copulas 552
- Hierarchical Archimedean Copula (HAC) (Experimental) 557
- Canonical Maximum Likelihood Estimation (CMLE) 559
- Exact Maximum Likelihood Estimation (MLE) 560
- Calibration Estimation 560
- Nonlinear Optimization Options 561
- Displayed Output 561
- OUTCOPULA= Data Set 563
- OUTPSEUDO=, OUT=, and OUTUNIFORM= Data Sets 563
- ODS Table Names 563
- ODS Graph Names 564

Examples: COPULA Procedure **566**
- Example 11.1: Copula Based VaR Estimation 566
- Example 11.2: Simulating Default Times 572

References **579**

Overview: COPULA Procedure

A multivariate distribution for a random vector contains a description of both the marginal distributions and their dependence structure. A copula approach to formulating a multivariate distribution provides a way to isolate the description of the dependence structure from the marginal distributions. A copula is a function that combines marginal distributions of variables into a specific multivariate distribution. All of the one-dimensional marginals in the multivariate distribution are the cumulative distribution functions of the factors. Copulas help perform large-scale multivariate simulation from separate models, each of which can be fitted using different, even nonnormal, distributional specifications.

The COPULA procedure enables you to fit multivariate distributions or copulas from a given sample data set. You can do the following:

- estimate the parameters for a specified copula type
- simulate a given copula
- plot dependent relationships among the variables

The following types of copulas are supported:

- normal copula
- *t* copula
- Clayton copula
- Gumbel copula
- Frank copula

Getting Started: COPULA Procedure

The following example illustrates the use of PROC COPULA. The data used are daily returns on several major stocks. The main purpose of this example is to estimate the joint distribution of stock returns and then simulate from this distribution a new sample of specified size.

Figure 11.1 shows the first 10 observations of the daily stock return data set.

Figure 11.1 First 10 Observations of Daily Returns

Obs	date	ret_msft	ret_ko	ret_ibm	ret_duk	ret_bp
1	01/03/2008	0.004182	0.010367	0.002002	0.003503	0.019114
2	01/04/2008	-0.027960	0.001913	-0.035861	-0.000582	-0.014536
3	01/07/2008	0.006732	0.023607	-0.010671	0.025611	0.017922
4	01/08/2008	-0.033435	0.004239	-0.024610	-0.002838	-0.016049
5	01/09/2008	0.029560	0.026680	0.007301	0.010814	-0.027078
6	01/10/2008	-0.003054	0.004441	0.016414	-0.001689	-0.004395
7	01/11/2008	-0.012255	-0.027346	-0.022546	-0.012408	-0.018473
8	01/14/2008	0.013958	0.008418	0.053857	0.003427	0.001166
9	01/15/2008	-0.011318	-0.010851	-0.010689	-0.017075	-0.040925
10	01/16/2008	-0.022587	-0.015021	-0.001955	0.002316	-0.021336

The following statements fit a normal copula to the returns data (with the FIT statement) and create a new SAS data set that contains parameter estimates of the model. The VAR statement specifies the list of variables, which in this case are the daily returns of five large company stocks.

```
/* Copula estimation */
proc copula data = returns;
   var ret_ibm ret_msft ret_bp ret_ko ret_duk;
   fit normal / outcopula=estimates;
run;
```

The first table in Figure 11.2 shows some general information about the copula fitting procedure: the number of observations, the name of the input data set, the type of model and the correlation matrix.

Figure 11.2 Copula Estimation: Fit Summary and Correlation Matrix

The COPULA Procedure

Model Fit Summary	
Number of Observations	603
Data Set	WORK.RETURNS
Copula Type	Normal

Correlation Matrix					
	ret_ibm	ret_msft	ret_bp	ret_ko	ret_duk
ret_ibm	1.0000	0.6232	0.5294	0.4725	0.4902
ret_msft	0.6232	1.0000	0.5229	0.5015	0.4567
ret_bp	0.5294	0.5229	1.0000	0.3980	0.4378
ret_ko	0.4725	0.5015	0.3980	1.0000	0.5283
ret_duk	0.4902	0.4567	0.4378	0.5283	1.0000

Next, the following statements restrict the data set to only those columns that contain correlation parameter estimates.

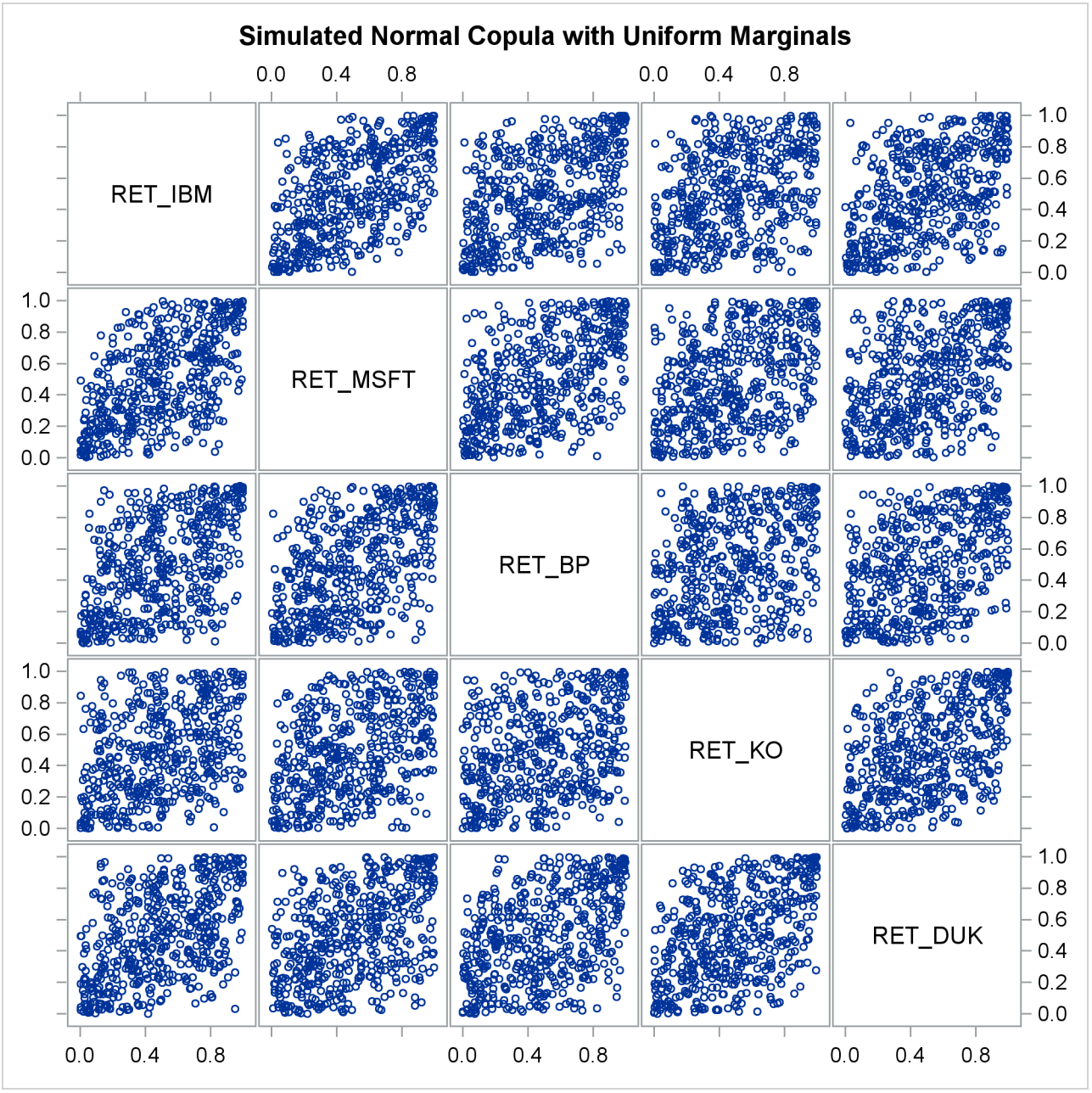
```
/* keep only correlation estimates */
data estimates;
   set estimates;
   keep ret_ibm ret_msft ret_bp ret_ko ret_duk;
run;
```

Then, in the following statements, the DEFINE statement specifies a normal copula named COP, and the CORR= option specifies that the data set Estimates be used as the source for the model parameters. The NDRAWS=500 option in the SIMULATE statement generates 500 observations from the normal copula. The OUTUNIFORM= option specifies the name of SAS data set to contain the simulated sample with uniform marginal distributions. Note that this syntax does not require the DATA= option.

```
/* Copula simulation of uniforms */
proc copula;
   var ret_ibm ret_msft ret_bp ret_ko ret_duk;
   define cop normal (corr = estimates);
   simulate cop / ndraws     = 500
                  seed       = 1234
                  outuniform = simulated_uniforms
                  plots=(datatype=uniform);
run;
```

The simulated data is contained in the new SAS data set, Simulated_Uniforms. A scatter plot matrix of uniform marginals contained in the data set is shown in Output 11.3.

Figure 11.3 Simulated Data, Uniform Marginals



The preceding sequence of PROC COPULA usage—first fit, then simulate given estimated parameters—is a legitimate sequence but has a limitation in that the second COPULA call does not generate the sample according to the empirical distribution of the raw data. It generates only marginally uniform series.

In the following statements, the FIT statement fits a *t* copula to the returns data and at the same time simulates the sample according to empirical marginal distributions:

```
/* Copula estimation and simulation of returns */
proc copula data = returns;
   var ret_ibm ret_msft ret_bp ret_ko ret_duk;
   fit T;
   simulate / ndraws = 1000
              seed   = 1234
              out    = simulated_returns;
run;
```

The output of the statements is similar in structure to the output displayed in Figure 11.2 with the addition of parameter estimates and inference statistics that are specific to the copula model as shown in Figure 11.4. For a *t* copula, the degrees of freedom are displayed (as in Figure 11.4); for Archimedean copulas, the parameter "theta" is displayed; and for a normal copula, this table is not printed.

Figure 11.4 Copula Estimation: Specific Parameter Estimates

The COPULA Procedure

Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|
DF	3.659299	0.320789	11.41	<.0001

The simulated data is contained in the new SAS data set, Simulated_Returns.

Syntax: COPULA Procedure

The COPULA procedure is controlled by the following statements:

PROC COPULA *options* **;**
VAR *variables* **;**
DEFINE *name copula-type* < *(parameter-value-options . . .)* > **;**
FIT *type* < **NAME=***name* > < **INIT=***(parameter-value-options)* > */ options* **;**
BOUNDS *bound1* < *, bound2 . . .* > **;**
SIMULATE < *copula-name-list* > */ options* **;**
BY *variables* **;**

Functional Summary

Table 11.1 summarizes the statements and options used with the COPULA procedure.

Table 11.1 COPULA Functional Summary

Description	Statement	Option
Data Set Options		
Specifies the input data set	COPULA	DATA=
Specifies the input data set that contains the correlation matrix for elliptical copulas	DEFINE	CORR=
Specifies the input data set that contains the correlation matrix defined in Kendall's tau for elliptical copulas	DEFINE	KENDALL=
Specifies the input data set that contains the correlation matrix defined in Spearman's rho for elliptical copulas	DEFINE	SPEARMAN=
Specifies the degrees of freedom for *t* copulas	DEFINE	DF=
Specifies the parameter value for Archimedean copulas	DEFINE	THETA=
Specifies the hierarchy for hierarchical Archimedean copulas	DEFINE	HIERARCHY=
Declaring the Role of Variables		
Specifies the names of the variables to use in copula fitting or in simulation	VAR	
Specifies BY-group processing	BY	
Plotting Options		
Prints a summary iteration listing	FIT	ITPRINT
Suppresses the normal printed output	FIT	NOPRINT
Requests all printing options	FIT	PRINTALL
Suppresses the correlation matrix printed output	FIT	NOCORR
Printing Control Options		
Displays plots for fitted copulas	FIT	PLOTS=
Displays plots for simulated copulas	SIMULATE	PLOTS=
Optimization Process Control Options		
Sets boundary restrictions on parameters	BOUNDS	
Selects the iterative minimization method to use	FIT	METHOD=
Sets initial values for parameters	FIT	INIT=
Copula Estimation Options		
Specifies the marginal distribution of the individual variables	FIT	MARGINALS=

Description	Statement	Option
Copula Simulation Options		
Specifies the marginal distribution of the simulated variables	SIMULATE	MARGINALS=
Specifies the random sample size	SIMULATE	NDRAWS=
Specifies the random number generator seed	SIMULATE	SEED=
Output Control Options		
Specifies the output data set to contain the fitted copula values	FIT	OUTCOPULA=
Specifies the output data set to contain pseudo-samples with the uniform marginal distribution	FIT	OUTPSEUDO=
Specifies the output data set to contain the random samples from the simulation	SIMULATE	OUT=
Specifies the output data set to contain the random samples from the simulation with uniform marginal distribution	SIMULATE	OUTUNIFORM=

PROC COPULA Statement

PROC COPULA < *option* > **;**

The PROC COPULA statement has the following option:

DATA= < *libref.* >*SAS-data-set*
specifies the input data set used to estimate parameters for the FIT statement. When the procedure is used for simulation only, the input data set is not required to run the procedure.If you do not specify *libref*, then the Work library is used. Work is the default temporary library that is automatically defined by SAS at the beginning of each SAS session or job.

BOUNDS Statement

BOUNDS *bound1* < *, bound2 . . .* > **;**

The BOUNDS statement specifies the lower and upper bounds for the parameters. You can use this statement only when maximum likelihood estimation is used for the specified copula. Each bound is composed of parameters, constants, and inequality operators in the following format:

operator item < *operator item* < *operator item . . .* >>

Each item is a constant, parameter, or list of parameters. Parameters associated with a regressor variable are referred to by the name of the corresponding regressor variable. Each operator is <, >, <=, or >=. The following example indicates that the lower and upper bounds for the parameter THETA are -5 and 10, respectively.

```
bounds -5 < THETA < 10;
```

If you do not specify bounds, the internal default values are used; the default values are described in the section "Details: COPULA Procedure" on page 547. For the normal and *t* copulas, the correlation matrix uses only the default parameter bounds, which are −1 and 1 for lower bound and upper bound, respectively.

BY Statement

BY *variables* **;**

The BY statement specifies groups in which separate FIT analyses for copula are performed. The *variables* must be present in the input data set and are excluded from the model fitting. The BY statement requires the VAR statement to be present.

DEFINE Statement

DEFINE *name copula-type* < *(parameter-value-options ...)* > **;**

The DEFINE statement specifies the relevant information of the copula used for the simulation.

name specifies the name of the copula definition, which can be used later in the SIMULATE statement.

copula-type specifies one of the following types of the copula:

NORMAL specifies the normal copula.

T specifies the *t* copula.

CLAYTON specifies the Clayton copula.

GUMBEL specifies the Gumbel copula.

FRANK specifies the Frank copula.

HACCLAYTON specifies the hierarchical Clayton copula.

HACGUMBEL specifies the hierarchical Gumbel copula.

HACFRANK specifies the hierarchical Frank copula.

These copula models are also described in the section "Details: COPULA Procedure" on page 547.

parameter-value-options

specify the input parameters used to simulate the specified copula. These options must be appropriate for the type of copula specified. The following options are valid:

CORR=*SAS-data-set*

specifies the data set that contains the correlation matrix to use for elliptical copulas. If the correlation matrix is valid but not submitted in order, then you must provide the variable names in the first column of the matrix and these names must match the variable names in the VAR statement. See Output 11.2.1 for an example of a

correlation matrix input in this form. If the correlation matrix is submitted in order, the first column of variable names is not required. This option can be used for the normal and *t* copulas.

KENDALL=*SAS-data-set*
specifies the data set that contains the correlation matrix defined in Kendall's tau. If the correlation matrix is valid but not submitted in order, then you must provide the variable names in the first column of the matrix and these names must match the variable names in the VAR statement. If the correlation matrix is submitted in order, the first column of variable names is not required. This option can be used for the normal and *t* copulas.

SPEARMAN=*SAS-data-set*
specifies the data set that contains the correlation matrix defined in Spearman's rho. If the correlation matrix is valid but not submitted in order, then you must provide the variable names in the first column of the matrix and these names must match the variable names in the VAR statement. If the correlation matrix is submitted in order, the first column of variable names is not required. This option can be used for the normal copula.

DF=*value*
specifies the degrees of freedom. This option can be used for the *t* copula.

THETA=*value*
specifies the parameter value for the Archimedean copulas.

HIERARCHY=*(name=(HAC-specification)(***THETA***=value))* (Experimental)
specifies the hierarchy for hierarchical Archimedean copulas. The argument usually consists of multiple specification lines, with each line specifying one copula in the hierarchy. *name* can be user-defined symbols, with the exception of the copula at the top of the hierarchy, which must be named ROOT. The *HAC-specification* is a list of symbols that can be either defined copula names or variable names from the VAR statement, depending on whether the element of the copula is a variable or an inner copula in the hierarchy. For example, you can use the following code to define a hierarchical Archimedean copula, with the hierarchy shown in Figure 11.5:

```
var u1-u4;
define cop hacclayton hierarchy=(
root = (c1 c2)(theta=1)
c1 = (u1 u2)(theta=3)
c2 = (u3 u4)(theta=5));
```

Note that as long as the specification is valid, the order of the specification lines does not matter. In the previous example, you could first list $c1$ and $c2$, and then define $root$.

The DEFINE statement is used with the SIMULATE statement.The FIT statement can also be used with the SIMULATE statement. The results of the FIT statement can be the input of the SIMULATE statement. Therefore, the SIMULATE statement can follow the FIT statement.If there is no FIT statement, then the DEFINE statement must precede SIMULATE statement. However, the FIT and DEFINE statements cannot both be used in the same procedure.

FIT Statement

FIT *type* < **NAME=***name* >< **INIT=***(parameter-value-options)* > */options* **;**

The FIT statement estimates the parameters for a specified copula type.

type
specifies the type of the copula to be estimated, which is one of the following:

NORMAL	fits the normal copula
T	fits the *t* copula
CLAYTON	fits the Clayton copula
GUMBEL	fits the Gumbel copula
FRANK	fits the Frank copula

NAME=*name*
specifies an identifier for the fit, which is stored as an ID variable in the OUTCOPULA= data set.

INIT=*(parameter-value-options)*
provides the initial values for the numerical optimization. For Archimedean copulas, the initial values of the parameter are computed using the calibration method. The initial value for the degrees-of-freedom parameter in the *t* copula is set to 2.0.

You can specify the following *options* after a slash (/):

METHOD=MLE | CAL
specifies the method used to estimate parameters. MLE represents canonical maximum likelihood estimation (CMLE) or maximum likelihood estimation (MLE). CAL is the calibration method that uses the correlation matrix (only Kendall's tau is implemented in this procedure). For the *t* copula, if METHOD=CAL, then the correlation matrix is estimated using the calibration method with Kendall's tau and the degrees of freedom are estimated by the MLE. For the normal copula, only MLE is supported and METHOD=CAL is ignored. The default for all copula types is METHOD=MLE.

OUTCOPULA=*SAS-data-set*
specifies the name of the output data set. Each fitted copula is written to the OUTCOPULA= data set. The data set is not created if this option is not specified.

OUTPSEUDO=*SAS-data-set*
specifies the output data set for saving the pseudo-samples with uniform marginal distributions. The pseudo-samples are obtained by transforming the individual variables of the original data with the empirical cumulative distribution functions (CDFs). The data set is not created if this option is not specified.

MARGINALS=UNIFORM | EMPIRICAL
specifies the marginal distribution of the individual variables. If MARGINALS=UNIFORM, then the copula is fitted with the input data without transformation. If MARGINALS=EMPIRICAL, the marginal empirical CDF is used to transform the data and the copula is fitted using the transformed data.

PLOTS<(*global-plot-options*)> < = *(specific-plot-options)*>
controls the plots that are produced by the COPULA procedure. By default, PROC COPULA produces a scatter plot matrix for variables (that is, it displays a symmetric matrix plot with the variables that are specified in the VAR statement).

You can specify the following *global-plot-options*:

UNPACKPANEL | UNPACK
requests scatter plots for pairs of variables. If you specify this option, PROC COPULA displays a scatter plot for each applicable pair of distinct variables that are specified in the VAR statement.

NVAR=ALL | *n*
specifies the maximum number of variables specified in the VAR statement to be displayed in the matrix plot. The NVAR=ALL option uses all variables that are specified in the VAR statement. By default, NVAR=5.

TAIL | CHI
requests that tail dependence plots (chi-plots) be plotted. If you specify this option with the UNPACK option on, PROC COPULA displays a chi-plot for each applicable pair of distinct variables that are specified in the VAR statement. If you specify this option without the UNPACK option, PROC COPULA displays a scatter plot matrix, the lower triangular section shows regular scatter plots between distinct pairs of variables that are specified in the VAR statement, the upper triangular section shows chi-plots for corresponding pairs of variables.

You can specify the following *specific-plot-options*:

DATATYPE=ORIGINAL | UNIFORM | BOTH
requests the data type to be plotted. DATA=ORIGINAL presents the data in its original marginal distribution; DATA=UNIFORM shows the transformed data with uniform marginal distribution; and DATA=BOTH plots both the original and uniform data types. If MARGINALS=UNIFORM, then the transformation is omitted and the DATA= option is ignored.

NONE
suppresses all plots.

Printing Options

ITPRINT
prints a summary iteration listing.

PRINTALL
default option.

NOCORR
suppresses the correlation matrix.

NOPRINT
suppresses all output.

SIMULATE Statement

SIMULATE < *copula-name-list* >*/options* **;**

The SIMULATE statement simulates data from a specified copula model. The copula name specification can be either the name of a defined copula as specified by *name* in the DEFINE statement or the name of a fitted copula specified in the NAME= option in the FIT statement copula specification.

MARGINALS=UNIFORM | EMPIRICAL
specifies how the marginal distributions are computed. If MARGINALS=UNIFORM, then the samples are drawn from the copula distribution and marginal distributions are uniform.

MARGINALS=EMPIRICAL can be used to explicitly specify that the marginal distributions are empirical CDF computed from the DATA= option in the PROC COPULA statement.

If the MARGINALS= option is not specified in the SIMULATE statement, then the marginal distributions used in the simulation depend on whether a preceding FIT statement was used: If there is no FIT statement, the marginal distributions depend on whether the PROC COPULA statement includes a DATA= option. If there is a preceding FIT statement, then the marginal distributions from that fit are used. If there is no FIT statement and there is no DATA= option, then MARGINALS=UNIFORM.

OUT=*SAS-data-set*
specifies the output data set for the random samples from the simulation. This data set is the SAS data set in the OUTUNIFORM= option transformed by the inverse empirical CDF. This option is useful only when an input data exists and MARGINALS=EMPIRICAL. The data set is not created if this option is not specified.

OUTUNIFORM=*SAS-data-set*
specifies the output data set for the result of the simulation in uniforms. This option can be used when MARGINALS=UNIFORM or when MARGINALS=EMPIRICAL. If MARGINALS=EMPIRICAL, then this option enables you to obtain the samples simulated from the joint distribution specified by the copula, with all marginal distributions being uniform. The data is not created if this option is not specified.

NDRAWS=*integer*
specifies the number of draws to generate for this simulation. The default is 100.

SEED=*integer*
specifies the seed for generating random numbers for the simulation. If the seed is not provided, a random number is used as the seed.

PLOTS<**(***global-plot-options***)**> **< =** *(specific-plot-options)***>**
controls the plots that are produced by the COPULA procedure. By default, the PROC COPULA produces a scatter plot matrix for variables. You can specify any of the following *global-plot-options*:

UNPACKPANEL | UNPACK
requests scatter plots for pairs of variables. If you specify this option, PROC COPULA displays a scatter plot for each applicable pair of distinct variables that are specified in the VAR statement.

NVAR=ALL | *n*
specifies the maximum number of variables specified in the VAR statement to be displayed in the matrix plot. The NVAR=ALL option uses all variables that are specified in the VAR statement. By default, NVAR=5.

TAIL | CHI
requests that tail dependence plots (chi-plots) be plotted. If you specify this option with the UNPACK option on, PROC COPULA displays a chi-plot for each applicable pair of distinct variables that are specified in the VAR statement. If you specify this option without the UNPACK option, PROC COPULA displays a scatter plot matrix, the lower triangular section shows regular scatter plots between distinct pairs of variables that are specified in the VAR statement, the upper triangular section shows chi-plots for corresponding pairs of variables.

You can specify the following *specific-plot-options*:

DATATYPE=ORIGINAL | UNIFORM | BOTH
requests the data type to be plotted. DATA=ORIGINAL presents the data in its original marginal distribution; DATA=UNIFORM shows the transformed data with uniform marginal distribution; and DATA=BOTH plots both the original and uniform data types. If MARGINALS=UNIFORM, then the transformation is omitted and the DATA= option is ignored. If there is no input data, then the simulated data can only have uniform marginal distributions; in this case, the DATA= option is ignored.

DISTRIBUTION=PDF | CDF
requests distributional graphs for the case of two variables. DISTRIBUTION=PDF specifies that the theoretical probability density function is provided with both a contour plot and a surface plot. DISTRIBUTION=CDF requests the graph for the theoretical cumulative distribution function of the copula.

NONE
suppresses all plots.

VAR Statement

VAR *variables* **;**

The VAR statement specifies the variable names in the input data set specified by the DATA= option in the PROC COPULA statement. The subset of variables in the data set is used for the copula models in the FIT statement. When there is no input data set, the VAR statement creates the names of the list of variables for the SIMULATE statement.

Details: COPULA Procedure

Sklar's Theorem

The copula models are tools for studying the dependence structure of multivariate distributions. The usual joint distribution function contains the information both about the marginal behavior of the individual random variables and about the dependence structure between the variables. The copula is introduced to decouple the marginal properties of the random variables and the dependence structures. A m-dimensional *copula* is a joint distribution function on $[0,1]^m$ with all marginal distributions being standard uniform. The common notation for a copula is $C(u_1,\ldots,u_m)$.

The Sklar (1959) theorem shows the importance of copulas in modeling multivariate distributions. The first part claims that a copula can be derived from any joint distribution functions, and the second part asserts the opposite: that is, any copula can be combined with any set of marginal distributions to result in a multivariate distribution function.

- Let F be a joint distribution function and $F_j, j = 1,\ldots,m$ be the marginal distributions. Then there exists a copula $C : [0,1]^m \rightarrow [0,1]$ such that

  $$F(x_1,\ldots,x_m) = C(F_1(x_1),\ldots,F_m(x_m))$$

  for all $x_1,\ldots,x_m$ in $[-\infty,\infty]$. Moreover, if the margins are continuous, then C is unique; otherwise C is uniquely determined on $\mathrm{Ran}F_1 \times \ldots \times \mathrm{Ran}F_m$, where $\mathrm{Ran}F_j = F_j([-\infty,\infty])$ is the range of F_j.

- The converse is also true. That is, if C is a copula and $F_1,\ldots,F_m$ are univariate distribution functions, then the multivariate function defined in the preceding equation is a joint distribution function with marginal distributions $F_j, j = 1,\ldots,m$.

Dependence Measures

There are three basic types of measures: linear correlation, rank correlation, and tail dependence. Linear correlation is given by

$$\rho \equiv \mathrm{corr}(X,Y) = \frac{\mathrm{cov}(X,Y)}{\sqrt{\mathrm{var}(X)}\sqrt{\mathrm{var}(Y)}}$$

The linear correlation coefficient carries very limited information about the joint properties of the variables. A well-known property is that uncorrelatedness does not imply independence, while independence implies noncorrelation. In addition, there exist distinct bivariate distributions that have the same marginal distribution and the same correlation coefficient. These results suggest that caution must be used when interpreting the linear correlation.

Another statistical measure of dependence is called rank correlation, which is nonparametric. Kendall's tau, for example, is the covariance between the sign statistic $X_1 - \tilde{X}_1$ and $X_2 - \tilde{X}_2$, where $(\tilde{X}_1, \tilde{X}_2)$ is an independent copy of (X_1, X_2):

$$\rho_\tau \equiv E[\mathrm{sign}(X_1 - \tilde{X}_1)(X_2 - \tilde{X}_2)]$$

The sign function (sometimes written as sgn) is defined by

$$\text{sign}(x) = \begin{cases} -1 & \text{if} x \leq 0 \\ 0 & \text{if} x = 0 \\ 1 & \text{if} x \geq 0 \end{cases}$$

Spearman's *rho* is the correlation between the transformed random variables:

$$\rho_S(X_1, X_2) \equiv \rho(F_1(X_1), F_2(X_2))$$

The variables are transformed by their distribution functions so that the transformed variables are uniformly distributed on $[0, 1]$. The rank correlations depend only on the copula of the random variables and are indifferent to the marginal distributions. Like linear correlation, the rank correlations have their limitations. In particular, there are different copulas that result in the same rank correlation.

A third measure focuses on only part of the joint properties between the variables. Tail dependence measures the dependence when both variables are at extreme values. Formally, they can be defined as the conditional probabilities of quantile exceedances. There are two types of tail dependence:

- The upper tail dependence, denoted λ_u, is

  $$\lambda_u(X_1, X_2) \equiv \lim_{q->1^-} P(X_2 > F_2^{-1}(q) | X_1 > F_1^{-1}(q))$$

  when the limit exists $\lambda_u \in [0, 1]$. Here F_j^{-1} is the quantile function (that is, the inverse of the CDF).
- The lower tail dependence is defined symmetrically.

Tail dependence is hard to detect by looking at a scatter plot of realizations of two random variables. One graphical way to detect tail dependence between two variables is by creating the chi plot of those two variables. The chi plot, as defined in Fisher and Switzer (2001), has characteristic patterns that depend on the dependence structure between the variables. The chi plot for the random variables X and Y is a scatter plot of the pairs (λ_i, χ_i) for each data point (x_i, y_i). λ_i is a measure of the distance of the data point (x_i, y_i) from the center of the data as measured by the median values of (x_i, y_i), and χ_i is a correlation coefficient between dichotomized values of X and Y. A positive λ_i means that x_i and y_i are either both large with respect to their median values or both small. A negative λ_i means that x_i or y_i is large with respect to its median, whereas the other value is small. Signs of tail dependence manifest as clusters of points that are significantly far from the χ axis around λ values of ± 1. If X and Y are uncorrelated, the χ values cluster around the λ axis.

Normal Copula

Let $u_j \sim U(0, 1)$ for $j = 1, \ldots, m$, where $U(0, 1)$ represents the uniform distribution on the $[0, 1]$ interval. Let Σ be the correlation matrix with $m(m-1)/2$ parameters satisfying the positive semidefiniteness constraint. The normal copula can be written as

$$C_\Sigma(u_1, u_2, \ldots u_m) = \mathbf{\Phi}_\Sigma\left(\Phi^{-1}(u_1), \ldots \Phi^{-1}(u_m)\right)$$

where Φ is the distribution function of a standard normal random variable and $\mathbf{\Phi}_\Sigma$ is the m-variate standard normal distribution with mean vector 0 and covariance matrix Σ. That is, the distribution $\mathbf{\Phi}_\Sigma$ is $N_m(0, \Sigma)$.

Simulation

For the normal copula, the input of the simulation is the correlation matrix Σ. The normal copula can be simulated by the following steps in which $\boldsymbol{U} = (U_1, \ldots, U_m)$ denotes one random draw from the copula:

1. Generate a multivariate normal vector $\boldsymbol{Z} \sim N(0, \Sigma)$ where Σ is an m-dimensional correlation matrix.
2. Transform the vector $\boldsymbol{Z}$ into $\boldsymbol{U} = (\Phi(Z_1), \ldots, \Phi(Z_m))^T$, where Φ is the distribution function of univariate standard normal.

The first step can be achieved by Cholesky decomposition of the correlation matrix $\Sigma = LL^T$ where L is a lower triangular matrix with positive elements on the diagonal. If $\tilde{\boldsymbol{Z}} \sim N(0, I)$, then $L\tilde{\boldsymbol{Z}} \sim N(0, \Sigma)$.

Fitting

To fit a normal copula is to estimate the covariance matrix Σ from an input sample data set. Given a random sample $\boldsymbol{u}_i = (u_{i,1}, \ldots, u_{i,m})^\top$ where $i = 1, \ldots, n$, the log-likelihood function is

$$\log L(\Sigma; \boldsymbol{u}_1, \ldots, \boldsymbol{u}_n) = \sum_{t=1}^{n} \log f_\Sigma(\Phi^{-1}(u_{t,1}), \ldots, \Phi^{-1}(u_{t,m})) - \sum_{t=1}^{n}\sum_{j=1}^{m} \log \phi(\Phi^{-1}(u_{t,j}))$$

Here f_Σ is the joint density of the multivariate normal with mean zero and variance Σ, and ϕ is the univariate density of the standard normal distribution. Note that the second term is not related to the parameters Σ and, therefore, can be ignored during the optimization. The restriction that Σ is a correlation matrix is very inconvenient, and it is common practice to circumvent this problem by first assuming that Σ has the covariance form. Therefore, Σ can be estimated by

$$\widehat{\Sigma} = \frac{1}{n}\sum_{i=1}^{n} \zeta_i \zeta_i^T$$

where

$$\zeta_i = \left(\Phi^{-1}(u_{i,1}), \Phi^{-1}(u_{i,2}), \ldots, \Phi^{-1}(u_{i,m})\right)^T$$

This estimate is consistent with the form of a covariance matrix but not necessarily with the form of a correlation matrix. The approximation to the original MLE problem can be obtained using the normalizing operator defined as follows:

$$\Delta(\Sigma) = \text{diag}(\sigma_{11}^{1/2}, \ldots, \sigma_{mm}^{1/2})$$
$$\mathcal{P}(\Sigma) = (\Delta(\Sigma))^{-1}\Sigma(\Delta(\Sigma))^{-1}$$

Student's *t* copula

Let $\Theta = \{(\nu, \Sigma) : \nu \in (1, \infty), \Sigma \in \mathbb{R}^{m\times m}\}$and let t_ν be a univariate t distribution with ν degrees of freedom.

The Student's t copula can be written as

$$C_\Theta(u_1, u_2, \ldots u_m) = \boldsymbol{t}_{\nu,\Sigma}\left(t_\nu^{-1}(u_1), t_\nu^{-1}(u_2), \ldots, t_\nu^{-1}(u_m)\right)$$

where $\boldsymbol{t}_{\nu,\Sigma}$ is the multivariate Student's t distribution with a correlation matrix Σ with ν degrees of freedom.

Simulation

The input parameters for the simulation are (ν, Σ). The t copula can be simulated by the following the two steps:

1. Generate a multivariate vector $\boldsymbol{X} \sim t_m(\nu, 0, \Sigma)$ following the centered t distribution with ν degrees of freedom and correlation matrix Σ.
2. Transform the vector $\boldsymbol{X}$ into $\boldsymbol{U} = (t_\nu(X_1), \ldots, t_\nu(X_m))^T$, where t_ν is the distribution function of univariate t distribution with ν degrees of freedom.

To simulate centered multivariate t random variables, you can use the property that $\boldsymbol{X} \sim t_m(\nu, 0, \Sigma)$ if $\boldsymbol{X} = \sqrt{\nu/s}\boldsymbol{Z}$, where $\boldsymbol{Z} \sim N(0, \Sigma)$ and the univariate random variable $s \sim \chi^2_\nu$.

Fitting

To fit a t copula is to estimate the covariance matrix Σ and degrees of freedom ν from a given multivariate data set. Given a random sample$\boldsymbol{u}_i = (u_{i,1}, \ldots, u_{i,m})^\top, i = 1, \ldots, n$ that has uniform marginal distributions, the log likelihood is

$$\begin{aligned}&\log L(\nu, \Sigma; u_{i,1}, \ldots, u_{i,m})\\ &= \sum_{i=1}^{n} \log g_{\nu,\Sigma}(t_\nu^{-1}(u_{i,1}), \ldots, t_\nu^{-1}(u_{i,m})) - \sum_{i=1}^{n}\sum_{j=1}^{m} \log g_\nu(t_\nu^{-1}(u_{i,j}))\end{aligned}$$

where ν denotes the degrees of freedom of the t copula, $g_{\nu,\Sigma}$ denotes the joint density function of the centered multivariate t distribution with parameters (ν, Σ), t_ν is the distribution function of a univariate t distribution with ν degrees of freedom, Σ is a correlation matrix, and g_ν is the density function of univariate t distribution with ν degrees of freedom.

The log likelihood can be maximized with respect to the parameters $\theta = (\nu, \Sigma) \in \Theta$ using numerical optimization. If you allow the parameters in Σ to be such that Σ is symmetric and with ones on the diagonal, then the MLE estimate for Σ might not be positive semidefinite. In that case, you need to apply the adjustment to convert the estimated matrix to positive semidefinite, as shown by McNeil, Frey, and Embrechts (2005), Algorithm 5.55.

When the dimension of the data m increases, the numerical optimization quickly becomes infeasible. It is common practice to estimate the correlation matrix Σ by calibration using Kendall's tau. Then, using this fixed Σ, the single parameter ν can be estimated by MLE. By proposition 5.37 in McNeil, Frey, and Embrechts (2005),

$$\rho_\tau(U_i, U_j) = \frac{2}{\pi}\arcsin\rho_{ij}$$

where ρ_τ is the Kendall's tau and ρ_{ij} is the off-diagonal elements of the correlation matrix Σ of the t copula. Therefore, an estimate for the correlation is

$$\hat{\rho}_{ij} = \sin\left(\frac{1}{2}\pi\hat{\rho}^\tau_{i,j}\right)$$

where $\hat{\rho}$ and $\hat{\rho}^{\tau}$ are the estimates of the sample correlation matrix and Kendall's tau, respectively. However, it is possible that the estimate of the correlation matrix $\hat{\Sigma}$ is not positive definite. In this case, there is a standard procedure that uses the eigenvalue decomposition to transform the correlation matrix into one that is positive definite. Let Σ be a symmetric matrix with ones on the diagonal, with off-diagonal entries in $[-1, 1]$. If Σ is not positive semidefinite, use Algorithm 5.55 from McNeil, Frey, and Embrechts (2005):

1. Compute the eigenvalue decomposition $\Sigma = EDE^T$, where D is a diagonal matrix that contains all the eigenvalues and E is an orthogonal matrix that contains the eigenvectors.
2. Construct a diagonal matrix $\tilde{D}$ by replacing all negative eigenvalues in D by a small value $\delta > 0$.
3. Compute $\tilde{\Sigma} = E\tilde{D}E^T$, which is positive definite but not necessarily a correlation matrix.
4. Apply the normalizing operator $\mathcal{P}$ on the matrix $\tilde{\Sigma}$ to obtain the correlation matrix desired.

The log likelihood function and its gradient function for a single observation are listed as follows, where $\boldsymbol{\zeta} = (\zeta_1, \ldots, \zeta_m)$, with $\zeta_j = t_\nu^{-1}(u_j)$, and g is the derivative of the $\log\Gamma$ function:

$$
\begin{aligned}
l = \log(c) &= -\frac{1}{2}\log(|\Sigma|) + \log\Gamma\left(\frac{\nu+m}{2}\right) + (m-1)\log\Gamma\left(\frac{\nu}{2}\right) - m\log\Gamma\left(\frac{\nu+1}{2}\right) \\
&\quad - \frac{\nu+m}{2}\log(1+\zeta^T\Sigma^{-1}\zeta/\nu) + \frac{\nu+1}{2}\sum_{j=1}^{m}\log\left(1+\frac{\zeta_j^2}{\nu}\right) \\
\frac{\partial l}{\partial \nu} &= \frac{1}{2}g\left(\frac{\nu+m}{2}\right) + \frac{m-1}{2}g\left(\frac{\nu}{2}\right) - \frac{m}{2}g\left(\frac{\nu+1}{2}\right) \\
&\quad - \frac{1}{2}\log(1+\zeta^T\Sigma^{-1}\zeta/\nu) + \frac{\nu+m}{2\nu^2}\frac{\zeta^T\Sigma^{-1}\zeta}{1+\zeta^T\Sigma^{-1}\zeta/\nu} \\
&\quad + \frac{1}{2}\sum_{j=1}^{m}\log(1+\zeta_j^2/\nu) - \frac{\nu+1}{2\nu^2}\sum_{j=1}^{m}\frac{\zeta_j^2}{1+\zeta_j^2/\nu} \\
&\quad - \frac{(\nu+m)}{\nu}\frac{\zeta^T\Sigma^{-1}(d\zeta/d\nu)}{1+\zeta^T\Sigma^{-1}\zeta/\nu} + \frac{\nu+1}{\nu}\sum\frac{\zeta_j(d\zeta_j/d\nu)}{1+\zeta_j^2/\nu}
\end{aligned}
$$

The derivative of the likelihood with respect to the correlation matrix Σ follows:

$$
\begin{aligned}
\frac{\partial l}{\partial \Sigma} &= -\frac{1}{2}(\Sigma^{-1})^T + \frac{\nu+m}{2}\frac{\Sigma^{-T}\zeta\zeta^T\Sigma^{-T}/\nu}{1+\zeta^T\Sigma^{-1}\zeta/\nu} \\
&= -\frac{1}{2}(\Sigma^{-1})^T + \frac{\nu+m}{2}\frac{\Sigma^{-T}\zeta\zeta^T\Sigma^{-T}}{\nu+\zeta^T\Sigma^{-1}\zeta}
\end{aligned}
$$

Archimedean Copulas

Overview of Archimedean Copulas

Let function $\phi : [0,1] \rightarrow [0,\infty)$ be a strict Archimedean copula generator function and suppose its inverse ϕ^{-1} is completely monotonic on $[0,\infty)$. A strict generator is a decreasing function $\phi : [0,1] \rightarrow [0,\infty)$ that satisfies $\phi(0) = \infty$ and $\phi(1) = 0$. A decreasing function $f(t) : [a,b] \rightarrow (-\infty,\infty)$ is completely monotonic if it satisfies

$$(-1)^k \frac{d^k}{dt^k} f(t) \geq 0, k \in \mathbb{N}, t \in (a,b)$$

An Archimedean copula is defined as follows:

$$C(u_1,u_2,\ldots,u_m) = \phi^{-1}\Big(\phi(u_1) + \cdots + \phi(u_m)\Big)$$

The Archimedean copulas available in the COPULA procedure are the Clayton copula, the Frank copula, and the Gumbel copula.

Clayton Copula

Let the generator function $\phi(u) = \theta^{-1}\left(u^{-\theta} - 1\right)$. A Clayton copula is defined as

$$C_\theta(u_1,u_2,\ldots,u_m) = \left[\sum_{i=1}^{m} u_i^{-\theta} - m + 1\right]^{-1/\theta}$$

with $\theta > 0$.

Frank Copula

Let the generator function be

$$\phi(u) = -\log\left[\frac{\exp(-\theta u) - 1}{\exp(-\theta) - 1}\right]$$

A Frank copula is defined as

$$C_\theta(u_1,u_2,\ldots,u_m) = \frac{1}{\theta}\log\left\{1 + \frac{\prod_{i=1}^{m}[\exp(-\theta u_i) - 1]}{[\exp(-\theta) - 1]^{m-1}}\right\}$$

with $\theta \in (-\infty,\infty)\backslash\{0\}$ for $m = 2$ and $\theta > 0$ for $m \geq 3$.

Gumbel Copula

Let the generator function $\phi(u) = (-\log u)^\theta$. A Gumbel copula is defined as

$$C_\theta(u_1,u_2,\ldots,u_m) = \exp\left\{-\left[\sum_{i=1}^{m}(-\log u_i)^\theta\right]^{1/\theta}\right\}$$

with $\theta > 1$.

Simulation

Suppose the generator of the Archimedean copula is ϕ. Then the simulation method using Laplace-Stieltjes transformation of the distribution function is given by Marshall and Olkin (1988) where $\tilde{F}(t) = \int_0^\infty e^{-tx} dF(x)$:

1. Generate a random variable V with the distribution function F such that $\tilde{F}(t) = \phi^{-1}(t)$.
2. Draw samples from independent uniform random variables $X_1, \ldots, X_m$.
3. Return $\boldsymbol{U} = (\tilde{F}(-\log(X_1)/V), \ldots \tilde{F}(-\log(X_m)/V))^T$.

The Laplace-Stieltjes transformations are as follows:

- For the Clayton copula, $\tilde{F} = (1+t)^{-1/\theta}$, and the distribution function F is associated with a Gamma random variable with shape parameter θ^{-1} and scale parameter one.
- For the Gumbel copula, $\tilde{F} = \exp(-t^{1/\theta})$, and F is the distribution function of the stable variable $\text{St}(\theta^{-1}, 1, \gamma, 0)$ with $\gamma = [\cos(\pi/(2\theta))]^\theta$.
- For the Frank copula with $\theta > 0$, $\tilde{F} = -\log\{1 - \exp(-t)[1 - \exp(-\theta)]\}/\theta$, and F is a discrete probability function $P(V = k) = (1 - \exp(-\theta))^k/(k\theta)$. This probability function is related to a logarithmic random variable with parameter value $1 - e^{-\theta}$.

For details about simulating a random variable from a stable distribution, see Theorem 1.19 in Nolan (2010). For details about simulating a random variable from a logarithmic series, see Chapter 10.5 in Devroye (1986).

For a Frank copula with $m = 2$ and $\theta < 0$, the simulation can be done through conditional distributions as follows:

1 Draw independent v_1, v_2 from a uniform distribution.

2 Let $u_1 = v_1$.

3 Let $u_2 = -\frac{1}{\theta}\log\left(1 + \frac{v_2(1-e^{-\theta})}{v_2(e^{-\theta v_1}-1)-e^{-\theta v_1}}\right)$.

Fitting

One method to estimate the parameters is to calibrate with Kendall's tau. The relation between the parameter θ and Kendall's tau is summarized in the following table for the three Archimedean copulas.

Table 11.2 Calibration Using Kendall's Tau

Copula Type	τ	**Formula for** θ
Clayton	$\theta/(\theta+2)$	$2\tau/(1-\tau)$
Gumbel	$1 - 1/\theta$	$1/(1-\tau)$
Frank	$1 - 4\theta^{-1}(1 - D_1(\theta))$	No closed form

In Table 11.2, $D_1(\theta) = \theta^{-1} \int_0^\theta t/(\exp(t)-1)dt$ for $\theta > 0$, and $D_1(\theta) = D_1(\theta) + 0.5\theta$ for $\theta < 0$. In addition, for the Frank copula, the formula for θ has no closed form. The numerical algorithm for root finding can be used to invert the function $\tau(\theta)$ to obtain θ as a function of τ.

Alternatively, you can use the MLE or the CMLE method to estimate the parameter θ given the data $\mathbf{u} = \{u_{i,j}\}$ and $i = 1, \ldots, n$, $j = 1, \ldots, m$. The log-likelihood function for each type of Archimedean copula is provided in the following sections.

Fitting the Clayton Copula

For the Clayton copula, the log-likelihood function is as follows (Cherubini, Luciano, and Vecchiato 2004, Chapter 7):

$$l = n\left[m\log(\theta) + \log\left(\Gamma\left(\frac{1}{\theta}+m\right)\right) - \log\left(\Gamma\left(\frac{1}{\theta}\right)\right)\right] - (\theta+1)\sum_{i,j}\log u_{ij}$$
$$-\left(\frac{1}{\theta}+m\right)\sum_i \log\left(\sum_j u_{ij}^{-\theta} - m + 1\right)$$

Let $g(\cdot)$ be the derivative of $\log(\Gamma(\cdot))$. Then the first order derivative is

$$\frac{dl}{d\theta} = n\left[\frac{m}{\theta} + g\left(\frac{1}{\theta}+m\right)\frac{-1}{\theta^2} - g\left(\frac{1}{\theta}\right)\frac{-1}{\theta^2}\right]$$
$$-\sum_{i,j}\log(u_{ij}) + \frac{1}{\theta^2}\sum_i \log\left(\sum_j u_{ij}^{-\theta} - m + 1\right)$$
$$-\left(\frac{1}{\theta}+m\right)\sum_i \frac{-\sum_j u_{ij}^{-\theta}\log(u_{ij})}{\sum_j u_{ij}^{-\theta} - m + 1}$$

The second order derivative is

$$\frac{d^2l}{d\theta^2} = n\left\{\frac{-m}{\theta^2} + g'\left(\frac{1}{\theta}+m\right)\frac{1}{\theta^4} + g\left(\frac{1}{\theta}+m\right)\frac{2}{\theta^3} - g'\left(\frac{1}{\theta}\right)\frac{1}{\theta^4} - g\left(\frac{1}{\theta}\right)\frac{2}{\theta^3}\right\}$$
$$-\frac{2}{\theta^3}\sum_i \log\left(\sum_j u_{ij}^{-\theta} - m + 1\right)$$
$$+\frac{2}{\theta^2}\sum_i \frac{-\sum_j u_{ij}^{-\theta}\log u_{ij}}{\sum_j u_{ij}^{-\theta} - m + 1}$$
$$-\left(\frac{1}{\theta}+m\right)\sum_i\left\{\frac{\sum_j u_{ij}^{-\theta}(\log u_{ij})^2}{\sum_j u_{ij}^{-\theta} - m + 1} - \left(\frac{\sum_j u_{ij}^{-\theta}\log u_{ij}}{\sum_j u_{ij}^{-\theta} - m + 1}\right)^2\right\}$$

Fitting the Gumbel Copula

A different parameterization $\alpha = \theta^{-1}$ is used for the following part, which is related to the fitting of the Gumbel copula. For Gumbel copula, you need to compute $\phi^{-1(m)}$. It turns out that for $k = 1, 2, \ldots, m$,

$$\phi^{-1(k)}(u) = (-1)^k \alpha \exp(-u^\alpha) u^{-k+\alpha} \Psi_{k-1}(u^\alpha)$$

where Ψ_{k-1} is a function that is described later. The copula density is given by

$$\begin{aligned} c &= \phi^{-1(m)}(x)\prod_k \phi'(u_k) \\ &= (-1)^m \alpha \exp\left(-x^\alpha\right) x^{-k+\alpha}\Psi_{m-1}\left(x^\alpha\right)\prod_k \phi'(u_k) \\ &= (-1)^m f_1 f_2 f_3 f_4 f_5 \end{aligned}$$

where $x = \sum_k \phi(u_k)$, $f_1 = \alpha$, $f_2 = \exp(-x^\alpha), f_3 = x^{-k+\alpha}, f_4 = \Psi_{m-1}(x^\alpha)$, and $f_5 = (-1)^m \prod_k \phi'(u_k)$.

The log density is

$$\begin{aligned} l &= \log(c) \\ &= \log(f_1) + \log(f_2) + \log(f_3) + \log(f_4) + \log\left((-1)^m f_5\right) \end{aligned}$$

Now the first order derivative of the log density has the decomposition

$$\frac{dl}{d\alpha} = \frac{1}{c}\frac{dc}{d\alpha} = \sum_{j=1}^{4}\frac{1}{f_j}\frac{df_j}{d\alpha} + \frac{d\sum_k \log\left(-\phi'(u_k)\right)}{d\alpha}$$

Some of the terms are given by

$$\begin{aligned} \frac{1}{f_1}\frac{df_1}{d\alpha} &= \frac{1}{\alpha} \\ \frac{1}{f_2}\frac{df_2}{d\alpha} &= -x^\alpha \log(x) - \alpha x^{\alpha-1}\frac{dx}{d\alpha} \\ \frac{1}{f_3}\frac{df_3}{d\alpha} &= \log(x) + (-k+\alpha)x^{-1}\frac{dx}{d\alpha} \end{aligned}$$

where

$$\frac{dx}{d\alpha} = \sum (-\log u_k)^{1/\alpha}\log(-\log u_k)\left(\frac{-1}{\alpha^2}\right)$$

The last term in the derivative of the $dl/d\alpha$ is

$$\begin{aligned} \log\left(-\phi'(u_k)\right) &= \log\left(\frac{1}{\alpha}(-\log u_k)^{\frac{1}{\alpha}-1}\frac{1}{u_k}\right) \\ &= -\log\alpha - \log(u_k) + \left(\frac{1}{\alpha}-1\right)\log(-\log(u_k)) \\ \frac{d\sum_k \log\left(-\phi'(u_k)\right)}{d\alpha} &= \sum_{k=1}^{m} -\frac{1}{\alpha} - \frac{1}{\alpha^2}\log(-\log(u_k)) \\ &= -\frac{m}{\alpha} - \frac{1}{\alpha^2}\sum_{k=1}^{m}\log(-\log(u_k)) \end{aligned}$$

Now the only remaining term is f_4, which is related to Ψ_{m-1}. Wu, Valdez, and Sherris (2007) show that $\Psi_k(x)$ satisfies a recursive equation

$$\Psi_k(x) = [\alpha(x-1)+k]\Psi_{k-1}(x) - \alpha x \Psi'_{k-1}(x)$$

with $\Psi_0(x) = 1$.

The preceding equation implies that $\Psi_{k-1}(x)$ is a polynomial of x and therefore can be represented as

$$\Psi_{k-1}(x) = \sum_{j=0}^{k-1} a_j(k-1,\alpha)\, x^j$$

In addition, its coefficient, denoted by $a_j(k-1,\alpha)$, is a polynomial of α. For simplicity, use the notation $a_j(\alpha) \equiv a_j(m-1,\alpha)$. Therefore,

$$f_4 = \Psi_{m-1}\left(x^\alpha\right) = \sum_{j=0}^{m-1} a_j(\alpha)\, x^{j\alpha}$$

$$\begin{aligned}\frac{df_4}{d\alpha} &= \frac{d\,\Psi_{m-1}(x^\alpha)}{d\alpha} \\ &= \sum_{j=0}^{m-1}\left[\frac{da_j(\alpha)}{d\alpha}x^{j\alpha} + a_j(\alpha)\,x^{j\alpha}\log(x)\,j + a_j(\alpha)(j\alpha)\,x^{j\alpha-1}\frac{dx}{d\alpha}\right]\end{aligned}$$

Fitting the Frank copula

For the Frank copula,

$$\phi^{-1(k)}(u) = -\frac{1}{\theta}\Psi_{k-1}\left((1+e^{-u}(e^{-\theta}-1))^{-1}\right)$$

When $\theta > 0$, a Frank copula has a probability density function

$$\begin{aligned}c &= \varphi^{-1(m)}(x)\prod_k \varphi'(u_k) \\ &= \frac{-1}{\theta}\Psi_{m-1}\left(\frac{1}{1+e^{-x}\left(e^{-\theta}-1\right)}\right)\prod_k \varphi'(u_k)\end{aligned}$$

where $x = \sum_k \varphi(u_k)$.

The log likelihood is

$$\log c = -\log(\theta) + \log\left(\Psi_{m-1}\left(\frac{1}{1+e^{-x}\left(e^{-\theta}-1\right)}\right)\right) + \sum \log\left(\varphi'(u_k)\right)$$

Denote

$$y = \frac{1}{1+e^{-x}\left(e^{-\theta}-1\right)}$$

Then the derivative of the log likelihood is

$$\frac{d\log c}{d\theta} = -\frac{1}{\theta} + \frac{1}{\Psi_{m-1}(y)}\frac{d\Psi_{m-1}}{d\theta} + \sum_k \frac{1}{\varphi'(u_k)}\frac{d\varphi'(u_k)}{d\theta}$$

The term in the last summation is

$$\frac{1}{\varphi'(u_k)}\frac{d\varphi'(u_k)}{d\theta} = \frac{1}{\theta\left(1-e^{\theta u_k}\right)}\left[1-e^{\theta u_k}+\theta u e^{\theta u_k}\right]$$

The function Ψ_{m-1} satisfies a recursive relation

$$\Psi_k(x) = x(x-1)\Psi'_{k-1}(x)$$

with $\Psi_0(x) = x-1$. Note that Ψ_{m-1} is a polynomial whose coefficients do not depend on θ; therefore,

$$\begin{aligned}\frac{d\Psi_{m-1}}{d\theta} &= \frac{d\Psi_{m-1}}{dy}\frac{dy}{d\theta}\\ &= \frac{d\Psi_{m-1}}{dy}\left[\frac{dy}{d\theta}+\frac{dy}{dx}\frac{dx}{d\theta}\right]\\ &= \frac{d\Psi_{m-1}}{dy}\left[\frac{e^{-x}e^{-\theta}}{\left[1+e^{-x}\left(e^{-\theta}-1\right)\right]^2}+\frac{e^{-x}\left(e^{-\theta}-1\right)}{\left[1+e^{-x}\left(e^{-\theta}-1\right)\right]^2}\frac{dx}{d\theta}\right]\end{aligned}$$

where

$$\begin{aligned}\frac{dx}{d\theta} &= \sum_k \frac{d\varphi(u_k)}{d\theta} = \sum_k\left[-\frac{u_k e^{-\theta u_k}}{1-e^{-\theta u_k}}+\frac{e^{-\theta}}{1-e^{\theta}}\right]\\ &= \sum_k\left[-\frac{u_k}{e^{\theta u_k}-1}+\frac{1}{e^{\theta}-1}\right]\end{aligned}$$

For the case of $m = 2$ and $\theta < 0$, the bivariate density is

$$\log c = \log(\theta(1-e^{-\theta})) - \theta(u_1+u_2) - \log((1-e^{-\theta}-(1-e^{-\theta u_1})(1-e^{-\theta u_2}))^2)$$

Hierarchical Archimedean Copula (HAC) (Experimental)

Adopting the notations of Savu and Trede (2010), let L denote the total level of hierarchies and let D denote the dimension of the HAC. There are n_l distinct copulas at each level $l, l = 1,\ldots,L$. These copulas are indexed by $(l, j), j = 1,\ldots,n_l$. At each level, there are also d_l variables, $0 \le d_l \le D$ and $\sum_l d_l = D$. In the first step, all the variables at the lowest level are grouped into n_1 subsets, each subset being an ordinary multivariate Archimedean copula

$$C_{1,j}(\boldsymbol{u}_{1,j}) = \phi_{1,j}^{-1}\left(\sum_{\boldsymbol{u}_{1,j}}\phi_{1,j}(\boldsymbol{u}_{1,j})\right), j = 1,\ldots,n_1$$

where $\phi_{1,j}$ is the generator of copula $C_{1,j}$, $\boldsymbol{u}_{1,j}$ denotes the variables that belong to copula $C_{1,j}$, and the sum $\sum_{u_{1,j}}$ is the sum over each variable in the subset $\boldsymbol{u}_{1,j}$. The copulas $C_{1,j}$ can be different Archimedean copulas for $j = 1, \ldots, n_1$. Then at the second level, the copulas $C_{1,j}$ that are derived in the first level are aggregated as if they are individual variables. Suppose there are n_2 copulas and d_2 variables,

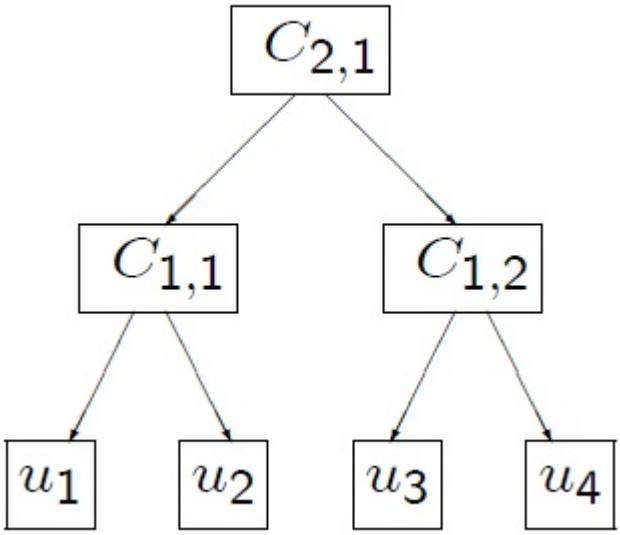
$$C_{2,j}(\boldsymbol{C}_{1,j}, \boldsymbol{u}_{2,j}) = \phi_{2,j}^{-1}\left(\sum_{\boldsymbol{C}_{1,j}} \phi_{2,j}(\boldsymbol{C}_{1,j}) + \sum_{\boldsymbol{u}_{2,j}} \phi_{2,j}(\boldsymbol{u}_{2,j})\right)$$

where $\phi_{2,j}$ denotes the generator of $C_{2,j}$ and $\boldsymbol{C}_{1,j}$ represents the subset of copulas in $C_{1,h}, h = 1, \ldots, n_1$, that is aggregated for copula $C_{2,j}$ for $j = 1, \ldots, n_2$. This structure continues until at level $l = L$ a single copula $C_{L,1}$ aggregates all the copulas at its previous level, $l = L - 1$.

A four-dimensional example that has total levels $L = 2$ and a structure shown in Figure 11.5 is defined as follows:

$$\begin{aligned} C_{2,1}(u_1, u_2, u_3, u_4) &= C_{2,1}\left(C_{1,1}(u_1, u_2), C_{1,2}(u_3, u_4)\right) \\ &= \phi_{2,1}^{-1}\left(\phi_{2,1} \circ \phi_{1,1}^{-1}\left(\phi_{1,1}(u_1) + \phi_{1,1}(u_2)\right) + \phi_{2,1} \circ \phi_{1,2}^{-1}\left(\phi_{1,2}(u_3) + \phi_{1,2}(u_4)\right)\right) \end{aligned}$$

Figure 11.5 Example Four-Dimensional Hierarchical Structure with Two Levels



Theorem 4.4 of McNeil (2008) states that the sufficient condition for a general hierarchical Archimedean structure to be a proper copula is that all appearing nodes of the form $\phi_{m,j} \circ \phi_{n,j}^{-1}$ have completely monotone derivatives. This condition places certain constraints on the copula parameters. In particular, if all the copulas in a hierarchical structure come from the Frank, Clayton, or Gumbel family, then $\theta_{m,j} \leq \theta_{n,j}$ for all j when $m < n$. Intuitively, this means that rank correlation must be increasing as you move down the hierarchical structure.

The hierarchical Archimedean copulas available in the COPULA procedure are the hierarchical versions of the Clayton, Frank, and Gumbel copulas.

Simulation

A slightly modified version of the recursive algorithm from McNeil (2008) works for all valid hierarchical structures that have Clayton, Frank, or Gumbel generators:

1. Start at $l = L$, and generate a random variable V with the distribution function F with Laplace transform $\phi_{L,1}^{-1}$.
2. For $l = L-1, \ldots, 1$, generate $u_{l,j}$ from its parent hierarchy. For $C_{l,j}$, recursively call this algorithm with the proper inner generators that correspond to the copula family.
3. Return $\boldsymbol{U} = (\phi_{L,1}^{-1}(-\log(u_1)/V), \ldots, \phi_{L,1}^{-1}(-\log(u_D)/V))^T$.

Let ϕ_1 be the outer generator and ϕ_2 the nested generator, and let θ_1 and θ_2 be the respective generator parameters. Let v be a draw from distribution function F with Laplace transform ϕ_1^{-1}. The inner copula generators $\phi_{12}(\cdot; v) = \exp(-v\phi_1 \circ \phi_2^{-1}(\cdot))$ and their corresponding Laplace transform distributions for the Clayton, Frank, and Gumbel family are summarized in Table 11.3.

Table 11.3 Inner Generators and Corresponding Distributions

Copula Type	$\phi_{12}(x; v)$	**Distribution with LT** $\phi_{12}(\cdot; v)$
Clayton	$\exp\left(v - v(1+x)^{\theta_1/\theta_2}\right)$	Tiled stable
Gumbel	$\exp(-vx^{\theta_1/\theta_2})$	$\text{Stable}\left(\frac{\theta_1}{\theta_2}, 1, \left(v\cos\frac{\theta_1\pi}{2\theta_2}\right)^{\theta_2/\theta_1}, 0\right)$
Frank	$\left(\frac{1}{1-e^{-\theta_1}}\left(1-\left(1-(1-e^{-\theta_2})\exp(-x)\right)^{\theta_1/\theta_2}\right)\right)^v$	No closed form

Note that when $\theta_1 = \theta_2$, the inner generators for the Clayton and Gumbel family both simplify to the generator of the independence copula, $\exp(-vx)$. For more information about simulating from the distribution with the Laplace transform given by the inner generator for the Frank family, see Hofert (2011). For more information about how to simulate from a tilted stable distribution, see McNeil (2008).

Canonical Maximum Likelihood Estimation (CMLE)

In the canonical maximum likelihood estimation (CMLE) method, it is assumed that the sample data $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{im})^\top$, $i = 1, \ldots, n$ have been transformed into uniform variates $\hat{\boldsymbol{u}}_i = (\hat{u}_{i1}, \ldots, \hat{u}_{im})$, $i = 1, \ldots, n$. One commonly used transformation is the nonparametric estimation of the CDF of the marginal distributions, which is closely related to empirical CDF,

$$\hat{u}_{i,j} = \hat{F}_{j,n}(x_{i,j})$$

where

$$\hat{F}_{j,n}(x) = \frac{1}{n+1}\sum_{i=1}^{n} I_{[x_{i,j} \le x]}$$

The transformed data $\hat{u}_{i,j}$ are used as if they had uniform marginal distributions; hence, they are called pseudo-samples. The function $\hat{F}_{j,n}$ is different from the standard empirical CDF in the scalar $1/(n+1)$, which is to ensure that the transformed data cannot be on the boundary of the unit interval $[0, 1]$. It is clear that

$$\hat{u}_{i,j} = \frac{1}{n+1}\text{rank}(x_{i,j})$$

where $\text{rank}(x_{i,j})$ is the rank among $i = 1, \ldots, n$ in increasing order.

Let $c(u_1, u_2, \ldots, u_m; \theta)$ be the density function of a copula $C(u_1, u_2, \ldots, u_m; \theta)$, and let θ be the parameter vector to be estimated. The parameter θ is estimated by maximum likelihood:

$$\hat{\theta} = \arg\max_{\theta \in \Theta} \sum_{i=1}^{n} \log c(\hat{u}_{i1}, \ldots, \hat{u}_{im}; \theta)$$

Exact Maximum Likelihood Estimation (MLE)

Suppose that the marginal distributions of vector elements $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{im})^\top$, $i = 1, \ldots, n$ are already known to be uniform. Then the parameter θ is estimated by exact maximum likelihood:

$$\hat{\theta} = \arg\max_{\theta \in \Theta} \sum_{i=1}^{n} \log c(x_{i1}, x_{i2}, \ldots, x_{im}; \theta)$$

Calibration Estimation

Instead of fitting the whole distribution as in MLE methods, you can directly use empirical estimates of distribution parameters. The unknown parameter that you want to estimate can be obtained by calibration using Kendall's tau. There exists a one-to-one map between the parameter at interest and Kendall's tau. Therefore, after you estimate the Kendall's tau, you can use the map to compute the parameter value. For example, the parameter matrix Σ in a t copula and the parameter θ in Archimedean copulas can be estimated in this manner. The most frequently used estimator of Kendall's tau is the rank correlation coefficient:

$$\hat{\rho}_\tau\left(X_i, X_j\right) = \binom{n}{2}^{-1} \sum_{1 \le t < s \le n} \text{sign}\left(\left(x_{t,i} - x_{s,i}\right)\left(x_{t,j} - x_{s,j}\right)\right)$$

The preceding formula is analogous to its population counterpart

$$\rho_\tau\left(X_i, X_j\right) = E\left[\text{sign}\left(\left(X_i - \tilde{X}_i\right)\left(X_j - \tilde{X}_j\right)\right)\right]$$

where $(\tilde{X}_i, \tilde{X}_j)$ has the same distribution but is independent of (X_i, X_j).

For Archimedean multivariate copulas there is only one parameter to estimate, τ (or its function θ), although for m variables there are $m(m-1)/2$ unique pairwise correlation coefficients. Denote the map from ρ_τ to θ by $\theta = \hat{\theta}(\rho_\tau)$. To aggregate the map, take simple arithmetic average:

$$\hat{\theta} = \frac{2}{m(m-1)} \sum_{1 \le i < j \le m} \hat{\theta}\left[\hat{\rho}_\tau\left(X_i, X_j\right)\right]$$

Nonlinear Optimization Options

PROC COPULA uses the nonlinear optimization (NLO) subsystem to perform nonlinear optimization tasks. In the PROC COPULA statement, you can specify nonlinear optimization options that are then passed to the NLO subsystem. For a list of all the nonlinear optimization options, see Chapter 7, "Nonlinear Optimization Methods."

Displayed Output

PROC COPULA produces displayed output described in the following sections.

Optimization Start and Resulting Parameter Estimates

If you specify the ITPRINT option in the PROC COPULA statement, PROC COPULA displays two tables, "Optimization Start Parameter Estimates" and "Optimization Results Parameter Estimates." Each table contains the following information for each model parameter:

- parameter number
- parameter name
- parameter estimate
- gradient of the objective function at the initial parameter values

In addition to this information, the table "Optimization Start Parameter Estimates" contains the following columns:

- lower-bound constraint
- upper-bound constraint

The value of the objective function at the parameter values is displayed below each table.

Iteration History for Parameter Estimates

If you specify the ITPRINT option in the PROC COPULA statement, PROC COPULA displays a table that contains the following information for each iteration. Note that some information is specific to the model-fitting method chosen (for example, Newton-Raphson, trust region, or quasi-Newton method).

- iteration number
- number of restarts since the fitting began
- number of function calls
- number of active constraints at the current solution

- value of the objective function (–1 times the log-likelihood value) at the current solution
- change in the objective function from previous iteration
- value of the maximum absolute gradient element
- step size (for Newton-Raphson and quasi-Newton methods)
- slope of the current search direction (for Newton-Raphson and quasi-Newton methods)
- lambda (for trust region method)
- radius value at current iteration (for trust region method)

Model Fit Summary

The "Model Fit Summary" table contains the following information:

- number of observations used
- number of missing values in data set, if any
- data set name
- type of model that was fit
- log-likelihood value at solution
- maximum absolute gradient at solution
- number of iterations
- optimization method
- value of Akaike's information criterion (AIC) at the solution (a smaller value indicates better fit)
- value of Schwarz-Bayesian criterion (SBC) at the solution (a smaller value indicates better fit)

Under the "Model Fit Summary" is a statement about whether the algorithm successfully converged.

Parameter Estimates

The "Parameter Estimates" table contains the estimates of the model parameters. For the normal copula, this table is not displayed because the only parameters are in the correlation matrix, which is displayed in the "Correlation Matrix" table. For the *t* copula, the parameter is the number of degrees of freedom; in the table it is called "DF." For Archimedean copulas such as Clayton, Frank, and Gumbel, the parameter is called "theta."

Correlation Matrix

The "Correlation Matrix" table contains the estimates of the model correlation matrix. This table is displayed only for elliptical copulas such as the normal and *t* copulas. Row and column names come from the list of variables defined in VAR statement.

OUTCOPULA= Data Set

The OUTCOPULA= data set consists of several rows. The first row (with _TYPE_='PARM') contains the parameter estimates in the model. For a *t* copula, the estimate is the number of degrees of freedom; for Archimedean copulas, the estimate is "theta." The second row (with _TYPE_='STD') contains the standard error for the parameter estimate in the model. These two rows do not appear for the normal copula.

If you use one of the elliptical copulas, *t* or normal, the rest of the data set contains the correlation matrix estimates. The correlation matrix appears in the observations with _TYPE_='CORR', and the _VARIABLE_ column contains the parameter names.

If METHOD=MLE and the nonlinear optimization subsystem is used, a _STATUS_ column is created that contains a character variable that indicates whether the optimization process reached convergence or failed to converge:

- 0 indicates that the convergence was reached
- 1 indicates that the maximum number of iterations allowed was exceeded
- 2 indicates a failure to improve the function value
- 3 indicates a failure to converge for one of the following reasons:
  - The objective function or its derivatives could not be evaluated or improved.
  - Linear constraints are dependent.
  - The algorithm failed to return to feasible region.
  - The number of iterations is greater than prespecified.

OUTPSEUDO=, OUT=, and OUTUNIFORM= Data Sets

The OUTPSEUDO=, OUT=, and OUTUNIF= data sets contain the same number of columns as specified in VAR statement. The names of the columns are taken from the same VAR statement list.

ODS Table Names

PROC COPULA assigns a name to each table it creates. You can use these names to denote the table when you use the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in Table 11.4.

Table 11.4 ODS Tables Produced in PROC COPULA

ODS Table Name	Description	Option
ODS Tables Created by the FIT Statement		
ConvergenceStatus	Convergence status	Default
Correlation	Correlation matrix estimates	Default with elliptical copulas
KendallCorrelation	Kendall Correlation matrix estimates	Default with elliptical copulas
SpearmanCorrelation	Spearman Correlation matrix estimates	Default with normal copula
FitSummary	Summary of nonlinear estimation	Default
ParameterEstimates	Parameter estimates	Default
ConvergenceStatus	Convergence status	ITPRINT
InputOptions	Input options	ITPRINT
IterHist	Iteration history	ITPRINT
IterStart	Optimization start	ITPRINT
IterStop	Optimization results	ITPRINT
ParameterEstimatesResults	Parameter estimates	ITPRINT
ParameterEstimatesStart	Parameter estimates	ITPRINT
ProblemDescription	Problem description	ITPRINT

ODS Graph Names

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*).

Before you create graphs, ODS Graphics must be enabled (for example, with the ODS GRAPHICS ON statement). For more information about enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" in that chapter.

The overall appearance of graphs is controlled by ODS styles. Styles and other aspects of using ODS Graphics are discussed in the section "A Primer on ODS Statistical Graphics" in that chapter.

PROC COPULA assigns a name to each graph it creates by using ODS. You can use these names to refer to the graphs when you use ODS. The names are listed in Table 11.5.

Table 11.5 ODS Graphics Produced by PROC COPULA

ODS Graph Name	Plot Description	Statement	PLOTS= Option
MatrixPlotOrig	Matrix panel of pairwise scatter plots of the original data	FIT	DATATYPE=BOTH, DATATYPE=ORIGINAL
MatrixPlotUnif	Matrix panel of pairwise scatter plots of the original data transformed into uniform marginals	FIT	DATATYPE=BOTH, DATATYPE=UNIFORM

Table 11.5 *continued*

ODS Graph Name	Plot Description	Statement	PLOTS= Option
MatrixPlotSOrig	Matrix panel of pairwise scatter plots of the simulated data	SIMULATE	DATATYPE=BOTH, DATATYPE=ORIGINAL
MatrixPlotSUnif	Matrix panel of pairwise scatter plots of the simulated data transformed into uniform marginals	SIMULATE	DATATYPE=BOTH, DATATYPE=UNIFORM
ScatterPlotOrig	Pairwise scatter plots of the original data	FIT	DATATYPE=BOTH UNPACK, DATATYPE=ORIGINAL UNPACK
ScatterPlotUnif	Pairwise scatter plots of the original data transformed into uniform marginals	FIT	DATATYPE=BOTH UNPACK, DATATYPE=UNIFORM UNPACK
ScatterPlotSOrig	Pairwise scatter plots of the simulated data	SIMULATE	DATATYPE=BOTH UNPACK, DATATYPE=ORIGINAL UNPACK
ScatterPlotSUnif	Pairwise scatter plots of the simulated data transformed into uniform marginals	SIMULATE	DATATYPE=BOTH UNPACK, DATATYPE=UNIFORM UNPACK
CdfContourPlot	Contour plot of theoretical bivariate CDF function	SIMULATE	DISTRIBUTION=CDF
CdfSurfacePlot	Surface plot of theoretical bivariate CDF function	SIMULATE	DISTRIBUTION=CDF
PdfContourPlot	Contour plot of theoretical bivariate PDF function	SIMULATE	DISTRIBUTION=PDF
PdfSurfacePlot	Surface plot of theoretical bivariate PDF function	SIMULATE	DISTRIBUTION=PDF
ChiPlotOrig	Tail dependence plot matrix with original data	FIT	
ChiPlotUnif	Tail dependence plot matrix with original data transformed into uniform marginals	FIT	
ChiPlotSOrig	Tail dependence plot matrix with simulated data	SIMULATE	
ChiPlotSUnif	Tail dependence plot matrix with simulated data transformed into uniform marginals	SIMULATE	

Table 11.5 *continued*

ODS Graph Name	Plot Description	Statement	PLOTS= Option
ChiPlot	Pairwise tail dependence plot of the data	FIT	UNPACK
ChiPlotS	Pairwise tail dependence plot of the simulated data	SIMULATE	UNPACK

Examples: COPULA Procedure

Example 11.1: Copula Based VaR Estimation

Value-at-risk (VaR) has become a de facto standard in financial risk management. The purpose of this measure is to give some quantitative insight to the riskiness of an asset portfolio. This measure is expressed generically in the following terms: What is the probability of losing no more than given percentage of a portfolio in a certain period of time? Or, what are the maximum possible losses at a given confidence level? The most simple and clearly wrong answer to this question is to compute the empirical quantile of past portfolio returns. The problem of this approach is that it does not take into account the dynamic nature of asset returns, the possibility of changing distribution, time memory, and, most importantly, cross-sectional dependence between individual assets in the portfolio.

This simple example of VaR computation takes into account at least cross-sectional dependence of the data. The end result is the prediction of the next-day maximum possible loss on the portfolio of stocks.

This example uses the daily returns on large stocks such as IBM, Microsoft, British Petroleum, Coca Cola, and Duke Energy. Output 11.1.1 shows the first 10 observations of the data.

Output 11.1.1 First 10 Observations of Daily Returns

Obs	date	ret_msft	ret_ko	ret_ibm	ret_duk	ret_bp
1	01/03/2008	0.004182	0.010367	0.002002	0.003503	0.019114
2	01/04/2008	-0.027960	0.001913	-0.035861	-0.000582	-0.014536
3	01/07/2008	0.006732	0.023607	-0.010671	0.025611	0.017922
4	01/08/2008	-0.033435	0.004239	-0.024610	-0.002838	-0.016049
5	01/09/2008	0.029560	0.026680	0.007301	0.010814	-0.027078
6	01/10/2008	-0.003054	0.004441	0.016414	-0.001689	-0.004395
7	01/11/2008	-0.012255	-0.027346	-0.022546	-0.012408	-0.018473
8	01/14/2008	0.013958	0.008418	0.053857	0.003427	0.001166
9	01/15/2008	-0.011318	-0.010851	-0.010689	-0.017075	-0.040925
10	01/16/2008	-0.022587	-0.015021	-0.001955	0.002316	-0.021336

The purpose of this exercise is to estimate one-day future losses of a stock portfolio. The simplest approach is to assume that the joint distribution of individual asset returns does not change with time. This might be close to the truth if only a small time interval is used. Then, a copula approach is used to estimate the joint distribution. Next, the new large sample of daily individual asset returns is simulated from the fitted joint distribution. These assets are then combined into a portfolio and its daily returns are computed. Finally, quantiles of simulated portfolio returns (which simply represent possible next-day losses of the portfolio) are examined.

So the first step is to cut off a small number of past return observations as in the following SAS data step:

```
/* Keep only the last 250 observations of the data */
data returns;
   set returns nobs=observ;
   if (_N_ > observ-250);
run;
```

The following statements fit a *t* copula to the returns data and at the same time simulate the sample from the fitted joint distribution:

```
/* Copula estimation and simulation of returns */
proc copula data = returns;
   var ret_ibm ret_msft ret_bp ret_ko ret_duk;
   * fit T-copula to stock returns;
   fit T /
          marginals = empirical
          method    = MLE
          plots      = (datatype = both);
   * simulate 10000 observations;
   * independent in time, dependent in cross-section;
   simulate /
            ndraws = 10000
            seed   = 1234
            out    = simulated_returns
            plots(unpack) = (datatype = original);
run;
```

The first line of COPULA procedure uses a VAR statement to specify the list of variables. In this example, these are daily returns of five large-company stocks.The next statement, FIT, requires some options. First, Student's *t* copula (T) is specified. After the slash, the MARGINALS=EMPIRICAL option specifies that an empirical distribution be fit. The choice of fitting method is MLE. The PLOTS=BOTH option requests that both original and transformed data graphs be organized into a symmetric panel.

Then, given the estimation results, the NDRAWS= option in the SIMULATE statement simulates 10,000 new observations for each asset return series. The SEED= option fixes the random number generator, the OUT= option specifies the name of SAS data set to contain the simulated sample, and the PLOT= option requests scatter plots of simulated returns in the original data scale.

The output of these statements is shown in Output 11.1.2.

Output 11.1.2 Copula Estimation

The COPULA Procedure

Model Fit Summary	
Number of Observations	250
Data Set	WORK.RETURNS
Copula Type	T
Log Likelihood	171.52667
Maximum Absolute Gradient	3.35807E-8
Number of Iterations	9
Optimization Method	Newton-Raphson
AIC	-321.05333
SBC	-282.31726

Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|
DF	6.713592	1.327787	5.06	<.0001

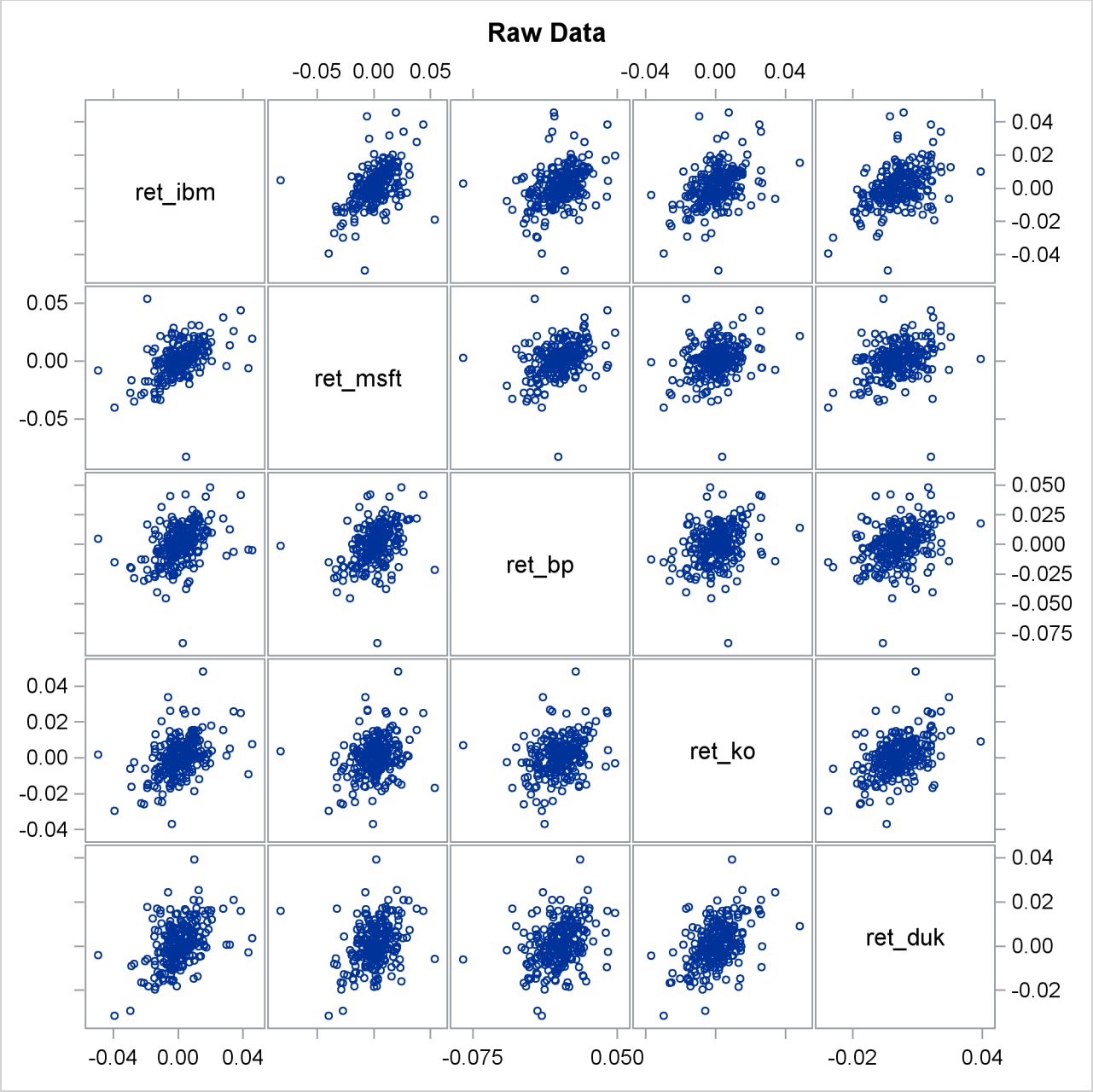
Correlation Matrix

	ret_ibm	ret_msft	ret_bp	ret_ko	ret_duk
ret_ibm	1.0000	0.5657	0.4663	0.4548	0.4740
ret_msft	0.5657	1.0000	0.4585	0.3234	0.3658
ret_bp	0.4663	0.4585	1.0000	0.3459	0.3576
ret_ko	0.4548	0.3234	0.3459	1.0000	0.4742
ret_duk	0.4740	0.3658	0.3576	0.4742	1.0000

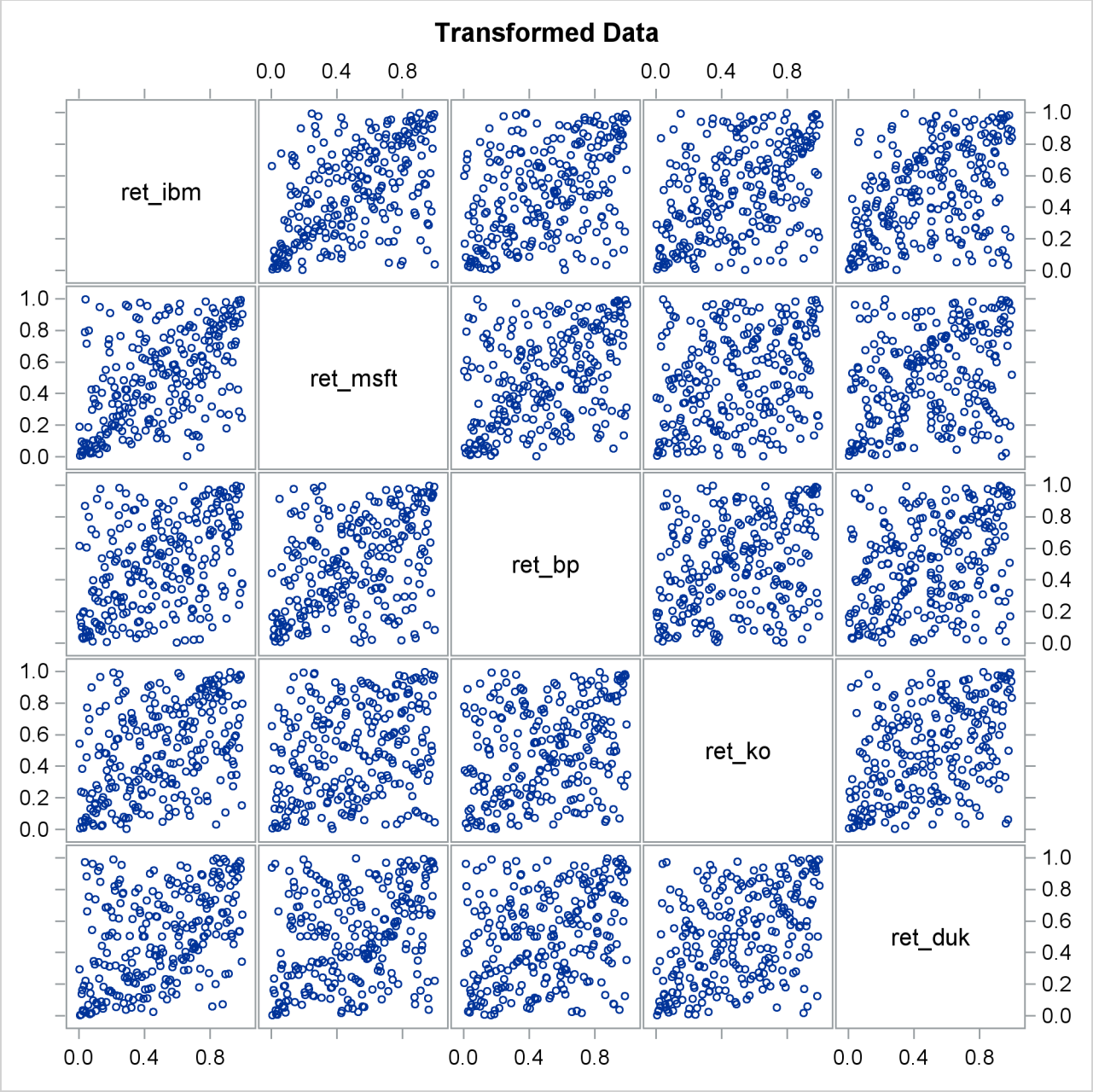
The first table in Output 11.1.2, "Model Fit Summary," provides some general description of copula model estimation. The second table, "Parameter Estimates," provides point estimates and inference on copula parameters. In this example the only parameter in this table is the number of degrees of freedom in the multivariate t distribution. The last table, "Correlation Matrix," contains estimates of copula model parameters.

The graphical output of the preceding statements is in Output 11.1.3 and in Output 11.1.4.

Output 11.1.3 Original Data



Output 11.1.4 Original Data Transformed into Uniform Marginals



Note that in Output 11.1.3 the most elliptical scatter plot, between IBM and MSFT, indicates the strongest dependence. Similarly, in Output 11.1.4 those graphs that are denser along the diagonal indicate the same thing.

Now the equally weighted next day portfolio return is computed. Each individual return is transformed into nominal scale first, then all returns are added up with equal weights, and the result is transformed into a net return by subtracting one.

```
/* compute equally weighted portfolio return */
data port_ret (drop = i ret);
   set simulated_returns;
   array returns{5} ret_ibm ret_msft ret_bp ret_ko ret_duk;
   ret =0;
   do i =1 to 5;
      ret = ret+ 0.2*exp(returns[i]);
   end;
   port_ret = ret-1;
run;
```

The final step is to compute empirical quantiles of simulated daily portfolio return. This is done with the help of PROC UNIVARIATE in the following statements:

```
/* compute descriptive statistics */
/* quantile table will give Value-at-Risk estimates for the portfolio */
proc univariate data = port_ret;
   var port_ret;
run;
```

Output 11.1.5 shows that with 99% confidence the potential loss on an equally weighted portfolio over the next day does not exceed 2.6% (the number in table is multiplied by 100). You can also say that there is no more than 5% chance of losing 1.5% of the portfolio value. These percentage measures are exactly the value-at-risk.

Output 11.1.5 Return Quantiles

The UNIVARIATE Procedure
Variable: port_ret

Quantiles (Definition 5)	
Level	**Quantile**
100% Max	0.048144752
99%	0.026628638
95%	0.015538196
90%	0.011573915
75% Q3	0.005801203
50% Median	0.000688897
25% Q1	-0.004953729
10%	-0.010636997
5%	-0.014677061
1%	-0.026629715
0% Min	-0.052757770

Example 11.2: Simulating Default Times

Suppose the correlation structure required for a normal copula function is already given. For example, it can be estimated from the historic data on default times in some set of industries, but this stage is not in the scope of this example. The correlation structure is saved in a SAS data set called inparm. The following statements and their output in Output 11.2.1 show that the correlation parameter is set at 0.8:

```
proc print data = inparm;
run;
```

Output 11.2.1 Copula Correlation Matrix

Obs	name	Y1	Y2
1	Y1	1.0	0.8
2	Y2	0.8	1.0

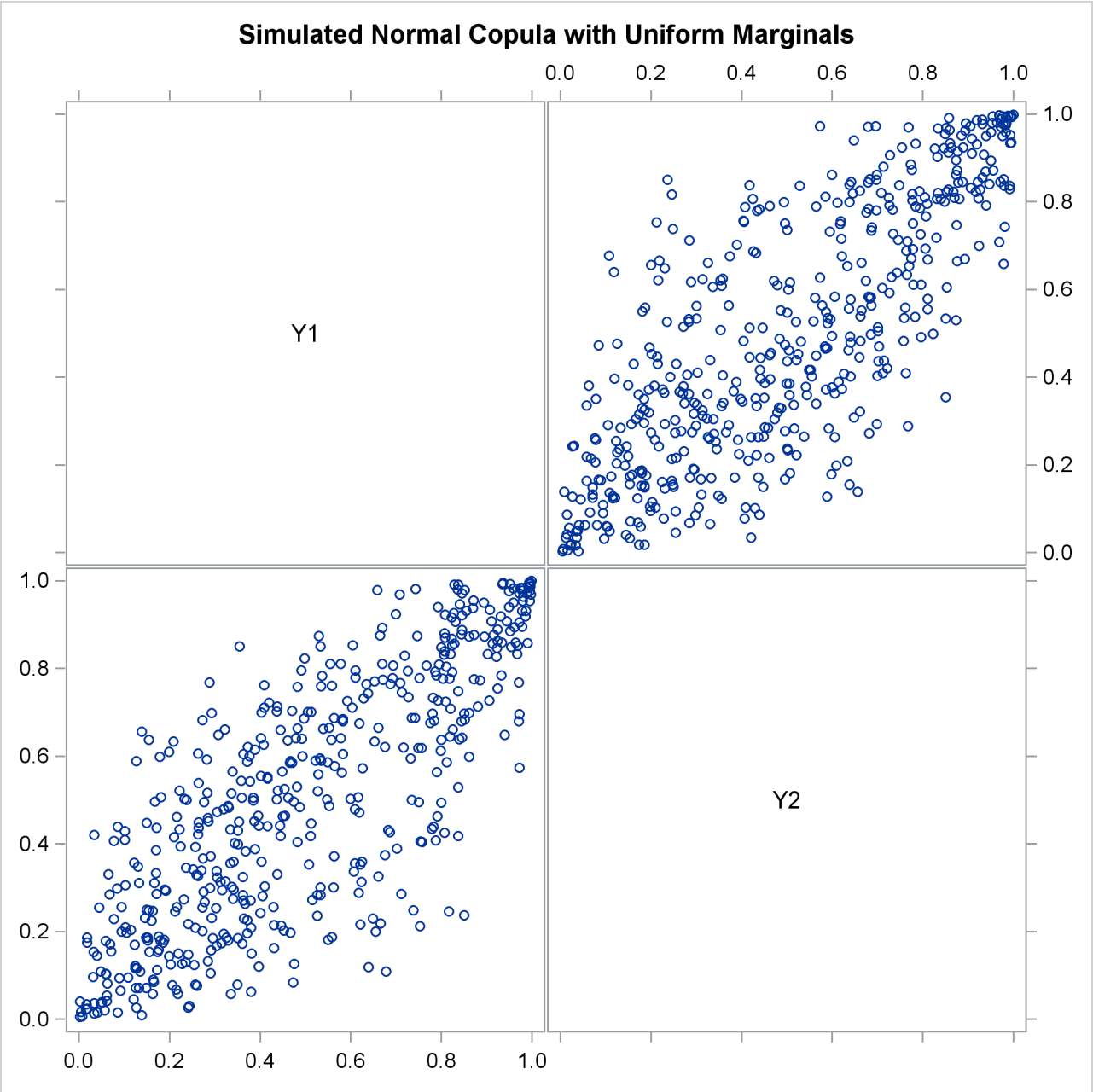
Now you use PROC COPULA to simulate the data. The VAR statement specifies the list of variables to contain simulated data. The DEFINE statement assigns the name COP and specifies a normal copula that reads the correlation matrix from the inparm data set.

The SIMULATE statement refers to the COP label defined in the VAR statement and specifies some options: the NDRAWS= option specifies a sample size, the SEED= option specifies 1234 as the random number generator seed, the OUTUNIFORM=NORMAL_UNIFDATA option names the output data set for the result of simulation in uniforms, and the PLOTS= option requests the matrix of data scatter plots and marginal distributions (DATATYPE=ORIGINAL) and theoretical cumulative distribution function contour and surface plots (DISTRIBUTION=CDF). Theoretical distribution graphs work only for the bivariate case.

```
/* simulate the data from bivariate normal copula */
proc copula ;
   var Y1-Y2;
   define cop normal (corr=inparm);
   simulate cop /
            ndraws     = 500
            seed       = 1234
            outuniform = normal_unifdata
            plots      = (datatype = original
                          distribution = cdf);
run;
```

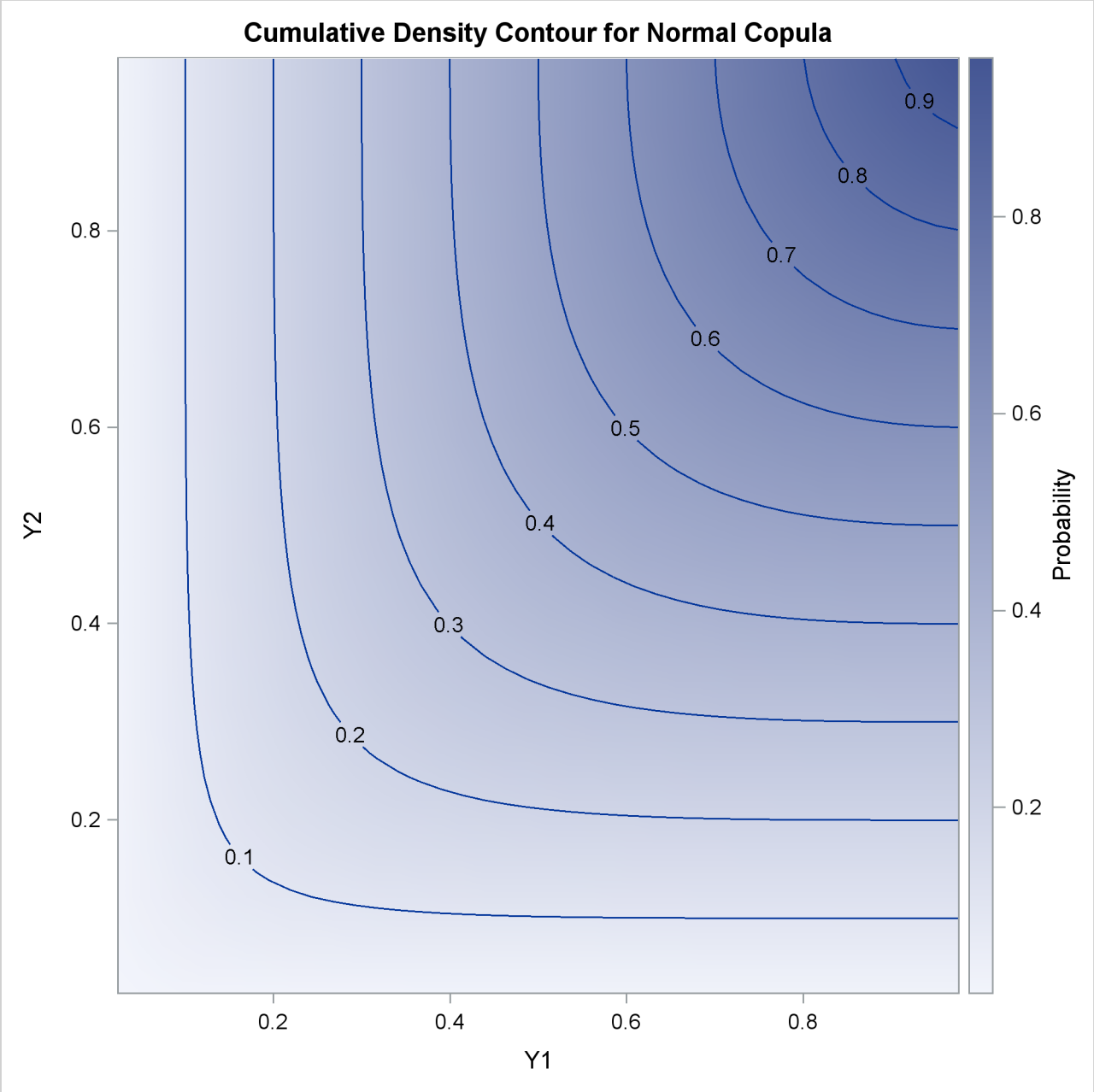
The graphical output is shown in Output 11.2.2 and in Output 11.2.3.

Output 11.2.2 Simulated Data, Uniform Marginals



Output 11.2.2 shows bivariate scatter plots of the simulated data. Also note that due to the high correlation parameter (0.8), the scatter plots are most dense around the 45 degree line, which indicates high dependence between the two variables.

Output 11.2.3 Joint Cumulative Distribution



Output 11.2.3 shows the theoretical CDF contour plot. If the correlation parameter were set to 0, then knowing copula properties you would expect perfectly parallel straight lines with the slope of –45 degrees. On the other hand, if the parameter were set to 1, you would expect perpendicular lines with corners lying on the diagonal.

The next DATA step transforms the variables from zero-one uniformly distributed to nonnegative exponentially distributed with parameter 0.5. Three indicator variables are added to the data set as well. SURVIVE1 and SURVIVE2 are equal to 1 if a respective company has remained in business for more than three years. SURVIVE is equal to 1 if both companies survived the same period together.

```
/* default time has exponential marginal distribution with parameter 0.5 */
data default;
   set normal_unifdata;
   array arr{2} Y1-Y2;
   array time{2} time1-time2;
   array surv{2} survive1-survive2;
   lambda = 0.5;
   do i=1 to 2;
      time[i] = -log(1-arr[i])/lambda;
      surv[i] = 0;
      if (time[i] >3) then surv[i]=1;
   end;
   survive = 0;
   if (time1 >3) && (time2 >3) then survive = 1;
run;
```

The first analysis step is to look at correlations between survival times of two companies. This step is performed with the following CORR procedure:

```
proc corr data = default plot=matrix kendall;
   var time1 time2;
run;
```

The output of this code is given in Output 11.2.4 and in Output 11.2.5.

Output 11.2.4 shows some descriptive statistics and two measures of correlation: Pearson and Kendall. Both of these measures indicate high and statistically significant dependence between life spans of two companies.

Output 11.2.4 Default Time Descriptive Statistics and Correlations

The CORR Procedure

2 Variables:	time1	time2

Simple Statistics

Variable	N	Mean	Std Dev	Median	Minimum	Maximum
time1	500	2.08347	2.23677	1.26496	0.00449	13.08462
time2	500	2.07547	2.19756	1.37603	0.01076	16.85567

Pearson Correlation Coefficients, N = 500
Prob > |r| under H0: Rho=0

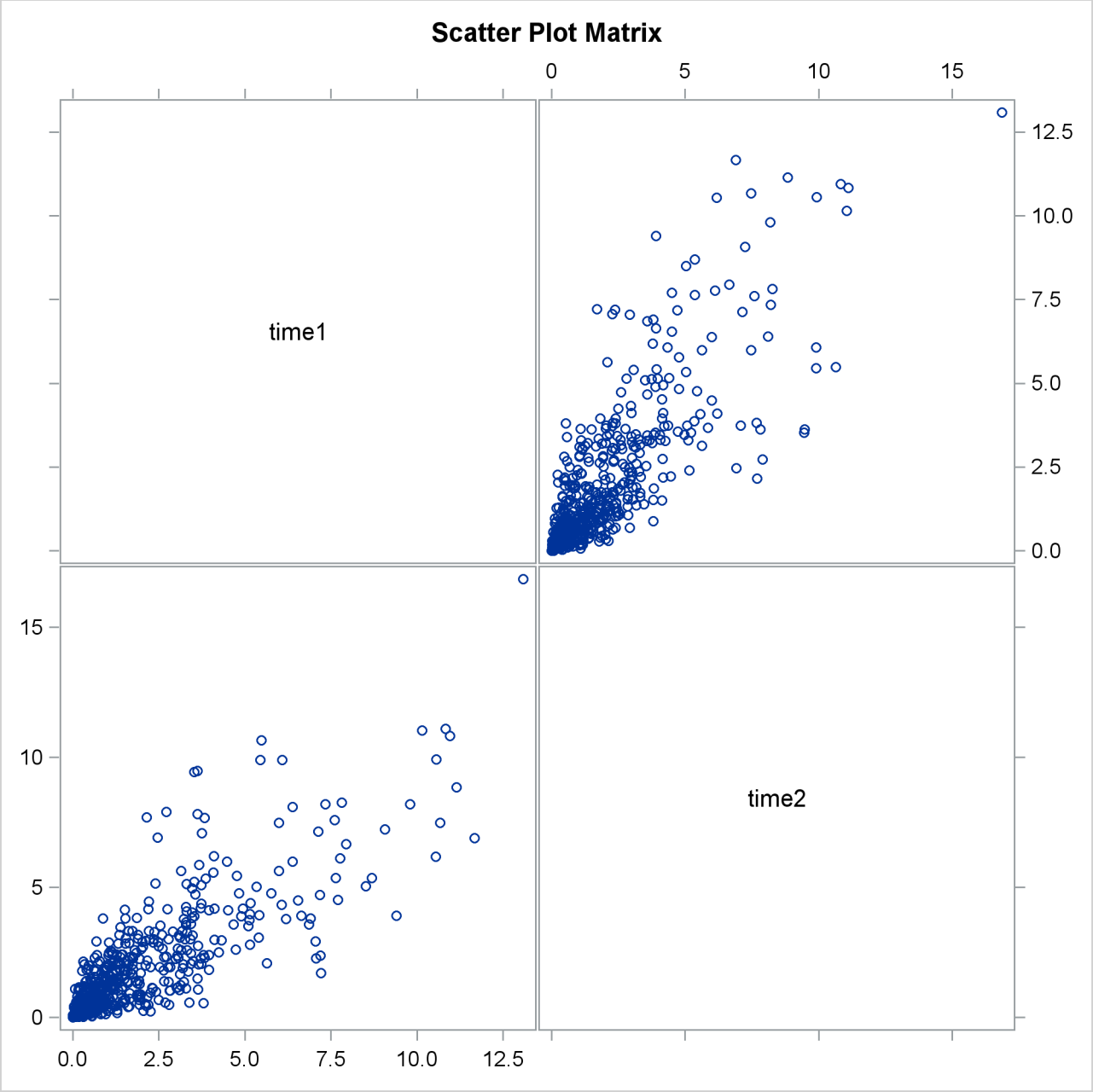
	time1	time2
time1	1.00000	0.80268 <.0001
time2	0.80268 <.0001	1.00000

Kendall Tau b Correlation Coefficients, N = 500
Prob > |tau| under H0: Tau=0

	time1	time2
time1	1.00000	0.59566 <.0001
time2	0.59566 <.0001	1.00000

Output 11.2.5 shows marginal distributions and scatter plots of simulated data. Distributions are noticeably close to exponential and scatter plots show a high degree of dependence.

Output 11.2.5 Default Times



The second and the last step is to empirically estimate the default probabilities of two companies. This is done in the following FREQ procedure:

```
proc freq data=default;
   table survive survive1-survive2;
run;
```

The result is shown in Output 11.2.6.

Output 11.2.6 Probabilities of Default

The FREQ Procedure

survive	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	415	83.00	415	83.00
1	85	17.00	500	100.00

survive1	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	374	74.80	374	74.80
1	126	25.20	500	100.00

survive2	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	390	78.00	390	78.00
1	110	22.00	500	100.00

Output 11.2.6 shows that the empirical default probabilities are 75% and 78%. Assuming that these companies are independent gives the probability estimate of both companies defaulting during the period of three years as: 0.75*0.78=0.59 (59%). Comparing this naive estimate with the much higher actual 83% joint default probability illustrates that neglecting the correlation between the two companies significantly underestimates the probability of default.

References

Cherubini, U., Luciano, E., and Vecchiato, W. (2004). *Copula Methods in Finance*. Chichester, UK: John Wiley & Sons.

Devroye, L. (1986). *Non-uniform Random Variate Generation*. New York: Springer-Verlag. http://luc.devroye.org/rnbookindex.html.

Fisher, N. I., and Switzer, P. (2001). "Graphical Assessment of Dependence: Is a Picture Worth 100 Tests?" *American Statistician* 55:233–239.

Galiani, S. S. (2003). *Copula Functions and Their Application in Pricing and Risk Managing Multiname Credit Derivative Products*. Master's thesis, King's College London. http://www.defaultrisk.com/pp_crdrv_41.htm.

Genest, C., Ghoudi, K., and Rivest, L. P. (1995). "A Semiparametric Estimation Procedure of Dependence Parameters in Multivariate Families of Distributions." *Biometrika* 82:543–552.

Hofert, M. (2011). "Efficiently Sampling Nested Archimedean Copulas." *Computational Statistics and Data Analysis* 55:57–70.

Joe, H. (1997). *Multivariate Models and Dependence Concepts*. London: Chapman & Hall.

Joe, H., and Xu, J. (1996). *The Estimation Method of Inference Functions for Margins for Multivariate Models*. Technical Report 166, University of British Columbia.

Marshall, A. W., and Olkin, I. (1988). "Families of Multivariate Distributions." *Journal of the American Statistical Association* 83:834–841.

McNeil, A., Frey, R., and Embrechts, P. (2005). *Quantitative Risk Management: Concepts, Techniques, and Tools*. Princeton, NJ: Princeton University Press.

McNeil, A. J. (2008). "Sampling Nested Archimedean Copulas." *Journal of Statistical Computation and Simulation* 78:567–581.

Mendes, B. V. M., de Melo, E. F. L., and Nelsen, R. B. (2007). "Robust Fits for Copula Models." *Communications in Statistics—Simulation and Computation* 36:997–1008.

Nelsen, R. B. (2006). *An Introduction to Copulas*. 2nd ed. New York: Springer.

Nolan, J. P. (2010). *Stable Distributions: Models for Heavy Tailed Data*. Boston: Birkhäuser.

Rüschendorf, L. (2009). "On the Distributional Transform, Sklar's Theorem, and the Empirical Copula Process." *Journal of Statistical Planning and Inference* 11:3921–3927.

Savu, C., and Trede, M. (2010). "Hierarchies of Archimedean Copulas." *Quantitative Finance* 10:295–304.

Sklar, A. (1959). "Fonctions de répartition à n dimensions et leurs marges." *Publications de l'Institut de Statistique de L'Université de Paris* 8:229–231.

Wu, F., Valdez, E., and Sherris, M. (2007). "Simulating from Exchangeable Archimedean Copulas." *Communications in Statistics—Simulation and Computation* 36:1019–1034.

Chapter 12
The COUNTREG Procedure

Contents

Overview: COUNTREG Procedure **582**
Getting Started: COUNTREG Procedure **584**
Syntax: COUNTREG Procedure **587**
- Functional Summary 588
- PROC COUNTREG Statement 591
- BAYES Statement 595
- BOUNDS Statement 601
- BY Statement 601
- CLASS Statement 601
- DISPMODEL Statement 603
- FREQ Statement 604
- INIT Statement 604
- MODEL Statement 605
- NLOPTIONS Statement 608
- OUTPUT Statement 608
- PERFORMANCE Statement 610
- PRIOR Statement 610
- RESTRICT Statement 611
- SCORE Statement 611
- SHOW Statement 613
- SPATIALDISPEFFECTS Statement (Experimental) 614
- SPATIALEFFECTS Statement (Experimental) 615
- SPATIALID Statement (Experimental) 615
- SPATIALZEROEFFECTS Statement (Experimental) 616
- STORE Statement 616
- TEST Statement 616
- WEIGHT Statement 618
- ZEROMODEL Statement 618

Details: COUNTREG Procedure **619**
- Specification of Regressors 619
- Missing Values 621
- Poisson Regression 622
- Conway-Maxwell-Poisson Regression 623
- Negative Binomial Regression 627
- Zero-Inflated Count Regression Overview 630
- Zero-Inflated Poisson Regression 630

Zero-Inflated Conway-Maxwell-Poisson Regression . . . 632
Zero-Inflated Negative Binomial Regression . . . 634
Spatial Lag of *X* Model (Experimental) . . . 636
Variable Selection . . . 637
Panel Data Analysis . . . 641
BY Groups and Scoring with an Item Store . . . 648
Parameter Naming Conventions for the RESTRICT, TEST, BOUNDS, and INIT Statements . . . 649
Computational Resources . . . 653
Nonlinear Optimization Options . . . 654
Covariance Matrix Types . . . 654
Displayed Output . . . 654
Bayesian Analysis . . . 656
Prior Distributions . . . 663
Automated MCMC Algorithm . . . 664
Marginal Likelihood . . . 666
OUTPUT OUT= Data Set . . . 669
OUTEST= Data Set . . . 670
ODS Table Names . . . 670
ODS Graphics . . . 671
Examples: COUNTREG Procedure . . . **673**
Example 12.1: Basic Models . . . 673
Example 12.2: ZIP and ZINB Models for Data That Exhibit Extra Zeros . . . 680
Example 12.3: Variable Selection . . . 690
References . . . **694**

Overview: COUNTREG Procedure

The COUNTREG (count regression) procedure analyzes regression models in which the dependent variable takes nonnegative integer or count values. The dependent variable is usually an *event count*, which refers to the number of times an event occurs. For example, an event count might represent the number of ship accidents per year for a given fleet. In count regression, the conditional mean $E(y_i|\mathbf{x}_i)$ of the dependent variable y_i is assumed to be a function of a vector of covariates $\mathbf{x}_i$.

The Poisson (log-linear) regression model is the most basic model that explicitly takes into account the nonnegative integer-valued aspect of the outcome. With this model, the probability of an event count is determined by a Poisson distribution, where the conditional mean of the distribution is a function of a vector of covariates. However, the basic Poisson regression model is limited because it forces the conditional mean of the outcome to equal the conditional variance. This assumption is often violated in real-life data. Negative binomial regression is an extension of Poisson regression in which the conditional variance can exceed the conditional mean. Also, a common characteristic of count data is that the number of zeros in the sample exceeds the number of zeros that are predicted by either the Poisson or negative binomial model. Zero-inflated Poisson (ZIP) and zero-inflated negative binomial (ZINB) models explicitly model the production of zero

counts to account for excess zeros and also enable the conditional variance of the outcome to differ from the conditional mean.

In zero-inflated models, additional zeros occur with probability φ_i, which is determined by a separate model, $\varphi_i = F(\mathbf{z}_i'\boldsymbol{\gamma})$, where F is the normal or logistic distribution function that results in a probit or logistic model and $\mathbf{z}_i$ is a set of covariates.

PROC COUNTREG supports the following models for count data:

- Poisson regression
- Conway-Maxwell-Poisson regression
- negative binomial regression with quadratic (NEGBIN2) and linear (NEGBIN1) variance functions (Cameron and Trivedi 1986)
- zero-inflated Poisson (ZIP) model (Lambert 1992)
- zero-inflated Conway-Maxwell-Poisson (ZICMP) model
- zero-inflated negative binomial (ZINB) model
- fixed-effects and random-effects Poisson models for panel data
- fixed-effects and random-effects negative binomial models for panel data
- all models in this list (except panel data models) that have spatial effects

The count data models have been used extensively in economics, political science, and sociology. For example, Hausman, Hall, and Griliches (1984) examine the effects of research and development expenditures on the number of patents obtained by US companies. Cameron and Trivedi (1986) study factors that affect the number of doctor visits that a group made during a two-week period. Greene (1994) studies the number of derogatory reports to a credit reporting agency for a group of credit card applicants. As a final example, Long (1997) analyzes the number of publications by PhD candidates in science in the final three years of their doctoral studies.

The COUNTREG procedure can use the maximum likelihood method and the Bayesian method. Initial starting values for the nonlinear optimizations are typically calculated by OLS. When a model that contains a dependent count variable is estimated using linear ordinary least squares (OLS) regression, the count nature of the dependent variable is ignored. This can lead to negative predicted counts and to parameter estimates that have undesirable properties in terms of statistical efficiency, consistency, and unbiasedness unless the mean of the counts is high, in which case the Gaussian approximation and linear regression might be satisfactory.

Getting Started: COUNTREG Procedure

The COUNTREG procedure is similar in use to other SAS regression model procedures. For example, the following statements are used to estimate a Poisson regression model:

```
proc countreg data=one ;
   model y = x / dist=poisson ;
run;
```

The response variable *y* is numeric and has nonnegative integer values. To allow for variance greater than the mean, specify the DIST=NEGBIN option to fit the negative binomial model instead of the Poisson.

The following example illustrates the use of PROC COUNTREG. The data are taken from Long (1997) and can be found in the SAS/ETS Sample Library. This study examines how factors such as gender (fem), marital status (mar), number of young children (kid5), prestige of the graduate program (phd), and number of articles published by the mentor (ment) of a doctoral candidate in science affect the number of articles (art) published by the scientist.

The first 10 observations are shown in Figure 12.1.

Figure 12.1 Article Count Data

Obs	art	fem	mar	kid5	phd	ment
1	3	0	1	2	1.38000	8.0000
2	0	0	0	0	4.29000	7.0000
3	4	0	0	0	3.85000	47.0000
4	1	0	1	1	3.59000	19.0000
5	1	0	1	0	1.81000	0.0000
6	1	0	1	1	3.59000	6.0000
7	0	0	1	1	2.12000	10.0000
8	0	0	1	0	4.29000	2.0000
9	3	0	1	2	2.58000	2.0000
10	3	0	1	1	1.80000	4.0000

The following SAS statements estimate the Poisson regression model:

```
proc countreg data=long97data;
   model art = fem mar kid5 phd ment / dist=poisson;
run;
```

The "Model Fit Summary" table, shown in Figure 12.2, lists several details about the model. By default, the COUNTREG procedure uses the Newton-Raphson optimization technique. The maximum log-likelihood value is shown, in addition to two information measures, Akaike's information criterion (AIC) and Schwarz's Bayesian information criterion (SBC), which can be used to compare competing Poisson models. Smaller values of these criteria indicate better models.

Figure 12.2 Estimation Summary Table for a Poisson Regression

The COUNTREG Procedure

Model Fit Summary	
Dependent Variable	art
Number of Observations	915
Data Set	WORK.LONG97DATA
Model	Poisson
Log Likelihood	-1651
Maximum Absolute Gradient	3.57454E-9
Number of Iterations	5
Optimization Method	Newton-Raphson
AIC	3314
SBC	3343

The parameter estimates of the model and their standard errors are shown in Figure 12.3. All covariates are significant predictors of the number of articles, except for the prestige of the program (phd), which has a p-value of 0.6271.

Figure 12.3 Parameter Estimates of Poisson Regression

Parameter Estimates					
Parameter	**DF**	**Estimate**	**Standard Error**	**t Value**	**Approx Pr > |t|**
Intercept	1	0.304617	0.102982	2.96	0.0031
fem	1	-0.224594	0.054614	-4.11	<.0001
mar	1	0.155243	0.061375	2.53	0.0114
kid5	1	-0.184883	0.040127	-4.61	<.0001
phd	1	0.012823	0.026397	0.49	0.6271
ment	1	0.025543	0.002006	12.73	<.0001

The following statements fit the negative binomial model. Although the Poisson model requires that the conditional mean equal the conditional variance, the negative binomial model allows for overdispersion; that is, the conditional variance can exceed the conditional mean.

```
proc countreg data=long97data;
   model art = fem mar kid5 phd ment / dist=negbin(p=2) method=qn;
run;
```

The fit summary is shown in Figure 12.4, and parameter estimates are listed in Figure 12.5.

Figure 12.4 Estimation Summary Table for a Negative Binomial Regression

The COUNTREG Procedure

Model Fit Summary	
Dependent Variable	art
Number of Observations	915
Data Set	WORK.LONG97DATA
Model	NegBin(p=2)
Log Likelihood	-1561
Maximum Absolute Gradient	5.72129E-7
Number of Iterations	16
Optimization Method	Quasi-Newton
AIC	3136
SBC	3170

Figure 12.5 Parameter Estimates of Negative Binomial Regression

Parameter Estimates					
Parameter	**DF**	**Estimate**	**Standard Error**	**t Value**	**Approx Pr > |t|**
Intercept	1	0.256144	0.138560	1.85	0.0645
fem	1	-0.216418	0.072672	-2.98	0.0029
mar	1	0.150489	0.082106	1.83	0.0668
kid5	1	-0.176415	0.053060	-3.32	0.0009
phd	1	0.015271	0.036040	0.42	0.6718
ment	1	0.029082	0.003470	8.38	<.0001
_Alpha	1	0.441620	0.052967	8.34	<.0001

The parameter estimate for _Alpha of 0.4416 is an estimate of the dispersion parameter in the negative binomial distribution. A t test for the hypothesis $H_0 : \alpha = 0$ is provided. It is highly significant, indicating overdispersion ($p < 0.0001$).

The null hypothesis $H_0 : \alpha = 0$ can be also tested against the alternative $\alpha > 0$ by using the likelihood ratio test, as described by Cameron and Trivedi (1998, pp. 45, 77–78). The likelihood ratio test statistic is equal to $-2(\mathcal{L}_P - \mathcal{L}_{NB}) = -2(-1651 + 1561) = 180$, where $\mathcal{L}_P$ and $\mathcal{L}_{NB}$ are the log likelihoods for the Poisson and negative binomial models, respectively. The likelihood ratio test is highly significant, providing strong evidence of overdispersion.

Syntax: COUNTREG Procedure

The following statements are available in the COUNTREG procedure:

PROC COUNTREG < *options* > ;
BAYES < *options* > ;
BOUNDS *bound1* < , *bound2 . . .* > ;
BY *variables* ;
CLASS *variables* ;
DISPMODEL *dependent-variable* ∼ < *dispersion-related-regressors* >< */options* > ;
FREQ *variable* ;
INIT *initvalue1* < , *initvalue2 . . .* > ;
MODEL *dependent* = < *regressors* >< */ options* > ;
NLOPTIONS < *options* > ;
OUTPUT < **OUT=***SAS-data-set* >< *output-options* > ;
PERFORMANCE *options* ;
PRIOR *variables* ∼ *distributions* ;
RESTRICT *restriction1* < , *restriction2 . . .* > ;
TEST *equation1* < , *equation2. . .* > */ test-options* ;
SCORE < **OUT=***SAS-data-set* > < *output-options* > ;
SHOW *options* ;
STORE < **OUT=** >*item-store-name* ;
WEIGHT *variable* < */options* > ;
ZEROMODEL *dependent-variable* ∼ < *zero-inflated-regressors* > < */options* > ;
SPATIALEFFECTS < *model-spatial-effect-regressors* > < */options* > ;
SPATIALDISPEFFECTS < *dispersion-spatial-effect-regressors* > < */options* > ;
SPATIALZEROEFFECTS < *zero-inflation-spatial-effect-regressors* > < */options* > ;
SPATIALID *variable* ;

You can specify only one MODEL statement. The CLASS statement must precede the MODEL statement. If you include the ZEROMODEL statement, it must appear after the MODEL statement. If you specify more than one FREQ or WEIGHT statement, the variable that is specified in the first instance is used.

Functional Summary

Table 12.1 summarizes statements and that you can use in the COUNTREG procedure.

Table 12.1 PROC COUNTREG Functional Summary

Description	Statement	Option
Data Set Options		
Specifies the input data set	COUNTREG	DATA=
Specifies the input spatial weights data set	COUNTREG	WMAT=
Specifies the identification variable for panel data analysis	COUNTREG	GROUPID=
Does not row-normalize the spatial weights matrix	COUNTREG	NONORMALIZE
Writes parameter estimates to an output data set	COUNTREG	OUTEST=
Requests that the procedure produce graphics via the Output Delivery System	COUNTREG	PLOTS=
Writes estimates to an output data set	OUTPUT	OUT=
Declaring the Role of Variables		
Specifies BY-group processing	BY	
Specifies classification variables	CLASS	
Specifies a frequency variable	FREQ	
Specifies a weight variable	WEIGHT	
Specifies a spatial ID variable	SPATIALID	
Item Store Control Options		
Displays the contents of the item store	SHOW	
Stores the model in an item store	STORE	
Restores the model from the item store	COUNTREG	RESTORE=
Printing Control Options		
Prints the correlation matrix of the estimates	MODEL	CORRB
Prints the covariance matrix of the estimates	MODEL	COVB
Prints a summary iteration listing	MODEL	ITPRINT
Suppresses the normal printed output	COUNTREG	NOPRINT
Requests all printing options	MODEL	PRINTALL
Option Process Control Options		
Specifies maximum number of iterations allowed	MODEL	MAXITER=
Selects the iterative minimization method to use	COUNTREG	METHOD=
Sets boundary restrictions on parameters	BOUNDS	
Sets initial values for parameters	INIT	
Sets linear restrictions on parameters	RESTRICT	
Sets the number of threads to use	PERFORMANCE	
Specifies the optimization options	NLOPTIONS	See Chapter 7, "Nonlinear Optimization Methods"

Description	Statement	Option
Model Estimation Options		
Specifies the dispersion variables	DISPMODEL	
Specifies the type of model	COUNTREG	DIST=
Specifies the type of covariance matrix	MODEL	COVEST=
Specifies the type of error components model for panel data	MODEL	ERRORCOMP=
Suppresses the intercept parameter	MODEL	NOINT
Specifies the offset variable	MODEL	OFFSET=
Specifies the parameterization for the Conway-Maxwell-Poisson (CMP) model	MODEL	PARAMETER=
Specifies the zero-inflated offset variable	ZEROMODEL	OFFSET=
Specifies the zero-inflated link function	ZEROMODEL	LINK=
Specifies variable selection	MODEL	SELECT=( )
Specifies variable selection	DISPMODEL	SELECT=( )
Specifies variable selection	ZEROMODEL	SELECT=( )
Specifies the spatial effects to be added to MODEL statement	SPATIALEFFECTS	
Specifies variable selection	SPATIALEFFECTS	SELECT=( )
Specifies the spatial effects for dispersion	SPATIALDISPEFFECTS	
Specifies variable selection	SPATIALDISPEFFECTS	SELECT=( )
Specifies the spatial effects for zero-inflation	SPATIALZEROEFFECTS	
Specifies variable selection	SPATIALZEROEFFECTS	SELECT=( )
Bayesian MCMC Options		
Controls the aggregation of multiple posterior chains	BAYES	AGGREGATION=
Automates the initialization of the MCMC algorithm	BAYES	AUTOMCMC()
Specifies the initial values of the MCMC algorithm	INIT	
Requests evaluation of the marginal likelihood	BAYES	MARGINLIKE
Specifies the maximum number of tuning phases	BAYES	MAXTUNE=
Specifies the minimum number of tuning phases	BAYES	MINTUNE=
Specifies the number of burn-in iterations	BAYES	NBI=
Specifies the number of iterations during the sampling phase	BAYES	NMC=
Specifies the number of threads to use during the sampling phase	BAYES	NTRDS=
Specifies the number of iterations during the tuning phase	BAYES	NTU=
Controls options for constructing the initial proposal covariance matrix	BAYES	PROPCOV=
Specifies the sampling scheme	BAYES	SAMPLING=
Specifies the random number generator seed	BAYES	SEED=
Prints the time required for the MCMC sampling	BAYES	SIMTIME
Controls the thinning of the Markov chain	BAYES	THIN=

Description	Statement	Option
Bayesian Summary Statistics and Convergence Diagnostics		
Displays convergence diagnostics	BAYES	DIAGNOSTICS=
Displays summary statistics of the posterior samples	BAYES	STATISTICS=
Bayesian Prior and Posterior Samples		
Specifies a SAS data set for the posterior samples	BAYES	OUTPOST=
Bayesian Analysis		
Specifies normal prior distribution	PRIOR	NORMAL(MEAN=, VAR=)
Specifies gamma prior distribution	PRIOR	GAMMA(SHAPE=, SCALE=)
Specifies inverse gamma prior distribution	PRIOR	IGAMMA(SHAPE=, SCALE=)
Specifies uniform prior distribution	PRIOR	UNIFORM(MIN=, MAX=)
Specifies beta prior distribution	PRIOR	BETA(SHAPE1=, SHAPE2=, MIN=, MAX=)
Specifies *t* prior distribution	PRIOR	T(LOCATION=, DF=)
Output Control Options		
Includes covariances in the OUTEST= data set	COUNTREG	COVOUT
Outputs the estimates of dispersion for the CMP model	OUTPUT	DISPERSION
Outputs the estimates of $\mathbf{g}_i'\boldsymbol{\delta}$ for the CMP model	OUTPUT	GDELTA=
Outputs the estimates of λ for the CMP model	OUTPUT	LAMBDA=
Outputs the estimates of ν for the CMP model	OUTPUT	NU=
Outputs the estimates of μ for the CMP model	OUTPUT	MU=
Outputs the estimates of mode for the CMP model	OUTPUT	MODE=
Outputs the probability that the response variable will take the current value	OUTPUT	PROB=
Outputs probabilities for particular response values	OUTPUT	PROBCOUNT( )
Outputs the expected value of the response variable	OUTPUT	PRED=
Outputs the estimates of variance for the CMP model	OUTPUT	VARIANCE=
Outputs estimates of $\mathbf{x}_i'\boldsymbol{\beta}$	OUTPUT	XBETA=
Outputs estimates of $\mathbf{z}_i'\boldsymbol{\gamma}$	OUTPUT	ZGAMMA=
Outputs the probability that the response variable will take a zero value as a result of the zero-generating process	OUTPUT	PROBZERO=
Specifies the output data set for scoring	SCORE	OUT=
Outputs the estimates of dispersion for the CMP model	SCORE	DISPERSION
Outputs the estimates of $\mathbf{g}_i'\boldsymbol{\delta}$ for the CMP model	SCORE	GDELTA=
Outputs the estimates of λ for the CMP model	SCORE	LAMBDA=
Outputs the estimates of ν for the CMP model	SCORE	NU=

Description	Statement	Option
Outputs the estimates of μ for the CMP model	SCORE	MU=
Outputs the estimates of mode for the CMP model	SCORE	MODE=
Outputs the probability that the response variable will take the current value	SCORE	PROB=
Outputs probabilities for particular response values	SCORE	PROBCOUNT( )
Outputs expected value of response variable	SCORE	PRED=
Outputs the estimates of variance for the CMP model	SCORE	VARIANCE=
Outputs estimates of $\mathbf{x}_i'\boldsymbol{\beta}$	SCORE	XBETA=
Outputs estimates of $\mathbf{z}_i'\boldsymbol{\gamma}$	SCORE	ZGAMMA=
Outputs the probability that the response variable will take a value of zero as a result of the zero-generating process	SCORE	PROBZERO=

PROC COUNTREG Statement

PROC COUNTREG < *options* > **;**

You can specify the following *options* in the PROC COUNTREG statement.

Data Set Options

DATA=*SAS-data-set*
specifies the input SAS data set. If the DATA= option is not specified, PROC COUNTREG uses the most recently created SAS data set.

GROUPID=*variable*
specifies an identification variable when a panel data model is estimated. The identification variable is used as a cross-sectional ID variable.

NONORMALIZE (Experimental)
does not row-normalize the spatial weights matrix that is specified in the WMAT= option. By default, the spatial weights matrix is required to be row-normalized; that is, the spatial weights matrix has unit row sum. Equivalently, this means that $w(\mathrm{s}_i, \mathrm{s}_j)$ is normalized by multiplying it by $\frac{1}{\sum_{j=1}^{n} w(\mathrm{s}_i, \mathrm{s}_j)}$, where n is the total number of spatial units. If the NONORMALIZE option is specified, spatial weights are used "as is" except for $w(\mathrm{s}_i, \mathrm{s}_i)$, which is always treated as 0. This implies that a spatial weight $w(\mathrm{s}_i, \mathrm{s}_j)$ cannot be missing for $i \neq j$ if the NONOMALIZE option is specified. If the NONOMALIZE option is not specified, missing spatial weights are replaced with 0.

WMAT=*SAS-data-set* (Experimental)
specifies the input SAS data set that contains spatial weights matrix. The spatial weights matrix is often known as the W matrix. The spatial weights $w(\mathrm{s}_i, \mathrm{s}_j)$ for two locations s_i and s_j must satisfy the following: $w(\mathrm{s}_i, \mathrm{s}_j) \geq 0$ and $w(\mathrm{s}_i, \mathrm{s}_i) = 0$, where $i, j = 1, 2, \ldots, n$ and n is the total number of spatial locations. However, it is not necessary that $w(\mathrm{s}_i, \mathrm{s}_j) = w(\mathrm{s}_j, \mathrm{s}_i)$. In addition, any nonzero

$w(s_i, s_i)$ is replaced with 0. For more information about missing spatial weights in W, see the section "NONORMALIZE" on page 591.

For a spatial weights data set that has n spatial units, the number of columns must be $n + 1$ if the SPATIALID statement specifies a spatial ID variable for the purpose of matching observations. For more information, see the section "SPATIALID Statement (Experimental)" on page 615. However, if the SPATIALID statement is not specified, the number of rows and columns in the spatial weights data set must be equal.

Item Store Control Options

RESTORE=*item-store-name*
specifies the source item store for processing. An *item-store-name* consists of a one- or two-level name, as with SAS data sets. As with data sets, an item store is associated by default with the Work library, and any item stores that are created in this library are deleted when the SAS session concludes.

Output Data Set Options

OUTEST=*SAS-data-set*
writes the parameter estimates to the specified output data set.

COVOUT
writes the covariance matrix for the parameter estimates to the OUTEST= data set. This option is valid only if the OUTEST= option is specified.

Printing Options

CORRB
prints the correlation matrix of the parameter estimates. This option can also be specified in the MODEL statement.

COVB
prints the covariance matrix of the parameter estimates. This option can also be specified in the MODEL statement.

NOPRINT
suppresses all printed output.

Estimation Control Options

COVEST=*value*
specifies the type of covariance matrix of the parameter estimates. The quasi-maximum-likelihood estimates are computed using COVEST=QML. By default, COVEST=HESSIAN. The supported covariance types are as follows:

HESSIAN	specifies the covariance from the Hessian matrix.
OP	specifies the covariance from the outer product matrix.
QML	specifies the covariance from the outer product and Hessian matrices.

Plot Control Options

PLOTS<*(global-plot-options)*> **< = ***plot-request* **| (***plot-requests***)>**
requests that the COUNTREG procedure produce statistical graphics via the Output Delivery System, provided that ODS GRAPHICS has been enabled. For general information about ODS Graphics, see Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*). The *global-plot-options* apply to all relevant plots that are generated by the COUNTREG procedure.

You can specify the following *global-plot-options*:

COUNTS(*value1* **<***value2***...>)**
supplies the plots PREDPROB and PREDPROFILE with particular values of the response variable. Each value should be a nonnegative integer. Nonintegers are rounded to the nearest integer. The *value* can also be a list of the form X TO Y BY Z. For example, COUNTS(0 1 2 TO 10 BY 2 15) specifies plotting for counts 0, 1, 2, 4, 6, 8, 10, and 15.

ONLY
suppresses the default plots. Only the plots that are specifically requested are produced.

UNPACKPANEL

UNPACK
displays each graph separately. (By default, some graphs can appear together in a single panel.)

You can specify the following *plot-requests*:

ALL
requests that all plots appropriate for the particular analysis be produced.

AUTOCORR< (LAGS=*n***) >**
displays the autocorrelation function plots of the parameters. This *plot-request* is available only for Bayesian analysis. The optional LAGS= suboption specifies the number (up to lag *n*) of autocorrelations to be plotted in the AUTOCORR plot. If this suboption is not specified, autocorrelations are plotted up to lag 50.

BAYESDIAG
displays the TRACE, AUTOCORR, and DENSITY plots. This *plot-request* is available only for Bayesian analysis.

BAYESSUM
displays the posterior distribution, prior distribution, and maximum likelihood estimates. This *plot-request* is available only for Bayesian analysis.

DENSITY< (FRINGE) >
displays the kernel density plots of the parameters. This *plot-request* is available only for Bayesian analysis. If you specify the FRINGE suboption, a fringe plot is created on the X axis of the kernel density plot.

DISPERSION
produces the overdispersion diagnostic plot.

NONE
suppresses all plots.

PREDPROB
produces the overall predictive probabilities of the specified count levels. You must also specify COUNTS in *global-plot-options*.

PREDPROFILE
produces the predictive probability profiles of specified count levels against model regressors. The regressor on the X axis is varied, whereas all other regressors are fixed at the mean of the observed data set.

PROFILELIKE
produces the profile likelihood functions of the model parameters. The model parameter on the X axis is varied, whereas all other parameters are fixed at their estimated maximum likelihood estimates.

TRACE< (SMOOTH) >
displays the trace plots of the parameters. This *plot-request* is available only for Bayesian analysis. The SMOOTH suboption displays a fitted penalized B-spline curve for each trace plot.

ZEROPROFILE | ZPPRO
produces the probability profiles of zero-inflation process selection and zero count prediction against model regressors. The regressor on the X axis is varied, whereas all other regressors are fixed at the mean of the observed data set.

Optimization Process Control Options

PROC COUNTREG uses the nonlinear optimization (NLO) subsystem to perform nonlinear optimization tasks. All the NLO options are available in the NLOPTIONS statement. For more information, see the "NLOPTIONS Statement" on page 608. In addition, you can specify the following option in the PROC COUNTREG statement:

METHOD=*value*

specifies the iterative minimization method to use. By default, METHOD=NRA.

CONGRA	specifies the conjugate-gradient method.
DBLDOG	specifies the double-dogleg method.
NMSIMP	specifies Nelder-Mead simplex method.
NRA	specifies the Newton-Raphson method.
NRRIDG	specifies the Newton-Raphson ridge method.
QN	specifies the quasi-Newton method.
TR	specifies the trust region method.

BAYES Statement

BAYES < *options* > **;**

The BAYES statement controls the Metropolis sampling scheme that is used to obtain samples from the posterior distribution of the underlying model and data. You can specify the following *options*.

AGGREGATION=WEIGHTED | NOWEIGHTED (Experimental)
specifies how multiple posterior samples should be aggregated.

WEIGHTED implements a weighted resampling scheme for the aggregation of multiple posterior chains. You can use this option when the posterior distribution is characterized by several very distinct posterior modes.

NOWEIGHTED aggregates multiple posterior chains without any adjustment. You can use this option when the posterior distribution is characterized by one or few relatively close posterior modes.

By default, AGGREGATION=NOWEIGHTED. For more information, see the section "Aggregation of Multiple Chains" on page 659.

AUTOMCMC< =(*automcmc-options***) >** (Experimental)
specifies an algorithm for the automated initialization of the MCMC sampling algorithm. For more information, see the section "Automated Initialization of MCMC" on page 660.

ACCURACY=(*accuracy-options***)**
customizes the behavior of the AUTOMCMC algorithm when you are searching for an accurate representation of the posterior distribution. By default, it implements the TARGETSTATS option. You can specify the following *accuracy-options*:

ATTEMPTS=*number*
specifies the maximum number of attempts that are required in order to obtain accurate samples from the posterior distribution. By default, ATTEMPTS=10.

TARGETESS=*number*
requests that the accuracy search be based on the effective sample size (ESS) analysis and specifies the minimum number of effective samples.

TARGETSTATS<=(*targetstats-option***)>**
requests that the accuracy search be based on the analysis of the posterior mean and a posterior quantile of interest. You can customize the behavior of the analysis of the posterior mean by adjusting the HEIDELBERGER suboptions. You can customize the behavior of the analysis of the posterior quantile by adjusting the RAFTERY suboptions. If you specify TARGETSTATS, you can also specify how the Raftery-Lewis test should be interpreted by using the following option:

RLLIMITS=(LB=*number* **UB=***number***)**
specifies a region where the search for the optimal sample size depends directly on the Raftery-Lewis test. By default, RLLIMITS=(LB=10000 UB=300000).

TOL=*value*
specifies the proportion of parameters that are required to be accurate. By default, TOL=0.95.

MAXNMC=*number*
specifies the maximum number of posterior samples that the AUTOMCMC option allows. By default, MAXNMC=700000.

RANDINIT< =(*randinit-options***) >**
specifies random starting points for the MCMC algorithm. The starting points can be sampled around the maximum likelihood estimate and around the prior mean. You can specify the following *randinit-options*:

MULTIPLIER=(*value***)**
specifies the radius of the area where the starting points are sampled. For the starting points that are sampled around the maximum likelihood estimate, the radius equals the standard deviation of the maximum likelihood estimate multiplied by the multiplier value. For the starting points that are sampled around the prior mean, the radius equals the standard deviation of the prior distribution multiplied by the multiplier value. By default, MULTIPLIER=2.

PROPORTION=(*value***)**
specifies the proportion of starting points that are sampled around the maximum likelihood estimate and around the prior mean. By default, PROPORTION=0, which implies that all the initial points are sampled around the maximum likelihood estimate. If you choose to sample starting points around the prior mean, the convergence of the MCMC algorithm could be very slow.

STATIONARITY=(*stationarity-options***)**
customizes the behavior of the AUTOMCMC algorithm when you are trying to sample from the posterior distribution. You can specify the following *stationarity-options*:

ATTEMPTS=*number*
specifies the maximum number of attempts that are required in order to obtain stationary samples from the posterior distribution. By default, ATTEMPTS=10.

TOL=*value*
specifies the proportion of parameters whose samples must be stationary. By default, TOL=0.95.

DIAGNOSTICS=ALL | NONE | *(keyword-list)*
DIAG=ALL | NONE | *(keyword-list)*
controls which diagnostics are produced. All the following diagnostics are produced by using DIAGNOSTICS=ALL. If you do not want any of these diagnostics, specify DIAGNOSTICS=NONE. If you want some but not all of the diagnostics, or if you want to change certain settings of these diagnostics, specify a subset of the following keywords. By default, DIAGNOSTICS=NONE.

AUTOCORR< (LAGS=*numeric-list***) >**
computes the autocorrelations at lags that are specified in the *numeric-list*. Elements in the *numeric-list* are truncated to integers, and repeated values are removed. If you do not specify the LAGS= option, autocorrelations of lags 1, 5, and 10 are computed.

AUTOMCMCSUM
produces a summary table for the AUTOMCMC (automatic MCMC) sampling tool is used.

ESS
computes Carlin's estimate of the effective sample size, the correlation time, and the efficiency of the chain for each parameter.

GEWEKE< (*geweke-options***) >**
computes the Geweke spectral density diagnostics, which are essentially a two-sample *t* test between the first f_1 portion and the last f_2 portion of the chain. The default is $f_1 = 0.1$ and $f_2 = 0.5$, but you can choose other fractions by using the following *geweke-options*:

FRAC1=*value*
specifies the fraction f_1 for the first window.

FRAC2=*value*
specifies the fraction f_2 for the second window.

HEIDELBERGER< (*heidel-options***) >**
computes the Heidelberger-Welch diagnostic for each variable, which consists of a stationarity test of the null hypothesis that the sample values form a stationary process. If the stationarity test is not rejected, a halfwidth test is then performed. Optionally, you can specify one or more of the following *heidel-options*:

SALPHA=*value*
specifies the α level ($0 < \alpha < 1$) for the stationarity test. By default, SALPHA=0.05.

HALPHA=*value*
specifies the α level ($0 < \alpha < 1$) for the halfwidth test. By default, HALPHA=0.1.

EPS=*value*
specifies a positive number ϵ such that if the halfwidth is less than ϵ times the sample mean of the retained iterates, the halfwidth test is passed. By default, EPS=0.05.

MCSE

MCERROR
computes the Monte Carlo standard error for each parameter. The Monte Carlo standard error, which measures the simulation accuracy, is the standard error of the posterior mean estimate and is calculated as the posterior standard deviation divided by the square root of the effective sample size.

RAFTERY< (*raftery-options***) >**
computes the Raftery-Lewis diagnostics, which evaluate the accuracy of the estimated quantile ($\hat{\theta}_Q$ for a given $Q \in (0, 1)$) of a chain. $\hat{\theta}_Q$ can achieve any degree of accuracy when the chain is allowed to run for a long time. The computation is stopped when the estimated probability $\hat{P}_Q = \Pr(\theta \leq \hat{\theta}_Q)$ reaches within $\pm R$ of the value Q with probability S; that is, $\Pr(Q - R \leq \hat{P}_Q \leq Q + R) = S$. The following *raftery-options* enable you to specify Q, R, S, and a precision level ϵ for the test:

ACCURACY | R=*value*
specifies a small positive number as the margin of error for measuring the accuracy of estimation of the quantile. By default, R=0.005.

EPSILON | EPS=*value*
specifies the tolerance level (a small positive number) for the stationary test. By default, EPS=0.001.

PROBABILITY | S=*value*
specifies the probability of attaining the accuracy of the estimation of the quantile. By default, S=0.95.

QUANTILE | Q=*value*
specifies the order (a value between 0 and 1) of the quantile of interest. By default, Q=0.025.

MARGINLIKE< (NSIM=*number***) >** (Experimental)
requests evaluation of the logarithm of the marginal likelihood. Two estimates are produced: the cross-entropy estimate and the harmonic mean. The cross-entropy estimate is based on an importance sampling algorithm for which you can specify the *number* of importance samples in the NSIM= suboption. The default is 10,000. For more information, see the section "Marginal Likelihood" on page 666.

MAXTUNE=*number*
specifies the maximum number of tuning phases. By default, MAXTUNE=24.

MINTUNE=*number*
specifies the minimum number of tuning phases. By default, MINTUNE=2.

NBI=*number*
specifies the number of burn-in iterations before the chains are saved. By default, NBI=1000.

NMC=*number*
specifies the number of iterations after the burn-in. By default, NMC=1000.

NTRDS=*number*

THREADS=*number*
specifies the number of threads to be used. The number of threads cannot exceed the number of computer cores available. Each core samples the number of iterations that is specified by the NMC= option. By default, NTRDS=1.

NTU=*number*
specifies the number of samples for each tuning phase. By default, NTU=500.

OUTPOST=*SAS-data-set*
names the SAS data set to contain the posterior samples. Alternatively, you can create the output data set by specifying an ODS OUTPUT statement as follows:

ODS OUTPUT POSTERIORSAMPLE=< *SAS-data-set*> **;**

PROPCOV=*value*
specifies the method to use in constructing the initial covariance matrix for the Metropolis-Hastings algorithm. The quasi-Newton (PROPCOV=QUANEW) and Nelder-Mead simplex (PROPCOV=NMSIMP) methods find numerically approximated covariance matrices at the optimum of the posterior density function with respect to all continuous parameters. The tuning phase starts at the optimized values; in some problems, this can greatly increase convergence performance. If the approximated covariance matrix is not positive definite, then an identity matrix is used instead.

You can specify the following *values*:

CONGRA
performs a conjugate-gradient optimization.

DBLDOG
performs a version of double-dogleg optimization.

NEWRAP
performs a Newton-Raphson optimization that combines a line-search algorithm with ridging.

NMSIMP
performs a Nelder-Mead simplex optimization.

NRRIDG
performs a Newton-Raphson optimization with ridging.

QUANEW
performs a quasi-Newton optimization.

TRUREG
performs a trust-region optimization.

SAMPLING=MULTIMETROPOLIS | UNIMETROPOLIS
specifies how to sample from the posterior distribution.

MULTIMETROPOLIS implements a Metropolis sampling scheme in a single block that contains all the parameters of the model.

UNIMETROPOLIS implements a Metropolis sampling scheme in multiple blocks, one for each parameter of the model.

By default, SAMPLING=MULTIMETROPOLIS.

SEED=*number*
specifies an integer seed in the range 1 to $2^{31} - 1$ for the random number generator in the simulation. Specifying a seed enables you to reproduce identical Markov chains for the same specification. If you do not specify the SEED= option, or if you specify SEED=0, a random seed is derived from the time of day, which is read from the computer's clock.

SIMTIME
prints the time required for the MCMC sampling.

STATISTICS< (*global-options*) **>=ALL** | **NONE** | *keyword* | *(keyword-list)*

STATS< (*global-options*) **>=ALL** | **NONE** | *keyword* | *(keyword-list)*

controls the number of posterior statistics that are produced. Specifying STATISTICS=ALL is equivalent to specifying STATISTICS=(CORR COV INTERVAL PRIOR SUMMARY). If you do not want any posterior statistics, specify STATISTICS=NONE. By default, STATISTICS=(SUMMARY INTERVAL).

You can specify the following *global-options*:

ALPHA=*numeric-list*

controls the probabilities of the credible intervals. The values in the *numeric-list* must be between 0 and 1. Each ALPHA= value produces a pair of 100(1–ALPHA)% equal-tail and HPD intervals for each parameter. By default, ALPHA=0.05, which yields the 95% credible intervals for each parameter.

PERCENT=*numeric-list*

requests the percentile points of the posterior samples. The values in the *numeric-list* must be between 0 and 100. By default, PERCENT=25, 50, 75, which yields the 25th, 50th, and 75th percentile points, respectively, for each parameter.

You can specify the following *keywords*:

CORR

produces the posterior correlation matrix.

COV

produces the posterior covariance matrix.

INTERVAL

produces equal-tail credible intervals and HPD intervals. The default is to produce the 95% equal-tail credible intervals and 95% HPD intervals, but you can use the ALPHA= *global-option* to request intervals of any probabilities.

NONE

suppresses printing of all summary statistics.

PRIOR

produces a summary table of the prior distributions that are used in the Bayesian analysis.

SUMMARY

produces the means, standard deviations, and percentile points (25th, 50th, and 75th) of the posterior samples. You can use the global PERCENT= *global-option* to request specific percentile points.

THIN=*number*

THINNING=*number*

controls the thinning of the Markov chain. Only one in every k samples is used when THIN=k, and if NBI=n_0 and NMC=n, the number of samples that are kept is

$$\left\lfloor \frac{n_0 + n}{k} \right\rfloor - \left\lfloor \frac{n_0}{k} \right\rfloor$$

where $\lfloor a \rfloor$ represents the integer part of the number a. By default, THIN=1.

BOUNDS Statement

BOUNDS *bound1* < *, bound2 . . .* > **;**

The BOUNDS statement imposes simple boundary constraints on the parameter estimates. BOUNDS statement constraints refer to the parameters that are estimated by the COUNTREG procedure. You can specify any number of BOUNDS statements as follows.

Each *bound* is composed of parameter names, constants, and inequality operators as follows:

item operator item < *operator item operator item . . .* >

Each *item* is a constant, a parameter name, or a list of parameter names. Each *operator* is <, >, <=, or >=.

Parameter names are as shown in the Parameter column of the "Parameter Estimates" table. Please refer to the section "Parameter Naming Conventions for the RESTRICT, TEST, BOUNDS, and INIT Statements" on page 649 for more information on how parameters are named in the BOUNDS statement.

You can use both the BOUNDS statement and the RESTRICT statement to impose boundary constraints; however, the BOUNDS statement provides a simpler syntax for specifying these kinds of constraints. See also the section "RESTRICT Statement" on page 611.

The following BOUNDS statement constrains the estimates of the parameter for `z` to be negative, the parameters for `x1` through `x10` to be between zero and one, and the parameter for `x1` in the zero-inflation model to be less than one:

```
bounds z < 0,
       0 < x1-x10 < 1,
       Inf_x1 < 1;
```

The BOUNDS statement is not supported if a BAYES statement is also specified. In Bayesian analysis, the restrictions on parameters are usually introduced through the prior distribution.

BY Statement

BY *variables* **;**

A BY statement can be used with PROC COUNTREG to obtain separate analyses on observations in groups defined by the BY variables. When a BY statement appears, the input data set should be sorted in the order of the BY variables.

CLASS Statement

CLASS *variable* < *options* > *. . .* < *variable* < *options* > > < */global-options* > **;**

The CLASS statement names the classification variables that are used to group (classify) data in the analysis. Classification variables can be either character or numeric.

Class levels are determined from the formatted values of the CLASS *variables*. Thus, you can use formats to group values into levels. See the discussion of the FORMAT procedure in the *SAS Language Reference: Dictionary* for details. The CLASS statement must precede the MODEL statement.

Most options can be specified either as individual variable *options* or as *global-options*. You can specify *options* for each variable by enclosing the options in parentheses after the variable name. You can also specify *global-options* for the CLASS statement by placing them after a slash (/). *Global-options* are applied to all the variables that are specified in the CLASS statement. If you specify more than one CLASS statement, the *global-options* specified in any one CLASS statement apply to all CLASS statements. However, individual CLASS variable *options* override the *global-options*. You can specify the following values for either an *option* or a *global-option*:

MISSING
treats missing values (., ._, .A, . . . , .Z for numeric variables and blanks for character variables) as valid values for the CLASS variable.

ORDER=DATA | FORMATTED | FREQ | INTERNAL
specifies the sort order for the levels of classification variables. This ordering determines which parameters in the model correspond to each level in the data, so the ORDER= option can be useful when you use the CONTRAST statement. By default, ORDER=FORMATTED. For ORDER=FORMATTED and ORDER=INTERNAL, the sort order is machine-dependent. When ORDER=FORMATTED is in effect for numeric variables for which you have supplied no explicit format, the levels are ordered by their internal values.

The following table shows how PROC COUNTREG interprets values of the ORDER= option.

Value of ORDER=	Levels Sorted By
DATA	Order of appearance in the input data set
FORMATTED	External formatted values, except for numeric variables with no explicit format, which are sorted by their unformatted (internal) values
FREQ	Descending frequency count; levels with more observations come earlier in the order
INTERNAL	Unformatted value

For more information about sort order, see the chapter on the SORT procedure in the *Base SAS Procedures Guide* and the discussion of BY-group processing in *SAS Language Reference: Concepts*.

PARAM=*keyword*
specifies the parameterization method for the classification variable or variables. You can specify any of the *keywords* shown in the following table; by default, PARAM=GLM.

Design matrix columns are created from CLASS variables according to the corresponding coding schemes:

Value of PARAM=	Coding
EFFECT	Effect coding
GLM	Less-than-full-rank reference cell coding (this *keyword* can be used only in a global option)
REFERENCE **REF**	Reference cell coding

All parameterizations are full rank, except for the GLM parameterization. The REF= option in the CLASS statement determines the reference level for effect and reference coding and for their orthogonal parameterizations. It also indirectly determines the reference level for a singular GLM parameterization through the order of levels.

REF=*'level'* | *keyword*
specifies the reference level for PARAM=EFFECT, PARAM=REFERENCE, and their orthogonalizations. When PARAM=GLM, the REF= option specifies a level of the classification variable to be put at the end of the list of levels. This level thus corresponds to the reference level in the usual interpretation of the linear estimates with a singular parameterization.

For an individual variable REF= option (but not for a global REF= option), you can specify the *level* of the variable to use as the reference level. Specify the formatted value of the variable if a format is assigned. For a global or individual variable REF= option, you can use one of the following *keywords*. By default, REF=LAST.

FIRST designates the first-ordered level as reference.

LAST designates the last-ordered level as reference.

DISPMODEL Statement

DISPMODEL *dependent-variable* ∼ *< dispersion-related-regressors ></ option>* **;**

The DISPMODEL statement specifies the *dispersion-related-regressors* that are used to model dispersion. This statement is ignored unless you specify DIST=CMPOISSON in the MODEL statement. The *dependent-variable* in the DISPMODEL statement must be the same as the *dependent-variable* in the MODEL statement.

The *dependent-variable* that appears in the DISPMODEL statement is directly used to model dispersion. Each of the q variables to the right of the tilde (∼) has a parameter to be estimated in the regression. For example, let $\mathbf{g}_i'$ be the ith observation's $1 \times (q+1)$ vector of values of the q dispersion explanatory variables (q_0 is set to 1 for the intercept term). Then the dispersion is a function of $\mathbf{g}_i'\boldsymbol{\delta}$, where $\boldsymbol{\delta}$ is the $(q+1)\times 1$ vector of parameters to be estimated, the dispersion model intercept is δ_0, and the coefficients for the q dispersion covariates are $\delta_1, \ldots, \delta_q$. If you specify DISP=CMPOISSON in the MODEL statement but do not include a DISPMODEL statement, then only the intercept term δ_0 is estimated. The "Parameter Estimates" table in the displayed output shows the estimates for the dispersion intercept and dispersion explanatory variables; they are labeled with the prefix "Disp_". For example, the dispersion intercept is labeled "Disp_Intercept". If you specify Age (a variable in your data set) as a dispersion explanatory variable, then the "Parameter Estimates" table labels the corresponding parameter estimate "Disp_Age". The following statements fit a

Conway-Maxwell-Poisson model by using the regressors SEX, ILLNESS, and INCOME and by using AGE as a dispersion-related regressor:

```
proc countreg data=docvisit;
   model doctorvisits=sex illness income / dist=cmpoisson;
   dispmodel doctorvisits ~ age /;
run;
```

You can specify the following *option* after a slash (/):

SELECT=INFO=<(*selection-options***)>**

SELECTVAR=INFO=<(*selection-options***)>**

requests that the variable selection method be based on an information criterion. For a list of *selection-options*, see the section "Options for Variable Selection Based on an Information Criterion" on page 606. For more information about this type of variable selection, see the section "Variable Selection Using an Information Criterion" on page 637.

FREQ Statement

FREQ *variable* **;**

The FREQ statement specifies a variable whose values represent the frequency of occurrence of each observation. PROC COUNTREG treats each observation as if it appears n times, where n is the value of the FREQ variable for the observation. If the frequency value is not an integer, it is truncated to an integer; if it is less than 1 or missing, the observation is not used in the model fitting. When the FREQ statement is not specified, each observation is assigned a frequency of 1. If you specify more than one FREQ statement, then the first statement is used.

INIT Statement

INIT *initvalue1* < *, initvalue2 . . .* > **;**

The INIT statement sets initial values for parameters in the optimization.

Each *initvalue* is written as a parameter or parameter list, followed by an optional equal sign (=), followed by a number:

parameter < = > *number*

Parameter names are as shown in the Parameter column of the "Parameter Estimates" table. Please refer to the section "Parameter Naming Conventions for the RESTRICT, TEST, BOUNDS, and INIT Statements" on page 649 for more information on how parameters are named in the INIT statement.

By default, initial values are determined by OLS regression. Initial values can be displayed with the ITPRINT option in the PROC statement.

If you also specify the BAYES statement, the INIT statement also initializes the Markov chain Monte Carlo (MCMC) algorithm. In particular, the INIT statement does one of the following:

- initializes the tuning phase (this also includes the PROPCOV= option)
- initializes the sampling phase, if there is no tuning phase

MODEL Statement

MODEL *dependent-variable = <regressors> </ options>* **;**

The MODEL statement specifies the *dependent-variable* and independent covariates (*regressors*) for the regression model. If you specify no *regressors*, PROC COUNTREG fits a model that contains only an intercept. The dependent count variable should take on only nonnegative integer values in the input data set. PROC COUNTREG rounds any positive noninteger count values to the nearest integer and ignores any observations that have a negative count.

You can specify only one MODEL statement. You can specify the following *options* after a slash (/).

DIST=*value*
specifies the type of model to be analyzed. If you specify this option in both the MODEL statement and the PROC COUNTREG statement, then only the value in the MODEL statement is used. You can specify the following *values*:

CMPOISSON | C | CMP specifies a Conway-Maxwell-Poisson regression model.

NEGBIN(P=1) specifies a negative binomial regression model that uses a linear variance function.

NEGBIN(P=2) | NEGBIN specifies a negative binomial regression model that uses a quadratic variance function.

POISSON | P specifies a Poisson regression model.

ZICMPOISSON | ZICMP specifies a zero-inflated Conway-Maxwell-Poisson regression. You must also specify the ZEROMODEL statement when you specify this model type.

ZINEGBIN | *ZINB* specifies a zero-inflated negative binomial regression. You must also specify the ZEROMODEL statement when you specify this model type.

ZIPOISSON | ZIP specifies a zero-inflated Poisson regression. You must also specify the ZEROMODEL statement when you specify this model type.

ERRORCOMP=FIXED | RANDOM
specifies the type of conditional panel model to be analyzed. You can specify the following values:

FIXED specifies a fixed-effect error component regression model.

RANDOM specifies a random-effect error component regression model.

NOINT
suppresses the intercept parameter.

OFFSET=*variable*
specifies a *variable* in the input data set to be used as an offset variable. The offset variable appears as a covariate in the model with its parameter restricted to 1. The offset variable cannot be the response variable, the zero-inflation offset variable (if any), or one of the explanatory variables. The "Model Fit Summary" table gives the name of the data set variable used as the offset variable; it is labeled as "Offset."

PARAMETER=MU | LAMBDA
specifies the parameterization for the Conway-Maxwell-Poisson model. The following parameterizations are supported:

LAMBDA estimates the original Conway-Maxwell-Poisson model (Shmueli et al. 2005).

MU reparameterizes λ as documented by Guikema and Coffelt (2008), where $\mu = \lambda^{1/\nu}$ and the integral part of μ represents the mode, which can be considered a measure of central tendency (mean).

By default, PARAMETER=MU.

Options for Variable Selection Based on an Information Criterion

For the MODEL, ZEROMODEL, DISPMODEL, SPATIALEFFECTS, SPATIALDISPEFFECTS, and SPATIALZEROEFFECTS statements, you can specify the following option after a slash (/) to control the variable selection process:

SELECT=INFO<(*selection-options***)>** (Experimental)
SELECTVAR=INFO<(*selection-options***)>**
requests that the variable selection method be based on an information criterion. For more information, see the section "Variable Selection Using an Information Criterion" on page 637. You can specify one or more of the following *selection-options*:

DIRECTION=FORWARD | BACKWARD
specifies the search algorithm to use in the variable selection method. You can specify the following values:

FORWARD specifies the search algorithm that starts with a base model and adds an additional variable at each step until either the model cannot be improved or one of the criteria for stopping has been met.

BACKWARD specifies the search algorithm that starts with the original model and removes a variable at each step until either the model cannot be improved or one of the criteria for stopping has been met.

By default, DIRECTION=FORWARD.

CRITER=AIC | SBC
specifies the information criterion to use in the variable selection. You can specify the following values:

AIC uses Akaike's information criterion to determine whether the current model is better than the previous model.

SBC uses the Schwarz-Bayesian information criterion to determine whether the current model is better than the previous model.

By default, CRITER=SBC.

LSTOP=*percentage*
specifies the *percentage* of decrease or increase in the AIC or SBC that is required for the algorithm to proceed; *percentage* must be a nonnegative number less than 1. By default, LSTOP=0.

MAXSTEPS=*number*
specifies the maximum *number* of steps to allow in the search algorithm. The default is infinite; that is, the algorithm does not stop until the stopping criterion is satisfied.

RETAIN(*variable1* **<***variable2***...>)**
requests that the variables named within parentheses be retained during the variable selection process.

Options for Penalized Variable Selection

For the MODEL statement, you can specify the following option instead of the SELECT=INFO option:

SELECT=PEN< (*selection-options***) >** (Experimental)
requests the penalized variable selection method. For more information, see the section "Variable Selection Using an Information Criterion" on page 637. You can specify one or more of the following *selection-options*:

GCV
specifies the generalized cross-validation (GCV) approach. For more information, see the section "The GCV Approach" on page 640.

GCV1
specifies the GCV1 approach. For more information, see the section "The GCV1 Approach" on page 641.

GCVLENGTH=*value*
specifies the number of different values to use for the generalized cross validation (GCV) tuning parameter. The value corresponds to λ

LAMBDA=*value*
specifies the *value* of lambda to use as the shrinkage parameter. When LAMBDA=0, no shrinkage is performed. As the value of LAMBDA increases, the coefficients are shrunk ever more strongly. By default, LAMBDA=0.

LLASTEPS=*value*
specifies the maximum number of iterations in the algorithm of local linear approximations. By default, LLASTEPS=5.

When SELECT=PEN, GCV1 is the default.

Printing Options

CORRB
prints the correlation matrix of the parameter estimates. The CORRB option can also be specified in the PROC COUNTREG statement.

COVB
prints the covariance matrix of the parameter estimates. The COVB can also be specified in the PROC COUNTREG statement.

ITPRINT
prints the objective function and parameter estimates at each iteration. The objective function is the negative log-likelihood function. The ITPRINT option can also be specified in the PROC COUNTREG statement.

PRINTALL
requests all printing options. The PRINTALL option can also be specified in the PROC COUNTREG statement.

NLOPTIONS Statement

NLOPTIONS < *options* > **;**

The NLOPTIONS statement provides the options to control the nonlinear optimization (NLO) subsystem to perform nonlinear optimization tasks. For a list of all the options of the NLOPTIONS statement, see Chapter 7, "Nonlinear Optimization Methods."

OUTPUT Statement

OUTPUT < **OUT=***SAS-data-set* > < *output-options* > **;**

The OUTPUT statement creates a new SAS data set that contains all the variables in the input data set and, optionally, the estimates of $\mathbf{x}_i'\boldsymbol{\beta}$, the expected value of the response variable, and the probability of the response variable taking on the current value or other values that you specify. In a zero-inflated model, you can additionally request that the output data set contain the estimates of $\mathbf{z}_i'\boldsymbol{\gamma}$ and the probability that the response is zero as a result of the zero-generating process. For the Conway-Maxwell-Poisson model, the estimates of $\mathbf{g}_i'\boldsymbol{\delta}$, λ, ν, μ, mode, variance, and dispersion are also available. Except for the probability of the current value, these statistics can be computed for all observations in which the regressors are not missing, even if the response is missing. By adding observations that have missing response values to the input data set, you can compute these statistics for new observations or for settings of the regressors that are not present in the data without affecting the model fit.

You can specify only one OUTPUT statement. You can specify the following *output-options*:

DISPERSION=*name*
names the variable that contains the value of dispersion for the Conway-Maxwell-Poisson distribution.

GDELTA=*name*
names the variable that contains estimates of $\mathbf{g}_i'\boldsymbol{\delta}$ for the Conway-Maxwell-Poisson distribution.

LAMBDA=*name*
names the variable that contains the estimate of λ for the Conway-Maxwell-Poisson distribution.

MODE=*name*
names the variable that contains the integral part of μ (mode) for the Conway-Maxwell-Poisson distribution.

MU=*name*
names the variable that contains the estimate of μ for the Conway-Maxwell-Poisson distribution.

NU=*name*
names the variable that contains the estimate of ν for the Conway-Maxwell-Poisson distribution.

OUT=*SAS-data-set*
names the output data set.

PRED=*name*

MEAN=*name*
names the variable that contains the predicted value of the response variable.

PROB=*name*
names the variable that contains the probability of the response variable taking the current value, $\Pr(Y = y_i)$.

PROBCOUNT(*value1* **<***value2***...>)**
outputs the probability that the response variable will take particular values. Each *value* should be a nonnegative integer. If you specify a noninteger, it is rounded to the nearest integer. The *value* can also be a list of the form X TO Y BY Z. For example, PROBCOUNT(0 1 2 TO 10 BY 2 15) requests predicted probabilities for counts 0, 1, 2, 4, 5, 6, 8, 10, and 15. This option is not available for the fixed-effects and random-effects panel models.

PROBZERO=*name*
names the variable that contains the value of φ_i, the probability of the response variable taking on the value of zero as a result of the zero-generating process. It is written to the output file only if the model is zero-inflated. This is not the overall probability of a zero response; that is provided by the PROBCOUNT(0) option.

VARIANCE=*name*
names the variable that contains the estimate of variance.

XBETA=*name*
names the variable that contains estimates of $\mathbf{x}_i'\boldsymbol{\beta}$.

ZGAMMA=*name*
names the variable that contains estimates of $\mathbf{z}_i'\boldsymbol{\gamma}$.

PERFORMANCE Statement

PERFORMANCE < *performance-options* > **;**

The PERFORMANCE statement controls the number of threads that are used in the optimization phase. You can also specify that multithreading not be used in the optimization phase by using the NOTHREADS option.

You can specify only one PERFORMANCE statement. The PERFORMANCE statement supports the following *performance-options*:

NTHREADS=*number*
specifies the number of threads to be used during optimization of the model.

NOTHREADS
specifies that no threads should be used during optimization of the model.

DETAILS
specifies that a timing table be included in the output.

If you use both the NTHREADS= and NOTHREADS options, then the NTHREADS= option is ignored. If you use a PERFORMANCE statement, then it overrides any global threading settings that might have been set using the CPUCOUNT=, THREADS, or NOTHREADS system option.

PRIOR Statement

PRIOR _REGRESSORS | *parameter-list* ∼ *distribution* **;**

The PRIOR statement specifies the prior distribution of the model parameters. You must specify a single parameter or a list of parameters, a tilde (∼), and then a distribution with its parameters. Multiple PRIOR statements are allowed.

You can specify the following *distributions*:

BETA(SHAPE1=*a*, SHAPE2=*b*, MIN=*m*, MAX=*M*)
specifies a beta distribution that has the parameters SHAPE1 and SHAPE2 and is defined between MIN and MAX.

GAMMA(SHAPE=*a*, SCALE=*b*)
specifies a gamma distribution that has the parameters SHAPE and SCALE.

IGAMMA(SHAPE=*a*, SCALE=*b*)
specifies an inverse gamma distribution that has the parameters SHAPE and SCALE.

NORMAL(MEAN=μ, VAR=σ^2)
specifies a normal distribution that has the parameters MEAN and VAR.

T(LOCATION=μ, DF=ν)
specifies a noncentral *t* distribution that has DF degrees of freedom and a location parameter equal to LOCATION.

UNIFORM(MIN=*m*, MAX=*M*)
specifies a uniform distribution that is defined between MIN and MAX.

For more information about how to specify *distributions*, see the section "Standard Distributions" on page 667.

You can specify the special keyword _REGRESSORS to select all the parameters that are used in the linear regression component of the model.

RESTRICT Statement

RESTRICT *restriction1* < *, restriction2 . . .* > **;**

The RESTRICT statement imposes linear restrictions on the parameter estimates. You can specify any number of RESTRICT statements.

Each *restriction* is written as an expression, followed by an equality operator (=) or an inequality operator (<, >, <=, >=), followed by a second expression:

expression operator expression

The *operator* can be =, <, >, <=, or >=.

Restriction expressions can be composed of parameter names, constants, and the operators times ($*$), plus ($+$), and minus ($-$). The restriction expressions must be a linear function of the parameters. Please refer to the section "Parameter Naming Conventions for the RESTRICT, TEST, BOUNDS, and INIT Statements" on page 649 for more information on how parameters are named in the RESTRICT statement.

Lagrange multipliers are reported in the "Parameter Estimates" table for all the active linear constraints. They are identified with the names Restrict1, Restrict2, and so on. The probabilities of these Lagrange multipliers are computed using a beta distribution (LaMotte 1994). Nonactive (nonbinding) restrictions have no effect on the estimation results and are not noted in the output.

The following RESTRICT statement constrains the negative binomial dispersion parameter α to 1, which restricts the conditional variance to be $\mu + \mu^2$:

```
restrict _Alpha = 1;
```

The RESTRICT statement is not supported if you also specify a BAYES statement. In Bayesian analysis, the restrictions on parameters are usually introduced through the prior distribution.

SCORE Statement

SCORE < **OUT=***SAS-data-set* > < *output-options* > **;**

The SCORE statement enables you to compute predicted values and other statistics for a SAS data set. As with the OUTPUT statement, the new data set that is created contains all the variables in the input data set and, optionally, the estimates of $\mathbf{x}_i'\boldsymbol{\beta}$, the expected value of the response variable, and the probability that the response variable will take the current value or other values that you specify. In a zero-inflated model, you can additionally request that the output data set contain the estimates of $\mathbf{z}_i'\boldsymbol{\gamma}$ and the probability that the response is zero as a result of the zero-generating process. For the Conway-Maxwell-Poisson model, the

estimates of $\mathbf{g}_i'\boldsymbol{\delta}$, λ, ν, μ, mode, variance, and dispersion are also available. Except for the probability of the current value, these statistics can be computed for all observations in which the regressors are not missing, even if the response is missing.

The following statements fit a Poisson model by using the DocVisit data set. Additional observations in the additionalPatients data set are used to compute expected values by using the SCORE statement. The data in the additionalPatients data set are not used during the fitting stage and are used only for scoring.

You score a data set in two separate steps. In the first step, you fit the model and use the STORE statement to preserve it in the DocVisitPoisson item store, as shown in the following statements:

```
proc countreg data=docvisit;
   model doctorvisits=sex illness income / dist=poisson;
   store docvisitPoisson;
run;
```

In the second step, you retrieve the content of the DocVisitPoisson item store and use it to calculate expected values by using the SCORE statement for the additionalPatients data set as follows:

```
proc countreg restore=docvisitPoisson data=additionalPatients;
score out=outScores mean=meanPoisson probability=prob;
run;
```

By retrieving the model from the item store and using it in a postprocessing step, you can separate the fitting and scoring stages and use data for scoring that might not be available at the time when the model was fitted.

You can specify only one SCORE statement. You can specify the following *output-options*:

DISPERSION=*name*
names the variable that contains the value of dispersion for the Conway-Maxwell-Poisson distribution.

GDELTA=*name*
names the variable that contains estimates of $\mathbf{g}_i'\boldsymbol{\delta}$ for the Conway-Maxwell-Poisson distribution.

LAMBDA=*name*
names the variable that contains the estimate of λ for the Conway-Maxwell-Poisson distribution.

MODE=*name*
names the variable that contains the integral part of μ (mode) for the Conway-Maxwell-Poisson distribution.

MU=*name*
names the variable that contains the estimate of μ for the Conway-Maxwell-Poisson distribution.

NU=*name*
names the variable that contains the estimate of ν for the Conway-Maxwell-Poisson distribution.

OUT=*SAS-data-set*
names the output data set.

PRED=*name*

MEAN=*name*
names the variable that contains the predicted value of the response variable.

PROB=*name*
names the variable that contains the probability that the response variable will take the current value, $\Pr(Y = y_i)$.

PROBCOUNT(*value1* **<***value2***...>)**
outputs the probability that the response variable will take particular values. Each value should be a nonnegative integer. Nonintegers are rounded to the nearest integer. The *value* can also be a list of the form X TO Y BY Z. For example, PROBCOUNT(0 1 2 TO 10 BY 2 15) requests predicted probabilities for counts 0, 1, 2, 4, 5, 6, 8, 10, and 15. This option is not available for the fixed-effects and random-effects panel models.

PROBZERO=*name*
names the variable that contains the value of φ_i, the probability of the response variable taking on the value of zero as a result of the zero-generating process. It is written to the output file only if the model is zero-inflated. This is not the overall probability of a zero response; that is provided by the PROBCOUNT(0) option.

VARIANCE=*name*
names the variable that contains the estimate of variance for the Conway-Maxwell-Poisson distribution.

XBETA=*name*
names the variable that contains estimates of $\mathbf{x}_i'\boldsymbol{\beta}$.

ZGAMMA=*name*
names the variable that contains estimates of $\mathbf{z}_i'\boldsymbol{\gamma}$.

SHOW Statement

SHOW *options* **;**

The SHOW statement displays the contents of the item store. You can use the SHOW statement to verify the contents of the item store before proceeding with the analysis.

Table 12.2 summarizes the *options* available in the SHOW statement.

Table 12.2 SHOW Statement Options

Option	Description
ALL	Displays all applicable contents
CLASSLEVELS	Displays the "Class Level Information" table
CORRELATION	Produces the correlation matrix of the parameter estimates
COVARIANCE	Produces the covariance matrix of the parameter estimates
EFFECTS	Displays information about the constructed effects
FITSTATS	Displays the fit statistics
PARAMETERS	Displays the parameter estimates
PROGRAM	Displays the SAS program that generates the item store

You can specify the following *options* after the SHOW statement:

ALL | _ALL_
displays all applicable contents.

CLASSLEVELS | CLASS
displays the "Class Level Information" table. This table is produced by the COUNTREG procedure by default if the model contains effects that depend on classification variables.

CORRELATION | CORR | CORRB
produces the correlation matrix of the parameter estimates.

COVARIANCE | COV | COVB
produces the covariance matrix of the parameter estimates.

EFFECTS
displays information about the effects in the model.

FITSTATS | FIT | FITSUMMARY
displays the fit statistics from the item store.

PARAMETERS
PARMS
displays the parameter estimates table from the item store.

PROGRAM
PROG
displays the SAS program that generates the item store, provided that this was stored at store generation time. The program does not include comments, titles, or some other global statements.

SPATIALDISPEFFECTS Statement (Experimental)

SPATIALDISPEFFECTS < *dispersion-spatial-effect-regressors* > < */options* > **;**

The SPATIALDISPEFFECTS statement adds the spatially weighted *dispersion-spatial-effect-regressors* to regressors that are specified in the DISPMODEL statement. For example, if you specify q variables $z_1, \ldots, z_q$ in the SPATIALDISPEFFECTS statement, then each of q spatially weighted variables $\mathbf{W}z_1, \ldots, \mathbf{W}z_q$ has a parameter to be estimated in the regression. Here, $\mathbf{W}z_1, \ldots, \mathbf{W}z_q$ denotes the matrix and vector product between $\mathbf{W}$ and a column vector whose entries are the values of $z_1, \ldots, z_q$, respectively. The spatial weights matrix $\mathbf{W}$ comes from the data set that is specified in the WMAT= option in the PROC COUNTREG statement.

The "Parameter Estimates" table in the displayed output shows the estimates for spatially weighted explanatory variables; they are labeled with the prefix "Dsp_W_". For example, if you specify z (a variable in your data set) as a spatial effect explanatory variable, then the "Parameter Estimates" table labels the corresponding parameter estimate "Dsp_W_z".

You can specify the following *option* after a slash (/):

SELECT=INFO=<(*selection-options***)>**

SELECTVAR=INFO=<(*selection-options***)>**

requests that the variable selection method be based on an information criterion. For a list of *selection-options*, see the section "Options for Variable Selection Based on an Information Criterion" on page 606. For more information about this type of variable selection, see the section "Variable Selection Using an Information Criterion" on page 637.

SPATIALEFFECTS Statement (Experimental)

SPATIALEFFECTS < *model-spatial-effect-regressors* > < */options* > **;**

The SPATIALEFFECTS statement adds the spatially weighted *model-spatial-effect-regressors* to regressors that are specified in the MODEL statement. For example, if you specify q variables $z_1, \ldots, z_q$ in the SPATIALEFFECTS statement, then each of q spatially weighted variables $\mathbf{W}z_1, \ldots, \mathbf{W}z_q$ has a parameter to be estimated in the regression. Here, $\mathbf{W}z_1, \ldots, \mathbf{W}z_q$ denotes the matrix and vector product between $\mathbf{W}$ and a column vector whose entries are the values of $z_1, \ldots, z_q$, respectively. The spatial weights matrix $\mathbf{W}$ comes from the data set that is specified in the WMAT= option in the PROC COUNTREG statement.

The "Parameter Estimates" table in the displayed output shows the estimates for spatially weighted *model-spatial-effect-regressors*; they are labeled with the prefix "W_". For example, if you specify z (a variable in your data set) as a spatial effect explanatory variable, then the "Parameter Estimates" table labels the corresponding parameter estimate "W_z".

You can specify the following *option* after a slash (/):

SELECT=INFO=<(*selection-options***)>**

SELECTVAR=INFO=<(*selection-options***)>**

requests that the variable selection method be based on an information criterion. For a list of *selection-options*, see the section "Options for Variable Selection Based on an Information Criterion" on page 606. For more information about this type of variable selection, see the section "Variable Selection Using an Information Criterion" on page 637.

SPATIALID Statement (Experimental)

SPATIALID *variable* **;**

For a spatial lag of X model, the SPATIALID statement specifies a variable that identifies a spatial unit for each observation in the two data sets that are provided by the DATA= and WMAT= options in the PROC COUNTREG statement. The variable also identifies the rows and columns of the WMAT= data set. The values of the spatial ID variable cannot be missing in either the DATA= data set or the WMAT= data set. When there are multiple SPATIALID statements, the first SPATIALID statement takes precedence over others that follow. In such a circumstance, the first SPATIALID statement applies to all spatial lag of X models.

The variable in the SPATIALID statement can be either numeric or character. However, the type of spatial ID variable in both the primary data set (specified in the DATA= option) and the spatial weights data set (specified in the WMAT= option) must be the same. When the spatial ID variable is numeric, its value needs to be an integer. If you specify a number that is not an integer, PROC COUNTREG uses the integer part of

that number for matching. When the variable is numeric, the first letter of column names in the WMAT= data set (which specifies a spatial unit) is discarded because a valid SAS variable name must start with a letter or an underscore. When a numeric column name (such as, 11) is in the WMAT= data set, the IMPORT procedure (in Base SAS) appends an underscore to the column name in order to make it a valid name (for example, 11 becomes _11).

SPATIALZEROEFFECTS Statement (Experimental)

SPATIALZEROEFFECTS < *zero-inflation-spatial-effect-regressors* > < */options* > **;**

The SPATIALZEROEFFECTS statement adds the spatially weighted *zero-inflation-spatial-effect-regressors* to regressors that are specified in the ZEROMODEL statement. For example, if you specify q variables $z_1, \ldots, z_q$ in the SPATIALZEROEFFECTS statement, then each of q spatially weighted variables $\mathbf{W}z_1, \ldots, \mathbf{W}z_q$ has a parameter to be estimated in the regression. Here, $\mathbf{W}z_1, \ldots, \mathbf{W}z_q$ denotes the matrix and vector product between $\mathbf{W}$ and a column vector whose entries are the values of $z_1, \ldots, z_q$, respectively. The spatial weights matrix $\mathbf{W}$ comes from the data set that is specified in the WMAT= option in the PROC COUNTREG statement.

The "Parameter Estimates" table in the displayed output shows the estimates for spatially weighted explanatory variables; they are labeled with the prefix "Inf_W_". For example, if you specify z (a variable in your data set) as a spatial effect explanatory variable, then the "Parameter Estimates" table labels the corresponding parameter estimate "Inf_W_z".

You can specify the following *option* after a slash (/):

SELECT=INFO=< (*selection-options***) >**

SELECTVAR=INFO=< (*selection-options***) >**

requests that the variable selection method be based on an information criterion. For a list of *selection-options*, see the section "Options for Variable Selection Based on an Information Criterion" on page 606. For more information about this type of variable selection, see the section "Variable Selection Using an Information Criterion" on page 637.

STORE Statement

STORE < **OUT=** >*item-store-name* **;**

The STORE statement saves the contents of the analysis to an item store in a binary format that cannot be modified. You can restore the stored information by specifying the RESTORE= option in the PROC COUNTREG statement and use it in postprocessing analysis.

TEST Statement

<label:> **TEST** *<'string'> equation1 < , equation2. . . > / test-options* **;**

The TEST statement performs Wald, Lagrange multiplier, and likelihood ratio tests of linear hypotheses about the regression parameters that are specified in the preceding MODEL statement.

You can add a label (which is printed in the output) to a TEST statement in two ways: add an unquoted *label* followed by a colon before the TEST keyword, or add a quoted *string* after the TEST keyword. The unquoted *label* cannot contain any spaces. If you include both an unquoted *label* and a quoted *string*, PROC COUNTREG uses the unquoted *label*. If you specify neither an unquoted *label* nor a quoted *string*, PROC COUNTREG automatically labels the tests.

Each *equation* specifies a linear hypothesis to be tested and consists of regression parameter names and relational operators. The regression parameter names are as shown in the Parameter column of the "Parameter Estimates" table. For more information about how parameters are named in the TEST statement, see the section "Parameter Naming Conventions for the RESTRICT, TEST, BOUNDS, and INIT Statements" on page 649. Only linear equality restrictions and tests are permitted in PROC COUNTREG. Test *equations* can consist only of algebraic operations that involve the addition symbol (+), subtraction symbol (-), and multiplication symbol (*).

All hypotheses in one TEST statement are tested jointly.

You can specify the following *test-options* after a slash (/):

ALL
requests Wald, Lagrange multiplier, and likelihood ratio tests.

LM
requests the Lagrange multiplier test.

LR
requests the likelihood ratio test.

WALD
requests the Wald test.

By default, the Wald test is performed.

The following statements illustrate the use of the TEST statement:

```
proc countreg;
   model y = x1 x2 x3;
   test x1 = 0, x2 * .5 + 2 * x3 = 0;
   test _int: test intercept = 0, x3 = 0;
run;
```

The first test investigates the joint hypothesis that

$$\beta_1 = 0$$

and

$$0.5\beta_2 + 2\beta_3 = 0$$

You cannot specify both the TEST statement and the BAYES statement.

WEIGHT Statement

WEIGHT *variable* < / *option* > **;**

The WEIGHT statement specifies a variable to supply weighting values to use for each observation in estimating parameters. The log likelihood for each observation is multiplied by the corresponding weight variable value.

If the weight of an observation is nonpositive, that observation is not used in the estimation.

You can specify the following *option* after a slash (/):

NONORMALIZE
does not normalize the weights. By default, the weights are normalized so that they add up to the actual sample size. Weights w_i are normalized by multiplying them by $\frac{n}{\sum_{i=1}^{n} w_i}$, where n is the sample size. If the weights are required to be used "as is," then specify the NONORMALIZE option.

ZEROMODEL Statement

ZEROMODEL *dependent variable* ~ < *zero-inflated regressors* > < /*options* > **;**

The ZEROMODEL statement is required if you specify either ZIP or ZINB in the DIST= option in the MODEL statement. If ZIP or ZINB is specified, then the ZEROMODEL statement must follow immediately after the MODEL statement. The dependent variable in the ZEROMODEL statement must be the same as the dependent variable in the MODEL statement.

The zero-inflated (ZI) regressors appear in the equation that determines the probability (φ_i) of a zero count. Each of these q variables has a parameter to be estimated in the regression. For example, let $\mathbf{z}_i'$ be the ith observation's $1 \times (q+1)$ vector of values of the q ZI explanatory variables (w_0 is set to 1 for the intercept term). Then φ_i is a function of $\mathbf{z}_i'\boldsymbol{\gamma}$, where $\boldsymbol{\gamma}$ is the $(q+1) \times 1$ vector of parameters to be estimated. (The ZI intercept is γ_0; the coefficients for the q ZI covariates are $\gamma_1, \ldots, \gamma_q$.) If this option is omitted, then only the intercept term γ_0 is estimated. The "Parameter Estimates" table in the displayed output gives the estimates for the ZI intercept and ZI explanatory variables; they are labeled with the prefix "Inf_". For example, the ZI intercept is labeled "Inf_intercept". If you specify Age (a variable in your data set) as a ZI explanatory variable, then the "Parameter Estimates" table labels the corresponding parameter estimate "Inf_Age".

You can specify the following options after a slash (/):

LINK=*value*
specifies the distribution function to use to compute probability of zeros. The following distribution functions are supported:

LOGISTIC	specifies the logistic distribution.
NORMAL	specifies the standard normal distribution.

If this option is omitted, then the default ZI link function is logistic.

OFFSET=*variable*

specifies a variable in the input data set to be used as a zero-inflated (ZI) offset variable. The ZI offset variable is included as a term, with its coefficient restricted to 1, in the equation that determines the probability (φ_i) of a zero count. The ZI offset variable cannot be the response variable, the offset variable (if any), or one of the explanatory variables. The name of the data set variable that is used as the ZI offset variable is displayed in the "Model Fit Summary" output, where it is labeled as "Inf_offset".

SELECT=INFO<(*option***)>**

SELECTVAR=INFO<(*option***)>**

requests that the variable selection method be based on an information criterion. For a list of *selection-options*, see the section "Options for Variable Selection Based on an Information Criterion" on page 606. For more information about this type of variable selection, see the section "Variable Selection Using an Information Criterion" on page 637.

Details: COUNTREG Procedure

Specification of Regressors

Each term in a model, called a *regressor*, is a variable or combination of variables. Regressors are specified in a special notation that uses variable names and operators. There are two kinds of variables: *classification* (*CLASS*) *variables* and *continuous variables*. There are two primary operators: *crossing* and *nesting*. A third operator, the *bar operator*, is used to simplify effect specification.

In the SAS System, *classification* (*CLASS*) *variables* are declared in the CLASS statement. (They can also be called *categorical*, *qualitative*, *discrete*, or *nominal variables*.) Classification variables can be either *numeric* or *character*. The values of a classification variable are called *levels*. For example, the classification variable Sex has the levels "male" and "female."

In a model, an independent variable that is not declared in the CLASS statement is assumed to be continuous. *Continuous variables*, which must be numeric, are used for covariates. For example, the heights and weights of subjects are continuous variables. A response variable is a *discrete count variable* and must also be numeric.

Types of Regressors

Seven different types of regressors are used in the COUNTREG procedure. In the following list, assume that A, B, C, D, and E are CLASS variables and that X1 and X2 are continuous variables:

- Regressors are specified by writing continuous variables by themselves: X1 X2.
- Polynomial regressors are specified by joining (crossing) two or more continuous variables with asterisks: X1*X1 X1*X2.
- Dummy regressors are specified by writing CLASS variables by themselves: A B C.
- Dummy interactions are specified by joining classification variables with asterisks: A*B B*C A*B*C.

- Nested regressors are specified by following a dummy variable or dummy interaction with a classification variable or list of classification variables enclosed in parentheses. The dummy variable or dummy interaction is nested within the regressor that is listed in parentheses: B(A) C(B*A) D*E(C*B*A). In this example, B(A) is read "B nested within A."
- Continuous-by-class regressors are written by joining continuous variables and classification variables with asterisks: X1*A.
- Continuous-nesting-class regressors consist of continuous variables followed by a classification variable interaction enclosed in parentheses: X1(A) X1*X2(A*B).

One example of the general form of an effect that involves several variables is

X1*X2*A*B*C(D*E)

This example contains an interaction between continuous terms and classification terms that are nested within more than one classification variable. The continuous list comes first, followed by the dummy list, followed by the nesting list in parentheses. Note that asterisks can appear within the nested list but not immediately before the left parenthesis.

The MODEL statement and several other statements use these effects. Some examples of MODEL statements that use various kinds of effects are shown in the following table, where a, b, and c represent classification variables. The variables x and z are continuous.

Specification	Type of Model
`model y=x;`	Simple regression
`model y=x z;`	Multiple regression
`model y=x x*x;`	Polynomial regression
`model y=a;`	Regression with one classification variable
`model y=a b c;`	Regression with multiple classification variables
`model y=a b a*b;`	Regression with classification variables and their interactions
`model y=a b(a) c(b a);`	Regression with classification variables and their interactions
`model y=a x;`	Regression with both continuous and classification variables
`model y=a x(a);`	Separate-slopes regression
`model y=a x x*a;`	Homogeneity-of-slopes regression

The Bar Operator

You can shorten the specification of a large factorial model by using the bar operator. For example, two ways of writing the model for a full three-way factorial model follow:

```
model Y = A B C   A*B A*C B*C   A*B*C;

model Y = A|B|C;
```

When the bar (|) is used, the right and left sides become effects, and the cross of them becomes an effect. Multiple bars are permitted. The expressions are expanded from left to right, using rules 2–4 given in Searle (1971, p. 390).

- Multiple bars are evaluated from left to right. For instance, A|B|C is evaluated as follows:

$$\begin{aligned} \text{A | B | C} &\rightarrow \{\text{ A | B }\}\text{ | C} \\ &\rightarrow \{\text{ A B A*B }\}\text{ | C} \\ &\rightarrow \text{A B A*B C A*C B*C A*B*C} \end{aligned}$$

- Crossed and nested groups of variables are combined. For example, A(B) | C(D) generates A*C(B D), among other terms.
- Duplicate variables are removed. For example, A(C) | B(C) generates A*B(C C), among other terms, and the extra C is removed.
- Effects are discarded if a variable occurs on both the crossed and nested parts of an effect. For instance, A(B) | B(D E) generates A*B(B D E), but this effect is discarded immediately.

You can also specify the maximum number of variables involved in any effect that results from bar evaluation by specifying that maximum number, preceded by an @ sign, at the end of the bar effect. For example, the specification A | B | C@2 would result in only those effects that contain two or fewer variables: in this case, A B A*B C A*C and B*C.

More examples of using the | and @ operators follow:

A | C(B)	is equivalent to	A C(B) A*C(B)
A(B) | C(B)	is equivalent to	A(B) C(B) A*C(B)
A(B) | B(D E)	is equivalent to	A(B) B(D E)
A | B(A) | C	is equivalent to	A B(A) C A*C B*C(A)
A | B(A) | C@2	is equivalent to	A B(A) C A*C
A | B | C | D@2	is equivalent to	A B A*B C A*C B*C D A*D B*D C*D
A*B(C*D)	is equivalent to	A*B(C D)

Missing Values

Any observation in the input data set that has a missing value for one or more of the regressors is ignored by PROC COUNTREG and not used in the model fit. PROC COUNTREG rounds any positive noninteger count values to the nearest integer. PROC COUNTREG ignores any observations that have a negative count, a zero or negative weight, or a frequency less than 1.

If there are observations in the input data set that have missing response values but with nonmissing regressors, PROC COUNTREG can compute several statistics and store them in an output data set by using the OUTPUT statement. For example, you can request that the output data set contain the estimates of $\mathbf{x}_i'\boldsymbol{\beta}$, the expected value of the response variable, and the probability that the response variable will take values that you specify.

In a zero-inflated model, you can additionally request that the output data set contain the estimates of $\mathbf{z}_i'\boldsymbol{\gamma}$, and the probability that the response is zero as a result of the zero-generating process. The presence of such observations (with missing response values) does not affect the model fit.

Poisson Regression

The most widely used model for count data analysis is Poisson regression. This assumes that y_i, given the vector of covariates $\mathbf{x}_i$, is independently Poisson-distributed with

$$P(Y_i = y_i|\mathbf{x}_i) = \frac{e^{-\mu_i}\mu_i^{y_i}}{y_i!}, \quad y_i = 0, 1, 2, \ldots$$

and the mean parameter (that is, the mean number of events per period) is given by

$$\mu_i = \exp(\mathbf{x}_i'\boldsymbol{\beta})$$

where $\boldsymbol{\beta}$ is a $(k + 1) \times 1$ parameter vector. (The intercept is β_0; the coefficients for the k regressors are $\beta_1, \ldots, \beta_k$.) Taking the exponential of $\mathbf{x}_i'\boldsymbol{\beta}$ ensures that the mean parameter μ_i is nonnegative. It can be shown that the conditional mean is given by

$$E(y_i|\mathbf{x}_i) = \mu_i = \exp(\mathbf{x}_i'\boldsymbol{\beta})$$

The name *log-linear model* is also used for the Poisson regression model because the logarithm of the conditional mean is linear in the parameters:

$$\ln[E(y_i|\mathbf{x}_i)] = \ln(\mu_i) = \mathbf{x}_i'\boldsymbol{\beta}$$

Note that the conditional variance of the count random variable is equal to the conditional mean in the Poisson regression model:

$$V(y_i|\mathbf{x}_i) = E(y_i|\mathbf{x}_i) = \mu_i$$

The equality of the conditional mean and variance of y_i is known as *equidispersion*.

The marginal effect of a regressor is given by

$$\frac{\partial E(y_i|\mathbf{x}_i)}{\partial x_{ji}} = \exp(\mathbf{x}_i'\boldsymbol{\beta})\boldsymbol{\beta}_j = E(y_i|\mathbf{x}_i)\beta_j$$

Thus, a one-unit change in the jth regressor leads to a *proportional* change in the conditional mean $E(y_i|\mathbf{x}_i)$ of β_j.

The standard estimator for the Poisson model is the maximum likelihood estimator (MLE). Because the observations are independent, the log-likelihood function is written as

$$\mathcal{L} = \sum_{i=1}^{N} w_i(-\mu_i + y_i \ln \mu_i - \ln y_i!) = \sum_{i=1}^{N} w_i(-e^{\mathbf{x}_i'\boldsymbol{\beta}} + y_i\mathbf{x}_i'\boldsymbol{\beta} - \ln y_i!)$$

where w_i is defined as follows:

1	if neither the WEIGHT nor FREQ statement is used.
W_i	where W_i are the nonnormalized values of the variable that are specified in the WEIGHT statement in which the NONORMALIZE option is specified.
$\frac{n}{\sum_{i=1}^{n} W_i} W_i$	where W_i are the nonnormalized values of the variable that is specified in the WEIGHT statement.
F_i	where F_i are the values of the variable that is specified in the FREQ statement.
$W_i F_i$	if both the WEIGHT statement, without the NONORMALIZE option, and the FREQ statement are specified.
$\frac{\sum_{i=1}^{n} F_i}{\sum_{i=1}^{n} F_i W_i} W_i F_i$	if both the FREQ and WEIGHT statements are specified.

The gradient and the Hessian are, respectively,

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{N} w_i (y_i - \mu_i)\mathbf{x}_i = \sum_{i=1}^{N} w_i (y_i - e^{\mathbf{x}_i' \boldsymbol{\beta}})\mathbf{x}_i$$

$$\frac{\partial^2 \mathcal{L}}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}'} = -\sum_{i=1}^{N} w_i \mu_i \mathbf{x}_i \mathbf{x}_i{}' = -\sum_{i=1}^{N} w_i e^{\mathbf{x}_i' \boldsymbol{\beta}} \mathbf{x}_i \mathbf{x}_i'$$

The Poisson model has been criticized for its restrictive property that the conditional variance must equal the conditional mean. Real-life data are often characterized by *overdispersion* (that is, the variance exceeds the mean). Allowing for overdispersion can improve model predictions because the Poisson restriction of equal mean and variance results in the underprediction of zeros when overdispersion exists. The most commonly used model that accounts for overdispersion is the negative binomial model. Conway-Maxwell-Poisson regression enables you to model both overdispersion and underdispersion.

Conway-Maxwell-Poisson Regression

The Conway-Maxwell-Poisson (CMP) distribution is a generalization of the Poisson distribution that enables you to model both underdispersed and overdispersed data. This distribution was originally proposed by Conway and Maxwell (1962), but its implementation to model under- and overdispersed count data is attributed to Shmueli et al. (2005).

Recall that y_i, given the vector of covariates $\mathbf{x}_i$, is independently Poisson-distributed as

$$P(Y_i = y_i | \mathbf{x}_i) = \frac{e^{-\lambda_i} \lambda_i^{y_i}}{y_i!}, \quad y_i = 0, 1, 2, \ldots$$

The Conway-Maxwell-Poisson distribution is defined as

$$P(Y_i = y_i | \mathbf{x}_i, \mathbf{z}_i) = \frac{1}{Z(\lambda_i, \nu_i)} \frac{\lambda_i^{y_i}}{(y_i!)^{\nu_i}}, \quad y_i = 0, 1, 2, \ldots$$

where the normalization factor is

$$Z(\lambda_i, \nu_i) = \sum_{n=0}^{\infty} \frac{\lambda_i^n}{(n!)^{\nu_i}}$$

and

$$\lambda_i = \exp(\mathbf{x}_i'\boldsymbol{\beta})$$

$$\nu_i = -\exp(\mathbf{g}_i'\delta)$$

The $\boldsymbol{\beta}$ vector is a $(k+1)\times 1$ parameter vector. (The intercept is β_0, and the coefficients for the k regressors are $\beta_1,\ldots,\beta_k$.) The δ vector is an $(m+1)\times 1$ parameter vector. (The intercept is represented by δ_0, and the coefficients for the m regressors are $\delta_1,\ldots,\delta_k$.) The covariates are represented by x_i and g_i vectors.

One of the restrictive properties of the Poisson model is that the conditional mean and variance must be equal:

$$E(y_i|\mathbf{x}_i) = V(y_i|\mathbf{x}_i) = \lambda_i = \exp(\mathbf{x}_i'\boldsymbol{\beta})$$

The CMP distribution overcomes this restriction by defining an additional parameter, ν, which governs the rate of decay of successive ratios of probabilities such that

$$P(Y_i = y_i - 1)/P(Y_i = y_i) = \frac{(y_i)^{\nu_i}}{\lambda_i}$$

The introduction of the additional parameter, ν, allows for flexibility in modeling the tail behavior of the distribution. If $\nu = 1$, the ratio is equal to the rate of decay of the Poisson distribution. If $\nu < 1$, the rate of decay decreases, enabling you to model processes that have longer tails than the Poisson distribution (overdispersed data). If $\nu > 1$, the rate of decay increases in a nonlinear fashion, thus shortening the tail of the distribution (underdispersed data).

There are several special cases of the Conway-Maxwell-Poisson distribution. If $\lambda < 1$ and $\nu \to \infty$, the Conway-Maxwell-Poisson results in the Bernoulli distribution. In this case, the data can take only the values 0 and 1, which represents an extreme underdispersion. If $\nu = 1$, the Poisson distribution is recovered with its equidispersion property. When $\nu = 0$ and $\lambda < 1$, the normalization factor is convergent and forms a geometric series,

$$Z(\lambda_i, 0) = \frac{1}{1-\lambda_i}$$

and the probability density function becomes

$$P(Y = y_i; \lambda_i, \nu_i = 0) = (1-\lambda_i)\lambda_i^{y_i}$$

The geometric distribution represents a case of severe overdispersion.

Mean, Variance, and Dispersion for the Conway-Maxwell-Poisson Model

The mean and the variance of the Conway-Maxwell-Poisson distribution are defined as

$$E[Y] = \frac{\partial \ln Z}{\partial \ln \lambda}$$

$$V[Y] = \frac{\partial^2 \ln Z}{\partial^2 \ln \lambda}$$

The Conway-Maxwell-Poisson distribution does not have closed-form expressions for its moments in terms of its parameters λ and ν. However, the moments can be approximated. Shmueli et al. (2005) use asymptotic expressions for Z to derive $E(Y)$ and $V(Y)$ as

$$E[Y] \approx \lambda^{1/\nu} + \frac{1}{2\nu} - \frac{1}{2}$$

$$V[Y] \approx \frac{1}{\nu}\lambda^{1/\nu}$$

In the Conway-Maxwell-Poisson model, the summation of infinite series is evaluated using a logarithmic expansion. The mean and variance are calculated as follows for the Shmueli et al. (2005) model:

$$E(Y) = \frac{1}{Z(\lambda,\nu)} \sum_{j=0}^{\infty} \frac{j\lambda^j}{(j!)^\nu}$$

$$V(Y) = \frac{1}{Z(\lambda,\nu)} \sum_{j=0}^{\infty} \frac{j^2\lambda^j}{(j!)^\nu} - E(Y)^2$$

The dispersion is defined as

$$D(Y) = \frac{V(Y)}{E(Y)}$$

Likelihood Function for the Conway-Maxwell-Poisson Model

The likelihood for a set of n independently and identically distributed variables $y_1, y_2, \ldots, y_n$ is written as

$$\begin{aligned} L(y_1, y_2, \ldots, y_n|\lambda,\nu) &= \frac{\prod_{i=1}^n \lambda^{y_i}}{(\prod_{i=1}^n y_i!)^\nu} Z(\lambda,\nu)^{-n} \\ &= \lambda^{\sum_{i=1}^n y_i} \exp\left(-\nu \sum_{i=1}^n \ln(y_i!)\right) Z(\lambda,\nu)^{-n} \\ &= \lambda^{S_1} \exp(-\nu S_2) Z(\lambda,\nu)^{-n} \end{aligned}$$

where S_1 and S_2 are sufficient statistics for $y_1, y_2, \ldots, y_n$. You can see from the preceding equation that the Conway-Maxwell-Poisson distribution is a member of the exponential family. The log-likelihood function can be written as

$$\mathcal{L} = -n \ln(Z(\lambda,\nu)) + \sum_{i=1}^n (y_i \ln(\lambda) - \nu \ln(y_i!))$$

The gradients can be written as

$$\mathcal{L}_\beta = \left(\sum_{k=1}^N y_k - n\frac{\lambda Z(\lambda,\nu)_\lambda}{Z(\lambda,\nu)}\right)\mathbf{x}$$

$$\mathcal{L}_\delta = \left(\sum_{k=1}^N \ln(y_k!) - n\frac{Z(\lambda,\nu)_\nu}{Z(\lambda,\nu)}\right)\nu\mathbf{z}$$

Conway-Maxwell-Poisson Regression: Guikema and Coffelt (2008) Reparameterization

Guikema and Coffelt (2008) propose a reparameterization of the Shmueli et al. (2005) Conway-Maxwell-Poisson model to provide a measure of central tendency that can be interpreted in the context of the generalized linear model. By substituting $\lambda = \mu^{\nu}$, the Guikema and Coffelt (2008) formulation is written as

$$P(Y = y_i; \mu, \nu) = \frac{1}{S(\mu, \nu)} \left(\frac{\mu^{y_i}}{y_i!} \right)^{\nu}$$

where the new normalization factor is defined as

$$S(\mu, \nu) = \sum_{j=0}^{\infty} \left(\frac{\mu^{j}}{j!} \right)^{\nu}$$

In terms of their new formulations, the mean and variance of Y are given as

$$E[Y] = \frac{1}{\nu} \frac{\partial \ln S}{\partial \ln \mu}$$

$$V[Y] = \frac{1}{\nu^2} \frac{\partial^2 \ln S}{\partial^2 \ln \mu}$$

They can be approximated as

$$E[Y] \approx \mu + \frac{1}{2}\nu - \frac{1}{2}$$

$$V[Y] \approx \frac{\mu}{\nu}$$

In the COUNTREG procedure, the mean and variance are calculated according to the following formulas for the Guikema and Coffelt (2008) model:

$$E(Y) = \frac{1}{Z(\lambda, \mu)} \sum_{j=0}^{\infty} \frac{j \mu^{\nu j}}{(j!)^{\nu}}$$

$$V(Y) = \frac{1}{Z(\lambda, \mu)} \sum_{j=0}^{\infty} \frac{j^2 \mu^{\nu j}}{(j!)^{\nu}} - E(Y)^2$$

In terms of the new parameter μ, the log-likelihood function is specified as

$$\mathcal{L} = \ln(S(\mu, \nu)) + \nu \sum_{i=1}^{N} (y_i \ln(\mu) - \ln(y_i!))$$

and the gradients are calculated as

$$\mathcal{L}_{\beta} = \left(\nu \sum_{i=1}^{N} y_i - \frac{\mu S(\mu, \nu)_{\mu}}{S(\mu, \nu)} \right) \mathbf{x}$$

$$\mathcal{L}_\delta = \left(\sum_{i=1}^{N} (y_i \ln(\mu) - \ln(y_i!)) - \frac{S(\mu,\nu)_\nu}{S(\mu,\nu)} \right) \nu \mathbf{g}$$

The default in the COUNTREG procedure is the Guikema and Coffelt (2008) specification. The Shmueli et al. (2005) model can be estimated by specifying the PARAMETER=LAMBDA option. If you specify DISP=CMPOISSON in the MODEL statement and you omit the DISPMODEL statement, the model is estimated according to the Lord, Guikema, and Geedipally (2008) specification, where ν represents a single parameter that does not depend on any covariates. The Lord, Guikema, and Geedipally (2008) specification makes the model comparable to the negative binomial model because it has only one parameter.

The dispersion is defined as

$$D(Y) = \frac{V(Y)}{E(Y)}$$

Using the Guikema and Coffelt (2008) specification results in the integral part of μ representing the mode, which is a reasonable approximation for the mean. The dispersion can be written as

$$D(Y) = \frac{V(Y)}{E(Y)} \approx \frac{\frac{\mu}{\nu}}{\mu + \frac{1}{2}\nu - \frac{1}{2}} \approx \frac{1}{\nu}$$

When $\nu < 1$, the variance can be shown to be greater than the mean and the dispersion greater than 1. This is a result of overdispersed data. When $\nu = 1$ and the mean and variance are equal, the dispersion is equal to 1 (Poisson model). When $\nu > 1$, the variance is smaller than the mean and the dispersion is less than 1. This is a result of underdispersed data.

All Conway-Maxwell-Poisson models in the COUNTREG procedure are parameterized in terms of dispersion, where

$$-\ln(\nu) = \delta_0 + \sum_{n=1}^{q} \delta_n g_n$$

Negative values of $\ln(\nu)$ indicate that the data are approximately overdispersed, and positive values of $\ln(\nu)$ indicate that the data are approximately underdispersed.

Negative Binomial Regression

The Poisson regression model can be generalized by introducing an unobserved heterogeneity term for observation i. Thus, the individuals are assumed to differ randomly in a manner that is not fully accounted for by the observed covariates. This is formulated as

$$E(y_i|\mathbf{x}_i, \tau_i) = \mu_i \tau_i = e^{\mathbf{x}_i' \boldsymbol{\beta} + \epsilon_i}$$

where the unobserved heterogeneity term $\tau_i = e^{\epsilon_i}$ is independent of the vector of regressors $\mathbf{x}_i$. Then the distribution of y_i conditional on $\mathbf{x}_i$ and τ_i is Poisson with conditional mean and conditional variance $\mu_i \tau_i$:

$$f(y_i|\mathbf{x}_i, \tau_i) = \frac{\exp(-\mu_i \tau_i)(\mu_i \tau_i)^{y_i}}{y_i!}$$

Let $g(\tau_i)$ be the probability density function of τ_i. Then, the distribution $f(y_i|\mathbf{x}_i)$ (no longer conditional on τ_i) is obtained by integrating $f(y_i|\mathbf{x}_i, \tau_i)$ with respect to τ_i:

$$f(y_i|\mathbf{x}_i) = \int_0^\infty f(y_i|\mathbf{x}_i, \tau_i) g(\tau_i) d\tau_i$$

An analytical solution to this integral exists when τ_i is assumed to follow a gamma distribution. This solution is the negative binomial distribution. When the model contains a constant term, it is necessary to assume that $E(e^{\epsilon_i}) = E(\tau_i) = 1$ in order to identify the mean of the distribution. Thus, it is assumed that τ_i follows a gamma(θ, θ) distribution with $E(\tau_i) = 1$ and $V(\tau_i) = 1/\theta$,

$$g(\tau_i) = \frac{\theta^\theta}{\Gamma(\theta)} \tau_i^{\theta-1} \exp(-\theta\tau_i)$$

where $\Gamma(x) = \int_0^\infty z^{x-1} \exp(-z) dz$ is the gamma function and θ is a positive parameter. Then, the density of y_i given $\mathbf{x}_i$ is derived as

$$\begin{aligned}
f(y_i|\mathbf{x}_i) &= \int_0^\infty f(y_i|\mathbf{x}_i, \tau_i) g(\tau_i) d\tau_i \\
&= \frac{\theta^\theta \mu_i^{y_i}}{y_i! \Gamma(\theta)} \int_0^\infty e^{-(\mu_i+\theta)\tau_i} \tau_i^{\theta+y_i-1} d\tau_i \\
&= \frac{\theta^\theta \mu_i^{y_i} \Gamma(y_i+\theta)}{y_i! \Gamma(\theta)(\theta+\mu_i)^{\theta+y_i}} \\
&= \frac{\Gamma(y_i+\theta)}{y_i! \Gamma(\theta)} \left(\frac{\theta}{\theta+\mu_i}\right)^\theta \left(\frac{\mu_i}{\theta+\mu_i}\right)^{y_i}
\end{aligned}$$

Making the substitution $\alpha = \frac{1}{\theta}$ $(\alpha > 0)$, the negative binomial distribution can then be rewritten as

$$f(y_i|\mathbf{x}_i) = \frac{\Gamma(y_i+\alpha^{-1})}{y_i! \Gamma(\alpha^{-1})} \left(\frac{\alpha^{-1}}{\alpha^{-1}+\mu_i}\right)^{\alpha^{-1}} \left(\frac{\mu_i}{\alpha^{-1}+\mu_i}\right)^{y_i}, \quad y_i = 0, 1, 2, \ldots$$

Thus, the negative binomial distribution is derived as a gamma mixture of Poisson random variables. It has conditional mean

$$E(y_i|\mathbf{x}_i) = \mu_i = e^{\mathbf{x}_i'\boldsymbol{\beta}}$$

and conditional variance

$$V(y_i|\mathbf{x}_i) = \mu_i[1 + \frac{1}{\theta}\mu_i] = \mu_i[1 + \alpha\mu_i] > E(y_i|\mathbf{x}_i)$$

The conditional variance of the negative binomial distribution exceeds the conditional mean. Overdispersion results from neglected unobserved heterogeneity. The negative binomial model with variance function $V(y_i|\mathbf{x}_i) = \mu_i + \alpha\mu_i^2$, which is quadratic in the mean, is referred to as the NEGBIN2 model (Cameron and Trivedi 1986). To estimate this model, specify DIST=NEGBIN(p=2) in the MODEL statement. The Poisson distribution is a special case of the negative binomial distribution where $\alpha = 0$. A test of the Poisson distribution can be carried out by testing the hypothesis that $\alpha = \frac{1}{\theta_i} = 0$. A Wald test of this hypothesis is provided (it is the reported t statistic for the estimated α in the negative binomial model).

The log-likelihood function of the negative binomial regression model (NEGBIN2) is given by

$$\begin{aligned}\mathcal{L} &= \sum_{i=1}^{N} w_i \left\{ \sum_{j=0}^{y_i-1} \ln(j+\alpha^{-1}) - \ln(y_i!) \right. \\ &\quad \left. -(y_i+\alpha^{-1})\ln(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})) + y_i\ln(\alpha) + y_i\mathbf{x}_i'\boldsymbol{\beta} \right\}\end{aligned}$$

$$\Gamma(y+a)/\Gamma(a) = \prod_{j=0}^{y-1}(j+a)$$

if y is an integer. See "Poisson Regression" on page 622 for the definition of w_i.

The gradient is

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{N} w_i \frac{y_i-\mu_i}{1+\alpha\mu_i}\mathbf{x}_i$$

and

$$\frac{\partial \mathcal{L}}{\partial \alpha} = \sum_{i=1}^{N} w_i \left\{ -\alpha^{-2} \sum_{j=0}^{y_i-1} \frac{1}{(j+\alpha^{-1})} + \alpha^{-2}\ln(1+\alpha\mu_i) + \frac{y_i-\mu_i}{\alpha(1+\alpha\mu_i)} \right\}$$

Cameron and Trivedi (1986) consider a general class of negative binomial models with mean μ_i and variance function $\mu_i + \alpha\mu_i^p$. The NEGBIN2 model, with $p = 2$, is the standard formulation of the negative binomial model. Models with other values of p, $-\infty < p < \infty$, have the same density $f(y_i|\mathbf{x}_i)$ except that α^{-1} is replaced everywhere by $\alpha^{-1}\mu^{2-p}$. The negative binomial model NEGBIN1, which sets $p = 1$, has variance function $V(y_i|\mathbf{x}_i) = \mu_i + \alpha\mu_i$, which is linear in the mean. To estimate this model, specify DIST=NEGBIN(p=1) in the MODEL statement.

The log-likelihood function of the NEGBIN1 regression model is given by

$$\begin{aligned}\mathcal{L} &= \sum_{i=1}^{N} w_i \left\{ \sum_{j=0}^{y_i-1} \ln\left(j+\alpha^{-1}\exp(\mathbf{x}_i'\boldsymbol{\beta})\right) \right. \\ &\quad \left. -\ln(y_i!) - \left(y_i+\alpha^{-1}\exp(\mathbf{x}_i'\boldsymbol{\beta})\right)\ln(1+\alpha) + y_i\ln(\alpha) \right\}\end{aligned}$$

See the section "Poisson Regression" on page 622 for the definition of w_i.

The gradient is

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{N} w_i \left\{ \left(\sum_{j=0}^{y_i-1} \frac{\mu_i}{(j\alpha+\mu_i)} \right)\mathbf{x}_i - \alpha^{-1}\ln(1+\alpha)\mu_i\mathbf{x}_i \right\}$$

and

$$\frac{\partial \mathcal{L}}{\partial \alpha} = \sum_{i=1}^{N} w_i \left\{ -\left(\sum_{j=0}^{y_i-1} \frac{\alpha^{-1}\mu_i}{(j\alpha+\mu_i)} \right) - \alpha^{-2}\mu_i\ln(1+\alpha) - \frac{(y_i+\alpha^{-1}\mu_i)}{1+\alpha} + \frac{y_i}{\alpha} \right\}$$

Zero-Inflated Count Regression Overview

The main motivation for zero-inflated count models is that real-life data frequently display overdispersion and excess zeros. Zero-inflated count models provide a way of modeling the excess zeros in addition to allowing for overdispersion. In particular, for each observation, there are two possible data generation processes. The result of a Bernoulli trial is used to determine which of the two processes is used. For observation i, Process 1 is chosen with probability φ_i and Process 2 with probability $1-\varphi_i$. Process 1 generates only zero counts. Process 2 generates counts from either a Poisson or a negative binomial model. In general,

$$y_i \sim \begin{cases} 0 & \text{with probability} \quad \varphi_i \\ g(y_i) & \text{with probability} \quad 1-\varphi_i \end{cases}$$

Therefore, the probability of $\{Y_i = y_i\}$ can be described as

$$\begin{aligned} P(y_i = 0|\mathbf{x}_i) &= \varphi_i + (1-\varphi_i)g(0) \\ P(y_i|\mathbf{x}_i) &= (1-\varphi_i)g(y_i), \quad y_i > 0 \end{aligned}$$

where $g(y_i)$ follows either the Poisson or the negative binomial distribution. You can specify the probability φ by using the PROBZERO= option in the OUTPUT statement.

When the probability φ_i depends on the characteristics of observation i, φ_i is written as a function of $\mathbf{z}_i'\boldsymbol{\gamma}$, where $\mathbf{z}_i'$ is the $1 \times (q+1)$ vector of zero-inflation covariates and $\boldsymbol{\gamma}$ is the $(q+1) \times 1$ vector of zero-inflation coefficients to be estimated. (The zero-inflation intercept is γ_0; the coefficients for the q zero-inflation covariates are $\gamma_1, \ldots, \gamma_q$.) The function F that relates the product $\mathbf{z}_i'\boldsymbol{\gamma}$ (which is a scalar) to the probability φ_i is called the zero-inflation link function,

$$\varphi_i = F_i = F(\mathbf{z}_i'\boldsymbol{\gamma})$$

In the COUNTREG procedure, the zero-inflation covariates are indicated in the ZEROMODEL statement. Furthermore, the zero-inflation link function F can be specified as either the logistic function,

$$F(\mathbf{z}_i'\boldsymbol{\gamma}) = \Lambda(\mathbf{z}_i'\boldsymbol{\gamma}) = \frac{\exp(\mathbf{z}_i'\boldsymbol{\gamma})}{1+\exp(\mathbf{z}_i'\boldsymbol{\gamma})}$$

or the standard normal cumulative distribution function (also called the probit function),

$$F(\mathbf{z}_i'\boldsymbol{\gamma}) = \Phi(\mathbf{z}_i'\boldsymbol{\gamma}) = \int_0^{\mathbf{z}_i'\boldsymbol{\gamma}} \frac{1}{\sqrt{2\pi}} \exp(-u^2/2)du$$

The zero-inflation link function is indicated in the LINK option in ZEROMODEL statement. The default ZI link function is the logistic function.

Zero-Inflated Poisson Regression

In the zero-inflated Poisson (ZIP) regression model, the data generation process that is referred to earlier as Process 2 is

$$g(y_i) = \frac{\exp(-\mu_i)\mu_i^{y_i}}{y_i!}$$

where $\mu_i = e^{\mathbf{x}_i'\boldsymbol{\beta}}$. Thus the ZIP model is defined as

$$\begin{aligned} P(y_i = 0|\mathbf{x}_i, \mathbf{z}_i) &= F_i + (1 - F_i)\exp(-\mu_i) \\ P(y_i|\mathbf{x}_i, \mathbf{z}_i) &= (1 - F_i)\frac{\exp(-\mu_i)\mu_i^{y_i}}{y_i!}, \quad y_i > 0 \end{aligned}$$

The conditional expectation and conditional variance of y_i are given by

$$E(y_i|\mathbf{x}_i, \mathbf{z}_i) = \mu_i(1 - F_i)$$

$$V(y_i|\mathbf{x}_i, \mathbf{z}_i) = E(y_i|\mathbf{x}_i, \mathbf{z}_i)(1 + \mu_i F_i)$$

Note that the ZIP model (as well as the ZINB model) exhibits overdispersion because $V(y_i|\mathbf{x}_i, \mathbf{z}_i) > E(y_i|\mathbf{x}_i, \mathbf{z}_i)$.

In general, the log-likelihood function of the ZIP model is

$$\mathcal{L} = \sum_{i=1}^{N} w_i \ln\left[P(y_i|\mathbf{x}_i, \mathbf{z}_i)\right]$$

After a specific link function (either logistic or standard normal) for the probability φ_i is chosen, it is possible to write the exact expressions for the log-likelihood function and the gradient.

ZIP Model with Logistic Link Function

First, consider the ZIP model in which the probability φ_i is expressed using a logistic link function—namely,

$$\varphi_i = \frac{\exp(\mathbf{z}_i'\boldsymbol{\gamma})}{1 + \exp(\mathbf{z}_i'\boldsymbol{\gamma})}$$

The log-likelihood function is

$$\begin{aligned} \mathcal{L} = & \sum_{\{i:y_i=0\}} w_i \ln\left[\exp(\mathbf{z}_i'\boldsymbol{\gamma}) + \exp(-\exp(\mathbf{x}_i'\boldsymbol{\beta}))\right] \\ & + \sum_{\{i:y_i>0\}} w_i \left[y_i\mathbf{x}_i'\boldsymbol{\beta} - \exp(\mathbf{x}_i'\boldsymbol{\beta}) - \sum_{k=2}^{y_i} \ln(k) \right] \\ & - \sum_{i=1}^{N} w_i \ln\left[1 + \exp(\mathbf{z}_i'\boldsymbol{\gamma})\right] \end{aligned}$$

See the section "Poisson Regression" on page 622 for the definition of w_i.

The gradient for this model is given by

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{\gamma}} = \sum_{\{i:y_i=0\}} w_i \left[\frac{\exp(\mathbf{z}_i'\boldsymbol{\gamma})}{\exp(\mathbf{z}_i'\boldsymbol{\gamma}) + \exp(-\exp(\mathbf{x}_i'\boldsymbol{\beta}))}\right]\mathbf{z}_i - \sum_{i=1}^{N} w_i \left[\frac{\exp(\mathbf{z}_i'\boldsymbol{\gamma})}{1 + \exp(\mathbf{z}_i'\boldsymbol{\gamma})}\right]\mathbf{z}_i$$

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{\beta}} = \sum_{\{i:y_i=0\}} w_i \left[\frac{-\exp(\mathbf{x}_i'\boldsymbol{\beta})\exp(-\exp(\mathbf{x}_i'\boldsymbol{\beta}))}{\exp(\mathbf{z}_i'\boldsymbol{\gamma}) + \exp(-\exp(\mathbf{x}_i'\boldsymbol{\beta}))}\right]\mathbf{x}_i + \sum_{\{i:y_i>0\}} w_i \left[y_i - \exp(\mathbf{x}_i'\boldsymbol{\beta})\right]\mathbf{x}_i$$

ZIP Model with Standard Normal Link Function

Next, consider the ZIP model in which the probability φ_i is expressed using a standard normal link function: $\varphi_i = \Phi(\mathbf{z}_i'\boldsymbol{\gamma})$. The log-likelihood function is

$$\begin{aligned}\mathcal{L} &= \sum_{\{i:y_i=0\}} w_i \ln\left\{\Phi(\mathbf{z}_i'\boldsymbol{\gamma}) + \left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right]\exp(-\exp(\mathbf{x}_i'\boldsymbol{\beta}))\right\} \\ &+ \sum_{\{i:y_i>0\}} w_i \left\{\ln\left[\left(1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right)\right] - \exp(\mathbf{x}_i'\boldsymbol{\beta}) + y_i\mathbf{x}_i'\boldsymbol{\beta} - \sum_{k=2}^{y_i}\ln(k)\right\}\end{aligned}$$

See the section "Poisson Regression" on page 622 for the definition of w_i.

The gradient for this model is given by

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\boldsymbol{\gamma}} &= \sum_{\{i:y_i=0\}} w_i \frac{\varphi(\mathbf{z}_i'\boldsymbol{\gamma})\left[1-\exp(-\exp(\mathbf{x}_i'\boldsymbol{\beta}))\right]}{\Phi(\mathbf{z}_i'\boldsymbol{\gamma}) + \left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right]\exp(-\exp(\mathbf{x}_i'\boldsymbol{\beta}))}\mathbf{z}_i \\ &- \sum_{\{i:y_i>0\}} w_i \frac{\varphi(\mathbf{z}_i'\boldsymbol{\gamma})}{\left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right]}\mathbf{z}_i\end{aligned}$$

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\boldsymbol{\beta}} &= \sum_{\{i:y_i=0\}} w_i \frac{-\left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right]\exp(\mathbf{x}_i'\boldsymbol{\beta})\exp(-\exp(\mathbf{x}_i'\boldsymbol{\beta}))}{\Phi(\mathbf{z}_i'\boldsymbol{\gamma}) + \left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right]\exp(-\exp(\mathbf{x}_i'\boldsymbol{\beta}))}\mathbf{x}_i \\ &+ \sum_{\{i:y_i>0\}} w_i\left[y_i - \exp(\mathbf{x}_i'\boldsymbol{\beta})\right]\mathbf{x}_i\end{aligned}$$

Zero-Inflated Conway-Maxwell-Poisson Regression

In the Conway-Maxwell-Poisson regression model, the data generation process is defined as

$$P(Y_i = y_i|\mathbf{x}_i,\mathbf{z}_i) = \frac{1}{Z(\lambda_i,\nu_i)}\frac{\lambda_i^{y_i}}{(y_i!)^{\nu_i}}, \quad y_i = 0,1,2,\ldots$$

where the normalization factor is

$$Z(\lambda_i,\nu_i) = \sum_{n=0}^{\infty}\frac{\lambda_i^n}{(n!)^{\nu_i}}$$

and

$$\lambda_i = \exp(\mathbf{x}_i'\boldsymbol{\beta})$$

$$\nu_i = -\exp(\mathbf{g}_i'\delta)$$

The zero-inflated Conway-Maxwell-Poisson model can be written as

$$\begin{aligned} P(y_i = 0|\mathbf{x}_i, \mathbf{z}_i) &= F_i + (1 - F_i)\frac{1}{Z(\lambda_i, \nu_i)} \\ P(y_i|\mathbf{x}_i, \mathbf{z}_i) &= (1 - F_i)\frac{1}{Z(\lambda_i, \nu_i)}\frac{\lambda_i^{y_i}}{(y_i!)^{\nu_i}}, \quad y_i > 0 \end{aligned}$$

The conditional expectation and conditional variance of y_i are given by

$$E(y_i|\mathbf{x}_i, \mathbf{z}_i) = (1 - F_i)\frac{1}{Z(\lambda, \nu)}\sum_{j=0}^{\infty}\frac{j\lambda^j}{(j!)^{\nu}}$$

$$V(y_i|\mathbf{x}_i, \mathbf{z}_i) = (1 - F_i)\frac{1}{Z(\lambda, \nu)}\sum_{j=0}^{\infty}\frac{j^2\lambda^j}{(j!)^{\nu}} - E(y_i|\mathbf{x}_i, \mathbf{z}_i)^2$$

The general form of the log-likelihood function for the Conway-Maxwell-Poisson zero-inflated model is

$$\mathcal{L} = \sum_{i=1}^{N} w_i \ln\left[P(y_i|\mathbf{x}_i, \mathbf{z}_i)\right]$$

Zero-Inflated Conway-Maxwell-Poisson Model with Logistic Link Function

In this model the probability φ_i is expressed by using a logistic link function as

$$\varphi_i = \Lambda(\mathbf{z}_i'\boldsymbol{\gamma}) = \frac{\exp(\mathbf{z}_i'\boldsymbol{\gamma})}{1 + \exp(\mathbf{z}_i'\boldsymbol{\gamma})}$$

The log-likelihood function is

$$\begin{aligned} \mathcal{L} &= \sum_{\{i:y_i=0\}} w_i \ln\left\{\Lambda(\mathbf{z}_i'\boldsymbol{\gamma}) + \left[1 - \Lambda(\mathbf{z}_i'\boldsymbol{\gamma})\right]\frac{1}{Z(\lambda_i, \nu_i)}\right\} \\ &+ \sum_{\{i:y_i>0\}} w_i \left\{\ln\left[\left(1 - \Lambda(\mathbf{z}_i'\boldsymbol{\gamma})\right)\right] - ln(Z(\lambda, \nu)) + (y_i \ln(\lambda) - \nu \ln(y_i!)\right\} \end{aligned}$$

Zero-Inflated Conway-Maxwell-Poisson Model with Normal Link Function

In this model, the probability φ_i is specified by using the standard normal distribution function (probit function): $\varphi_i = \Phi(\mathbf{z}_i'\boldsymbol{\gamma})$.

The log-likelihood function is written as

$$\begin{aligned} \mathcal{L} &= \sum_{\{i:y_i=0\}} w_i \ln\left\{\Phi(\mathbf{z}_i'\boldsymbol{\gamma}) + \left[1 - \Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right]\frac{1}{Z(\lambda_i, \nu_i)}\right\} \\ &+ \sum_{\{i:y_i>0\}} w_i \left\{\ln\left[\left(1 - \Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right)\right] - ln(Z(\lambda, \nu)) + (y_i \ln(\lambda) - \nu \ln(y_i!)\right\} \end{aligned}$$

Zero-Inflated Negative Binomial Regression

The zero-inflated negative binomial (ZINB) model in PROC COUNTREG is based on the negative binomial model with quadratic variance function (p=2). The ZINB model is obtained by specifying a negative binomial distribution for the data generation process referred to earlier as Process 2:

$$g(y_i) = \frac{\Gamma(y_i + \alpha^{-1})}{y_i!\Gamma(\alpha^{-1})} \left(\frac{\alpha^{-1}}{\alpha^{-1} + \mu_i}\right)^{\alpha^{-1}} \left(\frac{\mu_i}{\alpha^{-1} + \mu_i}\right)^{y_i}$$

Thus the ZINB model is defined to be

$$\begin{aligned} P(y_i = 0|\mathbf{x}_i, \mathbf{z}_i) &= F_i + (1 - F_i)\,(1 + \alpha\mu_i)^{-\alpha^{-1}} \\ P(y_i|\mathbf{x}_i, \mathbf{z}_i) &= (1 - F_i)\,\frac{\Gamma(y_i + \alpha^{-1})}{y_i!\Gamma(\alpha^{-1})} \left(\frac{\alpha^{-1}}{\alpha^{-1} + \mu_i}\right)^{\alpha^{-1}} \\ &\times \left(\frac{\mu_i}{\alpha^{-1} + \mu_i}\right)^{y_i}, \quad y_i > 0 \end{aligned}$$

In this case, the conditional expectation and conditional variance of y_i are

$$E(y_i|\mathbf{x}_i, \mathbf{z}_i) = \mu_i(1 - F_i)$$

$$V(y_i|\mathbf{x}_i, \mathbf{z}_i) = E(y_i|\mathbf{x}_i, \mathbf{z}_i)\,[1 + \mu_i(F_i + \alpha)]$$

Like the ZIP model, the ZINB model exhibits overdispersion because the conditional variance exceeds the conditional mean.

ZINB Model with Logistic Link Function

In this model, the probability φ_i is given by the logistic function—namely,

$$\varphi_i = \frac{\exp(\mathbf{z}_i'\boldsymbol{\gamma})}{1 + \exp(\mathbf{z}_i'\boldsymbol{\gamma})}$$

The log-likelihood function is

$$\begin{aligned} \mathcal{L} &= \sum_{\{i:y_i=0\}} w_i \ln\left[\exp(\mathbf{z}_i'\boldsymbol{\gamma}) + (1 + \alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{-\alpha^{-1}}\right] \\ &+ \sum_{\{i:y_i>0\}} w_i \sum_{j=0}^{y_i-1} \ln(j + \alpha^{-1}) \\ &+ \sum_{\{i:y_i>0\}} w_i \left\{-\ln(y_i!) - (y_i + \alpha^{-1})\ln(1 + \alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})) + y_i\ln(\alpha) + y_i\mathbf{x}_i'\boldsymbol{\beta}\right\} \\ &- \sum_{i=1}^{N} w_i \ln\left[1 + \exp(\mathbf{z}_i'\boldsymbol{\gamma})\right] \end{aligned}$$

See the section "Poisson Regression" on page 622 for the definition of w_i.

The gradient for this model is given by

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \boldsymbol{\gamma}} &= \sum_{\{i:y_i=0\}} w_i \left[\frac{\exp(\mathbf{z}_i'\boldsymbol{\gamma})}{\exp(\mathbf{z}_i'\boldsymbol{\gamma}) + (1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{-\alpha^{-1}}}\right]\mathbf{z}_i \\ &- \sum_{i=1}^{N} w_i \left[\frac{\exp(\mathbf{z}_i'\boldsymbol{\gamma})}{1+\exp(\mathbf{z}_i'\boldsymbol{\gamma})}\right]\mathbf{z}_i\end{aligned}$$

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}} &= \sum_{\{i:y_i=0\}} w_i \left[\frac{-\exp(\mathbf{x}_i'\boldsymbol{\beta})(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{-\alpha^{-1}-1}}{\exp(\mathbf{z}_i'\boldsymbol{\gamma}) + (1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{-\alpha^{-1}}}\right]\mathbf{x}_i \\ &+ \sum_{\{i:y_i>0\}} w_i \left[\frac{y_i - \exp(\mathbf{x}_i'\boldsymbol{\beta})}{1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})}\right]\mathbf{x}_i\end{aligned}$$

$$\frac{\partial \mathcal{L}}{\partial \alpha} = \sum_{\{i:y_i=0\}} w_i \frac{\alpha^{-2}\left[(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))\ln(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})) - \alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})\right]}{\exp(\mathbf{z}_i'\boldsymbol{\gamma})(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{(1+\alpha)/\alpha} + (1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))}$$

$$+ \sum_{\{i:y_i>0\}} w_i \left\{-\alpha^{-2}\sum_{j=0}^{y_i-1}\frac{1}{(j+\alpha^{-1})} + \alpha^{-2}\ln(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})) + \frac{y_i - \exp(\mathbf{x}_i'\boldsymbol{\beta})}{\alpha(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))}\right\}$$

ZINB Model with Standard Normal Link Function

For this model, the probability φ_i is specified using the standard normal distribution function (probit function): $\varphi_i = \Phi(\mathbf{z}_i'\boldsymbol{\gamma})$. The log-likelihood function is

$$\begin{aligned}\mathcal{L} &= \sum_{\{i:y_i=0\}} w_i \ln\left\{\Phi(\mathbf{z}_i'\boldsymbol{\gamma}) + \left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right](1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{-\alpha^{-1}}\right\} \\ &+ \sum_{\{i:y_i>0\}} w_i \ln\left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right] \\ &+ \sum_{\{i:y_i>0\}} w_i \sum_{j=0}^{y_i-1}\left\{\ln(j+\alpha^{-1})\right\} \\ &- \sum_{\{i:y_i>0\}} w_i \ln(y_i!) \\ &- \sum_{\{i:y_i>0\}} w_i (y_i+\alpha^{-1})\ln(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})) \\ &+ \sum_{\{i:y_i>0\}} w_i y_i \ln(\alpha) \\ &+ \sum_{\{i:y_i>0\}} w_i y_i \mathbf{x}_i'\boldsymbol{\beta}\end{aligned}$$

See the section "Poisson Regression" on page 622 for the definition of w_i.

The gradient for this model is given by

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\gamma}} = \sum_{\{i:y_i=0\}} w_i \left[\frac{\varphi(\mathbf{z}_i'\boldsymbol{\gamma}) \left[1 - (1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{-\alpha^{-1}}\right]}{\Phi(\mathbf{z}_i'\boldsymbol{\gamma}) + \left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right](1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{-\alpha^{-1}}} \right] \mathbf{z}_i$$

$$- \sum_{\{i:y_i>0\}} w_i \left[\frac{\varphi(\mathbf{z}_i'\boldsymbol{\gamma})}{1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})} \right] \mathbf{z}_i$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}} = \sum_{\{i:y_i=0\}} w_i \frac{-\left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right]\exp(\mathbf{x}_i'\boldsymbol{\beta})(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{-(1+\alpha)/\alpha}}{\Phi(\mathbf{z}_i'\boldsymbol{\gamma}) + \left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right](1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{-\alpha^{-1}}} \mathbf{x}_i$$

$$+ \sum_{\{i:y_i>0\}} w_i \left[\frac{y_i - \exp(\mathbf{x}_i'\boldsymbol{\beta})}{1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})} \right] \mathbf{x}_i$$

$$\frac{\partial \mathcal{L}}{\partial \alpha} = \sum_{\{i:y_i=0\}} w_i \frac{\left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right]\alpha^{-2}\left[(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))\ln(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})) - \alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})\right]}{\Phi(\mathbf{z}_i'\boldsymbol{\gamma})(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{(1+\alpha)/\alpha} + \left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right](1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))}$$

$$+ \sum_{\{i:y_i>0\}} w_i \left\{ -\alpha^{-2} \sum_{j=0}^{y_i-1} \frac{1}{(j+\alpha^{-1})} + \alpha^{-2}\ln(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})) + \frac{y_i - \exp(\mathbf{x}_i'\boldsymbol{\beta})}{\alpha(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))} \right\}$$

Spatial Lag of X Model (Experimental)

The spatial lag of X (SLX) model is illustrated by using the general framework for a zero-inflated model. According to the section "Zero-Inflated Count Regression Overview" on page 630, the data model for y_i can be formulated as

$$y_i \sim \begin{cases} 0 & \text{with probability} \quad \varphi_i \\ g(y_i) & \text{with probability} \quad 1-\varphi_i \end{cases}$$

and the general model for parameters can be written in matrix form as

$$\begin{aligned} \boldsymbol{\lambda} &= \exp(\mathbf{X}\boldsymbol{\beta}) \\ \boldsymbol{\varphi} &= F(\mathbf{Z}\boldsymbol{\gamma}) \\ \boldsymbol{\nu} &= -\exp(\mathbf{G}\boldsymbol{\delta}) \end{aligned}$$

where $\boldsymbol{\varphi} = (\varphi_1, \ldots, \varphi_n)'$, $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n)'$, and $\boldsymbol{\nu} = (\nu_1, \ldots, \nu_n)'$. In addition, $\mathbf{Z}_1$, $\mathbf{X}_1$, and $\mathbf{G}_1$ are design matrices, in which the ith row is $\mathbf{z}_i'$, $\mathbf{x}_i'$, and $\mathbf{g}_i'$ for $i = 1, 2, ..., n$, respectively.

In the spatial context, data are often collected over a predetermined set of spatial units, $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n$. In this case, both the dependent variable and the explanatory variables are spatially referenced. For example, $y_i = y(\mathbf{s}_i)$ denotes the dependent variable that is observed at location $\mathbf{s}_i$. For the SLX model, the data model for y_i remains the same. However, the parameter model becomes

$$\begin{aligned}
\boldsymbol{\lambda} &= \exp(\mathbf{X}_1\boldsymbol{\beta}_1 + \mathbf{W}\mathbf{X}_2\,\boldsymbol{\beta}_2) = \exp(\mathbf{X}\boldsymbol{\beta}) \\
\boldsymbol{\varphi} &= F(\mathbf{Z}_1\boldsymbol{\gamma}_1 + \mathbf{W}\mathbf{Z}_2\,\boldsymbol{\gamma}_2) = F(\mathbf{Z}\boldsymbol{\gamma}) \\
\boldsymbol{\nu} &= -\exp(\mathbf{G}_1\boldsymbol{\delta}_1 + \mathbf{W}\mathbf{G}_2\,\boldsymbol{\delta}_2) = -\exp(\mathbf{G}\boldsymbol{\delta})
\end{aligned}$$

where $\mathbf{W}$ is the spatial weights matrix, $\mathbf{X} = [\mathbf{X}_1\ \mathbf{W}\mathbf{X}_2]$, $\mathbf{Z} = [\mathbf{Z}_1\ \mathbf{W}\mathbf{Z}_2]$, and $\mathbf{G} = [\mathbf{G}_1\ \mathbf{W}\mathbf{G}_2]$. Moreover, $\boldsymbol{\beta}$ becomes a column vector by stacking $\boldsymbol{\beta}_1$ on top of $\boldsymbol{\beta}_2$, and similarly for $\boldsymbol{\gamma}$ and $\boldsymbol{\delta}$. For the sake of flexibility, $\mathbf{X}_2$ does not have to be the same as $\mathbf{X}_1$. Similar arguments apply to the DISPMODEL and ZEROMODEL statements. From the modeling perspective, the SLX model can be useful when spatial effects (as represented by the $\mathbf{W}\mathbf{X}_2$, $\mathbf{W}\mathbf{Z}_2$, and $\mathbf{W}\mathbf{G}_2$ terms) are important. The intercept term is always excluded from the design matrix $\mathbf{X}_2$, $\mathbf{Z}_2$, or $\mathbf{G}_2$.

A spatial weights matrix $\mathbf{W}$ is a square matrix, which often has nonnegative entries and its dimension is the total number of unique spatial units. Moreover, the diagonal elements of $\mathbf{W}$ are zeros because a spatial unit is not considered to be its own neighbor. Furthermore, the spatial weight w_{ij} between locations $\mathbf{s}_i$ and $\mathbf{s}_j$ describes how much influence the spatial unit $\mathbf{s}_j$ has on $\mathbf{s}_i$. In practice, $\mathbf{W}$ is often row-normalized; thus $\mathbf{W}x_1$ can be interpreted as the spatially weighted average of x_1.

In the SLX model, missing spatial weights are not allowed unless the NORMALIZE option is specified, in which case missing spatial weights are replaced by zeros. In addition, missing values are not allowed for the variables (including both dependent and explanatory variables) in the primary data set (which is specified in the DATA= option in the PROC COUNTREG statement).

The SPATIALEFFECTS, SPATIALZEROEFFECTS, and SPATIALDISPEFFECTS statements are used to include spatial effects in design matrices $\mathbf{X}_2$, $\mathbf{Z}_2$, and $\mathbf{G}_2$, respectively. Observations in the primary data set (specified in the DATA= option in the PROC COUNTREG statement) can be presented in different orders of spatial units than they are presented in the spatial weights data set (specified in the WMAT= option in the PROC COUNTREG statement). In this case, the SPATIALID statement enables you to use a spatial ID variable to associate the observations in the primary data set with those in the spatial weights data set. The SLX model is not supported for a panel data model.

Variable Selection

Variable Selection Methods

Variable Selection Using an Information Criterion

This type of variable selection uses either Akaike's information criterion (AIC) or the Schwartz Bayesian criterion (SBC) and either a forward selection method or a backward elimination method.

Forward selection starts from a small subset of variables. In each step, the variable that gives the largest decrease in the value of the information criterion specified in the CRITER= option (AIC or SBC) is added.

The process stops when the next candidate to be added does not reduce the value of the information criterion by more than the amount specified in the LSTOP= option in the MODEL statement.

Backward elimination starts from a larger subset of variables. In each step, one variable is dropped based on the information criterion chosen.

You can force a variable to be retained in the variable selection process by adding a RETAIN list to the SELECT=INFO (or SELECTVAR=INFO) option in your model. For example, suppose you add a RETAIN list to the SELECT=INFO option in your model as follows:

```
MODEL Art = Mar Kid5 Phd  / dist=negbin(p=2) SELECT=INFO(lstop=0.001 RETAIN(Phd));
```

Then this causes the variable selection process to consider only those models that contain Phd as a regressor. As a result, you are guaranteed that Phd will appear as one of the regressor variables in whatever model the variable selection process produces. The model that results is the "best" (relative to your selection criterion) of all the possible models that contain Phd.

When a ZEROMODEL statement is used in conjunction with a MODEL statement, then all the variables that appear in the ZEROMODEL statement are retained by default unless the ZEROMODEL statement itself contains a SELECT=INFO option.

For example, suppose you have the following:

```
MODEL Art = Mar Kid5 Phd  / dist=negbin(p=2) SELECT=INFO(lstop=0.001 RETAIN(Phd));
ZEROMODEL Art ~ Fem Ment / link=normal;
```

Then Phd is retained in the MODEL statement and all the variables in the ZEROMODEL statement (Fem and Ment) are retained as well. You can add an empty SELECT=INFO clause to the ZEROMODEL statement to indicate that all the variables in that statement are eligible for elimination (that is, need not be retained) during variable selection. For example:

```
MODEL Art = Mar Kid5 Phd  / dist=negbin(p=2) SELECT=INFO(lstop=0.001 RETAIN(Phd));
ZEROMODEL Art ~ Fem Ment / link=normal SELECT=INFO();
```

In this example, only Phd from the MODEL statement is guaranteed to be retained. All the other variables in the MODEL statement and all the variables in the ZEROMODEL statement are eligible for elimination.

Similarly, if your ZEROMODEL statement contains a SELECT=INFO option but your MODEL statement does not, then all the variables in the MODEL statement are retained, whereas only those variables listed in the RETAIN() list of the SELECT=INFO option for your ZEROMODEL statement are retained. For example:

```
MODEL Art = Mar Kid5 Phd  / dist=negbin(p=2) ;
ZEROMODEL Art ~ Fem Ment / link=normal SELECT=INFO(RETAIN(Ment));
```

Here, all the variables in the MODEL statement (Mar Kid5 Phd) are retained, but only the Ment variable in the ZEROMODEL statement is retained.

Variable Selection Using Penalized Likelihood

Variable selection in the linear regression context can be achieved by adding some form of penalty on the regression coefficients. One particular such form is L_1 norm penalty, which leads to LASSO:

$$\min_{\beta} \|Y - X\beta\|^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

This penalty method is becoming more popular in linear regression, because of the computational development in the recent years. However, how to generalize the penalty method for variable selection to the more general statistical models is not trivial. Some work has been done for the generalized linear models, in the sense that the likelihood depends on the data through a linear combination of the parameters and the data:

$$l(\beta|x) = l\left(x^T\beta\right)$$

In the more general form, the likelihood as a function of the parameters can be denoted by $l(\theta) = \sum_i l_i(\theta)$, where θ is a vector that can include any parameters and $l(\cdot)$ is the likelihood for each observation. For example, in the Poisson model, $\theta = (\beta_0, \beta_1, \ldots, \beta_p)$, and in the negative binomial model $\theta = (\beta_0, \beta_1, \ldots, \beta_p, \alpha)$. The following discussion introduces the penalty method, using the Poisson model as an example, but it applies similarly to the negative binomial model. The penalized likelihood function takes the form

$$Q(\beta) = \sum_i l_i(\beta) - n \sum_{j=1}^{p} p_{\lambda_j}\left(|\beta_j|\right)$$

The L_1 norm penalty function that is used in the calculation is specified as

$$p_\lambda(|\beta|) = \lambda$$

The main challenge for this penalized likelihood method is on the computation side. The penalty function is nondifferentiable at zero, posing a computational problem for the optimization. To get around this nondifferentiability problem, Fan and Li (2001) suggested a local quadratic approximation for the penalty function. However, it was later found that the numerical performance is not satisfactory in a few respects. Zou and Li (2008) proposed local linear approximation (LLA) to solve the problem (see page 639) numerically. The algorithm replaces the penalty function with a linear approximation around a fixed point $\beta^{(0)}$:

$$p_\lambda\left(|\beta_j|\right) \approx p_\lambda\left(\left|\beta_j^{(0)}\right|\right) + p'_\lambda\left(\left|\beta_j^{(0)}\right|\right)\left(|\beta_j| - \left|\beta_j^{(0)}\right|\right)$$

Then the problem can be solved iteratively. Start from $\beta^{(0)} = \hat{\beta}_M$, which denotes the usual MLE estimate. For iteration k,

$$\beta^{(k+1)} = \arg\max_{\beta} \left\{ \sum_i l_i(\beta) - n \sum_{j=1}^{p} p'_\lambda\left(\left|\beta_j^{(k)}\right|\right)|\beta_j| \right\}$$

The algorithm stops when $\|\beta^{(k+1)} - \beta^{(k)}\|$ is small. To save computing time, you can also choose a maximum number of iterations. This number can be specified by the LLASTEPS= option.

The objective function is nondifferentiable. The optimization problem can be solved using an optimization methods with constraints, by a variable transformation

$$\beta_j = \beta_j^+ - \beta_j^-, \beta_j^+ \geq 0, \beta_j^- \geq 0$$

For each fixed tuning parameter λ, you can solve the preceding optimization problem to obtain an estimate for β. Because of the property of the L_1 norm penalty, some of the coefficients in β can be exactly zero. The remaining question is to choose the best tuning parameter λ. You can use either of the approaches that are described in the following subsections.

The GCV Approach In the GCV approach, the generalized cross validation criteria (GCV) is computed for each value of λ on a predetermined grid $\{\lambda_1, \ldots, \lambda_L\}$; the value of λ that achieves the minimum of the GCV is the optimal tuning parameter. The maximum value λ_L can be determined by lemma 1 in Park and Hastie (2007) as follows. Suppose β_0 is free of penalty in the objective function. Let $\hat{\beta}_0$ be the MLE of β_0 by forcing the rest of the parameters to be zero. Then the maximum value of λ is

$$\begin{aligned}\lambda_L &= \arg\max_{\lambda} \left\{ \max_{\lambda} : \left| \frac{\partial l}{\partial \beta_j}(\hat{\beta}_0) \right| \leq n P'_\lambda(|\beta_j|), j = 1, \ldots, p \right\} \\ &= \arg\max_{\lambda} \left\{ \left| \frac{1}{n} \frac{\partial l}{\partial \beta_j}(\hat{\beta}_0) \right|, j = 1, \ldots, p \right\}\end{aligned}$$

You can compute the GCV by using the LASSO framework. In the last step of Newton-Raphson approximation, you have

$$\frac{1}{2} \min_{\beta} \left\| (\nabla^2 l(\beta^{(k)}))^{1/2} (\beta - \beta^{(k)}) + (\nabla^2 l(\beta^{(k)}))^{-1/2} \nabla l(\beta^{(k)}) \right\|^2 + n \sum_{j=1}^{p} p'_\lambda(|\beta_j^{(k)}|)|\beta_j|$$

The solution $\hat{\beta}$ satisfies

$$\hat{\beta} - \beta^{(k)} = -(\nabla^2 l(\beta^{(k)}) - 2W^-)^{-1} \left(\nabla l(\beta^{(k)}) - 2\mathbf{b} \right)$$

where

$$\begin{aligned}W^- &= n\text{diag}(W_1^-, \ldots, W_p^-) \\ W_j^- &= \begin{cases} \frac{p'_\lambda(|\beta_j^{(k)}|)}{|\beta_j|}, \text{if} \beta_j \neq 0 \\ 0, \text{if} \beta_j = 0 \end{cases} \\ \mathbf{b} &= n\text{diag}(p'_\lambda(|\beta_1^{(k)}|)\text{sgn}(\beta_1), \ldots, p'_\lambda(|\beta_p^{(k)}|)\text{sgn}(\beta_p))\end{aligned}$$

Note that the intercept term has no penalty on its absolute value, and therefore the W_j^- term that corresponds to the intercept is 0. More generally, you can make any parameter (such as the α in the negative binomial model) in the likelihood function free of penalty, and you treat them the same as the intercept.

The effective number of parameters is

$$\begin{aligned}e(\lambda) &= \text{tr} \left\{ \left(\nabla^2 l(\beta^{(k)}) \right)^{1/2} \left(\nabla^2 l(\beta^{(k)}) - 2W^- \right)^{-1} \left(\nabla^2 l(\beta^{(k)}) \right)^{1/2} \right\} \\ &= \text{tr} \left\{ \left(\nabla^2 l(\beta^{(k)}) - 2W^- \right)^{-1} \nabla^2 l(\beta^{(k)}) \right\}\end{aligned}$$

and the generalized cross validation error is

$$\mathrm{GCV}(\lambda) = \frac{l(\hat{\beta})}{n[1 - e(\lambda)/n]^2}$$

The GCV1 Approach Another form of GCV uses the number of nonzero coefficients as the degrees of freedom:

$$e_1(\lambda) = \sum_{j=0}^{p} \mathbf{1}_{[\beta_j \neq 0]}$$

$$\mathrm{GCV}_1(\lambda) = \frac{l(\hat{\beta})}{n[1 - e_1(\lambda)/n]^2}$$

The standard errors follow the sandwich formula:

$$\begin{aligned}\mathrm{cov}(\hat{\beta}) &= \left\{\nabla^2 l(\beta^{(k)}) - 2W^-\right\}^{-1} \widehat{\mathrm{cov}}\left(\nabla l(\beta^{(k)}) - 2\mathbf{b}\right) \left\{\nabla^2 l(\beta^{(k)}) - 2W^-\right\}^{-1} \\ &= \left\{\nabla^2 l(\beta^{(k)}) - 2W^-\right\}^{-1} \widehat{\mathrm{cov}}\left(\nabla l(\beta^{(k)})\right) \left\{\nabla^2 l(\beta^{(k)}) - 2W^-\right\}^{-1}\end{aligned}$$

It is common practice to report only the standard errors of the nonzero parameters.

Variable Selection with a NOINT Model

If you specify the NOINT option in your MODEL statement, the model produced by variable selection will always contain at least one effect from the original MODEL statement. If you request forward selection with a NOINT model and you do not retain any main model effect, then the only effects that will be candidates for the single-effect model that is derived in the first step will be the effects that are present in the original MODEL statement. For all subsequent steps, all effects from the MODEL, ZEROMODEL, DISPMODEL, and SPATIALEFFECTS statements will be candidates for inclusion in the model that is derived at that step in the process. Meanwhile, if you request backward selection with a NOINT model, you do not retain a specific main model effect, and a model that contains only one effect from the original MODEL statement is derived at a particular step, then that effect will remain in all the models that are evaluated in all subsequent steps.

Panel Data Analysis

Panel Data Poisson Regression with Fixed Effects

The count regression model for panel data can be derived from the Poisson regression model. Consider the multiplicative one-way panel data model,

$$y_{it} \sim \mathrm{Poisson}(\mu_{it})$$

where

$$\mu_{it} = \alpha_i \lambda_{it} = \alpha_i \exp(\mathbf{x}_{it}'\boldsymbol{\beta}), \quad i = 1, \ldots, N, \quad t = 1, \ldots, T$$

Here, α_i are the individual effects.

In the fixed-effects model, the α_i are unknown parameters. The fixed-effects model can be estimated by eliminating α_i by conditioning on $\sum_t y_{it}$.

In the random-effects model, the α_i are independent and identically distributed (iid) random variables, in contrast to the fixed effects model. The random-effects model can then be estimated by assuming a distribution for α_i.

In the Poisson fixed-effects model, conditional on λ_{it} and parameter α_i, y_{it} is iid Poisson-distributed with parameter $\mu_{it} = \alpha_i \lambda_{it} = \alpha_i \exp(\mathbf{x}'_{it}\boldsymbol{\beta})$, and x_{it} does not include an intercept. Then, the conditional joint density for the outcomes within the ith panel is

$$\begin{aligned} P[y_{i1},\ldots,y_{iT_i}|\sum_{t=1}^{T_i} y_{it}] &= P[y_{i1},\ldots,y_{iT_i},\sum_{t=1}^{T_i} y_{it}]/P[\sum_{t=1}^{T_i} y_{it}] \\ &= P[y_{i1},\ldots,y_{iT_i}]/P[\sum_{t=1}^{T_i} y_{it}] \end{aligned}$$

Because y_{it} is iid Poisson(μ_{it}), $P[y_{i1},\ldots,y_{iT_i}]$ is the product of T_i Poisson densities. Also, $(\sum_{t=1}^{T_i} y_{it})$ is Poisson($\sum_{t=1}^{T_i} \mu_{it}$). Then,

$$\begin{aligned} P[y_{i1},\ldots,y_{iT_i}|\sum_{t=1}^{T_i} y_{it}] &= \frac{\sum_{t=1}^{T_i}(\exp(-\mu_{it})\mu_{it}^{y_{it}}/y_{it}!)}{\exp(-\sum_{t=1}^{T_i}\mu_{it})\left(\sum_{t=1}^{T_i}\mu_{it}\right)^{\sum_{t=1}^{T_i} y_{it}}/\left(\sum_{t=1}^{T_i} y_{it}\right)!} \\ &= \frac{\exp(-\sum_{t=1}^{T_i}\mu_{it})\left(\prod_{t=1}^{T_i}\mu_{it}^{y_{it}}\right)\left(\prod_{t=1}^{T_i} y_{it}!\right)}{\exp(-\sum_{t=1}^{T_i}\mu_{it})\prod_{t=1}^{T_i}\left(\sum_{s=1}^{T_i}\mu_{is}\right)^{y_{it}}/\left(\sum_{t=1}^{T_i} y_{it}\right)!} \\ &= \frac{(\sum_{t=1}^{T_i} y_{it})!}{(\prod_{t=1}^{T_i} y_{it}!)}\prod_{t=1}^{T_i}\left(\frac{\mu_{it}}{\sum_{s=1}^{T_i}\mu_{is}}\right)^{y_{it}} \\ &= \frac{(\sum_{t=1}^{T_i} y_{it})!}{(\prod_{t=1}^{T_i} y_{it}!)}\prod_{t=1}^{T_i}\left(\frac{\lambda_{it}}{\sum_{s=1}^{T_i}\lambda_{is}}\right)^{y_{it}} \end{aligned}$$

Thus, the conditional log-likelihood function of the fixed-effects Poisson model is given by

$$\mathcal{L} = \sum_{i=1}^{N}\left[\ln\left((\sum_{t=1}^{T_i} y_{it})!\right) - \sum_{t=1}^{T_i}\ln(y_{it}!) + \sum_{t=1}^{T_i} y_{it}\ln\left(\frac{\lambda_{it}}{\sum_{s=1}^{T_i}\lambda_{is}}\right)\right]$$

The gradient is

$$\begin{aligned} \frac{\partial\mathcal{L}}{\partial\boldsymbol{\beta}} &= \sum_{i=1}^{N}\sum_{t=1}^{T_i} y_{it}x_{it} - \sum_{i=1}^{N}\sum_{t=1}^{T_i}\left[\frac{y_{it}\sum_{s=1}^{T_i}\left(\exp(\mathbf{x}'_{is}\boldsymbol{\beta})\mathbf{x}_{is}\right)}{\sum_{s=1}^{T_i}\exp(\mathbf{x}'_{is}\boldsymbol{\beta})}\right] \\ &= \sum_{i=1}^{N}\sum_{t=1}^{T_i} y_{it}(\mathbf{x}_{it} - \bar{\mathbf{x}}_i) \end{aligned}$$

where

$$\bar{\mathbf{x}}_i = \sum_{s=1}^{T_i} \left(\frac{\exp(\mathbf{x}_{is}'\boldsymbol{\beta})}{\sum_{k=1}^{T_i} \exp(\mathbf{x}_{ik}'\boldsymbol{\beta})} \right) \mathbf{x}_{is}$$

Panel Data Poisson Regression with Random Effects

In the Poisson random-effects model, conditional on λ_{it} and parameter α_i, y_{it} is iid Poisson-distributed with parameter $\mu_{it} = \alpha_i \lambda_{it} = \alpha_i \exp(\mathbf{x}_{it}'\boldsymbol{\beta})$, and the individual effects, α_i, are assumed to be iid random variables. The joint density for observations in all time periods for the ith individual, $P[y_{i1}, \ldots, y_{iT} | \lambda_{i1}, \ldots, \lambda_{iT_i}]$, can be obtained after the density $g(\alpha)$ of α_i is specified.

Let

$$\alpha_i \sim \text{iid gamma}(\theta, \theta)$$

so that $E(\alpha_i) = 1$ and $V(\alpha_i) = 1/\theta$:

$$g(\alpha_i) = \frac{\theta^\theta}{\Gamma(\theta)} \alpha_i^{\theta-1} \exp(-\theta\alpha_i)$$

Let $\lambda_i = (\lambda_{i1}, \ldots, \lambda_{iT_i})$. Because y_{it} is conditional on λ_{it} and parameter α_i is iid Poisson($\mu_{it} = \alpha_i \lambda_{it}$), the conditional joint probability for observations in all time periods for the ith individual, $P[y_{i1}, \ldots, y_{iT_i} | \lambda_i, \alpha_i]$, is the product of T_i Poisson densities:

$$\begin{aligned}
P[y_{i1}, \ldots, y_{iT_i} | \lambda_i, \alpha_i] &= \prod_{t=1}^{T_i} P[y_{it} | \lambda_i, \alpha_i] \\
&= \prod_{t=1}^{T_i} \left[\frac{\exp(-\mu_{it}) \mu_{it}^{y_{it}}}{y_{it}!} \right] \\
&= \left[\prod_{t=1}^{T_i} \frac{e^{-\alpha_i \lambda_{it}} (\alpha_i \lambda_{it})^{y_{it}}}{y_{it}!} \right] \\
&= \left[\prod_{t=1}^{T_i} \lambda_{it}^{y_{it}} / y_{it}! \right] \left(e^{-\alpha_i \sum_t \lambda_{it}} \alpha_i^{\sum_t y_{it}} \right)
\end{aligned}$$

Then, the joint density for the ith panel conditional on just the λ can be obtained by integrating out α_i:

$$
\begin{aligned}
P[y_{i1},\ldots,y_{iT_i}|\lambda_i] &= \int_0^\infty P[y_{i1},\ldots,y_{iT}|\lambda_i,\alpha_i]g(\alpha_i)d\alpha_i \\
&= \frac{\theta^\theta}{\Gamma(\theta)}\left[\prod_{t=1}^{T_i}\frac{\lambda_{it}^{y_{it}}}{y_{it}!}\right]\int_0^\infty \exp(-\alpha_i\sum_t\lambda_{it})\alpha_i^{\sum_t y_{it}}\alpha_i^{\theta-1}\exp(-\theta\alpha_i)d\alpha_i \\
&= \frac{\theta^\theta}{\Gamma(\theta)}\left[\prod_{t=1}^{T_i}\frac{\lambda_{it}^{y_{it}}}{y_{it}!}\right]\int_0^\infty \exp\left[-\alpha_i\left(\theta+\sum_t\lambda_{it}\right)\right]\alpha_i^{\theta+\sum_t y_{it}-1}d\alpha_i \\
&= \left[\prod_{t=1}^{T_i}\frac{\lambda_{it}^{y_{it}}}{y_{it}!}\right]\frac{\Gamma(\theta+\sum_t y_{it})}{\Gamma(\theta)} \\
&\quad \times\left(\frac{\theta}{\theta+\sum_t\lambda_{it}}\right)^\theta\left(\theta+\sum_t\lambda_{it}\right)^{-\sum_t y_{it}} \\
&= \left[\prod_{t=1}^{T_i}\frac{\lambda_{it}^{y_{it}}}{y_{it}!}\right]\frac{\Gamma(\alpha^{-1}+\sum_t y_{it})}{\Gamma(\alpha^{-1})} \\
&\quad \times\left(\frac{\alpha^{-1}}{\alpha^{-1}+\sum_t\lambda_{it}}\right)^{\alpha^{-1}}\left(\alpha^{-1}+\sum_t\lambda_{it}\right)^{-\sum_t y_{it}}
\end{aligned}
$$

where $\alpha(=1/\theta)$ is the overdispersion parameter. This is the density of the Poisson random-effects model with gamma-distributed random effects. For this distribution, $E(y_{it})=\lambda_{it}$ and $V(y_{it})=\lambda_{it}+\alpha\lambda_{it}^2$; that is, there is overdispersion.

Then the log-likelihood function is written as

$$
\begin{aligned}
\mathcal{L} &= \sum_{i=1}^N\left\{\sum_{t=1}^{T_i}\ln(\frac{\lambda_{it}^{y_{it}}}{y_{it}!})+\alpha^{-1}\ln(\alpha^{-1})-\alpha^{-1}\ln(\alpha^{-1}+\sum_{t=1}^{T_i}\lambda_{it})\right\} \\
&\quad +\sum_{i=1}^N\left\{-\left(\sum_{t=1}^{T_i}y_{it}\right)\ln\left(\alpha^{-1}+\sum_{t=1}^{T_i}\lambda_{it}\right)\right. \\
&\qquad \left.+\ln\left[\Gamma\left(\alpha^{-1}+\sum_{t=1}^{T_i}y_{it}\right)\right]-\ln(\Gamma(\alpha^{-1}))\right\}
\end{aligned}
$$

The gradient is

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}} &= \sum_{i=1}^{N}\left\{\sum_{t=1}^{T_i} y_{it}\mathbf{x}_{it} - \frac{\alpha^{-1}\sum_{t=1}^{T_i}\lambda_{it}\mathbf{x}_{it}}{\alpha^{-1}+\sum_{t=1}^{T_i}\lambda_{it}}\right\} \\ &\quad -\sum_{i=1}^{N}\left\{\left(\sum_{t=1}^{T_i} y_{it}\right)\frac{\sum_{t=1}^{T_i}\lambda_{it}\mathbf{x}_{it}}{\alpha^{-1}+\sum_{t=1}^{T_i}\lambda_{it}}\right\} \\ \frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}} &= \sum_{i=1}^{N}\left\{\sum_{t=1}^{T_i} y_{it}\mathbf{x}_{it} - \frac{(\alpha^{-1}+\sum_{t=1}^{T_i} y_{it})(\sum_{t=1}^{T_i}\lambda_{it}\mathbf{x}_{it})}{\alpha^{-1}+\sum_{t=1}^{T_i}\lambda_{it}}\right\}\end{aligned}$$

and

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \alpha} &= \sum_{i=1}^{N}\left\{-\alpha^{-2}\left[[1+\ln(\alpha^{-1})] - \frac{(\alpha^{-1}+\sum_{t=1}^{T_i} y_{it})}{(\alpha^{-1})+\sum_{t=1}^{T_i}\lambda_{it}} - \ln\left(\alpha^{-1}+\sum_{t=1}^{T_i}\lambda_{it}\right)\right]\right\} \\ &+ \sum_{i=1}^{N}\left\{-\alpha^{-2}\left[\frac{\Gamma'(\alpha^{-1}+\sum_{t=1}^{T_i} y_{it})}{\Gamma(\alpha^{-1}+\sum_{t=1}^{T_i} y_{it})} - \frac{\Gamma'(\alpha^{-1})}{\Gamma(\alpha^{-1})}\right]\right\}\end{aligned}$$

where $\lambda_{it} = \exp(\mathbf{x}'_{it}\boldsymbol{\beta})$, $\Gamma'(\cdot) = d\Gamma(\cdot)/d(\cdot)$ and $\Gamma'(\cdot)/\Gamma(\cdot)$ is the digamma function.

Panel Data Negative Binomial Regression with Fixed Effects

This section shows the derivation of a negative binomial model with fixed effects. Keep the assumptions of the Poisson-distributed dependent variable

$$y_{it} \sim Poisson\,(\mu_{it})$$

But now let the Poisson parameter be random with gamma distribution and parameters (λ_{it}, δ),

$$\mu_{it} \sim \Gamma\,(\lambda_{it}, \delta)$$

where one of the parameters is the exponentially affine function of independent variables $\lambda_{it} = \exp\left(\mathbf{x}'_{it}\beta\right)$. Use integration by parts to obtain the distribution of y_{it},

$$\begin{aligned}P\,[y_{it}] &= \int_0^{\infty}\frac{e^{-\mu_{it}}\mu_{it}^{y_{it}}}{y_{it}!}f\,(\mu_{it})\,d\mu_{it} \\ &= \frac{\Gamma\,(\lambda_{it}+y_{it})}{\Gamma\,(\lambda_{it})\,\Gamma\,(y_{it}+1)}\left(\frac{\delta}{1+\delta}\right)^{\lambda_{it}}\left(\frac{1}{1+\delta}\right)^{y_{it}}\end{aligned}$$

which is a negative binomial distribution with parameters (λ_{it}, δ). Conditional joint distribution is given as

$$\begin{aligned}P[y_{i1},\ldots,y_{iT_i}|\sum_{t=1}^{T_i} y_{it}] &= \left(\prod_{t=1}^{T_i}\frac{\Gamma\,(\lambda_{it}+y_{it})}{\Gamma\,(\lambda_{it})\,\Gamma\,(y_{it}+1)}\right) \\ &\quad\times\left(\frac{\Gamma\left(\sum_{t=1}^{T_i}\lambda_{it}\right)\Gamma\left(\sum_{t=1}^{T_i} y_{it}+1\right)}{\Gamma\left(\sum_{t=1}^{T_i}\lambda_{it}+\sum_{t=1}^{T_i} y_{it}\right)}\right)\end{aligned}$$

Hence, the conditional fixed-effects negative binomial log-likelihood is

$$\begin{aligned}\mathcal{L} &= \sum_{i=1}^{N}\left[\log\Gamma\left(\sum_{t=1}^{T_i}\lambda_{it}\right)+\log\Gamma\left(\sum_{t=1}^{T_i}y_{it}+1\right)-\log\Gamma\left(\sum_{t=1}^{T_i}\lambda_{it}+\sum_{t=1}^{T_i}y_{it}\right)\right]\\ &\quad+\sum_{i=1}^{N}\sum_{t=1}^{T_i}\left[\log\Gamma\left(\lambda_{it}+y_{it}\right)-\log\Gamma\left(\lambda_{it}\right)-\log\Gamma\left(y_{it}+1\right)\right]\end{aligned}$$

The gradient is

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\beta} &= \sum_{i=1}^{N}\left[\left(\frac{\Gamma'\left(\sum_{t=1}^{T_i}\lambda_{it}\right)}{\Gamma\left(\sum_{t=1}^{T_i}\lambda_{it}\right)}-\frac{\Gamma'\left(\sum_{t=1}^{T_i}\lambda_{it}+\sum_{t=1}^{T_i}y_{it}\right)}{\Gamma\left(\sum_{t=1}^{T_i}\lambda_{it}+\sum_{t=1}^{T_i}y_{it}\right)}\right)\sum_{t=1}^{T_i}\lambda_{it}\mathbf{x}_{it}\right]\\ &\quad+\sum_{i=1}^{N}\sum_{t=1}^{T_i}\left[\left(\frac{\Gamma'\left(\lambda_{it}+y_{it}\right)}{\Gamma\left(\lambda_{it}+y_{it}\right)}-\frac{\Gamma'\left(\lambda_{it}\right)}{\Gamma\left(\lambda_{it}\right)}\right)\lambda_{it}\mathbf{x}_{it}\right]\end{aligned}$$

Panel Data Negative Binomial Regression with Random Effects

This section describes the derivation of negative binomial model with random effects. Suppose

$$y_{it}\sim Poisson\left(\mu_{it}\right)$$

with the Poisson parameter distributed as gamma,

$$\mu_{it}\sim\Gamma\left(\nu_i\lambda_{it},\delta\right)$$

where its parameters are also random:

$$\nu_i\lambda_{it}=\exp\left(\mathbf{x}_{it}'\beta+\eta_{it}\right)$$

Assume that the distribution of a function of ν_i is beta with parameters (a,b):

$$\frac{\nu_i}{1+\nu_i}\sim Beta\left(a,b\right)$$

Explicitly, the beta density with [0, 1] domain is

$$f\left(z\right)=\left[B\left(a,b\right)\right]^{-1}z^{a-1}\left(1-z\right)^{b-1}$$

where $B\left(a,b\right)$ is the beta function. Then, conditional joint distribution of dependent variables is

$$P\left[y_{i1},\ldots,y_{iT_i}|\mathbf{x}_{i1},\ldots,\mathbf{x}_{iT_i},\nu_i\right]=\prod_{t=1}^{T_i}\frac{\Gamma\left(\lambda_{it}+y_{it}\right)}{\Gamma\left(\lambda_{it}\right)\Gamma\left(y_{it}+1\right)}\left(\frac{1}{1+\nu_i}\right)^{\lambda_{it}}\left(\frac{\nu_i}{1+\nu_i}\right)^{y_{it}}$$

Integrating out the variable ν_i yields the following conditional distribution function:

$$\begin{aligned} P[y_{i1},\ldots,y_{iT_i}|\mathbf{x}_{i1},\ldots,\mathbf{x}_{iT_i}] &= \int_0^1 \left[\prod_{t=1}^{T_i} \frac{\Gamma(\lambda_{it}+y_{it})}{\Gamma(\lambda_{it})\,\Gamma(y_{it}+1)} z_i^{\lambda_{it}} (1-z_i)^{y_{it}}\right] f(z_i)\,dz_i \\ &= \frac{\Gamma(a+b)\,\Gamma\left(a+\sum_{t=1}^{T_i}\lambda_{it}\right)\Gamma\left(b+\sum_{t=1}^{T_i}y_{it}\right)}{\Gamma(a)\,\Gamma(b)\,\Gamma\left(a+b+\sum_{t=1}^{T_i}\lambda_{it}+\sum_{t=1}^{T_i}y_{it}\right)} \\ &\quad \times \prod_{t=1}^{T_i} \frac{\Gamma(\lambda_{it}+y_{it})}{\Gamma(\lambda_{it})\,\Gamma(y_{it}+1)} \end{aligned}$$

Consequently, the conditional log-likelihood function for a negative binomial model with random effects is

$$\begin{aligned} \mathcal{L} &= \sum_{i=1}^{N}\left[\log\Gamma(a+b)+\log\Gamma\left(a+\sum_{t=1}^{T_i}\lambda_{it}\right)+\log\Gamma\left(b+\sum_{t=1}^{T_i}y_{it}\right)\right] \\ &\quad -\sum_{i=1}^{N}\left[\log\Gamma(a)+\log\Gamma(b)+\log\Gamma\left(a+b+\sum_{t=1}^{T_i}\lambda_{it}+\sum_{t=1}^{T_i}y_{it}\right)\right] \\ &\quad +\sum_{i=1}^{N}\sum_{t=1}^{T_i}\left[\log\Gamma(\lambda_{it}+y_{it})-\log\Gamma(\lambda_{it})-\log\Gamma(y_{it}+1)\right] \end{aligned}$$

The gradient is

$$\begin{aligned} \frac{\partial\mathcal{L}}{\partial\beta} &= \sum_{i=1}^{N}\left[\frac{\Gamma'\left(a+\sum_{t=1}^{T_i}\lambda_{it}\right)}{\Gamma\left(a+\sum_{t=1}^{T_i}\lambda_{it}\right)}\sum_{t=1}^{T_i}\lambda_{it}\mathbf{x}_{it}\right] \\ &\quad -\sum_{i=1}^{N}\left[\frac{\Gamma'\left(a+b+\sum_{t=1}^{T_i}\lambda_{it}+\sum_{t=1}^{T_i}y_{it}\right)}{\Gamma\left(a+b+\sum_{t=1}^{T_i}\lambda_{it}+\sum_{t=1}^{T_i}y_{it}\right)}\sum_{t=1}^{T_i}\lambda_{it}\mathbf{x}_{it}\right] \\ &\quad +\sum_{i=1}^{N}\sum_{t=1}^{T_i}\left[\left(\frac{\Gamma'(\lambda_{it}+y_{it})}{\Gamma(\lambda_{it}+y_{it})}-\frac{\Gamma'(\lambda_{it})}{\Gamma(\lambda_{it})}\right)\lambda_{it}\mathbf{x}_{it}\right] \end{aligned}$$

and

$$\begin{aligned} \frac{\partial\mathcal{L}}{\partial a} &= \sum_{i=1}^{N}\left[\frac{\Gamma'(a+b)}{\Gamma(a+b)}+\frac{\Gamma'\left(a+\sum_{t=1}^{T_i}\lambda_{it}\right)}{\Gamma\left(a+\sum_{t=1}^{T_i}\lambda_{it}\right)}\right] \\ &\quad -\sum_{i=1}^{N}\left[\frac{\Gamma'(a)}{\Gamma(a)}+\frac{\Gamma'\left(a+b+\sum_{t=1}^{T_i}\lambda_{it}+\sum_{t=1}^{T_i}y_{it}\right)}{\Gamma\left(a+b+\sum_{t=1}^{T_i}\lambda_{it}+\sum_{t=1}^{T_i}y_{it}\right)}\right] \end{aligned}$$

and

$$\frac{\partial \mathcal{L}}{\partial b} = \sum_{i=1}^{N}\left[\frac{\Gamma'(a+b)}{\Gamma(a+b)} + \frac{\Gamma'\left(b+\sum_{t=1}^{T_i} y_{it}\right)}{\Gamma\left(b+\sum_{t=1}^{T_i} y_{it}\right)}\right] - \sum_{i=1}^{N}\left[\frac{\Gamma'(b)}{\Gamma(b)} + \frac{\Gamma'\left(a+b+\sum_{t=1}^{T_i}\lambda_{it}+\sum_{t=1}^{T_i} y_{it}\right)}{\Gamma\left(a+b+\sum_{t=1}^{T_i}\lambda_{it}+\sum_{t=1}^{T_i} y_{it}\right)}\right]$$

BY Groups and Scoring with an Item Store

If you use the BY statement in conjunction with the ITEMSTORE statement when you fit your model, then the parameter estimates for each BY group are preserved in your item store.

You must use a BY statement if you want to score a data set by using an item store that was created when a BY statement was provided. The names of the BY variables in the data set to be scored (hereafter referred to as the *scored* data set) must match the names of the BY variables in the data set that is used to produce the item store (hereafter referred to as the *fitted* data set). The order of the names of the BY variables in your BY statement must match their order in the BY statement that was used when the item store was created.

The order in which the values of the BY variables appear in the scored data set does not have to match their order in the fitted data set. Furthermore, not all the values of the BY variables that are present in the fitted data set need to be present in the scored data set.

For example, suppose you have a data set named DocVisit that you use to fit a model by using a BY statement. Your BY variable is named AgeGroup, and there are four values for the AgeGroup variable (0, 1, 2, and 3) in the DocVisit data set.

In the first step, you use the following statements to fit your model by using the BY statement and generate an item store named DocVisitByAgeGroup:

```
PROC COUNTREG data=DocVisit;
model doctorvisits = sex illness income / dist=poisson;
store DocVisitByAgeGroup;
by AgeGroup;
run;
```

Now suppose you want to score a second data set named AdditionalPatients by using the DocVisitByAgeGroup item store. Then the AdditionalPatients data set must contain a variable named AgeGroup, and the values of this variable must be a subset of 0, 1, 2, and 3. Suppose that the values of the AgeGroup variable in the AdditionalPatients data set are 1 and 3.

In that case, you can score the data set by using this second step:

```
PROC COUNTREG data=AdditionalPatients restore=DocVisitByAgeGroup;
score out=OutScores mean=meanPoisson probability=prob;
by AgeGroup;
run;
```

Because the AdditionalPatients data set contains two BY groups, PROC COUNTREG first extracts the parameter estimates that are associated with the AgeGroup=1 BY group from the DocVisitByAgeGroup item store and uses them to score the first BY group in the AdditionalPatients data set. Then, PROC COUNTREG extracts the parameter estimates that are associated with the AgeGroup=3 BY group from the DocVisitByAgeGroup item store and uses them to score the second BY group in the AdditionalPatients data set.

What happens if your scored data set contains a value of the BY variable that is not present in the fitted data set? Modifying the preceding example slightly, suppose the values of the AgeGroup variable in the AdditionalPatients data set are 1, 2, 3, and 6. In that case, when the second step is submitted, PROC COUNTREG scores the BY groups in which AgeGroup equals 1, 2, or 3, but it does not attempt to score the BY group in which AgeGroup=6.

If you want to use the parameter estimates that are associated with a particular BY group in an item store to score a data set that contains no BY variable, it is fairly easy to do so. First, you create a new data set based on your original data set that includes an additional single-valued BY variable (whose value corresponds to the BY group in the item store in which you are interested). Second, you use the new data set and the BY statement to retrieve the parameter estimates of interest, which are then used to score the entire data set.

For example, suppose that the AdditionalPatients data set does not contain the AgeGroup variable. But suppose you happen to know that all the observations in the AdditionalPatients data set fall within the age group in which AgeGroup=2, as defined in the DocVisit data set. Then you could score the AdditionalPatients data set by using the following steps.

First, you would create a new data set named AdditionalPatientsWithByVar, which essentially adds a variable named AgeGroup, with its value set to 2, to each observation in the AdditionalPatients data set:

```
data AdditionalPatientsWithByVar;
set AdditionalPatients;
agegroup=2;
run;
```

Then, you would score the AdditionalPatientsWithByVar data set by using the DocVisitByAgeGroup item store along with the BY statement, as follows:

```
PROC COUNTREG data=AdditionalPatientsWithByVar restore=DocVisitByAgeGroup;
score out=OutScores mean=meanPoisson probability=prob;
by AgeGroup;
run;
```

Parameter Naming Conventions for the RESTRICT, TEST, BOUNDS, and INIT Statements

This section describes how you can refer to the parameters that are defined in the MODEL, ZEROMODEL, DISPMODEL, SPATIALEFFECTS, SPATIALDISPEFFECTS, and SPATIALZEROEFFECTS statements when you use the RESTRICT, TEST, BOUNDS, or INIT statement. The following examples use the RESTRICT statement, but the same remarks apply to naming parameters when you use the TEST, BOUNDS, or INIT statement. The names of the parameters are written to the OUTEST= data set.

To impose a restriction on a parameter that is related to a regressor in the MODEL statement, you simply use the name of the regressor itself to refer to its associated parameter. Suppose your model is defined in the following statement, where x1 through x5 are continuous variables:

```
model y = x1 x2 x5;
```

If you want to restrict the parameter that is associated with the regressor x5 to be greater than 1.7, then you use the following statement:

```
RESTRICT x5 > 1.7;
```

To impose a restriction on a parameter associated with a regressor in the ZEROMODEL statement, you can form the name of the parameter by prefixing Inf_ to the name of the regressor. Suppose your MODEL and ZEROMODEL statements are as follows:

```
model y = x1 x2 x5;
zeromodel y ~ x3 x5;
```

If you want to restrict the parameter related to the x5 regressor in the ZEROMODEL statement to be less than 1.0, then you refer to the parameter as Inf_x5 and provide the following statement:

```
RESTRICT Inf_x5 < 1.0;
```

Even though the regressor x5 appears in both the MODEL and ZEROMODEL statements, the parameter associated with x5 in the MODEL statement is, of course, different from the parameter associated with x5 in the ZEROMODEL statement. Thus, the name of a regressor that appears in a RESTRICT statement without any prefix refers to the parameter associated with that regressor in the MODEL statement, and the name of a regressor that appears in a RESTRICT statement with the prefix Inf_ refers to the parameter associated with that regressor in the ZEROMODEL statement. The parameter that is associated with the intercept in the ZEROMODEL is named Inf_Intercept.

In a similar way, you can form the name of a parameter associated with a regressor in the DISPMODEL statement by prefixing Dsp_ to the name of the regressor. The parameter associated with the intercept in the DISPMODEL is named Dsp_Intercept.

And you can form the name of a parameter associated with a regressor in the SPATIALEFFECTS statement by prefixing W_ to the name of the regressor. The parameter associated with the intercept in the SPATIALEFFECTS is named W_Intercept.

Referring to Class-Level Parameters

When your MODEL statement includes a classification variable, you can impose restrictions on the parameters associated with each of the levels that are related to the classification variable as follows.

Suppose your classification variable is named C and it has three levels: 0, 1, 2. Suppose your model is the following:

```
class C;
model y = x1 x2 C;
```

Adding a classification variable as a regressor to your model introduces additional parameters into your model, each of which is associated with one of the levels of the classification variable. You can form the name of the parameter associated with a particular level of your class variable by inserting the underscore character between the name of the classification variable and the value of the level. Thus, to restrict the parameter associated with level 0 of the classification variable C to always be greater than 0.7, you refer to the parameter as C_0 and provide the following statement:

```
RESTRICT C_0 > 0.7;
```

Referring to Parameters Associated with Interactions between Regressors

When a regressor in your model involves an interaction between other regressors, you can impose restrictions on the parameters associated with the interaction.

Suppose you have the following model:

```
model y = x1 x2 x3*x4;
```

You can form the name of the parameter associated with the interaction regressor x3*x4 by replacing the multiplication sign with an underscore. Thus, x3_x4 refers to the parameter that is associated with the interaction regressor x3*x4.

Referring to interactions between regressors and classification variables is handled in the same way. Suppose you have a classification variable that is named C and has three levels: 0, 1, 2. Suppose that your model is the following:

```
class C;
model y = x1 x2 C*x3;
```

The interaction between the continuous variable x3 and the classification variable C introduces three additional parameters, which are named x3_C_0, x3_C_1, and x3_C_2. Although the order of the terms in the interaction is C followed by x3, note how the name of the parameter associated with the interaction is formed by placing the name of the continuous variable x3 first, followed by an underscore, followed by the name of the classification variable C, followed by an underscore, and then followed by the level value. Depending on the parameterization you specify in your CLASS statement, for each interaction in your model that involves a classification variable, one of the parameters associated with that interaction might be dropped from your model prior to optimization.

The name of a parameter associated with a nested interaction is formed in a slightly different way. Suppose you have a classification variable that is named C and has three levels: 0, 1, 2. Suppose that your model is the following:

```
class C;
model y = x1 x2 x3(C);
```

The nested interaction between the continuous variable x3 and the classification variable C introduces three additional parameters, which are named x3_C__0, x3_C__1, and x3_C__2. Note how the name in each case is formed from the name of the regressor by replacing the left and right parentheses with underscores and then appending another underscore followed by the level value.

Referring to Class Level Parameters with Negative Values

When the value of a level is a negative number, you must replace the minus sign with an underscore when you form the name of the parameter that is associated with that particular level of the classification variable. For example, suppose your classification variable is named D and has four levels: –1, 0, 1, 2. Suppose your model is the following:

```
class D;
model y = x1 x2 D;
```

To restrict the parameter that is associated with level –1 of the classification variable D to always be less than 0.4, you refer to the parameter as D__1 (note that there are two underscores in this parameter name: one to connect the name of the classification variable to its value and the other to replace the minus sign in the value itself) and provide following statement:

```
RESTRICT D__1 < 0.4;
```

Dropping a Class Level Parameter to Avoid Collinearity

Depending on the parameterization you impose on your classification variable, one of the parameters associated with its levels might be dropped from your model prior to optimization in order to avoid collinearity. For example, when the default parameterization GLM is imposed, the parameter that is associated with the last level of your classification variable is dropped prior to optimization. If you attempt to impose a restriction on a dropped parameter by using the RESTRICT statement, PROC COUNTREG issues an error message in the log.

For example, suppose that your classification variable is named C and that it has three levels: 0, 1, 2. Suppose your model is the following:

```
class C;
model y = x1 x2 C;
```

Because no additional options are specified in the CLASS statement, GLM parameterization is assumed. This means that the parameter named C_2 (which is the parameter associated with the last level of your classification variable) is dropped from your model before the optimizer is invoked. Therefore, an error is issued if you attempt to restrict the C_2 parameter in any way by referring to it in a RESTRICT statement. For example, the following RESTRICT statement generates an error:

```
RESTRICT C_2 < 0.3;
```

Referring to Implicit Parameters

For certain model types, one or more implicit parameters are added to your model prior to optimization. You can impose restrictions on these implicit parameters.

For the Poisson model for which ERRORCOMP=RANDOM is specified, PROC COUNTREG automatically adds the _Alpha parameter to your model.

If no ERRORCOMP= option is specified for zero-inflated binomial and negative binomial models, then PROC COUNTREG adds the _Alpha parameter to the model. If ERRORCOMP=RANDOM is specified for the zero-inflated binomial and negative binomial models, then PROC COUNTREG adds two implicit parameters to the model: _Alpha and _Beta.

For Conway-Maxwell Poisson models that do not include a DISPMODEL statement, the _lnNu parameter is added to the model.

Whenever your model type dictates the addition of one or more of these implicit parameters, you can impose restrictions on the implicit parameters by referring to them by name in a RESTRICT statement. For example, if your model type implies the existence of the _Alpha parameter, you can restrict _Alpha to be greater than 0.2 as follows:

```
RESTRICT _Alpha > 0.2;
```

Computational Resources

The time and memory that PROC COUNTREG requires are proportional to the number of parameters in the model and the number of observations in the data set being analyzed. Less time and memory are required for smaller models and fewer observations. Also affecting these resources are the method that is chosen to calculate the variance-covariance matrix and the optimization method. All optimization methods available through the METHOD= option have similar memory use requirements.

The processing time might differ for each method, depending on the number of iterations and functional calls needed. The data set is read into memory to save processing time. If not enough memory is available to hold the data, the COUNTREG procedure stores the data in a utility file on disk and rereads the data as needed from this file. When this occurs, the execution time of the procedure increases substantially. The gradient and the variance-covariance matrix must be held in memory. If the model has p parameters including the intercept, then at least $8 * (p + p * (p + 1)/2)$ bytes are needed. If the quasi-maximum likelihood method is used to estimate the variance-covariance matrix (COVEST=QML), an additional $8 * p * (p + 1)/2$ bytes of memory are needed.

Time is also a function of the number of iterations needed to converge to a solution for the model parameters. The number of iterations that are needed cannot be known in advance. The MAXITER= option can be used to limit the number of iterations that PROC COUNTREG does. The convergence criteria can be altered by nonlinear optimization options available in the PROC COUNTREG statement. For a list of all the nonlinear optimization options, see Chapter 7, "Nonlinear Optimization Methods."

Nonlinear Optimization Options

PROC COUNTREG uses the nonlinear optimization (NLO) subsystem to perform nonlinear optimization tasks. In the PROC COUNTREG statement, you can specify nonlinear optimization options that are then passed to the NLO subsystem. For a list of all the nonlinear optimization options, see Chapter 7, "Nonlinear Optimization Methods."

Covariance Matrix Types

The COUNTREG procedure enables you to specify the estimation method for the covariance matrix. The COVEST=HESSIAN option estimates the covariance matrix based on the inverse of the Hessian matrix, COVEST=OP uses the outer product of gradients, and COVEST=QML produces the covariance matrix based on both the Hessian and outer product matrices. The default is COVEST=HESSIAN.

Although all three methods produce asymptotically equivalent results, they differ in computational intensity and produce results that might differ in finite samples. The COVEST=OP option provides the covariance matrix that is typically the easiest to compute. In some cases, the OP approximation is considered more efficient than the Hessian or QML approximation because it contains fewer random elements. The QML approximation is computationally the most complex because both the outer product of gradients and the Hessian matrix are required. In most cases, OP or Hessian approximation is preferred to QML. The need to use QML approximation arises in some cases when the model is misspecified and the information matrix equality does not hold.

Displayed Output

PROC COUNTREG produces the following displayed output.

Class Level Information

If you specify the CLASS statement, the COUNTREG procedure displays a table that contains the following information:

- classification variable name
- number of levels of the classification variable
- list of values of the classification variable

Iteration History for Parameter Estimates

If you specify the ITPRINT or PRINTALL option in the PROC COUNTREG statement, PROC COUNTREG displays a table that contains the following information for each iteration. Some information is specific to the model-fitting procedure that you choose (for example, Newton-Raphson, trust region, quasi-Newton).

- iteration number

- number of restarts since the fitting began
- number of function calls
- number of active constraints at the current solution
- value of the objective function (–1 times the log-likelihood value) at the current solution
- change in the objective function from previous iteration
- value of the maximum absolute gradient element
- step size (for Newton-Raphson and quasi-Newton methods)
- slope of the current search direction (for Newton-Raphson and quasi-Newton methods)
- lambda (for trust region method)
- radius value at current iteration (for trust region method)

Model Fit Summary

The "Model Fit Summary" table contains the following information:

- dependent (count) variable name
- number of observations used
- number of missing values in data set, if any
- data set name
- type of model that was fit
- parameterization for the Conway-Maxwell-Poisson model
- offset variable name, if any
- zero-inflated link function, if any
- zero-inflated offset variable name, if any
- log-likelihood value at solution
- maximum absolute gradient at solution
- number of iterations
- AIC value at solution (a smaller value indicates better fit)
- SBC value at solution (a smaller value indicates better fit)

Under the "Model Fit Summary" is a statement about whether the algorithm successfully converged.

Parameter Estimates

The "Parameter Estimates" table gives the estimates of the model parameters. In zero-inflated (ZI) models, estimates are also given for the ZI intercept and ZI regressor parameters labeled with the prefix "Inf_". For example, the ZI intercept is labeled "Inf_intercept". If you specify "Age" as a ZI regressor, then the "Parameter Estimates" table labels the corresponding parameter estimate "Inf_Age". If you do not list any ZI regressors, then only the ZI intercept term is estimated.

If the DISPMODEL statement is specified for the Conway-Maxwell-Poisson model, the estimates are given for the dispersion intercept, and parameters are labeled with the prefix "Dsp_". For example, the dispersion model intercept is labeled "Dsp_Intercept". If you specify "Education" as a dispersion model regressor, then the "Parameter Estimates" table labels the corresponding parameter estimate "Dsp_Education". If you do not list any dispersion regressors, then only the dispersion intercept is estimated.

"_Alpha" is the negative binomial dispersion parameter. The t statistic given for "_Alpha" is a test of overdispersion.

Last Evaluation of the Gradient

If you specify the model option ITPRINT, the COUNTREG procedure displays the last evaluation of the gradient vector.

Covariance of Parameter Estimates

If you specify the COVB option in the MODEL statement or in the PROC COUNTREG statement, the COUNTREG procedure displays the estimated covariance matrix, defined as the inverse of the information matrix at the final iteration.

Correlation of Parameter Estimates

If you specify the CORRB option in the MODEL statement or in the PROC COUNTREG statement, PROC COUNTREG displays the estimated correlation matrix. It is based on the Hessian matrix that is used in the final iteration.

Bayesian Analysis

To perform Bayesian analysis, you must specify a BAYES statement. Unless otherwise stated, all options in this section are options in the BAYES statement.

By default, PROC COUNTREG uses the random walk Metropolis algorithm to obtain posterior samples. For information about implementing the Metropolis algorithm in PROC COUNTREG, such as blocking the parameters and tuning the covariance matrices, see the sections "Blocking of Parameters" on page 657 and "Tuning the Proposal Distribution" on page 657.

The Bayes theorem states that

$$p(\theta|\mathbf{y}) \propto \pi(\theta)L(y|\theta)$$

where θ is a parameter or a vector of parameters and $\pi(\theta)$ is the product of the prior densities that are specified in the PRIOR statement. The term $L(y|\theta)$ is the likelihood that is associated with the MODEL statement.

Blocking of Parameters

In a multivariate parameter model, all the parameters are updated in one single block (by default or when you specify the SAMPLING=MULTIMETROPOLIS option). This could be inefficient, especially when parameters have vastly different scales. As an alternative, you could update the parameters one at a time (by specifying SAMPLING=UNIMETROPOLIS).

Tuning the Proposal Distribution

One key factor in achieving high efficiency of a Metropolis-based Markov chain is finding a good proposal distribution for each block of parameters. This process is called *tuning*. The tuning phase consists of a number of loops that are controlled by the options MINTUNE= and MAXTUNE=. The MINTUNE= option controls the minimum number of tuning loops and has a default value of 2. The MAXTUNE= option controls the maximum number of tuning loops and has a default value of 24. Each loop iterates the number of times that are specified by the NTU= option, which has a default of 500. At the end of every loop, PROC COUNTREG examines the acceptance probability for each block. The acceptance probability is the percentage of samples, specified by the NTU= option, that have been accepted. If this probability does not fall within the acceptable tolerance range (see the following section), the proposal distribution is modified before the next tuning loop begins.

A good proposal distribution should resemble the actual posterior distribution of the parameters. Large sample theory states that the posterior distribution of the parameters approaches a multivariate normal distribution (see Gelman et al. 2004, Appendix B; Schervish 1995, Section 7.4). That is why a normal proposal distribution often works well in practice. The default proposal distribution in PROC COUNTREG is the normal distribution.

Scale Tuning

The acceptance rate is closely related to the sampling efficiency of a Metropolis chain. For a random walk Metropolis, a high acceptance rate means that most new samples occur right around the current data point. Their frequent acceptance means that the Markov chain is moving rather slowly and not exploring the parameter space fully. A low acceptance rate means that the proposed samples are often rejected; hence the chain is not moving much. An efficient Metropolis sampler has an acceptance rate that is neither too high nor too low. The scale c in the proposal distribution $q(\cdot|\cdot)$ effectively controls this acceptance probability. Roberts, Gelman, and Gilks (1997) show that if both the target and proposal densities are normal, the optimal acceptance probability (TargetAcceptance) for the Markov chain should be around 0.45 in a one-dimensional problem and should asymptotically approach 0.234 in higher-dimensional problems. The corresponding optimal scale is 2.38, which is the initial scale that is set for each block.

Because of the nature of stochastic simulations, it is impossible to fine-tune a set of variables so that the Metropolis chain has exactly the desired acceptance rate that you want. In addition, Roberts and Rosenthal (2001) empirically demonstrate that an acceptance rate between 0.15 and 0.5 is at least 80% efficient, so there is really no need to fine-tune the algorithms to reach an acceptance probability that is within a small tolerance of the optimal values. PROC COUNTREG works with a probability range, determined by TargetAcceptance $\pm$ 0.075. If the observed acceptance rate in a given tuning loop is less than the lower bound of the range, the scale is reduced; if the observed acceptance rate is greater than the upper bound of the range, the scale is increased. During the tuning phase, a scale parameter in the normal distribution is adjusted as a function of the observed acceptance rate and the target acceptance rate. PROC COUNTREG

uses the updating scheme[1]

$$c_{\text{new}} = \frac{c_{\text{cur}} \cdot \Phi^{-1}(p_{\text{opt}}/2)}{\Phi^{-1}(p_{\text{cur}}/2)}$$

where c_{cur} is the current scale, p_{cur} is the current acceptance rate, and p_{opt} is the optimal acceptance probability.

Covariance Tuning

To tune a covariance matrix, PROC COUNTREG takes a weighted average of the old proposal covariance matrix and the recent observed covariance matrix, based on the number of samples (as specified by the NTU= option) in the current loop. The formula to update the covariance matrix is

$$\text{COV}_{\text{new}} = 0.75\ \text{COV}_{\text{cur}} + 0.25\ \text{COV}_{\text{old}}$$

There are two ways to initialize the covariance matrix:

- The default is an identity matrix that is multiplied by the initial scale of 2.38 and divided by the square root of the number of estimated parameters in the model. A number of tuning phases might be required before the proposal distribution is tuned to its optimal stage, because the Markov chain needs to spend time learning about the posterior covariance structure. If the posterior variances of your parameters vary by more than a few orders of magnitude, if the variances of your parameters are much different from 1, or if the posterior correlations are high, then the proposal tuning algorithm might have difficulty forming an acceptable proposal distribution.
- Alternatively, you can use a numerical optimization routine, such as the quasi-Newton method, to find a starting covariance matrix. The optimization is performed on the joint posterior distribution, and the covariance matrix is a quadratic approximation at the posterior mode. In some cases this is a better and more efficient way of initializing the covariance matrix. However, there are cases, such as when the number of parameters is large, in which the optimization could fail to find a matrix that is positive definite. In those cases, the tuning covariance matrix is reset to the identity matrix.

A by-product of the optimization routine is that it also finds the maximum a posteriori (MAP) estimates with respect to the posterior distribution. The MAP estimates are used as the initial values of the Markov chain.

For more information, see the section "INIT Statement" on page 604.

Initial Values of the Markov Chains

You can assign initial values to any parameters. (For more information, see the section "INIT Statement" on page 604) If you use the optimization option PROPCOV=, then PROC COUNTREG starts the tuning at the optimized values. This option overwrites the provided initial values. If you specify the RANDINIT option, the information that the INIT statement provides is overwritten.

[1] Roberts, Gelman, and Gilks (1997) and Roberts and Rosenthal (2001) demonstrate that the relationship between acceptance probability and scale in a random walk Metropolis scheme is $p = 2\Phi\left(-\sqrt{I}c/2\right)$, where c is the scale, p is the acceptance rate, Φ is the CDF of a standard normal, and $I \equiv E_f[(f'(x)/f(x))^2]$, $f(x)$ is the density function of samples (Roberts, Gelman, and Gilks 1997; Roberts and Rosenthal 2001). This relationship determines the updating scheme, with I replaced by the identity matrix to simplify calculation.

Aggregation of Multiple Chains

When you want to exploit the possibility of running several MCMC instances at the same time (that is, the value of the NTRDS= option is greater than 1), you face the problem of aggregating the chains. In ordinary applications, each MCMC instance can easily obtain stationary samples from the entire posterior distribution. In these applications, you can use the option AGGREGATION=NOWEIGHTED. This option piles one chain on top of another and makes no particular adjustment. However, when the posterior distribution is characterized by multiple distinct posterior modes, some of the MCMC instances fail to obtain stationary samples from the entire posterior distribution. You can use the option AGGREGATION=WEIGHTED when the posterior samples from each MCMC instance approximate well only a part of the posterior distribution.

The main idea behind the option AGGREGATION=WEIGHTED is to consider the entire posterior distribution to be similar to a mixture distribution. When you are sampling with multiple threads, each MCMC instance samples from one of the mixture components. Then the samples from each mixture component are aggregated together using a resampling scheme in which weights are proportional to the nonnormalized posterior distribution.

Description of the Algorithm

The preliminary step of the aggregation that is implied by the option AGGREGATION=WEIGHTED is to run several (K) independent instances of the MCMC algorithm. Each instance searches for a set of stationary samples. Notice that the concept of stationarity is weaker: each instance might be able to explore not the entire posterior but only portions of it. In the following, each column represents the output from one MCMC instance:

$$\begin{pmatrix} x_{11} \\ x_{21} \\ \cdots \\ x_{n1} \end{pmatrix} \begin{pmatrix} x_{12} \\ x_{22} \\ \cdots \\ x_{n2} \end{pmatrix} \cdots \begin{pmatrix} x_{1K} \\ x_{2K} \\ \cdots \\ x_{nK} \end{pmatrix} \sim \text{globally or locally sampled from the posterior}$$

If the length of each chain is less than n, you can augment the corresponding chain by subsampling the chain itself. Each chain is then sorted with respect to the nonnormalized posterior density: $\pi(x_{[1].}) \leq \pi(x_{[2].}) \leq \cdots \pi(x_{[n].})$. Therefore,

$$\begin{pmatrix} x_{11} \\ x_{21} \\ \cdots \\ x_{n1} \end{pmatrix} \begin{pmatrix} x_{12} \\ x_{22} \\ \cdots \\ x_{n2} \end{pmatrix} \cdots \begin{pmatrix} x_{1K} \\ x_{2K} \\ \cdots \\ x_{nK} \end{pmatrix} \rightarrow \begin{pmatrix} x_{[1]1} \\ x_{[2]1} \\ \cdots \\ x_{[n]1} \end{pmatrix} \begin{pmatrix} x_{[1]2} \\ x_{[2]2} \\ \cdots \\ x_{[n]2} \end{pmatrix} \cdots \begin{pmatrix} x_{[1]K} \\ x_{[2]K} \\ \cdots \\ x_{[n]K} \end{pmatrix}$$

The final step is to use a multinomial sampler to resample each row i with weights proportional to the nonnormalized posterior densities:

$$\widetilde{x}_{(i-1)K+1}, \widetilde{x}_{(i-1)K+2}, \ldots, \widetilde{x}_{(i-1)K+K} \sim \text{Multinom}\left[x_{[i]1}, x_{[i]2}, \ldots, x_{[i]K}; \pi(x_{[i]1}), \pi(x_{[i]2}), \ldots, \pi(x_{[i]K})\right]$$

The resulting posterior sample,

$$\widetilde{x}_1, \widetilde{x}_2, \ldots, \widetilde{x}_K, \ldots, \widetilde{x}_{(i-1)K+1}, \widetilde{x}_{(i-1)K+2}, \ldots, \widetilde{x}_{(i-1)K+K}, \ldots, \widetilde{x}_{(n-1)K+1}, \widetilde{x}_{(n-1)K+2}, \ldots, \widetilde{x}_{nK}$$

is a good approximation of the posterior distribution that is characterized by multiple modes.

Automated Initialization of MCMC

The MCMC methods can generate samples from the posterior distribution. The correct implementation of these methods often requires the stationarity analysis, convergence analysis, and accuracy analysis of the posterior samples. These analyses usually imply the following:

- initialization of the proposal distribution
- initialization of the chains (starting values)
- determination of the burn-in
- determination of the length of the chains

In more general terms, this determination is equivalent to deciding whether the samples are drawn from the posterior distribution (stationarity analysis) and whether the number of samples is large enough to accurately approximate the posterior distribution (accuracy analysis). You can use the AUTOMCMC option to automate and facilitate the stationary analysis and the accuracy analysis.

Description of the Algorithm

The algorithm has two phases. In the first phase, the stationarity phase, the algorithm tries to generate stationary samples from the posterior distribution. In the second phase, the accuracy phase, the algorithm searches for an accurate representation of the posterior distribution. The algorithm implements the following tools:

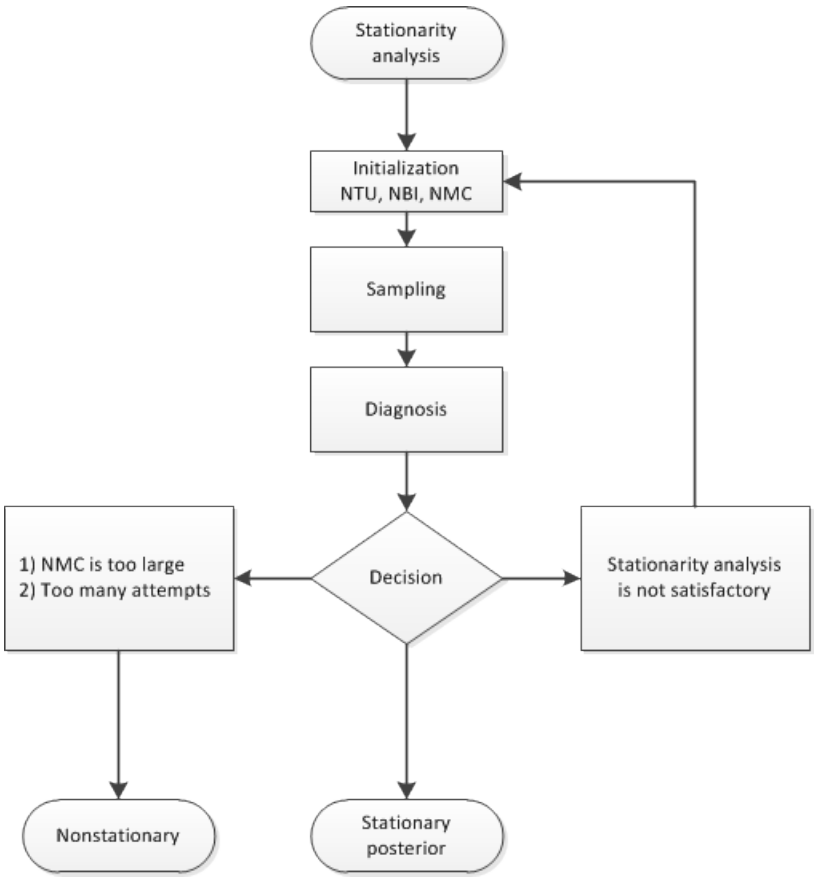
- Geweke test to check stationarity
- Heidelberger-Welch test to check stationarity and provide a proxy for the burn-in
- Heidelberger-Welch halfwidth test to check the accuracy of the posterior mean
- Raftery-Lewis test to check the accuracy of a specified percentile (indirectly providing a proxy for the number of required samples)
- effective sample size analysis to determine a proxy for the number of required samples

During the stationarity phase, the algorithm searches for stationarity. The number of attempts that the algorithm makes is determined by the ATTEMPTS= option. During each attempt, a preliminary tuning stage chooses a proposal distribution for the MCMC sampler. At the end of the preliminary tuning phase, the algorithm analyzes tests for the stationarity of the samples. If the percentage of successful stationary tests is greater than or equal to the percentage that is indicated by the TOL= option, then the posterior sample is considered to be stationary. If the sample cannot be considered stationary, then the algorithm attempts to achieve stationarity by changing some of the initialization parameters as follows:

- increasing the number of tuning samples (NTU= option)
- increasing the number of posterior samples (NMC= option)
- increasing the burn-in (NBI= option)

Figure 12.6 shows a flowchart of the AUTOMCMC algorithm as it searches for stationarity.

Figure 12.6 Flowchart of the AUTOMCMC Algorithm: Stationarity Analysis



You can initialize NMC=M, NBI=B, and NTU=T during the stationarity phase by specifying the NMC=, NBI=, and NTU= options in the BAYES statement. You can also change the minimum stationarity acceptance ratio of successful stationarity tests that are needed to exit the stationarity phase. By default, TOL=0.95. For example:

```
proc countreg data=dataset;
   ...;
   bayes nmc=M nbi=B ntu=T automcmc=( stationarity=(tol=0.95) );
   ...;
run;
```

During the accuracy phase, the algorithm attempts to determine how many posterior samples are needed. The number of attempts is determined by the ATTEMPTS= option. You can choose between two different approaches to study the accuracy:

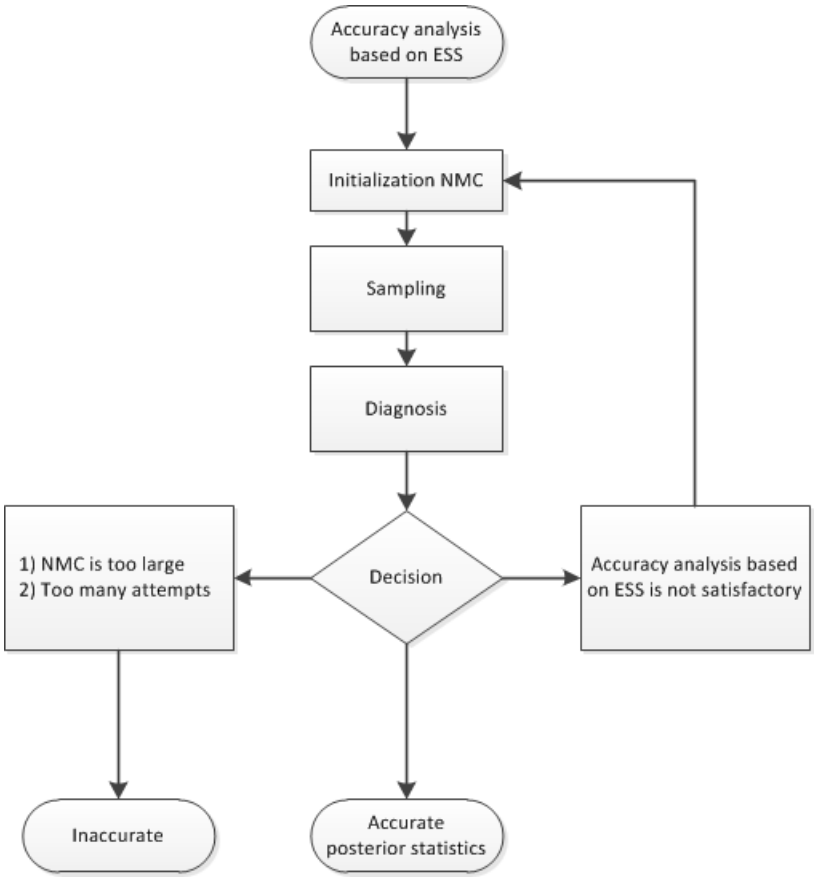
- accuracy analysis based on the effective sample size (ESS)
- accuracy analysis based on the Heidelberger-Welch halfwidth test and the Raftery-Lewis test

If you choose the effective sample size approach, you must provide the minimum number of effective samples that are needed. You can also change the tolerance for the ESS accuracy analysis (by default, TOL=0.95). For example:

```
proc countreg data=dataset;
   ...;
   bayes automcmc=(targetess=N accuracy=(tol=0.95));
   ...;
run;
```

Figure 12.7 shows a flowchart of the AUTOMCMC algorithm based on the effective sample size approach to determine whether the samples provide an accurate representation of the posterior distribution.

Figure 12.7 Flowchart of the AUTOMCMC Algorithm: Accuracy Analysis Based on the ESS



If you choose the accuracy analysis based on the Heidelberger-Welch halfwidht test and the Raftery-Lewis test (the default option), then you might want to choose a posterior quantile of interest for the Raftery-Lewis test (by default, 0.025). You can also change the tolerance for the accuracy analysis (by default, TOL=0.95). Notice that the Raftery-Lewis test produces a proxy for the number of posterior samples that are required. In each attempt, the current number of posterior samples is compared to this proxy. If the proxy is greater than the current NMC, then the algorithm reinitializes itself. To control this reinitialization, you can use the option RLLIMITS=(LB=*lb* UB=*ub*). In particular, there are three cases

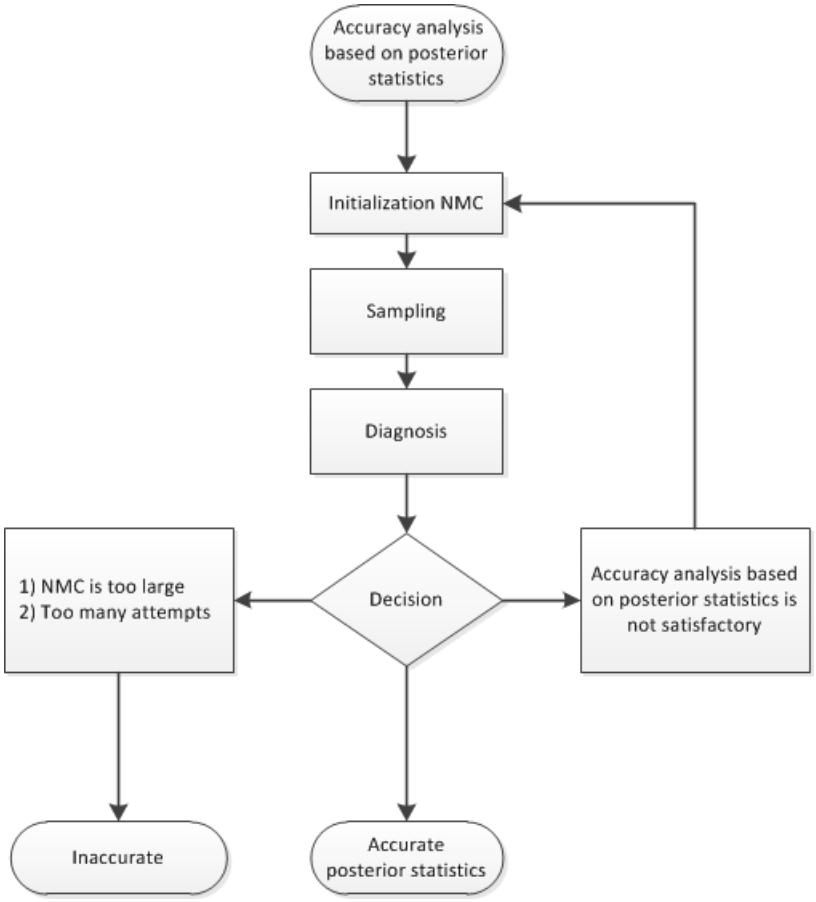
- If the proxy is greater than *ub*, then NMC is set equal to *ub*.
- If the proxy is less than *lb*, then NMC is set equal to *lb*.
- If *lb* is less than the proxy, which is less than *ub*, then NMC is set equal to the proxy.

For example:

```
proc countreg data=dataset;
   ...;
   bayes automcmc=( accuracy=(tol=0.95 targetstats=(rllimits=(lb=k1 ub=k2))) )
        raftery(q=0.025);
   ...;
run;
```

Figure 12.8 shows a flowchart of the AUTOMCMC algorithm based on the Heidelberger-Welch halfwidth test and the Raftery-Lewis test approach to determine whether the posterior samples provide an accurate representation of the posterior distribution.

Figure 12.8 Flowchart of the AUTOMCMC Algorithm: Accuracy Analysis Based on the Heidelberger-Welch Halfwidth Test and the Raftery-Lewis Test



Prior Distributions

The PRIOR statement is used to specify the prior distribution of the model parameters. You must specify a list of parameters, a tilde (~), and then a distribution and its parameters. You can specify multiple PRIOR statements to define independent priors. Parameters that are associated with a regressor variable are referred to by the name of the corresponding regressor variable.

You can specify the special keyword _REGRESSORS to consider all the regressors of a model. If multiple prior statements affect the same parameter, the prior that is specified is used. For example, in a regression that uses three regressors (X1, X2, X3), the following statements imply that the prior on X1 is NORMAL(MEAN=0, VAR=1), the prior on X2 is GAMMA(SHAPE=3, SCALE=4), and the prior on X3 is UNIFORM(MIN=0, MAX=1):

```
...
prior _Regressors ~ uniform(min=0, max=1);
prior X1 X2 ~ gamma(shape=3, scale=4);
prior X1 ~ normal(mean=0, var=1);
...
```

If a parameter is not associated with a PRIOR statement or if some of the prior hyperparameters are missing, then the default choices shown in Table 12.3 are considered.

Table 12.3 Default Values for Prior Distributions

PRIOR *distribution*	**Hyperparameter**$_1$	**Hyperparameter**$_2$	**Min**	**Max**	**Parameters Default Choice**
NORMAL	MEAN=0	VAR=1E6	$-\infty$	∞	Regression-Location-Threshold
IGAMMA	SHAPE=2.000001	SCALE=1	> 0	∞	Scale
GAMMA	SHAPE=1	SCALE=1	0	∞	
UNIFORM			$-\infty$	∞	
BETA	SHAPE1=1	SHAPE2=1	$-\infty$	∞	
T	LOCATION=0	DF=3	$-\infty$	∞	

For density specifications, see the section "Standard Distributions" on page 667.

Automated MCMC Algorithm

The main purpose of the automated MCMC algorithm is to provide the user with the opportunity to obtain a good approximation of the posterior distribution without initializing the MCMC algorithm: initial values, proposal distributions, burn-in, and number of samples.

The automated MCMC algorithm has two phases: tuning and sampling. The tuning phase has two main concerns: the choice of a good proposal distribution and the search for the stationary region of the posterior distribution. In the sampling phase, the algorithm decides how many samples are necessary to obtain good approximations of the posterior mean and some quantiles of interest.

Tuning Phase

During the tuning phase, the algorithm tries to search for a good proposal distribution and, at the same time, to reach the stationary region of the posterior. The choice of the proposal distribution is based on the analysis of the acceptance rates. This is similar to what is done in PROC MCMC; for more information, see Chapter 73, "The MCMC Procedure" (*SAS/STAT User's Guide*). For the stationarity analysis, the main idea is to run two tests, Geweke (Ge) and Heidleberger-Welch (HW), on the posterior chains at the end of each attempt. For more information, see Chapter 7, "Introduction to Bayesian Analysis Procedures" (*SAS/STAT User's Guide*). If the stationarity hypothesis is rejected, then the number of tuning samples is increased and

the tests are repeated in the next attempt. After 10 attempts, the tuning phase ends regardless of the results. The tuning parameters for the first attempt are fixed:

0	burn-in (nbi)
1000	tuning samples (ntu)
10000	MCMC samples (nmc)

For the remaining attempts, the tuning parameters are adjusted dynamically. More specifically, each parameter is assigned an acceptance ratio (AR) of the stationarity hypothesis,

$\mathrm{AR}_i = 0$	if both tests reject the stationarity hypothesis
$\mathrm{AR}_i = 0.5$	if only one test rejects the stationarity hypothesis
$\mathrm{AR}_i = 1$	if neither test rejects the stationarity hypothesis

for $i = 1, \ldots, k$. Then, an overall stationarity average (SA) for all parameters ratios is evaluated,

$$\mathrm{SA} = \sum_{i=1}^{k} \frac{\mathrm{AR}_i}{k}$$

and the number of tuning samples is updated accordingly:

ntu = ntu + 2000	if	$\mathrm{SA} < 70\%$
ntu = ntu + 1000	if	$70\% \leq \mathrm{SA} < 100\%$
ntu = ntu	if	$\mathrm{SA} = 100\%$

The Heidelberger-Welch test also provides an indication of how much burn-in should be used. The algorithm requires this burn-in to be nbi(HW) = 0. If that is not the case, the burn-in is updated accordingly, nbi = nbi + nbi(HW), and a new tuning attempt is made. This choice is motivated by the fact that the burn-in must be discarded in order to reach the stationary region of the posterior distribution.

The number of samples that the Raftery-Lewis nmc(RL) diagnostic tool predicts is also considered: nmc = nmc + nmc(RL). For more information, see Chapter 7, "Introduction to Bayesian Analysis Procedures" (*SAS/STAT User's Guide*). However, in order to exit the tuning phase, nmc(RL) = 0 is not required.

Sampling Phase

The main purpose of the sampling phase is to make sure that the mean and a quantile of interest are evaluated accurately. This can be tested by implementing the Heidelberger-Welch halfwidth test and by using the Raftery-Lewis diagnostic tool. In addition, the requirements that are defined in the tuning phase are also checked: the Geweke and Heidelberger-Welch tests must not reject the stationary hypothesis, and the burn-in that the Heidelberger-Welch test predicts must be zero.

The sampling phase has a maximum of 10 attempts. If the algorithm exceeds this limit, the sampling phase ends and indications of how to improve sampling are given. The sampling phase first updates the burn-in with the information provided by the Heidelberger-Welch test: nbi = nbi + nbi(HW). Then, it determines the difference between the actual number of samples and the number of samples that are predicted by the Raftery-Lewis test: $\Delta[\mathrm{nmc(RL)}] = \mathrm{nmc(RL)} - \mathrm{nmc}$. The new number of samples is updated accordingly:

nmc = nmc + 1000	if	$0 < \Delta[\mathrm{nmc(RL)}] \leq 10000$
nmc = nmc + Δ[nmc(RL)]	if	$10000 < \Delta[\mathrm{nmc(RL)}] \leq 300000$
nmc = nmc + 300000	if	$300000 < \Delta[\mathrm{nmc(RL)}]$

Finally, the Heidelberger-Welch halfwidth test checks whether the sample mean is accurate. If the mean of any parameters is not considered to be accurate, the number of samples is increased:

$$\begin{array}{lll} \text{nmc} = \text{nmc} + \{10000 - \Delta[\text{nmc(RL)}]\} & \text{if} & 10000 - \Delta[\text{nmc(RL)}] \geq 0 \\ \text{nmc} = \text{nmc} & \text{if} & 10000 - \Delta[\text{nmc(RL)}] < 0 \end{array}$$

Marginal Likelihood

The Bayes theorem states that

$$p(\theta|\mathbf{y}) \propto \pi(\theta)L(y|\theta)$$

where θ is a vector of parameters and $\pi(\theta)$ is the product of the prior densities, which are specified in the PRIOR statement. The term $L(y|\theta)$ is the likelihood associated with the MODEL statement. The function $\pi(\theta)L(y|\theta)$ is the nonnormalized posterior distribution over the parameter vector θ. The normalized posterior distribution, or simply the posterior distribution, is

$$p(\theta|\mathbf{y}) = \frac{\pi(\theta)L(y|\theta)}{\int_\theta \pi(\theta)L(y|\theta)d\theta}$$

The denominator $m(y) = \int_\theta \pi(\theta)L(y|\theta)d\theta$, also called the "marginal likelihood," is a quantity of interest because it represents the probability of the data after the effect of the parameter vector has been averaged out. Due to its interpretation, the marginal likelihood can be used in various applications, including model averaging and variable or model selection.

A natural estimate of the marginal likelihood is provided by the harmonic mean,

$$m(y) = \left\{ \frac{1}{n} \sum_{i=1}^{n} \frac{1}{L(y|\theta_i)} \right\}^{-1}$$

where θ_i is a sample draw from the posterior distribution. This estimator has proven to be unstable in practical applications.

An alternative and more stable estimator can be obtained by using an importance sampling scheme. The auxiliary distribution for the importance sampler can be chosen through the cross-entropy theory (Chan and Eisenstat 2015). In particular, given a parametric family of distributions, the auxiliary density function is chosen to be the one closest, in terms of the Kullback-Leibler divergence, to the probability density that would give a zero variance estimate of the marginal likelihood. In practical terms, this is equivalent to the following algorithm:

1. Choose a parametric family, $f(., \beta)$, for the parameters of the model: $f(\theta|\beta)$
2. Evaluate the maximum likelihood estimator of β by using the posterior samples $\theta_1, \ldots, \theta_n$ as data
3. Use $f(\theta^*|\hat{\beta}_{mle})$ to generate the importance samples: $\theta_1^*, \ldots, \theta_{n^*}^*$
4. Estimate the marginal likelihood:

$$m(y) = \frac{1}{n^*} \sum_{j=1}^{n^*} \frac{L(y|\theta_j^*)\pi(\theta_j^*)}{f(\theta_j^*|\hat{\beta}_{mle})}$$

The parametric family for the auxiliary distribution is chosen to be Gaussian. The parameters that are subject to bounds are transformed accordingly

- If $-\infty < \theta < \infty$, then $p = \theta$.
- If $m \leq \theta < \infty$, then $q = \log(\theta - m)$.
- If $-\infty < \theta \leq M$, then $r = \log(M - \theta)$.
- If $m \leq \theta \leq M$, then $s = \log(\theta - m) - \log(M - \theta)$.

Assuming independence for the parameters that are subject to bounds, the auxiliary distribution to generate importance samples is

$$\begin{pmatrix} \mathbf{p} \\ \mathbf{q} \\ \mathbf{r} \\ \mathbf{s} \end{pmatrix} \sim \mathbf{N}\left[\begin{pmatrix} \mu_p \\ \mu_q \\ \mu_r \\ \mu_s \end{pmatrix}, \begin{pmatrix} \Sigma_p & 0 & 0 & 0 \\ 0 & \Sigma_q & 0 & 0 \\ 0 & 0 & \Sigma_r & 0 \\ 0 & 0 & 0 & \Sigma_r \end{pmatrix} \right]$$

where $\mathbf{p}$, $\mathbf{q}$, $\mathbf{r}$ and $\mathbf{s}$ are vectors containing the transformations of the unbounded, bounded-below, bounded-above and bounded-above-and-below parameters. Also, given the imposed independence structure, Σ_p can be a non-diagonal matrix while Σ_q, Σ_r and Σ_s are imposed to be diagonal matrices.

Standard Distributions

Table 12.4 through Table 12.9 show all the distribution density functions that PROC COUNTREG recognizes. You specify these distribution densities in the PRIOR statement.

Table 12.4 Beta Distribution

PRIOR statement	BETA(SHAPE1=a, SHAPE2=b, MIN=m, MAX=M) Note: Commonly $m = 0$ and $M = 1$.
Density	$\frac{(\theta-m)^{a-1}(M-\theta)^{b-1}}{B(a,b)(M-m)^{a+b-1}}$
Parameter restriction	$a > 0, \quad b > 0, \quad -\infty < m < M < \infty$
Range	$\begin{cases} [m, M] & \text{when } a = 1, b = 1 \\ [m, M) & \text{when } a = 1, b \neq 1 \\ (m, M] & \text{when } a \neq 1, b = 1 \\ (m, M) & \text{otherwise} \end{cases}$
Mean	$\frac{a}{a+b} \times (M - m) + m$
Variance	$\frac{ab}{(a+b)^2(a+b+1)} \times (M - m)^2$

Mode	$\begin{cases} \frac{a-1}{a+b-2} \times M + \frac{b-1}{a+b-2} \times m & a > 1, b > 1 \\ m \text{ and } M & a < 1, b < 1 \\ m & \begin{cases} a < 1, b \geq 1 \\ a = 1, b > 1 \end{cases} \\ M & \begin{cases} a \geq 1, b < 1 \\ a > 1, b = 1 \end{cases} \\ \text{not unique} & a = b = 1 \end{cases}$
Defaults	SHAPE1=SHAPE2=1, MIN $\rightarrow -\infty$, MAX $\rightarrow \infty$

Table 12.5 Gamma Distribution

PRIOR statement	GAMMA(SHAPE=a, SCALE=b)
Density	$\frac{1}{b^a \Gamma(a)} \theta^{a-1} e^{-\theta/b}$
Parameter restriction	$a > 0, b > 0$
Range	$[0, \infty)$
Mean	ab
Variance	ab^2
Mode	$(a-1)b$
Defaults	SHAPE=SCALE=1

Table 12.6 Inverse Gamma Distribution

PRIOR statement	IGAMMA(SHAPE=a, SCALE=b)
Density	$\frac{b^a}{\Gamma(a)} \theta^{-(a+1)} e^{-b/\theta}$
Parameter restriction	$a > 0, b > 0$
Range	$0 < \theta < \infty$
Mean	$\frac{b}{a-1}, \quad a > 1$
Variance	$\frac{b^2}{(a-1)^2(a-2)}, \quad a > 2$
Mode	$\frac{b}{a+1}$
Defaults	SHAPE=2.000001, SCALE=1

Table 12.7 Normal Distribution

PRIOR statement	NORMAL(MEAN=μ, VAR=σ^2)
Density	$\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(\theta-\mu)^2}{2\sigma^2}\right)$
Parameter restriction	$\sigma^2 > 0$

Range	$-\infty < \theta < \infty$
Mean	μ
Variance	σ^2
Mode	μ
Defaults	MEAN=0, VAR=1000000

Table 12.8 *t* **Distribution**

PRIOR statement	T(LOCATION=μ, DF=ν)
Density	$\frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)\sqrt{\pi\nu}}\left[1+\frac{(\theta-\mu)^2}{\nu}\right]^{-\frac{\nu+1}{2}}$
Parameter restriction	$\nu > 0$
Range	$-\infty < \theta < \infty$
Mean	μ, for $\nu > 1$
Variance	$\frac{\nu}{\nu-2}$, for $\nu > 2$
Mode	μ
Defaults	LOCATION=0, DF=3

Table 12.9 **Uniform Distribution**

PRIOR statement	UNIFORM(MIN=m, MAX=M)
Density	$\frac{1}{M-m}$
Parameter restriction	$-\infty < m < M < \infty$
Range	$\theta \in [m, M]$
Mean	$\frac{m+M}{2}$
Variance	$\frac{(M-m)^2}{12}$
Mode	Not unique
Defaults	MIN$\rightarrow -\infty$, MAX$\rightarrow \infty$

OUTPUT OUT= Data Set

The OUTPUT statement creates a new SAS data set that contains all the variables in the input data set and, optionally, the estimates of $\mathbf{x}_i'\boldsymbol{\beta}$, the expected value of the response variable, and the probability that the response variable will take the current value or other values that you specify. In a zero-inflated model, you can also request that the output data set contain the estimates of $\mathbf{z}_i'\boldsymbol{\gamma}$, and the probability that the response is zero as a result of the zero-generating process. In a Conway-Maxwell-Poisson model, you can also request that the output data set contains estimates of $\mathbf{g}_i'\boldsymbol{\delta}$, λ, ν, μ, mode, variance and dispersion.

Except for the probability of the current value, these statistics can be computed for all observations in which the regressors are not missing, even if the response is missing. By adding observations that have missing response values to the input data set, you can compute these statistics for new observations or for settings of the regressors that are not present in the data without affecting the model fit.

OUTEST= Data Set

The OUTEST= data set is has two rows: the first row (with _TYPE_='PARM') contains each of the parameter estimates in the model, and the second row (with _TYPE_='STD') contains the standard errors for the parameter estimates in the model.

If you specify the COVOUT option in the PROC COUNTREG statement, the OUTEST= data set also contains the covariance matrix for the parameter estimates. The covariance matrix appears in the observations for which _TYPE_='COV', and the _NAME_ variable labels the rows with the parameter names.

The names of the parameters are used as variable names. These are the same names that are used in the INIT, BOUNDS, and RESTRICT statements.

ODS Table Names

PROC COUNTREG assigns a name to each table that it creates. You can use these names to denote the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in Table 12.10.

Table 12.10 ODS Tables Produced in PROC COUNTREG

ODS Table Name	Description	Option
ODS Tables Created by the MODEL Statement		
ClassLevels	Class levels	Default
FitSummary	Summary of nonlinear estimation	Default
ConvergenceStatus	Convergence status	Default
ParameterEstimates	Parameter estimates	Default
CovB	Covariance of parameter estimates	COVB
CorrB	Correlation of parameter estimates	CORRB
InputOptions	Input options	ITPRINT
IterStart	Optimization start	ITPRINT
IterHist	Iteration history	ITPRINT
IterStop	Optimization results	ITPRINT
ParameterEstimatesResults	Parameter estimates	ITPRINT
ParameterEstimatesStart	Parameter estimates	ITPRINT
ProblemDescription	Problem description	ITPRINT
ODS Tables Created by the BAYES Statement		
AutoCorr	Autocorrelation statistics for each parameter	Default
AutoMcmcSummary	Automatic MCMC summary	DIAGNOSTICS=AUTOSUM
Corr	Correlation matrix of the posterior samples	STATS=COR

Table 12.10 *continued*

ODS Table Name	Description	Option
Cov	Covariance matrix of the posterior samples	STATS=COV
ESS	Effective sample size for each parameter	Default
MCSE	Monte Carlo standard error for each parameter	Default
Geweke	Geweke diagnostics for each parameter	Default
Heidelberger	Heidelberger-Welch diagnostics for each parameter	DIAGNOSTICS=HEIDEL
LogMarginLike	Marginal likelihood	MARGINLIKE
PostIntervals	Equal-tail and HPD intervals for each parameter	Default
PosteriorSample	Posterior samples	(ODS output data set only)
PostSummaries	Posterior summaries	Default
PriorSummaries	Prior summaries	STATS=PRIOR
Raftery	Raftery-Lewis diagnostics for each parameter	DIAGNOSTICS=RAFTERY

ODS Graphics

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*).

Before you create graphs, ODS Graphics must be enabled (for example, with the ODS GRAPHICS ON statement). For more information about enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" in that chapter.

The overall appearance of graphs is controlled by ODS styles. Styles and other aspects of using ODS Graphics are discussed in the section "A Primer on ODS Statistical Graphics" in that chapter.

This section describes the use of ODS Graphics to create graphics by using the COUNTREG procedure.

To request these graphs, you must specify the ODS GRAPHICS ON statement. There is no default plot for the COUNTREG procedure. If, in addition to the ODS GRAPHICS statement, you specify the ALL option in the PROC COUNTREG statement, then all applicable plots are created.

ODS Graph Names

PROC COUNTREG assigns a name to each graph that it creates using ODS. You can use these names to refer to the graphs when using ODS. The names are listed in Table 12.11.

Table 12.11 ODS Graphics Produced in PROC AUTOREG

ODS Table Name	Description	Plots= Option
PredProbPlot	Predictive probability plot	PLOTS(COUNTS)=PREDPROB
ProfileLikPlot	Profile likelihood functions	PLOTS(UNPACK)=PROFILELIKE or PROLIK
OverDispersion	Overdispersion diagnostic plot	PLOTS=DISPERSION
ZpProfilePlot	Zero-probability and zero-inflation profile plot	PLOTS(UNPACK)=ZEROPROFILE or ZPPRO
PredProfilePlot	Predictive probability profile plot	PLOTS(UNPACK COUNTS)=PREDPRO or PREDPROFILE

Table 12.12 Graphs Produced by PROC CONTREG When a BAYES Statement Is Included

ODS Graph Name	Plot Description	Statement and Option
Bayesian Diagnostic Plots		
ADPanel	Autocorrelation function and density panel	PLOTS=(AUTOCORR DENSITY)
AutocorrPanel	Autocorrelation function panel	PLOTS=AUTOCORR
AutocorrPlot	Autocorrelation function plot	PLOTS(UNPACK)=AUTOCORR
DensityPanel	Density panel	PLOTS=DENSITY
DensityPlot	Density plot	PLOTS(UNPACK)=DENSITY
TAPanel	Trace and autocorrelation function panel	PLOTS=(TRACE AUTOCORR)
TADPanel	Trace, density, and autocorrelation function panel	PLOTS=(TRACE AUTOCORR DENSITY) PLOTS=BAYESDIAG
TDPanel	Trace and density panel	PLOTS=(TRACE DENSITY)
TracePanel	Trace panel	PLOTS=TRACE
TracePlot	Trace plot	PLOTS(UNPACK)=TRACE
Bayesian Summary Plots		
BayesSumPlot	Prior/posterior densities and MLE	PLOTS=BAYESSUM

Examples: COUNTREG Procedure

Example 12.1: Basic Models

Data Description and Objective

The data set DocVisit contains information for approximately 5,000 Australian individuals about the number and possible determinants of doctor visits that were made during a two-week interval. This data set contains a subset of variables that are taken from the Racd3 data set used by Cameron and Trivedi (1998). The DocVisit data set can be found in the SAS/ETS Sample Library.

The variable Doctorco represents doctor visits. Additional variables in the data set that you want to evaluate as determinants of doctor visits include Sex (coded 0=male, 1=female), Age (age in years divided by 100), Illness (number of illnesses during the two-week interval, with five or more coded as five), Income (annual income in Australian dollars divided by 1,000), and Hscore (a score on a general health questionnaire, in which a high score indicates bad health). Summary statistics for these variables are computed in the following statements and presented in Output 12.1.1.

```
proc means data=docvisit;
   var doctorco sex age illness income hscore;
run;
```

Output 12.1.1 Summary Statistics

The MEANS Procedure

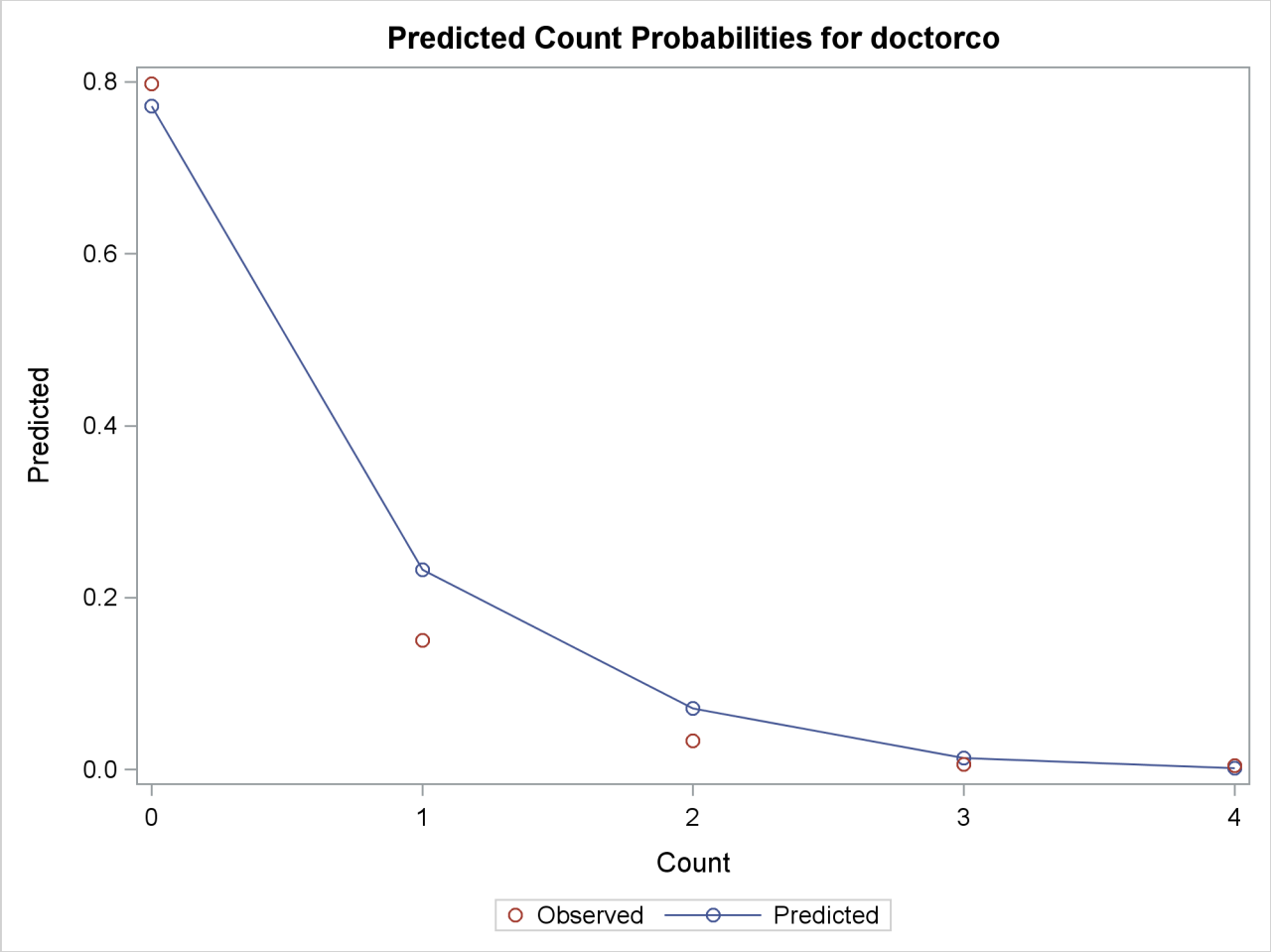
Variable	N	Mean	Std Dev	Minimum	Maximum
doctorco	5190	0.3017341	0.7981338	0	9.0000000
sex	5190	0.5206166	0.4996229	0	1.0000000
age	5190	0.4063854	0.2047818	0.1900000	0.7200000
illness	5190	1.4319846	1.3841524	0	5.0000000
income	5190	0.5831599	0.3689067	0	1.5000000
hscore	5190	1.2175337	2.1242665	0	12.0000000

Poisson Model

The following statements fit a Poisson model to the data by using the covariates Sex, Illness, Income, and Hscore:

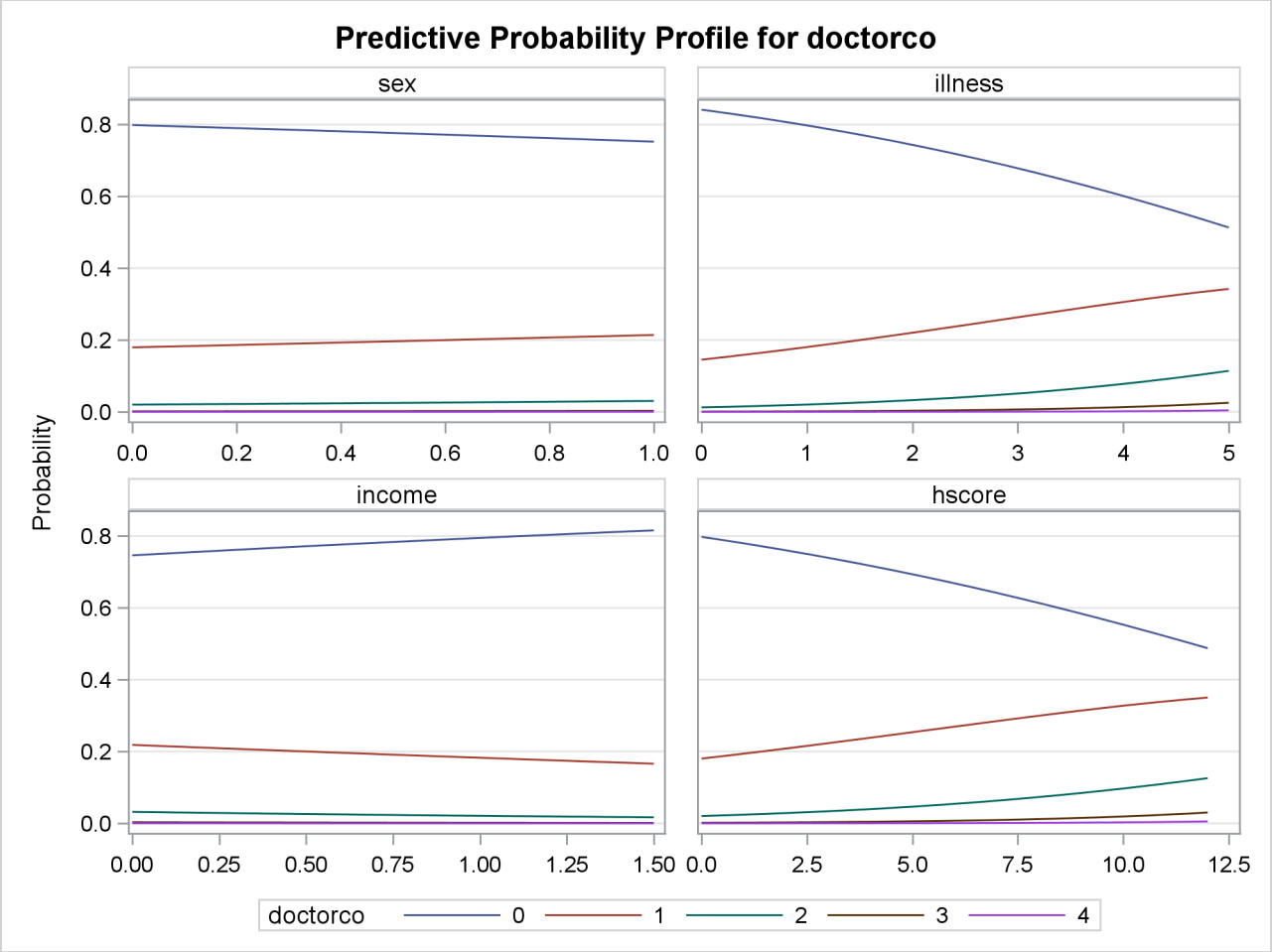
```
proc countreg data=docvisit plots(only counts(0 to 4 by 1))=(predprob predpro);
   model doctorco=sex illness income hscore / dist=poisson printall;
run;
```

Output 12.1.2 Mean Predicted Count Probabilities



Output 12.1.2 shows the predicted probabilities of count levels 0 to 4 from the Poisson model. Most of the observed counts are in the range 0 to 4 and account for more than 99% of the entire data set. One factor that would be interesting to explore is how the model-predicted probabilities of those count levels react to different regressor values. Output 12.1.3 shows the predictive profiles of the count levels in question against the first three regressors in the model. In each panel, the regressor in question is varied while all other regressors are fixed at their observed mean and the model parameters are fixed at their MLE.

Output 12.1.3 Profile Function of Predictive Probabilities



In this example, the DIST= option in the MODEL statement specifies the Poisson distribution. In addition, the PRINTALL option displays the correlation and covariance matrices for the parameters, log-likelihood values, and convergence information in addition to the parameter estimates. The parameter estimates for this model are shown in Output 12.1.4.

Output 12.1.4 Parameter Estimates of Poisson Model

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	-1.855552	0.074545	-24.89	<.0001
sex	1	0.235583	0.054362	4.33	<.0001
illness	1	0.270326	0.017080	15.83	<.0001
income	1	-0.242095	0.077829	-3.11	0.0019
hscore	1	0.096313	0.009089	10.60	<.0001

Using the CLASS Statement

If some regressors are categorical in nature (meaning that these variables can take only a few discrete qualitative values), specify them in the CLASS statement. In this example, Sex is categorical because it takes only two values. A CLASS variable can be numeric or character.

Consider the following extension:

```
proc countreg data=docvisit;
   class sex;
   model doctorco=sex illness income hscore / dist=poisson;
run;
```

The partial output is given in Output 12.1.5.

Output 12.1.5 Parameter Estimates of Poisson Model with CLASS statement

The COUNTREG Procedure

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	-1.619969	0.063985	-25.32	<.0001
sex 0	1	-0.235583	0.054362	-4.33	<.0001
sex 1	0	0	.	.	.
illness	1	0.270326	0.017080	15.83	<.0001
income	1	-0.242095	0.077829	-3.11	0.0019
hscore	1	0.096313	0.009089	10.60	<.0001

If the CLASS statement is present, the COUNTREG procedure creates as many indicator or dummy variables as there are categories in a CLASS variable and uses them as independent variables. In order to avoid collinearity with the intercept, the last-created dummy variable is assigned a zero coefficient by default. This means that only the dummy variable that is associated with the first level of Sex (male=0) is used as a regressor. Consequently, the estimated coefficient for this dummy variable is the negative of the one for the original Sex variable in Output 12.1.4, because the reference level has switched from male to female.

Now consider a more practical task. The previous example implicitly assumes that each additional illness during the two-week interval has the same effect. In other words, this variable is thought of as a continuous variable. But this variable has only six values, and it is quite possible that the number of illnesses has a nonlinear effect on doctor visits. In order to check this conjecture, the following statements specify the Illness variable in the CLASS statement so that it is represented in the model by a set of six dummy variables that can account for any type of nonlinearity:

```
proc countreg data=docvisit;
   class sex illness;
   model doctorco=sex illness income hscore / dist=poisson;
run;
```

The parameter estimates are displayed in Output 12.1.6.

Output 12.1.6 Parameter Estimates of Poisson Model with CLASS statement

The COUNTREG Procedure

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	-0.385930	0.088062	-4.38	<.0001
sex 0	1	-0.219118	0.054190	-4.04	<.0001
sex 1	0	0	.	.	.
illness 0	1	-1.934983	0.121267	-15.96	<.0001
illness 1	1	-0.698307	0.089732	-7.78	<.0001
illness 2	1	-0.471100	0.090742	-5.19	<.0001
illness 3	1	-0.488481	0.099127	-4.93	<.0001
illness 4	1	-0.272372	0.107593	-2.53	0.0114
illness 5	0	0	.	.	.
income	1	-0.253583	0.077441	-3.27	0.0011
hscore	1	0.094590	0.009025	10.48	<.0001

The Estimate column shows the difference between each ILLNESS parameter and ILLNESS=5. Note that these estimates for different Illness categories do not increase linearly but instead show a relatively large jump from zero illnesses to one illness, followed by relatively smaller increases.

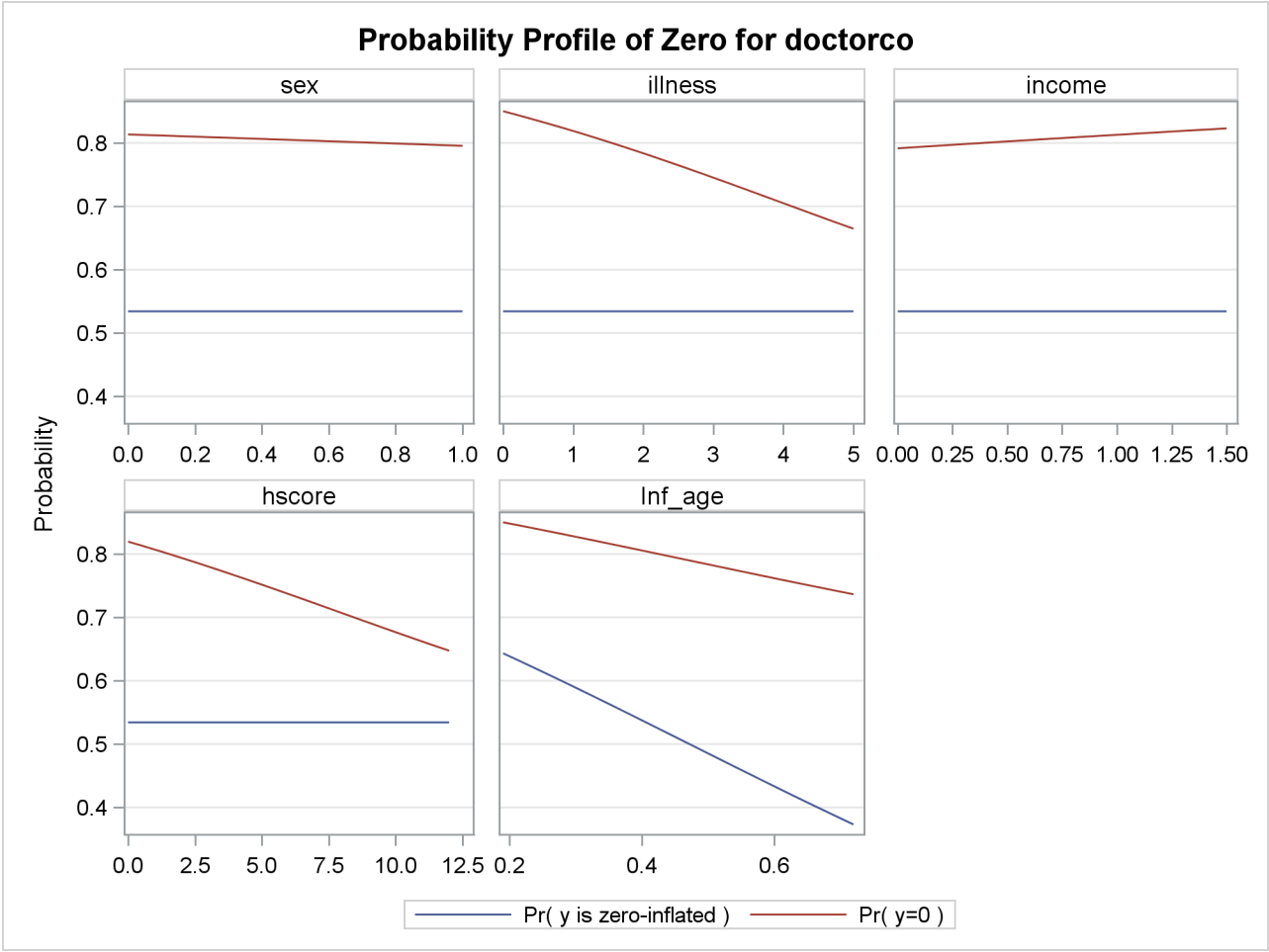
Zero-Inflated Poisson (ZIP) Model

Suppose you suspect that the population of individuals can be viewed as two distinct groups: a low-risk group, consisting of individuals who never go to the doctor, and a high-risk group, consisting of individuals who do go to the doctor. You might suspect that the data have this structure both because the sample variance of Doctorco (0.64) exceeds its sample mean (0.30), which suggests overdispersion, and because a large fraction of the Doctorco observations (80%) have the value zero. Estimating a zero-inflated model is one way to deal with overdispersion that results from excess zeros.

Suppose also that you suspect that the covariate Age has an impact on whether an individual belongs to the low-risk group. For example, younger individuals might have illnesses of much lower severity when they do get sick and be less likely to visit a doctor, all other factors being equal. The following statements estimate a zero-inflated Poisson regression, with Age as a covariate in the zero-generation process:

```
proc countreg data=docvisit plots(only)=(dispersion zeroprofile);
   model doctorco=sex illness income hscore / dist=zip;
   zeromodel doctorco ~ age;
run;
```

Output 12.1.7 Profile Function of Zero Process Selection and Zero Prediction



You might be interested in exploring how the zero process selection probability reacts to different regressor values. Output 12.1.7 displays this information in much the same fashion as Output 12.1.3. Because Sex, Illness, Income, and Hscore do not appear in the ZEROMODEL statement, the zero-inflation selection probability does not change for different values of those regressors. However, the plot shows that Age positively affects the zero process selection probability in a linear fashion.

In this case, the ZEROMODEL statement that follows the MODEL statement specifies that both an intercept and the variable Age be used to estimate the likelihood of zero doctor visits. Output 12.1.8 shows the resulting parameter estimates.

Output 12.1.8 Parameter Estimates for ZIP Model

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	-1.033387	0.096973	-10.66	<.0001
sex	1	0.122511	0.062566	1.96	0.0502
illness	1	0.237478	0.019997	11.88	<.0001
income	1	-0.143945	0.087810	-1.64	0.1012
hscore	1	0.088386	0.010043	8.80	<.0001
Inf_Intercept	1	0.986557	0.131339	7.51	<.0001
Inf_age	1	-2.090924	0.270580	-7.73	<.0001

The estimates of the zero-inflated intercept (Inf_Intercept) and the zero-inflated regression coefficient for Age (Inf_age) are approximately 0.99 and –2.09, respectively. Because the zero-inflation model uses a logistic link by default, you can estimate the probabilities for individuals of ages 20, 50, and 70 as follows:

$$\text{20 years: } \frac{e^{(0.99-2.09*.20)}}{1+e^{(0.99-2.09*.20)}} = 0.64$$

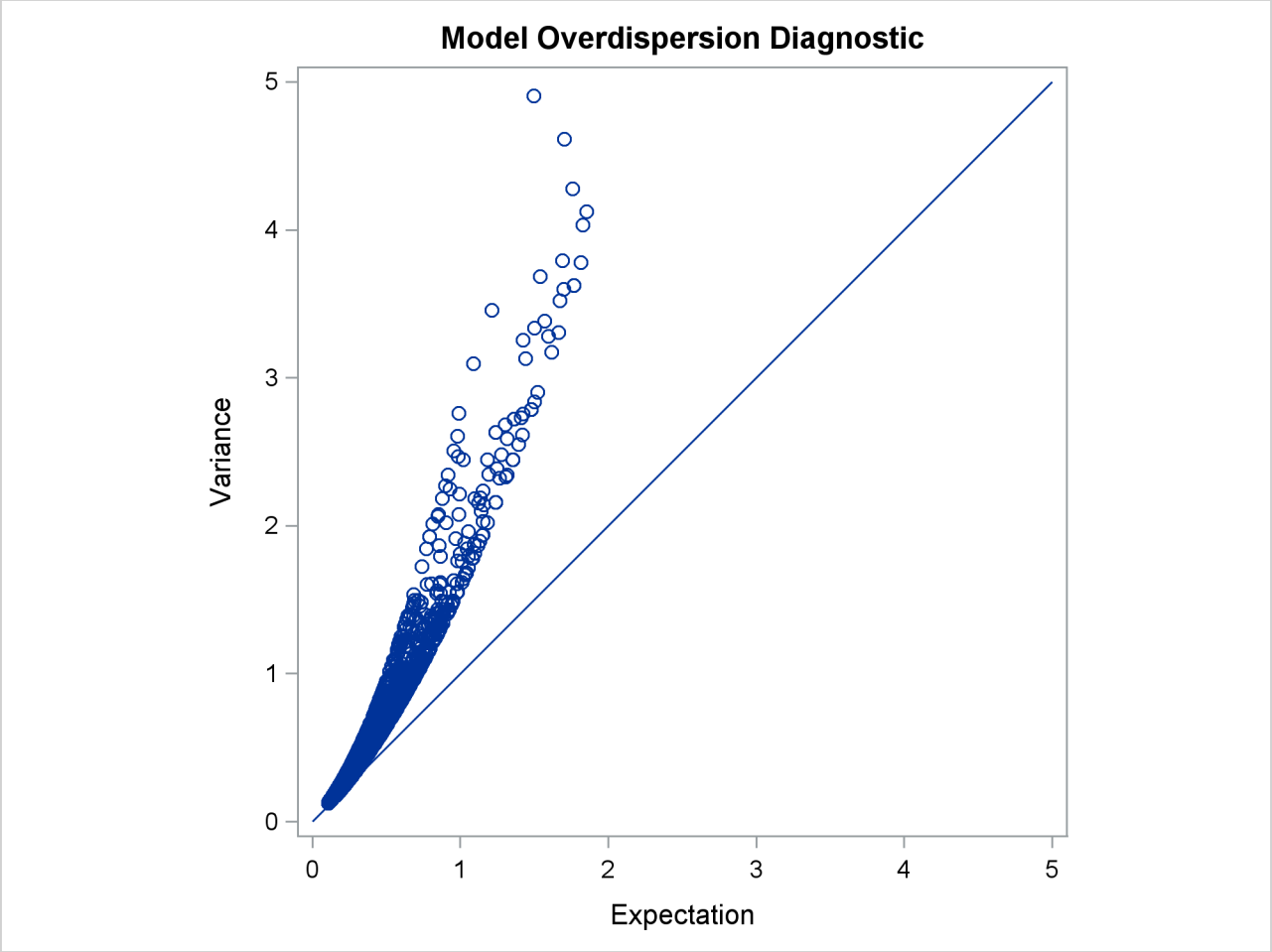
$$\text{50 years: } \frac{e^{(0.99-2.09*.50)}}{1+e^{(0.99-2.09*.50)}} = 0.49$$

$$\text{70 years: } \frac{e^{(0.99-2.09*.70)}}{1+e^{(0.99-2.09*.70)}} = 0.38$$

That is, the estimated probability of belonging to the low-risk group is about 0.64 for a 20-year-old individual, 0.49 for a 50-year-old individual, and only 0.38 for a 70-year-old individual. This supports the suspicion that older individuals are more likely to have a positive number of doctor visits.

Alternative models to account for the overdispersion are the negative binomial and the zero-inflated negative binomial models, which can be fit using the DIST=NEGBIN and DIST=ZINB options, respectively.

Output 12.1.9 Overdispersion Diagnostic Plot



Output 12.1.9 plots the conditional variance against the conditional mean and can be used as a diagnostic tool to check the level of overdispersion in a model.

Example 12.2: ZIP and ZINB Models for Data That Exhibit Extra Zeros

In the study by Long (1997) of the number of published articles by scientists (see the section "Getting Started: COUNTREG Procedure" on page 584), the observed proportion of scientists who publish no articles is 0.3005. The following statements use PROC FREQ to compute the proportion of scientists who publish each observed number of articles. Output 12.2.1 shows the results.

```
proc freq data=long97data;
   table art / out=obs;
run;
```

Output 12.2.1 Proportion of Scientists Who Publish a Certain Number of Articles

The FREQ Procedure

art	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	275	30.05	275	30.05
1	246	26.89	521	56.94
2	178	19.45	699	76.39
3	84	9.18	783	85.57
4	67	7.32	850	92.90
5	27	2.95	877	95.85
6	17	1.86	894	97.70
7	12	1.31	906	99.02
8	1	0.11	907	99.13
9	2	0.22	909	99.34
10	1	0.11	910	99.45
11	1	0.11	911	99.56
12	2	0.22	913	99.78
16	1	0.11	914	99.89
19	1	0.11	915	100.00

PROC COUNTREG is then used to fit Poisson and negative binomial models to the data. For each model, the PROBCOUNT option computes the probability that the number of published articles is m, for $m = 0$ to 10. The following statements compute the estimates for Poisson and negative binomial models. The MEAN procedure is then used to compute the average probability of a zero response.

```
proc countreg data=long97data;
   model art=fem mar kid5 phd ment / dist=poisson;
   output out=predpoi probcount(0 to 10);
run;

proc means mean data=predpoi;
   var p_0;
run;
```

The output from the Poisson model for the COUNTREG and MEAN procedures is shown in Output 12.2.2.

Output 12.2.2 Poisson Model Estimation

The COUNTREG Procedure

Model Fit Summary	
Dependent Variable	art
Number of Observations	915
Data Set	WORK.LONG97DATA
Model	Poisson
Log Likelihood	-1651
Maximum Absolute Gradient	3.57454E-9
Number of Iterations	5
Optimization Method	Newton-Raphson
AIC	3314
SBC	3343

Algorithm converged.

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	0.304617	0.102982	2.96	0.0031
fem	1	-0.224594	0.054614	-4.11	<.0001
mar	1	0.155243	0.061375	2.53	0.0114
kid5	1	-0.184883	0.040127	-4.61	<.0001
phd	1	0.012823	0.026397	0.49	0.6271
ment	1	0.025543	0.002006	12.73	<.0001

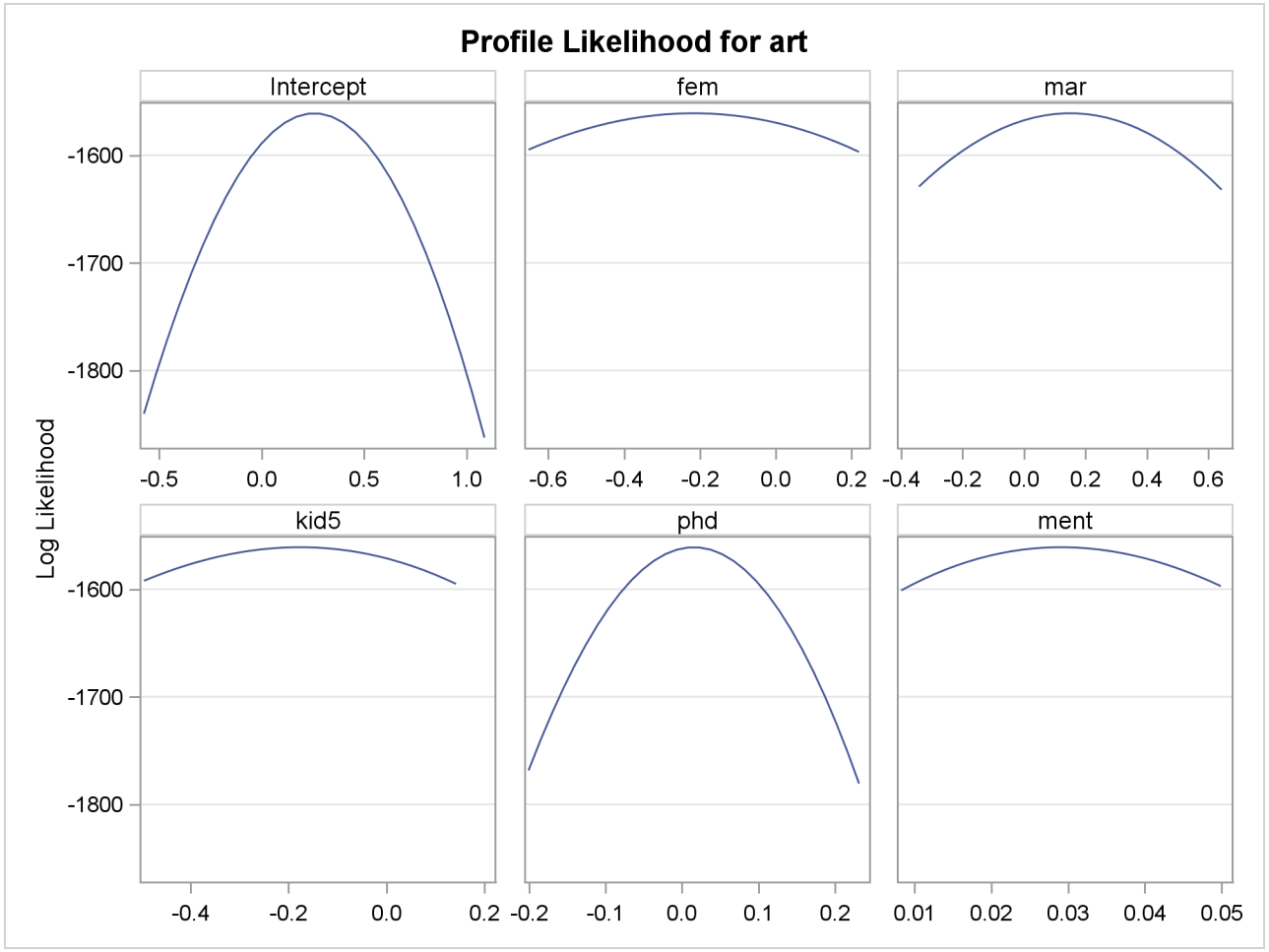
The MEANS Procedure

Analysis Variable : P_0 Probability of art taking level=0
Mean
0.2092071

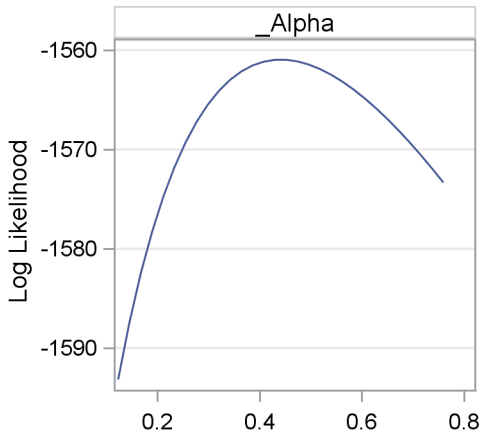
The following statements show the syntax for the negative binomial model:

```
proc countreg data=long97data plots(only)=profilelike;
   model art=fem mar kid5 phd ment / dist=negbin(p=2) method=qn;
   output out=prednb probcount(0 to 10);
run;

proc means mean data=prednb;
   var p_0;
run;
```

Output 12.2.3 Profile Likelihood Functions



Output 12.2.3 *continued*



Output 12.2.3 show the profile likelihood functions of the negative binomial model for the Long (1997) data set, in which each model parameter is varied while holding all others fixed at the MLE. This can serve as a diagnostic tool for model performance, because a large number of flat profile likelihood functions indicates poor optimization results and the resulting MLE should be used with caution.

Output 12.2.4 shows the results of the preceding statements.

Output 12.2.4 Negative Binomial Model Estimation

The COUNTREG Procedure

Model Fit Summary	
Dependent Variable	art
Number of Observations	915
Data Set	WORK.LONG97DATA
Model	NegBin(p=2)
Log Likelihood	-1561
Maximum Absolute Gradient	5.72129E-7
Number of Iterations	16
Optimization Method	Quasi-Newton
AIC	3136
SBC	3170

Algorithm converged.

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	0.256144	0.138560	1.85	0.0645
fem	1	-0.216418	0.072672	-2.98	0.0029
mar	1	0.150489	0.082106	1.83	0.0668
kid5	1	-0.176415	0.053060	-3.32	0.0009
phd	1	0.015271	0.036040	0.42	0.6718
ment	1	0.029082	0.003470	8.38	<.0001
_Alpha	1	0.441620	0.052967	8.34	<.0001

The MEANS Procedure

Analysis Variable : P_0 Probability of art taking level=0
Mean
0.3035957

For each model, the predicted proportion of zero articles can be calculated as the average predicted probability of zero articles across all scientists. Under the Poisson model, the predicted proportion of zero articles is 0.2092, which considerably underestimates the observed proportion. The negative binomial more closely estimates the proportion of zeros (0.3036). Also, the test of the dispersion parameter, _Alpha, in the negative binomial model indicates significant overdispersion ($p < 0.0001$). As a result, the negative binomial model is preferred to the Poisson model.

Another way to account for the large number of zeros in this data set is to fit a zero-inflated Poisson (ZIP) or a zero-inflated negative binomial (ZINB) model. In the following statements, DIST=ZIP requests the ZIP model. In the ZEROMODEL statement, you can specify the predictors, $\mathbf{z}$, for the process that generates the additional zeros. The ZEROMODEL statement also specifies the model for the probability φ. By default, a logistic model is used for φ. You can change the default by using the LINK= option. In this particular ZIP model, all variables that are used to model the article counts are also used to model φ.

```
proc countreg data=long97data;
   model art = fem mar kid5 phd ment / dist=zip;
   zeromodel art ~ fem mar kid5 phd ment;
   output out=predzip probcount(0 to 10);
run;

proc means data=predzip mean;
   var p_0;
run;
```

The parameters of the ZIP model are displayed in Output 12.2.5. The first set of parameters gives the estimates of $\boldsymbol{\beta}$ in the model for the Poisson process mean. Parameters that have the prefix "Inf_" are the estimates of $\boldsymbol{\gamma}$ in the logistic model for φ.

Output 12.2.5 ZIP Model Estimation

The COUNTREG Procedure

Model Fit Summary	
Dependent Variable	art
Number of Observations	915
Data Set	WORK.LONG97DATA
Model	ZIP
ZI Link Function	Logistic
Log Likelihood	-1605
Maximum Absolute Gradient	2.08804E-7
Number of Iterations	16
Optimization Method	Newton-Raphson
AIC	3234
SBC	3291

Algorithm converged.

Output 12.2.5 *continued*

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	0.640838	0.121306	5.28	<.0001
fem	1	-0.209145	0.063405	-3.30	0.0010
mar	1	0.103751	0.071111	1.46	0.1446
kid5	1	-0.143320	0.047429	-3.02	0.0025
phd	1	-0.006166	0.031008	-0.20	0.8424
ment	1	0.018098	0.002295	7.89	<.0001
Inf_Intercept	1	-0.577060	0.509383	-1.13	0.2573
Inf_fem	1	0.109747	0.280082	0.39	0.6952
Inf_mar	1	-0.354013	0.317611	-1.11	0.2650
Inf_kid5	1	0.217101	0.196481	1.10	0.2692
Inf_phd	1	0.001272	0.145262	0.01	0.9930
Inf_ment	1	-0.134114	0.045244	-2.96	0.0030

The MEANS Procedure

Analysis Variable : P_0 Probability of art taking level=0
Mean
0.2985679

The proportion of zeros that are predicted by the ZIP model is 0.2986, which is much closer to the observed proportion than the Poisson model. But Output 12.2.7 shows that both models deviate from the observed proportions at one, two, and three articles.

The ZINB model is specified by the DIST=ZINB option. All variables are again used to model both the number of articles and φ. The METHOD=QN option specifies that the quasi-Newton method be used to fit the model rather than the default Newton-Raphson method. These options are implemented in the following statements:

```
proc countreg data=long97data;
   model art=fem mar kid5 phd ment / dist=zinb method=qn;
   zeromodel art ~ fem mar kid5 phd ment;
   output out=predzinb probcount(0 to 10);
run;

proc means data=predzinb mean;
   var p_0;
run;
```

The estimated parameters of the ZINB model are shown in Output 12.2.6. The test for overdispersion again indicates a preference for the negative binomial version of the zero-inflated model ($p < 0.0001$). The ZINB model also does a good job of estimating the proportion of zeros (0.3119), and it follows the observed proportions well, though possibly not as well as the negative binomial model.

Output 12.2.6 ZINB Model Estimation

The COUNTREG Procedure

Model Fit Summary	
Dependent Variable	art
Number of Observations	915
Data Set	WORK.LONG97DATA
Model	ZINB
ZI Link Function	Logistic
Log Likelihood	-1550
Maximum Absolute Gradient	0.00385
Number of Iterations	81
Optimization Method	Quasi-Newton
AIC	3126
SBC	3189

Algorithm converged.

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	0.416749	0.143596	2.90	0.0037
fem	1	-0.195506	0.075592	-2.59	0.0097
mar	1	0.097582	0.084452	1.16	0.2479
kid5	1	-0.151731	0.054206	-2.80	0.0051
phd	1	-0.000700	0.036270	-0.02	0.9846
ment	1	0.024786	0.003493	7.10	<.0001
Inf_Intercept	1	-0.191646	1.322782	-0.14	0.8848
Inf_fem	1	0.635937	0.848890	0.75	0.4538
Inf_mar	1	-1.499456	0.938639	-1.60	0.1102
Inf_kid5	1	0.628429	0.442773	1.42	0.1558
Inf_phd	1	-0.037726	0.308001	-0.12	0.9025
Inf_ment	1	-0.882288	0.316217	-2.79	0.0053
_Alpha	1	0.376681	0.051029	7.38	<.0001

The MEANS Procedure

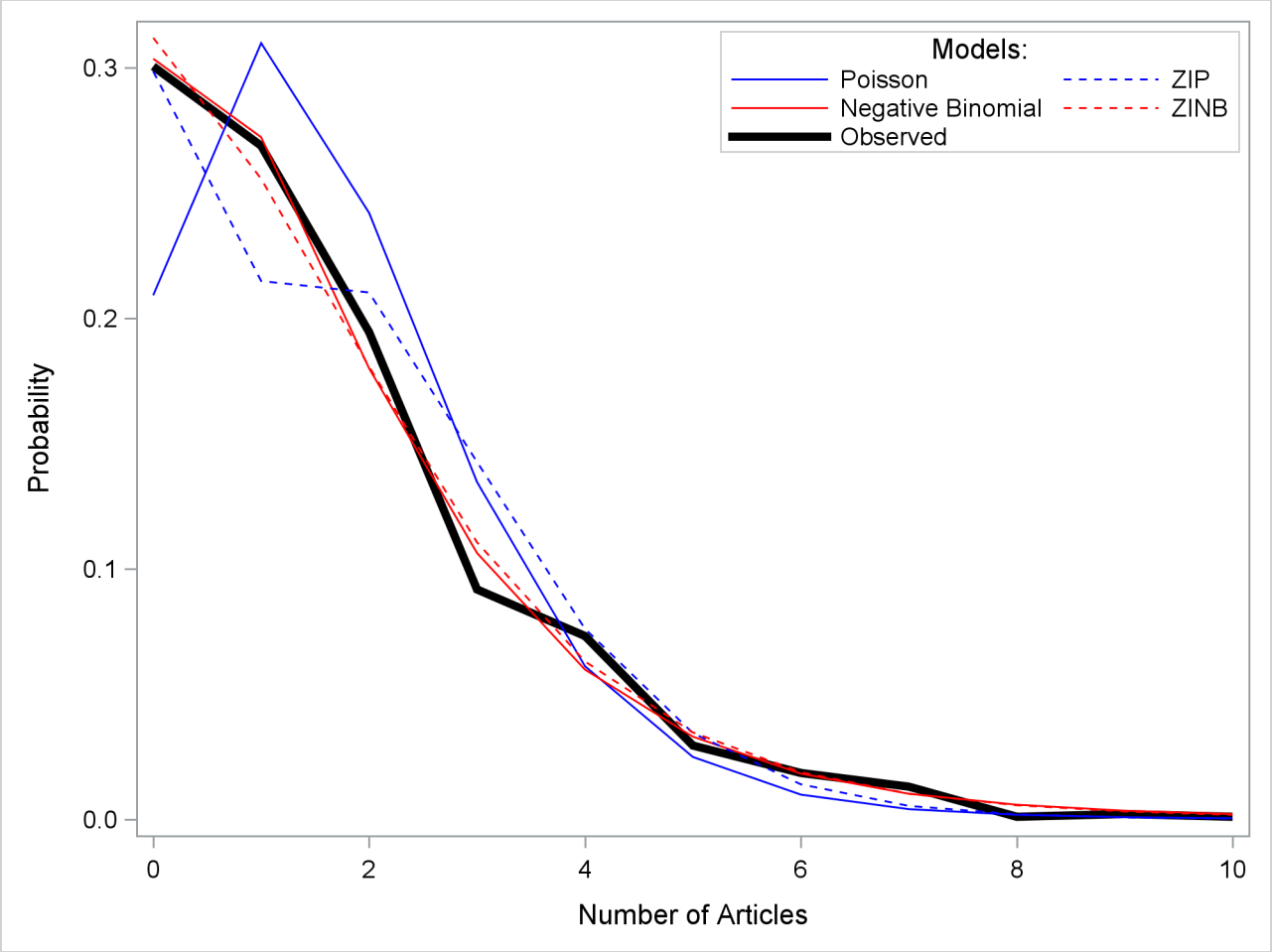
Analysis Variable : P_0 Probability of art taking level=0
Mean
0.3119491

The following statements compute the average predicted count probability across all scientists for each count 0, 1, . . . , 10. The averages for each model, along with the observed proportions, are then arranged for plotting by PROC SGPLOT.

```
proc summary data=predpoi;
   var p_0-p_10;
   output out=mnpoi mean(p_0-p_10)=mn0-mn10;
run;
proc summary data=prednb;
   var p_0-p_10;
   output out=mnnb mean(p_0-p_10)=mn0-mn10;
run;
proc summary data=predzip;
   var p_0-p_10;
   output out=mnzip mean(p_0-p_10)=mn0-mn10;
run;
proc summary data=predzinb;
   var p_0-p_10;
   output out=mnzinb mean(p_0-p_10)=mn0-mn10;
run;


data means;
   set mnpoi mnnb mnzip mnzinb;
   drop _type_ _freq_;
run;

proc transpose data=means out=tmeans;
run;

data allpred;
   merge obs(where=(art<=10)) tmeans;
   obs=percent/100;
run;

proc sgplot;
   yaxis label='Probability';
   xaxis label='Number of Articles';
   series y=obs  x=art / name='obs' legendlabel='Observed'
      lineattrs=(color=black thickness=4px);
   series y=col1 x=art / name='poi' legendlabel='Poisson'
      lineattrs=(color=blue);
   series y=col2 x=art/ name='nb' legendlabel='Negative Binomial'
      lineattrs=(color=red);
   series y=col3 x=art/ name='zip' legendlabel='ZIP'
      lineattrs=(color=blue pattern=2);
   series y=col4 x=art/ name='zinb' legendlabel='ZINB'
      lineattrs=(color=red pattern=2);
   discretelegend 'poi' 'zip' 'nb' 'zinb' 'obs' / title='Models:'
      location=inside position=ne across=2 down=3;
run;
```

For each of the four fitted models, Output 12.2.7 shows the average predicted count probability for each article count across all scientists. The Poisson model clearly underestimates the proportion of zero articles published, whereas the other three models are quite accurate at zero. All the models do well at the larger numbers of articles.

Output 12.2.7 Average Predicted Count Probability



Example 12.3: Variable Selection

This example demonstrates two algorithms of automatic variable selection in the COUNTREG procedure. Automatic variable selection is most effective when the number of possible candidates for explaining the variation of some variable is large. For clarity of exposition, this example uses only a small number of variables. The data set Article published by Long (1997) contains six variables. (This data set is also used in "Example 12.2: ZIP and ZINB Models for Data That Exhibit Extra Zeros" on page 680.) The dependent variable Art records the number of articles that were published by a doctoral student in the last three years of his or her program. Explanatory variables include sex of the student (Fem), marital status (Mar), number of children (Kid5), prestige of the program (Phd), and publishing activity of the academic adviser (Ment). All these variables intuitively suggest their affect on the students' primary academic output.

First, for comparison purposes, estimate the simple Poisson model. The choice of model is specified by DIST= option in the MODEL statement, as follows:

```
proc countreg data = long97data;
   model art = fem mar kid5 phd ment / dist = poisson;
run;
```

The output of these statements is shown in Figure 12.3.1.

Output 12.3.1 Poisson Model for the Number of Published Articles

The COUNTREG Procedure

Model Fit Summary	
Dependent Variable	art
Number of Observations	915
Data Set	WORK.LONG97DATA
Model	Poisson
Log Likelihood	-1651
Maximum Absolute Gradient	3.57454E-9
Number of Iterations	5
Optimization Method	Newton-Raphson
AIC	3314
SBC	3343

Algorithm converged.

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	0.304617	0.102982	2.96	0.0031
fem	1	-0.224594	0.054614	-4.11	<.0001
mar	1	0.155243	0.061375	2.53	0.0114
kid5	1	-0.184883	0.040127	-4.61	<.0001
phd	1	0.012823	0.026397	0.49	0.6271
ment	1	0.025543	0.002006	12.73	<.0001

Note that the Newton-Raphson optimization algorithm took five steps to converge. All parameters, except for one, are significant at a 1% or 5% level, whereas Phd is not significant even at the 10% level.

In this case, it might be easy to identify the variables that have limited explanatory power. However, if the number of variables were large, the manual selection could be time-consuming and inaccurate. For a large number of variables, you would be better off in applying one of the automatic algorithms of variable selection. The following statements use the penalized likelihood method, which is indicated by SELECT=PEN option in the MODEL statement:

```
proc countreg data = long97data method = qn;
   model art = fem mar kid5 phd ment / dist   = poisson
                                       select = PEN;
run;
```

The output of these statements is shown in Output 12.3.2.

Output 12.3.2 Poisson Model for the Number of Published Articles with Penalized Likelihood Method

The COUNTREG Procedure

Model Fit Summary	
Dependent Variable	art
Number of Observations	915
Data Set	WORK.LONG97DATA
Model	Poisson
Log Likelihood	-1651
Maximum Absolute Gradient	6.66114E-6
Number of Iterations	7
Optimization Method	Quasi-Newton
AIC	3312
SBC	3336

Algorithm converged.

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	0.345174	0.060125	5.74	<.0001
fem	1	-0.225303	0.054615	-4.13	<.0001
mar	1	0.152175	0.061067	2.49	0.0127
kid5	1	-0.184993	0.040139	-4.61	<.0001
ment	1	0.025761	0.001950	13.21	<.0001

The "Parameter Estimates" table shows that the variable Phd was dropped from the model.

The next statements use the information criterion by specifying the SELECT=INFO option. The direction of the search is chosen to be forward, and the information criterion is AIC. In order to achieve the same selection of variables as for the penalized likelihood method, 0.001 is specified for the percentage of decrease in the information criterion necessary for the algorithm to stop.

```
proc countreg data = long97data;
   model art = fem mar kid5 phd ment / dist      = poisson
                                       select    = INFO
                                     ( direction = forward
                                       criter    = AIC
                                       lstop     = 0.001 );
run;
```

The output of these statements is shown in Figure 12.3.3.

Output 12.3.3 Poisson Model for the Number of Published Articles with Search Method Using Information Criterion

The COUNTREG Procedure

Variable Selection Information

Step	Effect Entered	Effect Removed	AIC	SBC
0	Base Model		3487.146950	3491.965874
1	ment		3341.286487	3350.924335
2	fem		3330.744604	3345.201376
3	kid5		3316.593036	3335.868733
4	mar		3312.348824	3336.443445

Model Fit Summary

Dependent Variable	art
Number of Observations	915
Data Set	WORK.LONG97DATA
Model	Poisson
Log Likelihood	-1651
Maximum Absolute Gradient	1.28369E-9
Number of Iterations	0
Optimization Method	Newton-Raphson
AIC	3312
SBC	3336

Algorithm converged.

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	0.345174	0.060125	5.74	<.0001
fem	1	-0.225303	0.054615	-4.13	<.0001
mar	1	0.152175	0.061067	2.49	0.0127
kid5	1	-0.184993	0.040139	-4.61	<.0001
ment	1	0.025761	0.001950	13.21	<.0001

From the output, it is clear that the same set of variables was chosen as the result of information criterion algorithm. Note that the forward optimization algorithm starts with the constant as the only explanatory variable.

References

Abramowitz, M., and Stegun, I. A., eds. (1970). *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.* 9th printing. New York: Dover.

Amemiya, T. (1985). *Advanced Econometrics*. Cambridge, MA: Harvard University Press.

Cameron, A. C., and Trivedi, P. K. (1986). "Econometric Models Based on Count Data: Comparisons and Applications of Some Estimators and Tests." *Journal of Applied Econometrics* 1:29–53.

Cameron, A. C., and Trivedi, P. K. (1998). *Regression Analysis of Count Data*. Cambridge: Cambridge University Press.

Chan, J. C. C., and Eisenstat, E. (2015). "Marginal Likelihood Estimation with the Cross-Entropy Method." *Econometric Reviews* 34:256–285.

Conway, R. W., and Maxwell, W. L. (1962). "A Queuing Model with State Dependent Service Rates." *Journal of Industrial Engineering* 12:132–136.

Fan, J., and Li, R. (2001). "Variable Selection via Nonconcave Penalized Likelihood and Its Oracle Properties." *Journal of the American Statistical Association* 96:1348–1360.

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (2004). *Bayesian Data Analysis*. 2nd ed. London: Chapman & Hall.

Godfrey, L. G. (1988). *Misspecification Tests in Econometrics*. Cambridge: Cambridge University Press.

Greene, W. H. (1994). "Accounting for Excess Zeros and Sample Selection in Poisson and Negative Binomial Regression Models." Working Paper 94-10, Department of Economics, Leonard N. Stern School of Business, New York University. http://ideas.repec.org/p/ste/nystbu/94-10.html.

Greene, W. H. (2000). *Econometric Analysis*. 4th ed. Upper Saddle River, NJ: Prentice-Hall.

Guikema, S. D., and Coffelt, J. P. (2008). "A Flexible Count Data Regression Model for Risk Analysis." *Risk Analysis* 28:213–223.

Hausman, J. A., Hall, B. H., and Griliches, Z. (1984). "Econometric Models for Count Data with an Application to the Patents-R&D Relationship." *Econometrica* 52:909–938.

King, G. (1989a). "A Seemingly Unrelated Poisson Regression Model." *Sociological Methods and Research* 17:235–255.

King, G. (1989b). *Unifying Political Methodology: The Likelihood Theory and Statistical Inference*. Cambridge: Cambridge University Press.

Lambert, D. (1992). "Zero-Inflated Poisson Regression with an Application to Defects in Manufacturing." *Technometrics* 34:1–14.

LaMotte, L. R. (1994). "A Note on the Role of Independence in *t* Statistics Constructed from Linear Statistics in Regression Models." *American Statistician* 48:238–240.

Long, J. S. (1997). *Regression Models for Categorical and Limited Dependent Variables*. Thousand Oaks, CA: Sage Publications.

Lord, D., Guikema, S. D., and Geedipally, S. R. (2008). "Application of the Conway-Maxwell-Poisson Generalized Linear Model for Analyzing Motor Vehicle Crashes." *Accident Analysis and Prevention* 40:1123–1134.

Park, M. Y., and Hastie, T. J. (2007). "$l_1$1-Regularization Path Algorithm for Generalized Linear Models." *Journal of the Royal Statistical Society, Series B* 69:659–677.

Roberts, G. O., Gelman, A., and Gilks, W. R. (1997). "Weak Convergence and Optimal Scaling of Random Walk Metropolis Algorithms." *Annals of Applied Probability* 7:110–120.

Roberts, G. O., and Rosenthal, J. S. (2001). "Optimal Scaling for Various Metropolis-Hastings Algorithms." *Statistical Science* 16:351–367.

Schervish, M. J. (1995). *Theory of Statistics*. New York: Springer-Verlag.

Searle, S. R. (1971). *Linear Models*. New York: John Wiley & Sons.

Shmueli, G., Minka, T. P., Kadane, J. B., Borle, S., and Boatwright, P. (2005). "A Useful Distribution for Fitting Discrete Data: Revival of the Conway-Maxwell-Poisson Distribution." *Journal of the Royal Statistical Society, Series C* 54:127–142.

Winkelmann, R. (2000). *Econometric Analysis of Count Data*. Berlin: Springer-Verlag.

Zou, H., and Li, R. (2008). "One-Step Sparse Estimates in Nonconcave Penalized Likelihood Models." *Annals of Statistics* 36:1509–1533.

Chapter 13
The DATASOURCE Procedure

Contents

Overview: DATASOURCE Procedure **698**
Getting Started: DATASOURCE Procedure **700**
Structure of a SAS Data Set Containing Time Series Data 700
Reading Data Files 700
Subsetting Input Data Files 701
Controlling the Frequency of Data – The INTERVAL= Option 701
Selecting Time Series Variables – The KEEP and DROP Statements 701
Controlling the Time Range of Data – The RANGE Statement 704
Reading in Data Files Containing Cross Sections 704
Obtaining Descriptive Information on Cross Sections 705
Subsetting a Data File Containing Cross Sections 709
Renaming Time Series Variables 710
Changing the Lengths of Numeric Variables 711
Syntax: DATASOURCE Procedure **712**
PROC DATASOURCE Statement 714
KEEP Statement 716
DROP Statement 717
KEEPEVENT Statement 718
DROPEVENT Statement 718
WHERE Statement 719
RANGE Statement 719
ATTRIBUTE Statement 720
FORMAT Statement 721
LABEL Statement 721
LENGTH Statement 721
RENAME Statement 722
Details: DATASOURCE Procedure **722**
Variable Lists 722
OUT= Data Set 723
OUTCONT= Data Set 725
OUTBY= Data Set 726
OUTALL= Data Set 727
OUTEVENT= Data Set 727
Data Elements Reference: DATASOURCE Procedure 728
Examples: DATASOURCE Procedure **754**
Example 13.1: BEA National Income and Product Accounts 754

Example 13.2: BLS Consumer Price Index Surveys 757
Example 13.3: BLS State and Area Employment, Hours, and Earnings Surveys 761
Example 13.4: DRI/McGraw-Hill Format CITIBASE Files 764
Example 13.5: DRI Data Delivery Service Database 768
Example 13.6: PC Format CITIBASE Database 771
Example 13.7: Quarterly COMPUSTAT Data Files 773
Example 13.8: Annual COMPUSTAT Data Files, V9.2 New Filetype CSAUC3 775
Example 13.9: CRSP Daily NYSE/AMEX Combined Stocks 778
References **783**

Overview: DATASOURCE Procedure

The DATASOURCE procedure extracts time series and event data from many different kinds of data files distributed by various data vendors and stores them in a SAS data set. Once stored in a SAS data set, the time series and event variables can be processed by other SAS procedures.

The DATASOURCE procedure has statements and options to extract only a subset of time series data from an input data file. It gives you control over the frequency of data to be extracted, time series variables to be selected, cross sections to be included, and time range of data to be output.

The DATASOURCE procedure can create auxiliary data sets containing descriptive information on the time series variables and cross sections. More specifically, the OUTCONT= option names a data set containing information on time series variables, the OUTBY= option names a data set that reports information on cross-sectional variables, and the OUTALL= option names a data set that combines both time series variables and cross-sectional information.

In addition to the auxiliary data sets, two types of primary output data sets are the OUT= and OUTEVENT= data sets. The OUTEVENT= data set contains event variables but excludes periodic time series data. The OUT= data set contains periodic time series data and any event variables referenced in the KEEP statement.

The output variables in the output and auxiliary data sets can be assigned various attributes by the DATASOURCE procedure. These attributes are labels, formats, new names, and lengths. While the first three attributes in this list are used to enhance the output, the length attribute is used to control the memory and disk-space usage of the DATASOURCE procedure.

Data files currently supported by the DATASOURCE procedure include the following:

- U.S. Bureau of Economic Analysis data files:
  - National Income and Product Accounts
  - National Income and Product Accounts PC format
  - S-pages
- U.S. Bureau of Labor Statistics data files:
  - Consumer Price Index Surveys

  - Producer Price Index Survey
  - National Employment, Hours, and Earnings Survey
  - State and Area Employment, Hours, and Earnings Survey

- Standard & Poor's Compustat Services Financial Database Files:

  - COMPUSTAT Annual
  - COMPUSTAT 48 Quarter
  - COMPUSTAT Full Coverage Annual
  - COMPUSTAT Full Coverage 48 Quarter

- Center for Research in Security Prices (CRSP) data files:

  - Daily Binary Format Files
  - Monthly Binary Format Files
  - Daily Character Format Files
  - Monthly Character Format Files

- Global Insight, formerly DRI/McGraw-Hill data files:

  - Basic Economics Data (formerly CITIBASE)
  - DRI Data Delivery Service files
  - CITIBASE Data Files
  - DRI Data Delivery Service Time Series
  - PC Format CITIBASE Databases

- FAME Information Services Databases

- Haver Analytics data files

  - United States Economic Indicators
  - Specialized Databases
  - Financial Indicators
  - Industry
  - Industrial Countries
  - Emerging Markets
  - International Organizations
  - Forecasts and As Reported Data
  - United States Regional

- International Monetary Fund's Economic Information System data files:

  - International Financial Statistics

  - Direction of Trade Statistics
  - Balance of Payment Statistics
  - Government Finance Statistics

- Organization for Economic Cooperation and Development:
  - Annual National Accounts
  - Quarterly National Accounts
  - Main Economic Indicators

Getting Started: DATASOURCE Procedure

Structure of a SAS Data Set Containing Time Series Data

SAS procedures require time series data to be in a specific form recognizable by the SAS System. This form is a two-dimensional array, called a SAS data set, whose columns correspond to series variables and whose rows correspond to measurements of these variables at certain time periods.

The time periods at which observations are recorded can be included in the data set as a time ID variable. The DATASOURCE procedure does include a time ID variable by the name of DATE.

For example, the following data set in Table 13.1, extracted from a DRIBASIC data file, gives the foreign exchange rates for Japan, Switzerland, and the United Kingdom, respectively.

Table 13.1 The Form of SAS Data Sets Required by Most SAS/ETS Procedures

Time ID Variable	Time Series Variables		
DATE	EXRJAN	EXRSW	EXRUK
SEP1987	143.290	1.50290	164.460
OCT1987	143.320	1.49400	166.200
NOV1987	135.400	1.38250	177.540
DEC1987	128.240	1.33040	182.880
JAN1988	127.690	1.34660	180.090
FEB1988	129.170	1.39160	175.820

Reading Data Files

The DATASOURCE procedure is designed to read data from many different files and to place them in a SAS data set. For example, if you have a DRI Basic Economics data file you want to read, use the following statements:

```
proc datasource filetype=dribasic infile=citifile out=dataset;
run;
```

Here, the FILETYPE= option indicates that you want to read DRI's Basic Economics data file, the INFILE= option specifies the fileref CITIFILE of the external file you want to read, and the OUT= option names the SAS data set to contain the time series data.

Subsetting Input Data Files

When only a subset of a data file is needed, it is inefficient to extract all the data and then subset it in a subsequent DATA step. Instead, you can use the DATASOURCE procedure options and statements to extract only needed information from the data file.

The DATASOURCE procedure offers the following subsetting capabilities:

- the INTERVAL= option controls the frequency of data output
- the KEEP or DROP statement selects a subset of time series variables
- the RANGE statement restricts the time range of data
- the WHERE statement selects a subset of cross sections

Controlling the Frequency of Data – The INTERVAL= Option

The OUT= data set contains only data with the same frequency. If the data file you want to read contains time series data with several frequencies, you can indicate the frequency of data you want to extract with the INTERVAL= option. For example, the following statements extract all monthly time series from the DRIBASIC file CITIFILE:

```
proc datasource filetype=dribasic infile=citifile
                interval=month  out=dataset;
run;
```

When the INTERVAL= option is not given, the default frequency defined for the FILETYPE= type file is used. For example, the statements in the previous section extract yearly series since INTERVAL=YEAR is the default frequency for DRI's Basic Economic Data files.

To extract data for several frequencies, you need to execute the DATASOURCE procedure once for each frequency.

Selecting Time Series Variables – The KEEP and DROP Statements

If you want to include specific series in the OUT= data set, list them in a KEEP statement. If, on the other hand, you want to exclude some variables from the OUT= data set, list them in a DROP statement. For example, the following statements extract monthly foreign exchange rates for Japan (EXRJAN), Switzerland (EXRSW), and the United Kingdom (EXRUK) from a DRIBASIC file CITIFILE:

```
proc datasource filetype=dribasic infile=citifile
                interval=month  out=dataset;
   keep  exrjan exrsw exruk;
run;
```

The KEEP statement also allows input names to be quoted strings. If the name of a series in the input file contains blanks or special characters that are not valid SAS name syntax, put the series name in quotes to select it. Another way to allow the use of special characters in your SAS variable names is to use the SAS options statement to designate VALIDVARNAME=ANY. This option will allow PROC DATASOURCE to include special characters in your SAS variable names. The following is an example of extracting series from a FAME database by using the DATASOURCE procedure.

```
proc datasource filetype=fame dbname='fame_nys /disk1/prc/prc'
                interval=weekday out=outds outcont=attrds;
   range '1jan90'd to '1feb90'd;
   keep cci.close
        '{ibm.high,ibm.low,ibm.close}'
        'mave(ibm.close,30)'
        'crosslist({gm,f,c},{volume})'
        'cci.close+ibm.close';
   rename 'mave(ibm.close,30)' = ibm30day
          'cci.close+ibm.close' = cci_ibm;
run;
```

The resulting output data set OUTDS contains the following series: DATE, CCI_CLOS, IBM_HIGH, IBM_LOW, IBM_CLOS, IBM30DAY, GM_VOLUM, F_VOLUME, C_VOLUME, CCI_IBM.

Obviously, to be able to use KEEP and DROP statements, you need to know the name of time series variables available in the data file. The OUTCONT= option gives you this information. More specifically, the OUTCONT= option creates a data set containing descriptive information on the same frequency time series. This descriptive information includes series names, a flag indicating if the series is selected for output, series variable types, lengths, position of series in the OUT= data set, labels, format names, format lengths, format decimals, and a set of FILETYPE= specific descriptor variables.

For example, the following statements list some of the monthly series available in the CITIFILE and are shown in Figure 13.1.

```
/*-- Selecting Time Series Variables -- The KEEP and DROP Statements --*/
filename citifile "%sysget(DATASRC_DATA)citiaf.dat" RECFM=F LRECL=80;
proc datasource filetype=dribasic infile=citifile
                interval=month  outcont=vars;
   drop e: ;
run;

title1 'Some Time Series Variables Available in CITIFILE';
proc print data=vars;
run;
```

Figure 13.1 Listing of the OUTCONT= Data Set

Some Time Series Variables Available in CITIFILE

Obs	NAME	KEPT	SELECTED	TYPE	LENGTH	VARNUM	LABEL
1	BUS	1	1	1	5	.	INDEX OF NET BUSINESS FORMATION, (1967=100;SA)
2	CCBPY	1	1	1	5	.	RATIO, CONSUMER INSTAL CREDIT TO PERSONAL INCOME (%,SA)(BCD-95)
3	CCI30M	1	1	1	5	.	CONSUMER INSTAL.LOANS: DELINQUENCY RATE,30 DAYS & OVER, (%,SA)
4	CCIPY	1	1	1	5	.	RATIO, CONSUMER INSTAL CREDIT TO PERSONAL INCOME (%,SA)(BCD-95)
5	COCI77	1	1	1	5	.	CONSTRUCTION COST INDEX: DEPT OF COMMERCE COMPOSITE(1977=100,NSA)
6	CONU	1	1	1	5	.	CONSTRUCT.PUT IN PLACE: PRIV NEW HOUSING UNITS (MIL$,SAAR)
7	DLEAD	1	1	1	5	.	COMPOSITE INDEX OF 12 LEADING INDICATORS(67=100,SA)
8	F6CMB	1	1	1	5	.	DEPOSITORY INST RESERVES: TOTAL BORROWINGS AT RES BANKS(MIL$,NSA)
9	F6EDM	1	1	1	5	.	U.S.MDSE EXPORTS: MANUFACTURED GOODS (MIL$,NSA)
10	WTNO8	1	1	1	5	.	MFG & TRADE SALES:MERCHANT WHOLESALERS,OTHR NONDUR GDS,82$
11	WTNR	1	1	1	5	.	MERCHANT WHOLESALERS' SALES: NONDURABLE GOODS (MIL$,SA)
12	WTR	1	1	1	5	.	MERCHANT WHOLESALERS' SALES: TOTAL (MIL$,SA)

Obs	FORMAT	FORMATL	FORMATD	CODE
1		0	0	BUS
2		0	0	CCBPY
3		0	0	CCI30M
4		0	0	CCIPY
5		0	0	COCI77
6		0	0	CONU
7		0	0	DLEAD
8		0	0	F6CMB
9		0	0	F6EDM
10		0	0	WTNO8
11		0	0	WTNR
12		0	0	WTR

Controlling the Time Range of Data – The RANGE Statement

The RANGE statement is used to control the time range of observations included in the output data set. Figure 13.2 shows an example extracting the foreign exchange rates from September 1985 to February 1987, you can use the following statements:

```
/*-- Controlling the Time Range of Data - The RANGE Statement --*/
filename citifile "%sysget(DATASRC_DATA)citiaf.dat" RECFM=F LRECL=80;
proc datasource filetype=dribasic infile=citifile
                interval=month  out=dataset;
   keep  exrjan exrsw exruk;
   range from 1985:9 to 1987:2;
run;

title1 'Printout of the OUT= Data Set';
proc print data=dataset;
run;
```

Figure 13.2 Subset Obtained by KEEP and RANGE Statements

Printout of the OUT= Data Set

Obs	DATE	EXRJAN	EXRSW	EXRUK
1	SEP1985	236.530	2.37490	136.420
2	OCT1985	214.680	2.16920	142.150
3	NOV1985	204.070	2.13060	143.960
4	DEC1985	202.790	2.10420	144.470
5	JAN1986	199.890	2.06600	142.440
6	FEB1986	184.850	1.95470	142.970
7	MAR1986	178.690	1.91500	146.740
8	APR1986	175.090	1.90160	149.850
9	MAY1986	167.030	1.85380	152.110
10	JUN1986	167.540	1.84060	150.850
11	JUL1986	158.610	1.74450	150.710
12	AUG1986	154.180	1.66160	148.610
13	SEP1986	154.730	1.65370	146.980
14	OCT1986	156.470	1.64330	142.640
15	NOV1986	162.850	1.68580	142.380
16	DEC1986	162.050	1.66470	143.930
17	JAN1987	154.830	1.56160	150.540
18	FEB1987	153.410	1.54030	152.800

Reading in Data Files Containing Cross Sections

Some data files group time series data with respect to cross-section identifiers; for example, International Financial Statistics files, distributed by IMF, group data with respect to countries (COUNTRY). Within each country, data are further grouped by Control Source Code (CSC), Partner Country Code (PARTNER), and Version Code (VERSION).

If a data file contains cross-section identifiers, the DATASOURCE procedure adds them to the output data set as BY variables. For example, the data set in Table 13.2 contains three cross sections:

- Cross-section one is identified by (COUNTRY='112' CSC='F' PARTNER=' ' VERSION='Z').
- Cross-section two is identified by (COUNTRY='146' CSC='F' PARTNER=' ' VERSION='Z').
- Cross-section three is identified by (COUNTRY='158' CSC='F' PARTNER=' ' VERSION='Z').

Table 13.2 The Form of a SAS Data Set Containing BY Variables

BY Variables				Time ID Variable	Time Series Variables	
COUNTRY	CSC	PARTNER	VERSION	DATE	EFFEXR	EXRINDEX
112	F		Z	SEP1987	9326	12685
112	F		Z	OCT1987	9393	12813
112	F		Z	NOV1987	9626	13694
112	F		Z	DEC1987	9675	14099
112	F		Z	JAN1988	9581	13910
112	F		Z	FEB1988	9493	13549
146	F		Z	SEP1987	12046	16192
146	F		Z	OCT1987	12067	16266
146	F		Z	NOV1987	12558	17596
146	F		Z	DEC1987	12759	18301
146	F		Z	JAN1988	12642	18082
146	F		Z	FEB1988	12409	17470
158	F		Z	SEP1987	13841	16558
158	F		Z	OCT1987	13754	16499
158	F		Z	NOV1987	14222	17505
158	F		Z	DEC1987	14768	18423
158	F		Z	JAN1988	14933	18565
158	F		Z	FEB1988	14915	18331

Note that the data sets in Table 13.1 and Table 13.2 use two different ways of representing time series data for three different countries: the United Kingdom (COUNTRY='112'), Switzerland (COUNTRY='146'), and Japan (COUNTRY='158'). The first representation (Table 13.1) incorporates each country's name into the series names, while the second representation (Table 13.2) represents countries as different cross sections by using the BY variable named COUNTRY. See "Time Series and SAS Data Sets" in Chapter 4, "Working with Time Series Data."

Obtaining Descriptive Information on Cross Sections

If you want to know the unique set of values BY variables assume for each cross section in the data file, use the OUTBY= option. For example, the following statements list some of the cross sections available for an IFS file, and are shown in Figure 13.3.

```
filename ifsfile "%sysget(DATASRC_DATA)imfifs1.dat" RECFM=F LRECL=88;
proc datasource
   filetype=imfifsp infile=ifsfile
   outselect=on ebcdic
   interval=month
   outby=xsection;
run;

title1 'Some Cross Sections Available in IFSFILE';
proc print data=xsection;
run;
```

Figure 13.3 Listing of the OUTBY= Data Set

Some Cross Sections Available in IFSFILE

Obs	COUNTRY	CSC	PARTNER	VERSION	ST_DATE	END_DATE	NTIME	NOBS	NSERIES	NSELECT	CNTYNAME
1	111	F		Z	JAN1957	SEP1986	357	357	6	3	UNITED STATES
2	112	F		Z	JAN1957	SEP1986	357	357	6	3	UNITED KINGDOM
3	146	F		Z	JAN1957	SEP1986	357	357	6	3	SWITZERLAND
4	158	F		Z	JAN1957	SEP1986	357	357	6	3	JAPAN
5	186	F		Z	JAN1957	SEP1986	357	357	6	3	TURKEY

The OUTBY= data set reports the total number of series, NSERIES, defined in each cross section, NSELECT of which represent the selected variables. If you want to see the descriptive information on each of these NSELECT variables for each cross section, specify the OUTALL= option. For example, the following statements print descriptive information on all monthly series defined for all cross sections (COUNTRY='111', COUNTRY='112', COUNTRY='146', COUNTRY='158', and COUNTRY='186') which are shown in Figure 13.4.

```
filename datafile "%sysget(DATASRC_DATA)imfifs1.dat" RECFM=F LRECL=88;

title3 'Time Series Defined in Cross Section';
proc datasource filetype=imfifsp
   outselect=on ebcdic
   interval=month
   outall=ifsall;
run;

title4 'Cross Sections Available in OUTALL=IFSALL Data Set';
proc print
   data=ifsall;
run;
```

Figure 13.4 Listing of the OUTALL= Data Set

Some Cross Sections Available in IFSFILE

Time Series Defined in Cross Section
Cross Sections Available in OUTALL=IFSALL Data Set

Obs	COUNTRY	CSC	PARTNER	VERSION	NAME	KEPT	SELECTED	TYPE	LENGTH	VARNUM
1	111	F		Z	F___AA	1	1	1	5	.
2	111	F		Z	F___AC	1	1	1	5	.
3	111	F		Z	F___AE	1	1	1	5	.
4	112	F		Z	F___AA	1	1	1	5	.
5	112	F		Z	F___AC	1	1	1	5	.
6	112	F		Z	F___AE	1	1	1	5	.
7	146	F		Z	F___AA	1	1	1	5	.

Obs	BLKNUM	LABEL	FORMAT	FORMATL	FORMATD	ST_DATE	END_DATE	NTIME	NOBS
1	1	MARKET RATE CONVERSION FACTOR		0	0	JAN1957	SEP1986	357	357
2	2	MARKET RATE CONVERSION FACTOR		0	0	JAN1957	SEP1986	357	357
3	3	MARKET RATE CONVERSION FACTOR		0	0	JAN1957	SEP1986	357	357
4	4	MARKET RATE CONVERSION FACTOR		0	0	JAN1957	SEP1986	357	357
5	5	MARKET RATE CONVERSION FACTOR		0	0	JAN1957	SEP1986	357	357
6	6	MARKET RATE CONVERSION FACTOR		0	0	JAN1957	SEP1986	357	357
7	7	MARKET RATE CONVERSION FACTOR		0	0	JAN1957	SEP1986	357	357

Obs	CNTYNAME	SUBJECT	SCDATA	DATATYPE	DU_CODE	DU_NAME	NDEC	BASEYEAR	SOURCE
1	UNITED STATES		S	E	U	U	5		
2	UNITED STATES		S	F	U	U	5		
3	UNITED STATES		S	A	U	U	5		
4	UNITED KINGDOM		S	E	U	U	6		
5	UNITED KINGDOM		S	F	U	U	5		
6	UNITED KINGDOM		S	A	U	U	6		
7	SWITZERLAND		S	E	U		4		

Figure 13.4 *continued*

Some Cross Sections Available in IFSFILE

Time Series Defined in Cross Section
Cross Sections Available in OUTALL=IFSALL Data Set

Obs	COUNTRY	CSC	PARTNER	VERSION	NAME	KEPT	SELECTED	TYPE	LENGTH	VARNUM
8	146	F		Z	F___AC	1	1	1	5	.
9	146	F		Z	F___AE	1	1	1	5	.
10	158	F		Z	F___AA	1	1	1	5	.
11	158	F		Z	F___AC	1	1	1	5	.
12	158	F		Z	F___AE	1	1	1	5	.
13	186	F		Z	F___AA	1	1	1	5	.

Obs	BLKNUM	LABEL	FORMAT	FORMATL	FORMATD	ST_DATE	END_DATE	NTIME	NOBS
8	8	MARKET RATE CONVERSION FACTOR		0	0	JAN1957	SEP1986	357	357
9	9	MARKET RATE CONVERSION FACTOR		0	0	JAN1957	SEP1986	357	357
10	10	MARKET RATE CONVERSION FACTOR		0	0	JAN1957	SEP1986	357	357
11	11	MARKET RATE CONVERSION FACTOR		0	0	JAN1957	SEP1986	357	357
12	12	MARKET RATE CONVERSION FACTOR		0	0	JAN1957	SEP1986	357	357
13	13	MARKET RATE CONVERSION FACTOR		0	0	JAN1957	SEP1986	357	357

Obs	CNTYNAME	SUBJECT	SCDATA	DATATYPE	DU_CODE	DU_NAME	NDEC	BASEYEAR	SOURCE
8	SWITZERLAND			S	F	U	6		
9	SWITZERLAND			S	A	U	4		
10	JAPAN			S	E	U	3		
11	JAPAN			S	F	U	6		
12	JAPAN			S	A	U	3		
13	TURKEY			S	E	U	3		

Figure 13.4 *continued*

Some Cross Sections Available in IFSFILE

Time Series Defined in Cross Section
Cross Sections Available in OUTALL=IFSALL Data Set

Obs	COUNTRY	CSC	PARTNER	VERSION	NAME	KEPT	SELECTED	TYPE	LENGTH	VARNUM
14	186	F		Z	F___AC	1	1	1	5	.
15	186	F		Z	F___AE	1	1	1	5	.

Obs	BLKNUM	LABEL	FORMAT	FORMATL	FORMATD	ST_DATE	END_DATE	NTIME	NOBS
14	14	MARKET RATE CONVERSION FACTOR		0	0	JAN1957	SEP1986	357	357
15	15	MARKET RATE CONVERSION FACTOR		0	0	JAN1957	SEP1986	357	357

Obs	CNTYNAME	SUBJECT	SCDATA	DATATYPE	DU_CODE	DU_NAME	NDEC	BASEYEAR	SOURCE
14	TURKEY		S	F	U		5		
15	TURKEY		S	A	U		3		

The OUTCONT= data set contains one observation for each time series variable with the descriptive information summarized over BY groups. When the data file contains no cross sections, the OUTCONT= and OUTALL= data sets are equivalent, except that the OUTALL= data set also reports time ranges of available data. The OUTBY= data set in this case contains a single observation reporting the number of series and time ranges for the whole data file.

Subsetting a Data File Containing Cross Sections

Data files containing cross sections can be subsetted by controlling which cross sections to include in the output data set. Selecting a subset of cross sections is accomplished using the WHERE statement. The WHERE statement gives a condition that the BY variables must satisfy for a cross section to be selected. For example, the following statements extract the monthly market rate conversion factors for the United Kingdom (COUNTRY='112') and Switzerland (COUNTRY='146') for the period from September 1985 to February 1986.

```
filename datafile "%sysget(DATASRC_DATA)imfifs1.dat" RECFM=F LRECL=88;

title3 'Time Series Defined in Selected Cross Sections';
proc datasource filetype=imfifsp
   outselect=on ebcdic
   interval=month
   out=ifs;

   where country in ('146', '112') and partner=' ';
   keep F___AA F___AC;
   range from '01sep85'd to '01feb86'd;
run;
```

```
title4 'OUTALL=IFS Data Set';
proc print
   data=ifs;
run;
```

Renaming Time Series Variables

Sometimes the time series variable names as given by data vendors are not descriptive enough, or you may prefer a different naming convention. In such cases, you can use the RENAME statement to assign more meaningful names to time series variables. You can also use LABEL statements to associate descriptive labels with your series variables.

For example, the series names for market rate conversion factor (F___AA) and market rate conversion factor (F___AC) used by IMF can be given more descriptive names and labels by the following statements and are shown in Figure 13.5 and Figure 13.6.

```
filename ifsfile "%sysget(DATASRC_DATA)imfifs1.dat" RECFM=F LRECL=88;

proc datasource filetype=imfifsp infile=ifsfile
                interval=month
                out=market outcont=mrktvars;
   where country in ('112','146','158') and partner=' ';
   keep  f___aa f___ac;
   range from '01jun85'd to '01feb86'd;
   rename  f___aa=alphmkt  f___ac=charmkt;
   label   f___aa='F___AA: Market Rate Conversion Factor Used in Alpha Test'
           f___ac='F___AC: Market Rate Conversion Used in Charlie Test';
run;

title1 'Printout of OUTCONT= Showing New NAMEs and LABELs';
proc print data=mrktvars ;
   var  name label length;
run;

title1 'Contents of OUT= Showing New NAMEs and LABELs';
proc contents data=market;
run;
```

The RENAME statement allows input names to be quoted strings. If the name of a series in the input file contains blanks or special characters that are not in valid SAS name syntax, use the SAS option VALIDVARNAME=ANY or put the series name in quotes to rename it. See the FAME example using rename in the "Selecting Time Series Variables – The KEEP and DROP Statements" on page 701 section.

Figure 13.5 Renaming and Labeling Variables

Printout of OUTCONT= Showing New NAMEs and LABELs

Obs	NAME	LABEL	LENGTH
1	alphmkt	F___AA: Market Rate Conversion Factor Used in Alpha Test	5
2	charmkt	F___AC: Market Rate Conversion Used in Charlie Test	5

Figure 13.6 Renaming and Labeling Variables

Alphabetic List of Variables and Attributes					
#	**Variable**	**Type**	**Len**	**Format**	**Label**
1	COUNTRY	Char	3		COUNTRY CODE
2	CSC	Char	1		CONTROL SOURCE CODE
5	DATE	Num	4	MONYY7.	Date of Observation
3	PARTNER	Char	3		PARTNER COUNTRY CODE
4	VERSION	Char	1		VERSION CODE
6	alphmkt	Num	5		F___AA: Market Rate Conversion Factor Used in Alpha Test
7	charmkt	Num	5		F___AC: Market Rate Conversion Used in Charlie Test

Notice that even though you changed the names of F___AA and F___AC to alphmkt and charmkt, respectively, you still use their old names in the KEEP and LABEL statements because renaming takes place at the output stage.

Changing the Lengths of Numeric Variables

The length attribute indicates the number of bytes the SAS System uses for storing the values of variables in output data sets. Therefore, the shorter the variable lengths, the more efficient the disk-space usage. However, there is a trade-off. The lengths of numeric variables are closely tied to their precision, and reducing their lengths arbitrarily can cause precision loss.

The DATASOURCE procedure uses default lengths for series variables appropriate to each file type. For example, the default lengths for numeric variables are 5 for IMFIFSP type files. In some cases, however, you may want to assign different lengths. Assigning lengths less than the defaults reduces memory and disk-space usage at the expense of precision. Specifying lengths longer than the defaults increases the precision but causes the DATASOURCE procedure to use more memory and disk space. The following statements define a default length of 4 for all numeric variables in the IFSFILE and then assign a length of 6 to the exchange rate index. Output is shown in Figure 13.7 and Figure 13.8.

```
filename ifsfile "%sysget(DATASRC_DATA)imfifs1.dat" RECFM=F LRECL=88;

proc datasource filetype=imfifsp infile=ifsfile
                interval=month
                out=market outcont=mrktvars;
   where country in ('112','146','158') and partner=' ';
   keep  f___aa f___ac;
   range from '01jun85'd to '01feb86'd;
   rename  f___aa=alphmkt  f___ac=charmkt;
   label   f___aa='F___AA: Market Rate Conversion Factor Used in Alpha Test'
           f___ac='F___AC: Market Rate Conversion Used in Charlie Test';
   length _numeric_ 4;
   length f___aa 6;
run;

title1 'Printout of OUTCONT= Showing New NAMEs and LABELs';
proc print data=mrktvars ;
   var  name label length;
run;
```

```
title1 'Contents of OUT= Showing New NAMEs and LABELs';
proc contents data=market;
run;
```

Figure 13.7 Changing the Lengths of Numeric Variables

Printout of OUTCONT= Showing New NAMEs and LABELs

Obs	NAME	LABEL	LENGTH
1	alphmkt	F___AA: Market Rate Conversion Factor Used in Alpha Test	6
2	charmkt	F___AC: Market Rate Conversion Used in Charlie Test	4

Figure 13.8 Changing the Lengths of Numeric Variables

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Label
1	COUNTRY	Char	3		COUNTRY CODE
2	CSC	Char	1		CONTROL SOURCE CODE
5	DATE	Num	4	MONYY7.	Date of Observation
3	PARTNER	Char	3		PARTNER COUNTRY CODE
4	VERSION	Char	1		VERSION CODE
6	alphmkt	Num	6		F___AA: Market Rate Conversion Factor Used in Alpha Test
7	charmkt	Num	4		F___AC: Market Rate Conversion Used in Charlie Test

The default lengths of the character variables are set to the minimum number of characters that can hold the longest possible value.

Syntax: DATASOURCE Procedure

The DATASOURCE procedure uses the following statements:

PROC DATASOURCE *options* ;
KEEP *variable-list* ;
DROP *variable-list* ;
KEEPEVENT *event-list* ;
DROPEVENT *event-list* ;
WHERE *where-expression* ;
RANGE FROM *from TO to* ;
ATTRIBUTE *variable-list attribute-list . . .* ;
FORMAT *variable-list format . . .* ;
LABEL *variable="label" . . .* ;
LENGTH *variable-list length . . .* ;
RENAME *old-name=new-name . . .* ;

The PROC DATASOURCE statement is required. All the rest of the statements are optional.

The DATASOURCE procedure uses two kinds of statements, subsetting statements and attribute statements. Subsetting statements provide selection of time series data over selected time periods and cross sections from the input data file. Attribute statements control the attributes of the variables in the output SAS data set.

The subsetting statements are the KEEP, DROP, KEEPEVENT, and DROPEVENT statements (which select output variables); the RANGE statement (which selects time ranges); and the WHERE statement (which selects cross sections). The attribute statements are the ATTRIBUTE, FORMAT, LABEL, LENGTH, and RENAME statements.

The statements and options used by PROC DATASOURCE are summarized in Table 13.3.

Table 13.3 Summary of Syntax

Option	Description
Input Data File Options	
FILETYPE=	type of input data file to read
INFILE=	fileref(s) of the input data
LRECL=	lrecl(s) of the input data
RECFM=	recfm(s) of the input data
ASCII	character set of the incoming data
EBCDIC	character set of the incoming data
Output Data Set Options	
OUT=	write the extracted time series data
OUTALL=	information on time series and cross sections
OUTBY=	information on only cross sections
OUTCONT=	information on only time series variables
OUTEVENT=	write event-oriented data
OUTSELECT=	control reporting of all or only selected series and cross sections
INDEX	create single indexes from BY variables for the OUT= data set
ALIGN=	control the alignment of SAS date values
Subsetting Option and Statements	
INTERVAL=	select periodicity of series to extract
KEEP	time series to include in the OUT= data set
DROP	time series to exclude from the OUT= data set
KEEPEVENT	events to include in the OUTEVENT= data set
DROPEVENT	events to exclude from the OUTEVENT= data set
WHERE	select cross sections for output
RANGE	time range of observations to be output
Assigning Attributes Options and Statements	
FORMAT	assign formats to variables in the output data sets
ATTRIBUTE FORMAT=	assign formats to variables in the output data sets
LABEL	assign labels to variables in the output data sets
ATTRIBUTE LABEL=	assign labels to variables in the output data sets
LENGTH	control the lengths of variables in the output data sets
ATTRIBUTE LENGTH=	control the lengths of variables in the output data sets
RENAME	assign new names to variables in the output data sets

PROC DATASOURCE Statement

PROC DATASOURCE *options* **;**

The following options can be used in the PROC DATASOURCE statement:

ALIGN= *option*
controls the alignment of SAS dates used to identify output observations. The ALIGN= option allows the following values: BEGINNING | BEG | B, MIDDLE | MID | M, and ENDING | END | E. BEGINNING is the default.

ASCII
specifies the incoming data is ASCII. This option is used when the native character set of your host machine is EBCDIC.

DBNAME= '*database name*'
specifies the FAME database to access. Only use this option with the filetype=FAME option. The character string you specify in the DBNAME= option is passed through to FAME. Specify the value of this option as you would in accessing the database from within FAME software.

EBCDIC
specifies the incoming data is ebcdic. This option is needed when the native character set of your host machine is ASCII.

FAMEPRINT
prints the FAME command file generated by PROC DATASOURCE and the log file produced by the FAME component of the interface system. Only use this option with the filetype=FAME option.

FILETYPE= *entry*
DBTYPE= *dbtype*
specifies the kind of input data file to process. See "Data Elements Reference: DATASOURCE Procedure" on page 728 for a list of supported file types. The FILETYPE= option is required.

INDEX
creates a set of single indexes from BY variables for the OUT= data set. Under some circumstances, creating indexes for a SAS data set may increase the efficiency in locating observations when BY or WHERE statements are used in subsequent steps. Refer to *SAS Language Reference: Concepts* for more information on SAS indexes. The INDEX option is ignored when no OUT= data set is created or when the data file does not contain any BY variables. The INDEX= data set option can be used to override the index variable definitions.

INFILE= *fileref*
INFILE= *(fileref1 fileref2 . . . filerefn)*
specifies the *fileref* assigned to the input data file. The default value is DATAFILE. The fileref used in the INFILE= option (or if no INFILE= option is specified, the fileref DATAFILE) must be associated with the physical data file in a FILENAME statement. (On some operating systems, the fileref assignment can be made with the system's control language, and a FILENAME statement may not be needed. Refer to *SAS Statements: Reference* for more details on the FILENAME statement. Physical data files can reside on DVD, CD-ROM, or other media.

For some file types, the data are distributed over several files. In this case, the INFILE= option is required, and it lists in parentheses the filerefs for each of the files making up the database. The order in which these FILEREFS are listed is important and must conform to the specifics of each file type as explained in "Data Elements Reference: DATASOURCE Procedure" on page 728.

LRECL= *lrecl*

LRECL= *(lrecl1 lrecl2 . . . lrecln)*

The logical record length in bytes of the infile. Only use this if you need to override the default LRECL of the file. For some file types, the data are distributed over several files. In this case, the LRECL= option lists in parentheses the LRECLs for each of the files making up the database. The order in which these LRECLs are listed is important and must conform to the specifics of each file type as explained in "Data Elements Reference: DATASOURCE Procedure" on page 728.

RECFM= *recfm*

RECFM= *(recfm1 recfm2 . . . recfmn)*

The record format of the infile. Only use this if you need to override the default record format of the file. For some file types, the data are distributed over several files. In this case, the RECFM= option lists in parentheses the RECFMs for each of the files making up the database. The order in which these RECFMs are listed is important and must conform to the specifics of each file type as explained in "Data Elements Reference: DATASOURCE Procedure" on page 728. The possible values of RECFM are

- F or FIXED for fixed length records
- N or BIN for binary records
- D or VAR for varying length records
- U or DEF for host default record format
- DOM_V or DOMAIN_VAR or BIN_V or BIN_VAR for UNIX binary record format

INTERVAL= *interval*

FREQUENCY= *interval*

TYPE= *interval*

specifies the periodicity of series selected for output to the OUT= data set. The OUT= data set created by PROC DATASOURCE can contain only time series with the same periodicity. Some data files contain time series with different periodicities; for example, a file can contain both monthly series and quarterly series. Use the INTERVAL= option to indicate which periodicity you want. If you want to extract series with different periodicities, use different PROC DATASOURCE invocations with the desired INTERVAL= options.

Common values for INTERVAL= are YEAR, QUARTER, MONTH, WEEK, and DAY. The values allowed, as well as the default value of the INTERVAL= option, depend on the file type. See "Data Elements Reference: DATASOURCE Procedure" on page 728 for the INTERVAL= values appropriate to the data file type you are reading.

OUT= *SAS-data-set*

names the output data set for the time series extracted from the data file. If none of the output data set options are specified, including the OUT= data set itself, an OUT= data set is created and named according to the DATA*n* convention. However, when you create any of the other output data sets, such as OUTCONT=, OUTBY=, OUTALL=, or OUTEVENT=, you must explicitly specify the OUT= data set; otherwise, it will not be created. See "OUT= Data Set" on page 723 for further details.

OUTALL= *SAS-data-set*
writes information on the contents of the input data file to an output data set. The OUTALL= data set includes descriptive information, time ranges, and observation counts for all the time series within each BY group. By default, no OUTALL= data set is created.

The OUTALL= data set contains the Cartesian product of the information output by the OUTCONT= and OUTBY= options. In data files for which there are no cross sections, the OUTALL= and OUTCONT= data sets are almost equivalent, except that OUTALL= data set also reports time ranges and observation counts of series. See "OUTALL= Data Set" on page 727 for further details.

OUTBY= *SAS-data-set*
writes information on the BY variables to an output data set. The OUTBY= data set contains the list of cross sections in the database delimited by the unique set of values that the BY variables assume. Unless the OUTSELECT=OFF option is present, only the selected BY groups are written to the OUTBY= data set. If you omit the OUTBY= option, no OUTBY= data set is created. See "OUTBY= Data Set" on page 726 for further details.

OUTCONT= *SAS-data-set*
writes information on the contents of the input data file to an output data set. By default, the OUTCONT= data set includes descriptive information on all of the unique series of the selected periodicity in the data file. When the OUTSELECT=OFF option is omitted, the OUTCONT= data set includes observations only for the series selected for output to the OUT= data set. By default, no OUTCONT= data set is created. See "OUTCONT= Data Set" on page 725 for further details.

OUTEVENT= *SAS-data-set*
names the output data set to output event-oriented time series data. This option can only be used when CRSP stock files are being processed. For all other file types, it will be ignored. See "OUTEVENT= Data Set" on page 727 for further details.

OUTSELECT= ON | OFF
determines whether to output all observations (OUTSELECT=OFF) or only those corresponding to the selected time series and selected BY groups (OUTSELECT=ON) to OUTCONT=, OUTBY=, and OUTALL= data sets. The default is OUTSELECT=ON. The OUTSELECT= option is only relevant when any one of the auxiliary data sets is specified. The option writes observations to OUTCONT=, OUTBY=, and OUTALL= data sets for only the selected time series and selected BY groups if it is set ON. The OUTSELECT= option is only relevant when any one of the OUTCONT=, OUTBY=, and OUTALL= options is specified. The default is OUTSELECT=ON.

KEEP Statement

KEEP *variable-list* **;**

The KEEP statement specifies which variables in the data file are to be included in the OUT= data set. Only the time series and event variables can be specified in a KEEP statement. All the BY variables and the time ID variable DATE are always included in the OUT= data set; they cannot be referenced in a KEEP statement. If they are referenced, a warning message is given and the reference is ignored.

The variable list can contain variable names or name range specifications. See "Variable Lists" on page 722 for details.

There is a default KEEP list for each file type. Usually, descriptor type variables, like footnotes, are not included in the default KEEP list. If you give a KEEP statement, the default list becomes undefined.

Only one KEEP or one DROP statement can be used. KEEP and DROP are mutually exclusive.

You can also use the KEEP= data set option to control which variables to include in the OUT= data set. However, the KEEP statement differs from the KEEP= data set option in several respects:

- The KEEP statement selection is applied before variables are read from the data file, while the KEEP= data set option selection is applied after variables are read and as they are written to the OUT= data set. Therefore, using the KEEP statement instead of the KEEP= data set option is much more efficient.
- If the KEEP statement causes no series variables to be selected, then no observations are output to the OUT= data set.
- The KEEP statement variable specifications are applied to each cross section independently. This behavior may produce variables different from those produced by the KEEP= data set option when order-range variable list specifications are used.

DROP Statement

DROP *variable-list* **;**

The DROP statement specifies that some variables be excluded from the OUT= data set. Only the time series and event variables can be specified in a DROP statement. None of the BY variables or the time ID variable DATE can be excluded from the OUT= data set. If they are referenced in a DROP statement, a warning message is given and the reference is ignored. Use the WHERE statement for selection based on BY variables, and use the RANGE statement for date selections.

The variable list can contain variable names or name range specifications. See "Variable Lists" on page 722 for details.

Only one DROP or one KEEP statement can be used. KEEP and DROP are mutually exclusive.

There is a default DROP or KEEP list for each file type. Usually, descriptor type variables, like footnotes, are not included in the default KEEP list. If you specify a DROP statement, the default list becomes undefined.

You can also use the DROP= data set option to control which variables to exclude from the OUT= data set. However, the DROP statement differs from the DROP= data set option in several aspects:

- The DROP statement selection is applied before variables are read from the data file, while the DROP= data set option selection is applied after variables are read and as they are written to the OUT= data set. Therefore, using the DROP statement instead of the DROP= data set option is much more efficient.
- If the DROP statement causes all series variables to be excluded, then no observations are output to the OUT= data set.
- The DROP statement variable specifications are applied to each cross section independently. This behavior may produce variables different from those produced by the DROP= data set option when order-range variable list specifications are used.

KEEPEVENT Statement

KEEPEVENT *variable-list* **;**

The KEEPEVENT statement specifies which event variables in the data file are to be included in the OUTEVENT= data set. As a result, the KEEPEVENT statement is valid only for data files containing event-oriented time series data. All the BY variables, the time ID variable DATE, and the event-grouping variable EVENT are always included in the OUTEVENT= data set. These variables cannot be referenced in the KEEPEVENT statement. If any of these variables are referenced, a warning message is given and the reference is ignored.

The variable list can contain variable names or name range specifications. See "Variable Lists" on page 722 for details.

Only one KEEPEVENT or one DROPEVENT statement can be used. KEEPEVENT and DROPEVENT are mutually exclusive.

You can also use the KEEP= data set option to control which event variables to include in the OUTEVENT= data set. However, the KEEPEVENT statement differs from the KEEP= data set option in several respects:

- The KEEPEVENT statement selection is applied before variables are read from the data file, while the KEEP= data set option selection is applied after variables are read and as they are written to the OUTEVENT= data set. Therefore, using the KEEPEVENT statement instead of the KEEP= data set option is much more efficient.
- If the KEEPEVENT statement causes no event variables to be selected, then no observations are output to the OUTEVENT= data set.

DROPEVENT Statement

DROPEVENT *variable-list* **;**

The DROPEVENT statement specifies that some event variables be excluded from the OUTEVENT= data set. As a result, the DROPEVENT statement is valid only for data files containing event-oriented time series data. All the BY variables, the time ID variable DATE, and the event-grouping variable EVENT are always included in the OUTEVENT= data set. These variables cannot be referenced in the DROPEVENT statement. If any of these variables are referenced, a warning message is given and the reference is ignored.

The variable list can contain variable names or name range specifications. See "Variable Lists" on page 722 for details.

Only one DROPEVENT or one KEEPEVENT statement can be used. DROPEVENT and KEEPEVENT are mutually exclusive.

You can also use the DROP= data set option to control which event variables to exclude from the OUTEVENT= data set. However, the DROPEVENT statement differs from the DROP= data set option in several respects:

- The DROPEVENT statement selection is applied before variables are read from the data file, while the DROP= data set option selection is applied after variables are read and as they are written to the OUTEVENT= data set. Therefore, using the DROPEVENT statement instead of the DROP= data set option is much more efficient.
- If the DROPEVENT statement causes all series variables to be excluded, then no observations are output to the OUTEVENT= data set.

WHERE Statement

WHERE *where-expression* **;**

The WHERE statement specifies conditions that BY variables must satisfy in order for a cross section to be included in the OUT= and OUTEVENT= data sets. By default, all BY groups are selected.

The *where-expression* must refer only to BY variables defined for the file type you are reading. The "Data Elements Reference: DATASOURCE Procedure" on page 728 lists the names of the BY variables for each file type.

For example, DOTS (Direction of Trade Statistics) files, distributed by the International Monetary Fund, have four BY variables: COUNTRY, CSC, PARTNER, and VERSION. Both COUNTRY and PARTNER are three-digit country codes. To select the direction of trade statistics of the United States (COUNTRY='111') with Turkey (COUNTRY='186'), Japan (COUNTRY='158'), and the oil exporting countries group (COUNTRY='985'), you should specify

```
where country='111' and partner in ('186','158','985');
```

You can use any SAS language operators and special WHERE expression operators in the WHERE statement condition. Refer to *SAS Language Reference: Concepts* for a more detailed discussion of WHERE expressions.

If you want to see the names of the BY variables and the values they assume for each cross section, you can first run PROC DATASOURCE with only the OUTBY= option. The information contained in the OUTBY= data set will aid you in selecting the appropriate BY groups for subsequent PROC DATASOURCE steps.

RANGE Statement

RANGE FROM *from TO to* **;**

The RANGE statement selects the time range of observations written to the OUT= and OUTEVENT= data sets. The *from* and *to* values can be SAS date, time, or datetime constants, or they can be specified as *year* or *year* **:** *period*, where *year* is a two-digit or four-digit year, and *period* (when specified) is a period within the year corresponding to the INTERVAL= option. (For example, if INTERVAL=QTR, then *period* refers to quarters.) When *period* is omitted, the beginning of the year is assumed for the *from* value, and the end of the year is assumed for the *to* value.

If a two-digit year is specified, PROC DATASOURCE uses the current value of the YEARCUTOFF option to determine the century of your data. Warnings are issued in the SAS log whenever DATASOURCE needs to determine the century from a two-digit year specification.

The default YEARCUTOFF value is 1926. To use a different YEARCUTOFF value, specify

```
options yearcutoff=yyyy;
```

where YYYY is the YEARCUTOFF value you want to use. See *SAS System Options: Reference* for a more detailed discussion of the YEARCUTOFF option.

Both the FROM and TO specifications are optional, and both the FROM and TO keywords are optional. If the FROM limit is omitted, the output observations start with the minimum date for which data are available for any selected series. Similarly, if the TO limit is omitted, the output observations end with the maximum date for which data are available.

The following are some examples of RANGE statements:

```
range from 1980 to 1990;
range 1980 - 1990;
range from 1980;
range 1980;
range to 1990;
range to 1990:2;
range from '31aug89'd to '28feb1990'd;
```

The RANGE statement applies to each BY group independently. If all the selected series contain no data in the specified range for a given BY group, then there will be no observations for that BY group in the OUT= and OUTEVENT= data sets.

If you want to know the time ranges for which periodic time series data are available, you can first run PROC DATASOURCE with the OUTBY= or OUTALL= option. The OUTBY= data set reports the union of the time ranges over all the series within each BY group, while the OUTALL= data set gives time ranges for each series separately in each BY group.

ATTRIBUTE Statement

ATTRIBUTE *variable-list attribute-list . . .* **;**

The ATTRIBUTE statement assigns formats, labels, and lengths to variables in the output data sets.

The *variable-list* can contain variable names and variable name range specifications. See "Variable Lists" on page 722 for details. The attributes specified in the following attribute list apply to all variables in the variable list.

An *attribute-list* consists of one or more of the following options:

FORMAT= *format*
associates a format with variables in *variable-list*. The *format* can be either a standard SAS format or a format defined with the FORMAT procedure. The default formats for variables depend on the file type.

LABEL= *"label"*
assigns a label to the variables in the variable list. The default labels for variables depend on the file type. Labels can be up to 256 bytes in length.

LENGTH= *length*
specifies the number of bytes used to store the values of variables in the variable list. The default lengths for numeric variables depend on the file type. Usually default lengths are set to 5 bytes.

The length specification also controls the amount of memory that PROC DATASOURCE uses to hold variable values while processing the input data file. Thus, specifying a LENGTH= value smaller than the default will reduce both the disk space taken up by the output data sets and the amount of memory used by the PROC DATASOURCE step, at the cost of precision of output data values.

FORMAT Statement

FORMAT *variable-list format . . .* **;**

The FORMAT statement assigns formats to variables in output data sets. The *variable-list* can contain variable names and variable name range specifications. See "Variable Lists" on page 722 for details. The format specified applies to all variables in the variable list.

A single FORMAT statement can assign the same format to several variables or different formats to different variables. The FORMAT statement can use standard SAS formats or formats defined using the FORMAT procedure.

Any later format specification for a variable, using either the FORMAT statement or the FORMAT= option in the ATTRIBUTE statement, always overrides the previous one.

LABEL Statement

LABEL *variable = "label" . . .* **;**

The LABEL statement assigns SAS variable labels to variables in the output data sets. You can give labels for any number of variables in a single LABEL statement. The default labels for variables depend on the file type. Extra-long labels (> 256 bytes) reside in the OUTCONT data set as the DESCRIPT variable.

Any later label specification for a variable, using either the LABEL statement or the LABEL= option in the ATTRIBUTE statement, always overrides the previous one.

LENGTH Statement

LENGTH *variable-list length . . .* **;**

The LENGTH statement, like the LENGTH= option in the ATTRIBUTE statement, specifies the number of bytes used to store values of variables in output data sets. The default lengths for numeric variables depend on the file type. Usually default lengths are set to 5 bytes.

The default lengths of character variables are defined as the minimum number of characters that can hold the longest possible value.

For some file types, the LENGTH statement also controls the amount of memory used to store values of numeric variables while processing the input data file. Thus, specifying LENGTH values smaller than the default will reduce both the disk space taken up by the output data sets and the amount of memory used by the PROC DATASOURCE step, at the cost of precision of output data values.

Any later length specification for a variable, using either the LENGTH statement or the LENGTH= option in the ATTRIBUTE statement, always overrides the previous one.

RENAME Statement

RENAME *old-name = new-name . . .* **;**

The RENAME statement is used to change the names of variables in the output data sets. Any number of variables can be renamed in a single RENAME statement. The most recent RENAME specification overrides any previous ones for a given variable. The *new-name* is limited to 32 characters. Renaming of variables is done at the output stage. Therefore, you need to use the old variable names in all other PROC DATASOURCE statements. For example, the series variable names DATA1-DATA350 used with annual COMPUSTAT files are not very descriptive, so you may choose to rename them to reflect the financial aspect they represent. You may rename "DATA51" as "INVESTTAX" with the RENAME statement

```
rename data51=investtax;
```

since it contains investment tax credit data. However, in all other DATASOURCE statements, you must use the old name, DATA51.

Details: DATASOURCE Procedure

Variable Lists

Variable lists used in PROC DATASOURCE statements can consist of any combination of variable names and name range specifications. Items in variable lists can have the following forms:

- a name, such as PZU.
- an alphabetic range *name1-name2*. For example, A-DZZZZZZZ specifies all variables with names starting with A, B, C, or D.
- a prefix range *prefix* :. For example, IP: selects all variables with names starting with the letters IP.
- an order range *name1–name2*. For example, GLR72–GLRD72 specifies all variables in the input data file between GLR72 and GRLD72 inclusive.

- a numeric order range *name1*-NUMERIC-*name2*. For example, GLR72-NUMERIC-GLRD72 specifies all numeric variables between GLR72 and GRLD72 inclusive.
- a character order range *name1*-CHARACTER-*name2*. For example, GLR72-CHARACTER-GLRD72 specifies all character variables between GLR72 and GRLD72 inclusive.
- one of the keywords _NUMERIC_, _CHARACTER_, or _ALL_. The keyword _NUMERIC_ specifies all numeric variables, _CHARACTER_ specifies all character variables, and _ALL_ specifies all variables.

To determine the order of series in a data file, run PROC DATASOURCE with the OUTCONT= option, and print the output data set. Note that order and alphabetic range specifications are inclusive, meaning that the beginning and ending names of the range are also included in the variable list.

For order ranges, the names used to define the range must actually name variables in the input data file. For alphabetic ranges, however, the names used to define the range need not be present in the data file.

Note that variable specifications are applied to each cross section independently. This may cause the order-range variable list specification to behave differently than its DATA step and data set option counterparts. This is because PROC DATASOURCE knows which variables are defined for which cross sections, while the DATA step applies order range specification to the whole collection of time series variables.

If the ending variable name in an order range specification is not in the current cross section, all variables starting from the beginning variable to the last variable defined in that cross section get selected. If the first variable is not in the current cross section, then order range specification has no effect for that cross section.

The variable names used in variable list specifications can refer either to series names appearing in the input data file or to the SAS names assigned to series data fields internally if the series names are not recorded to the INFILE= file. When the latter is the case, internally defined variable names are listed in "Data Elements Reference: DATASOURCE Procedure" on page 728 later in this chapter.

The following are examples of the use of variable lists:

```
keep  ip: pw112-pw117 pzu;
drop  data1-data99  data151-data350;
length data1-numeric-aftnt350  ucode 4;
```

The first statement keeps all the variables starting with IP:, all the variables between PW112 and PW117 including PW112 and PW117 themselves, and a single variable PZU. The second statement drops all the variables that fall alphabetically between DATA1 and DATA99, and between DATA151 and DATA350. Finally, the third statement assigns a length of 4 bytes to all the numeric variables defined between DATA1 and AFTNT350, and UCODE. Variable lists can not exceed 200 characters in length.

OUT= Data Set

The OUT= data set can contain the following variables:

- the BY variables, which identify cross-sectional dimensions when the input data file contains time series replicated for different values of the BY variables. Use the BY variables in a WHERE statement

to process the OUT= data set by cross sections. The order in which BY variables are defined in the OUT= data set corresponds to the order in which the data file is sorted.

- DATE, a SAS date-, time-, or datetime-valued variable that reports the time period of each observation. The values of the DATE variable may span different time ranges for different BY groups. The format of the DATE variable depends on the INTERVAL= option.
- the periodic time series variables, which are included in the OUT= data set only if they have data in at least one selected BY group and they are not discarded by a KEEP or DROP statement
- the event variables, which are included in the OUT= data set if they are not discarded by a KEEP or DROP statement. By default, these variables are not output to OUT= data set.

The values of BY variables remain constant in each cross section. Observations within each BY group correspond to the sampling of the series variables at the time periods indicated by the DATE variable.

You can create a set of single indexes for the OUT= data set by using the INDEX option, provided there are BY variables. Under some circumstances, this may increase the efficiency of subsequent PROC and DATA steps that use BY and WHERE statements. However, there is a cost associated with creation and maintenance of indexes. The *SAS Language Reference: Concepts* lists the conditions under which the benefits of indexes outweigh the cost.

With data files containing cross sections, there can be various degrees of overlap among the series variables. One extreme is when all the series variables contain data for all the cross sections. In this case, the output data set is very compact. In the other extreme case, however, the set of time series variables are unique for each cross section, making the output data set very sparse, as depicted in Table 13.4.

Table 13.4 The OUT= Data Set Containing Unique Series for Each BY Group

BY Variables BY1 . . . BY*P*	Series in first BY group F1 F2 F3 . . . FN	Series in second BY group S1 S2 S3 . . . SM	. . . . . .	Series in last BY group T1 T2 T3 . . . TK
BY group 1	DATA is here			
BY group 2		DATA is here	data is missing everywhere except on diagonal	
⋮			DATA is here	
BY group N				DATA is here

The data in Table 13.4 can be represented more compactly if cross-sectional information is incorporated into series variable names.

OUTCONT= Data Set

The OUTCONT= data set contains descriptive information for the time series variables. This descriptive information includes various attributes of the time series variables. The OUTCONT= data set contains the following variables:

- NAME, a character variable that contains the series name
- KEPT, a numeric variable that indicates whether the series was selected for output by the DROP or KEEP statements. KEPT is usually the same as SELECTED, but can differ if a WHERE statement is used.
- SELECTED, a numeric variable that indicates whether the series is selected for output to the OUT= data set. The series is included in the OUT= data set (SELECTED=1) if it is kept (KEPT=1) and it has data for at least one selected BY group.
- TYPE, a numeric variable that indicates the type of the time series variable. TYPE=1 for numeric series; TYPE=2 for character series.
- LENGTH, a numeric variable that gives the number of bytes allocated for the series variable in the OUT= data set
- VARNUM, a numeric variable that gives the variable number of the series in the OUT= data set. If the series variable is not selected for output (SELECTED=0), then VARNUM has a missing value. Likewise, if no OUT= option is given, VARNUM has all missing values.
- LABEL, a character variable that contains the label of the series variable. LABEL contains only the first 256 characters of the labels. If they are longer than 256 characters, then the variable, DESCRIPT, is defined to hold the whole length of series labels. Note that if a data file assigns different labels to the same series variable within different cross sections, only the first occurrence of labels will be transferred to the LABEL column.
- the variables FORMAT, FORMATL, and FORMATD, which give the format name, length, and number of format decimals, respectively
- the GENERIC variables, whose values may vary from one series to another, but whose values remain constant across BY groups for the same series

By default, the OUTCONT= data set contains observations for only the selected series where SELECTED=1. If the OUTSELECT=OFF option is specified, the OUTCONT= data set contains one observation for each unique series of the specified periodicity contained in the input data file.

If you do not know what series are in the data file, you can run PROC DATASOURCE with the OUTCONT= option and OUTSELECT=OFF. The information contained in the OUTCONT= data set can then help you to determine which time series data you want to extract.

OUTBY= Data Set

The OUTBY= data set contains information on the cross sections contained in the input data file. These cross sections are represented as BY groups in the OUT= data set. The OUTBY= data set contains the following variables:

- the BY variables, whose values identify the different cross sections in the data file. The BY variables depend on the file type.
- BYSELECT, a numeric variable that reports the outcome of the WHERE statement condition for the BY variable values for this observation. The value of BYSELECT is 1 for BY groups selected by the WHERE statement for output to the OUT= data set and is 0 for BY groups that are excluded by the WHERE statement. BYSELECT is added to the data set only if a WHERE statement is given. When there is no WHERE statement, then all the BY groups are selected.
- ST_DATE, a numeric variable that gives the starting date for the BY group. The starting date is the earliest of the starting dates of all the series that have data for the current BY group.
- END_DATE, a numeric variable that gives the ending date for the BY group. The ending date is the latest of the ending dates of all the series that have data for the BY group.
- NTIME, a numeric variable that gives the number of time periods between ST_DATE and END_DATE, inclusive. Usually, this is the same as NOBS, but they differ when time periods are not equally spaced and when the OUT= data set is not specified. NTIME is a maximum limit on NOBS.
- NOBS, a numeric variable that gives the number of time series observations in the OUT= data set between ST_DATE and END_DATE inclusive. When a given BY group is discarded by a WHERE statement, the NOBS variable corresponding to this BY group becomes 0, since the OUT= data set does not contain any observations for this BY group. Note that BYSELECT=0 for every discarded BY group.
- NINRANGE, a numeric variable that gives the number of observations in the range (*from,to*) defined by the RANGE statement. This variable is only added to the OUTBY= data set when the RANGE statement is specified.
- NSERIES, a numeric variable that gives the total number of unique time series variables having data for the BY group
- NSELECT, a numeric variable that gives the total number of selected time series variables having data for the BY group
- the generic variables, whose values remain constant for all the series in the current BY group

In this list, you can only control the attributes of the BY and GENERIC variables.

The variables NOBS, NTIME, and NINRANGE give observation counts, while the variables NSERIES and NSELECT give series counts.

By default, observations for only the selected BY groups (where BYSELECT=1) are output to the OUTBY= data set, and the date and time range variables are computed over only the selected time series variables.

If the OUTSELECT=OFF option is specified, the OUTBY= data set contains an observation for each BY group, and the date and time range variables are computed over all the time series variables.

For file types that have no BY variables, the OUTBY= data set contains one observation giving ST_DATE, END_DATE, NTIME, NOBS, NINRANGE, NSERIES, and NSELECT for all the series in the file.

If you do not know the BY variable names or their possible values, you can do an initial run of PROC DATASOURCE with the OUTBY= option. The information contained in the OUTBY= data set can help you design your WHERE expression and RANGE statement for the subsequent executions of PROC DATASOURCE to obtain different subsets of the same data file.

OUTALL= Data Set

The OUTALL= data set combines and expands the information provided by the OUTCONT= and OUTBY= data sets. That is, the OUTALL= data set not only reports the OUTCONT= information separately for each BY group, but also reports the OUTBY= information separately for each series. Each observation in the OUTBY= data set gets expanded to NSERIES or NSELECT observations in the OUTALL= data set, depending on whether the OUTSELECT=OFF option is specified.

By default, only the selected BY groups and series are included in the OUTALL= data set. If the OUTSELECT=OFF option is specified, then all the series within all the BY groups are reported.

The OUTALL= data set contains all the variables defined in the OUTBY= and OUTCONT= data sets and also contains the GENERIC variables (whose values can vary from one series to another and from one BY group to another). Another additional variable is BLKNUM, which gives the data block number in the data file containing the series variable.

The OUTALL= data set is useful when BY groups do not contain the same time series variables or when the time ranges for series change across BY groups.

You should be careful in using the OUTALL= option, since the OUTALL= data set can get very large for many file types. Some file types have the same series and time ranges for each BY group; the OUTALL= option should not be used with these file types. For example, you should not specify the OUTALL= option with COMPUSTAT files, since all the BY groups contain the same series variables.

The OUTALL= and OUTCONT= data sets are equivalent when there are no BY variables, except that the OUTALL= data set contains extra information about the time ranges and observation counts of the series variables.

OUTEVENT= Data Set

The OUTEVENT= data set is used to output event-oriented time series data. Events occurring at discrete points in time are recorded along with the date they occurred. Only CRSP stock files contain event-oriented time series data. For all other types of files, the OUTEVENT= option is ignored.

The OUTEVENT= data set contains the following variables:

- the BY variables, which identify cross-sectional dimensions when the input data file contains time series replicated for different values of the BY variables. Use the BY variables in a WHERE statement

to process the OUTEVENT= data set by cross sections. The order in which BY variables are defined in the OUTEVENT= data set corresponds to the order in which the data file is sorted.

- DATE, a SAS date-, time- or datetime-valued variable that reports the discrete time periods at which events occurred. The format of the DATE variable depends on the INTERVAL= option, and should accurately report the date based on the SAS YEARCUTOFF option. The default value for YEARCUTOFF is 1920. The dates used can span up to 250 years.
- EVENT, a character variable that contains the event group name. The EVENT variable is another cross-sectional variable.
- the event variables, which are included in the OUTEVENT= data set only if they have data in at least one selected BY group, and are not discarded by a KEEPEVENT or DROPEVENT statement

Note that each event group contains a nonoverlapping set of event variables; therefore, the OUTEVENT= data set is very sparse. You should exercise care when selecting event variables to be included in the OUTEVENT= data set.

Also note that even though the OUTEVENT= data set cannot contain any periodic time series variables, the OUT= data set can contain event variables if they are explicitly specified in a KEEP statement. In summary, you can specify event variables in a KEEP statement, but you cannot specify periodic time series variables in a KEEPEVENT statement.

While variable selection for OUT= and OUTEVENT= data sets are controlled by a different set of statements (KEEP versus KEEPEVENT or DROP versus DROPEVENT), cross-section and range selections are controlled by the same statements, so in summary, the WHERE and the RANGE statements are effective for both output data sets.

Data Elements Reference: DATASOURCE Procedure

PROC DATASOURCE can process only certain kinds of data files. For certain time series databases, the DATASOURCE procedure has built-in information on the layout of files composing the database. PROC DATASOURCE knows how to read only these kinds of data files. To access these databases, you must indicate the data file type in the FILETYPE= option. For more detailed information, see the corresponding document for each filetype. (See "References".) The currently supported file types are summarized in Table 13.5.

Table 13.5 Supported File Types

Supplier	FILETYPE=	Description
BEA	BEANIPA	National Income and Product Accounts
	BEANIPAD	National Income and Product Accounts PC Format
BLS	BLSCPI	Consumer Price Index Surveys
	BLSWPI	Producer Price Index Survey
	BLSEENA	National Employment, Hours, and Earnings Survey
	BLSEESA	State and Area Employment,Hours,and Earnings Survey

Table 13.5 *continued*

Supplier	FILETYPE=	Description
GLOBAL	DRIBASIC	Basic Economic (formerly CITIBASE) Data Files
INSIGHT	CITIBASE	CITIBASE Data Files
(DRI)	DRIDDS	DRI Data Delivery Service Time Series
(DRI)	CITIDISK	PC Format CITIBASE Databases
CRSP	CRY2DBS	Y2K Daily Binary Security File Format
	CRY2DBI	Y2K Daily Binary Calendar&Indices File Format
	CRY2DBA	Y2K Daily Binary File Annual Data Format
	CRY2MBS	Y2K Monthly Binary Security File Format
	CRY2MBI	Y2K Monthly Binary Calendar&Indices File Format
	CRY2MBA	Y2K Monthly Binary File Annual Data Format
	CRY2DCS	Y2K Daily Character Security File Format
	CRY2DCI	Y2K Daily Character Calendar&Indices File Format
	CRY2DCA	Y2K Daily Character File Annual Data Format
	CRY2MCS	Y2K Monthly Character Security File Format
	CRY2MCI	Y2K Monthly Character Calendar&Indices File Format
	CRY2MCA	Y2K Monthly Character File Annual Data Format
	CRY2DIS	Y2K Daily IBM Binary Security File Format
	CRY2DII	Y2K Daily IBM Binary Calendar&Indices File Format
	CRY2DIA	Y2K Daily IBM Binary File Annual Data Format
	CRY2MIS	Y2K Monthly IBM Binary Security File Format
	CRY2MII	Y2K Monthly IBM Binary Calendar&Indices File Format
	CRY2MIA	Y2K Monthly IBM Binary File Annual Data Format
	CRY2MVS	Y2K Monthly VAX Binary Security File Format
	CRY2MVI	Y2K Monthly VAX Binary Calendar&Indices File Format
	CRY2MVA	Y2K Monthly VAX Binary File Annual Data Format
	CRY2DVS	Y2K Daily VAX Binary Security File Format
	CRY2DVI	Y2K Daily VAX Binary Calendar&Indices File Format
	CRY2DVA	Y2K Daily VAX Binary File Annual Data Format
	CRSPDBS	CRSP Daily Binary Security File Format
	CRSPDBI	CRSP Daily Binary Calendar&Indices File Format
	CRSPDBA	CRSP Daily Binary File Annual Data Format
	CRSPMBS	CRSP Monthly Binary Security File Format
	CRSPMBI	CRSP Monthly Binary Calendar&Indices File Format
	CRSPMBA	CRSP Monthly Binary File Annual Data Format
	CRSPDCS	CRSP Daily Character Security File Format
	CRSPDCI	CRSP Daily Character Calendar&Indices File Format
	CRSPDCA	CRSP Daily Character File Annual Data Format
	CRSPMCS	CRSP Monthly Character Security File Format
	CRSPMCI	CRSP Monthly Character Calendar&Indices File Format
	CRSPMCA	CRSP Monthly Character File Annual Data Format
	CRSPDIS	CRSP Daily IBM Binary Security File Format
	CRSPDII	CRSP Daily IBM Binary Calendar&Indices File Format
	CRSPDIA	CRSP Daily IBM Binary File Annual Data Format
	CRSPMIS	CRSP Monthly IBM Binary Security File Format

Table 13.5 *continued*

Supplier	FILETYPE=	Description
CRSP	CRSPMII	CRSP Monthly IBM Binary Calendar&Indices File Format
	CRSPMIA	CRSP Monthly IBM Binary File Annual Data Format
	CRSPMVS	CRSP Monthly VAX Binary Security File Format
	CRSPMVI	CRSP Monthly VAX Binary Calendar&Indices File Format
	CRSPMVA	CRSP Monthly VAX Binary File Annual Data Format
	CRSPDVS	CRSP Daily VAX Binary Security File Format
	CRSPDVI	CRSP Daily VAX Binary Calendar&Indices File Format
	CRSPDVA	CRSP Daily VAX Binary File Annual Data Format
	CRSPMUS	CRSP Monthly UNIX Binary Security File Format or utility dump of CRSPAccess Monthly Security File Format
	CRSPMUI	CRSP Monthly UNIX Binary Calendar&Indices File Format or utility dump of CRSPAccess Monthly Cal&Indices Format
	CRSPMUA	CRSP Monthly UNIX Binary File Annual Data Format or utility dump of CRSPAccess Monthly Annual Data Format
	CRSPDUS	CRSP Daily UNIX Binary Security File Format or utility dump of CRSPAccess Daily Security Format
	CRSPDUI	CRSP Daily UNIX Binary Calendar&Indices File Format or utility dump of CRSPAccess Daily Calendar&Indices Format
	CRSPDUA	CRSP Daily UNIX Binary File Annual Data Format or utility dump of CRSPAccess Daily Annual Data Format
CRSP	CRSPMOS	CRSP Monthly Old Character Security File Format
	CRSPMOI	CRSP Monthly Old Character Calendar&Indices File Format
	CRSPMOA	CRSP Monthly Old Character File Annual Data Format
	CRSPDOS	CRSP Daily Old Character Security File Format
	CRSPDOI	CRSP Daily Old Character Calendar&Indices File Format
	CRSPDOA	CRSP Daily Old Character File Annual Data Format
	CR95MIS	CRSP 1995 Monthly IBM Binary Security File Format
	CR95MII	CRSP 1995 Monthly IBM Binary Calendar&Indices File Format
	CR95MIA	CRSP 1995 Monthly IBM Binary File Annual Data Format
	CR95DIS	CRSP 1995 Daily IBM Binary Security File Format
	CR95DII	CRSP 1995 Daily IBM Binary Calendar&Indices File Format
	CR95DIA	CRSP 1995 Daily IBM Binary File Annual Data Format
	CR95MVS	CRSP 1995 Monthly VAX Binary Security File Format
	CR95MVI	CRSP 1995 Monthly VAX Binary Calendar&Indices File Format
	CR95MVA	CRSP 1995 Monthly VAX Binary File Annual Data Format
	CR95DVS	CRSP 1995 Daily VAX Binary Security File Format
	CR95DVI	CRSP 1995 Daily VAX Binary Calendar&Indices File Format
	CR95DVA	CRSP 1995 Daily VAX Binary File Annual Data Format

Table 13.5 *continued*

Supplier	FILETYPE=	Description
	CR95MUS	CRSP 1995 Monthly UNIX Binary Security File Format
	CR95MUI	CRSP 1995 Monthly UNIX Binary Calendar&Indices File Format
	CR95MUA	CRSP 1995 Monthly UNIX Binary File Annual Data Format
	CR95DUS	CRSP 1995 Daily UNIX Binary Security File Format
	CR95DUI	CRSP 1995 Daily UNIX Binary Calendar&Indices File Format
	CR95DUA	CRSP 1995 Daily UNIX Binary File Annual Data Format
	CR95MSS	CRSP 1995 Monthly VMS Binary Security File Format
	CR95MSI	CRSP 1995 Monthly VMS Binary Calendar&Indices File Format
	CR95MSA	CRSP 1995 Monthly VMS Binary File Annual Data Format
	CR95DSS	CRSP 1995 Daily VMS Binary Security File Format
	CR95DSI	CRSP 1995 Daily VMS Binary Calendar&Indices File Format
	CR95DSA	CRSP 1995 Daily VMS Binary File Annual Data Format
	CR95MAS	CRSP 1995 Monthly ALPHA Binary Security File Format
	CR95MAI	CRSP 1995 Monthly ALPHA Binary Calendar&Indices Format
	CR95MAA	CRSP 1995 Monthly ALPHA Binary File Annual Data Format
	CR95DAS	CRSP 1995 Daily ALPHA Binary Security File Format
	CR95DAI	CRSP 1995 Daily ALPHA Binary Calendar&Indices File Format
	CR95DAA	CRSP 1995 Daily ALPHA Binary File Annual Data Format
FAME	FAME	FAME Information Services Databases
HAVER	HAVER	Haver Analytics Data Files
IMF	IMFIFSP	International Financial Statistics, Packed Format
	IMFDOTSP	Direction of Trade Statistics, Packed Format
	IMFBOPSP	Balance of Payment Statistics, Packed Format
	IMFGFSP	Government Finance Statistics, Packed Format
OECD	OECDANA	OECD Annual National Accounts Format
	OECDQNA	OECD Quarterly National Accounts Format
	OECDMEI	OECD Main Economic Indicators Format
S&P	CSAIBM	COMPUSTAT Annual, IBM 360&370 Format
	CS48QIBM	COMPUSTAT 48 Quarter, IBM 360&370 Format
	CSAUC	COMPUSTAT Annual, Universal Character Format
	CS48QUC	COMPUSTAT 48 Quarter, Universal Character Format
	CSAIY2	Y2K COMPUSTAT Annual, IBM 360&370 Format
	CSQIY2	Y2K COMPUSTAT 48 Quarter, IBM 360&370 Format
	CSAUCY2	Y2K COMPUSTAT Annual, Universal Character Format
	CSQUCY2	Y2K COMPUSTAT 48 Quarter, Universal Character Format

Data supplier abbreviations used in Table 13.5 are summarized in Table 13.6.

Table 13.6 Data Supplier Abbreviations

Abbreviation	Supplier
BEA	Bureau of Economic Analysis, U.S. Department of Commerce
BLS	Bureau of Labor Statistics, U.S. Department of Labor
CRSP	Center for Research in Security Prices
DRI	Global Insight (formerly DRI/McGraw-Hill)
FAME	FAME Information Services, Inc.
GLOBAL INSIGHT	Global Insight, Inc.
HAVER	Haver Analytics Inc.
IMF	International Monetary Fund
OECD	Organization for Economic Cooperation and Development
S&P	Standard & Poor's Compustat Services Inc.

BEA Data Files

The Bureau of Economic Analysis, U.S. Department of Commerce, supplies national income, product accounting, and various other macroeconomic data at the regional, national, and international levels in the form of data files with various formats and on various media.

The following BEA data file types are supported.

FILETYPE=BEANIPA–National Income and Product Accounts Format

Table 13.7 FILETYPE=BEANIPA–National Income and Product Accounts Format

Metadata Field Types	Metadata Fields	Metadata Labels
Data Files	Database is stored in a single file.	
INTERVAL=	YEAR (default), QUARTER, MONTH	
BY Variables	PARTNO	Part Number of Publication, Integer Portion of the Table Number, 1–9 (character)
	TABNUM	Table Number Within Part, Decimal Portion of the Table Number, 1–24 (character)
Series Variables	Series variable names are constructed by concatenating table number suffix, line and column numbers within each table. An underscore (_) prefix is also added for readability.	

FILETYPE=BEANIPAD–National Income and Product Accounts PC Format

The PC format National Income and Product Accounts files contain the same information as the BEANIPA files described previously.

Table 13.8 FILETYPE=BEANIPAD–National Income and Product Accounts PC Format

Metadata Field Types	Metadata Fields	Metadata Labels
Data Files	Database is stored in a single file.	
INTERVAL=	YEAR (default), QUARTER, MONTH	
BY Variables	PARTNO	Part Number of Publication, Integer Portion of the Table Number, 1–9 (character)
	TABNUM	Table Number Within Part, Decimal Portion of the Table Number, 1–24 (character)
Series Variables	Series variable names are constructed by concatenating table number suffix, line and column numbers within each table. An underscore (_) prefix is also added for readability.	

BLS Data Files

The Bureau of Labor Statistics, U.S. Department of Labor, compiles and distributes data on employment, expenditures, prices, productivity, injuries and illnesses, and wages.

The following BLS file types are supported.

FILETYPE=BLSCPI–Consumer Price Index Surveys (=CU,CW)

Table 13.9 FILETYPE=BLSCPI–Consumer Price Index Surveys (=CU,CW)

Metadata Field Types	Metadata Fields	Metadata Labels
Data Files	Database is stored in a single file.	
INTERVAL=	YEAR, SEMIYEAR1.6, MONTH (default)	
BY Variables	SURVEY	Survey type: CU=All Urban Consumers, CW=Urban Wage Earners and Clerical Workers (character)
	SEASON	Seasonality: S=Seasonally adjusted, U=Unadjusted (character)
	AREA	Geographic Area (character)
	BASPTYPE	Index Base Period Type, S=Standard, A=Alternate Reference (character)
	BASEPER	Index Base Period (character)
Series Variables	Series variable names are the same as consumer item codes listed in the Series Directory shipped with the data.	
Missing Codes	A data value of 0 is interpreted as MISSING.	

FILETYPE=BLSWPI–Producer Price Index Survey (WP)

Table 13.10 FILETYPE=BLSWPI–Producer Price Index Survey (WP)

Metadata Field Types	Metadata Fields	Metadata Labels
Data Files	Database is stored in a single file.	
INTERVAL=	YEAR, MONTH (default)	
BY Variables	SEASON	Seasonality: S=Seasonally adjusted, U=Unadjusted (character)
	MAJORCOM	Major Commodity Group (character)
Sorting Order	BY SEASON MAJORCOM	
Series Variables	Series variable names are the same as commodity codes but prefixed by an underscore (_).	
Missing Codes	A data value of 0 is interpreted as MISSING.	

FILETYPE=BLSEENA–National Employment, Hours, and Earnings Survey

Table 13.11 FILETYPE=BLSEENA–National Employment, Hours, and Earnings Survey

Metadata Field Types	Metadata Fields	Metadata Labels
Data Files	Database is stored in a single file.	
INTERVAL=	YEAR, QUARTER, MONTH (default)	
BY Variables	SEASON	Seasonality: S=Seasonally adjusted, U=Unadjusted (character)
	DIVISION	Major Industrial Division (character)
	INDUSTRY	Industry Code (character)
Sorting Order	BY SEASON DIVISION INDUSTRY	
Series Variables	Series variable names are the same as data type codes prefixed by EE.	
	EE01	Total Employment
	EE02	Employment of Women
	EE03	Employment of Production or Nonsupervisory Workers
	EE04	Average Weekly Earnings of Production Workers
	EE05	Average Weekly Hours of Production Workers
	EE06	Average Hourly Earnings of Production Workers
	EE07	Average Weekly Overtime Hours of Production Workers
	EE40	Index of Aggregate Weekly Hours
	EE41	Index of Aggregate Weekly Payrolls
	EE47	Hourly Earnings Index; 1977 Weights; Current Dollars
	EE48	Hourly Earnings Index; 1977 Weights; Base 1977 Dollars

Table 13.11 (BLSEENA–National Employment, Hours, and Earnings Survey Format continued)

Metadata Field Types	Metadata Fields	Metadata Labels
	EE49	Average Hourly Earnings; Base 1977 Dollars
	EE50	Gross Average Weekly Earnings; Current Dollars
	EE51	Gross Average Weekly Earnings; Base 1977 Dollars
	EE52	Spendable Average Weekly Earnings; No Dependents; Current Dollars
	EE53	Spendable Average Weekly Earnings; No Dependents; Base 1977 Dollars
	EE54	Spendable Average Weekly Earnings; 3 Dependents; Current Dollars
	EE55	Spendable Average Weekly Earnings; 3 Dependents; Base 1977 Dollars
	EE60	Average Hourly Earnings Excluding Overtime
	EE61	Index of Diffusion; 1-month Span; Base 1977
	EE62	Index of Diffusion; 3-month Span; Base 1977
	EE63	Index of Diffusion; 6-month Span; Base 1977
	EE64	Index of Diffusion; 12-month Span; Base 1977
Missing Codes	Series data values are set to MISSING when their status codes are 1.	

FILETYPE=BLSEESA–State and Area Employment, Hours, and Earnings Survey

Table 13.12 FILETYPE=BLSEESA–State and Area Employment, Hours, and Earnings Survey

Metadata Field Types	Metadata Fields	Metadata Labels
Data Files	Database is stored in a single file.	
INTERVAL=	YEAR, MONTH (default)	
BY Variables	STATE	State FIPS codes (numeric)
	AREA	Area codes (character)
	DIVISION	Major industrial division (character)
	INDUSTRY	Industry code (character)
	DETAIL	Private/Government detail
Sorting Order	BY STATE AREA DIVISION INDUSTRY DETAIL	
Series Variables	Series variable names are the same as data type codes prefixed by SA.	
	SA1	All employees
	SA2	Women workers
	SA3	Production workers
	SA4	Average weekly earnings
	SA5	Average weekly hours
Missing Codes	Series data values are set to MISSING when their status codes are 1.	

Global Insight DRI Data Files

The DRIBASIC (formerly CITIBASE) database contains economic and financial indicators of the U.S. and international economies gathered from various government and private sources by DRI/McGraw-Hill, Inc. There are over 8000 yearly, quarterly, monthly, weekly, and daily time series.

Global Insight, formerly DRI/McGraw-Hill, distributes Basic Economic data files on various media. Old DRIDDS data files can be read by DATASOURCE using the DRIDDS filetype.

The following DRI file types are supported.

FILETYPE=DRIBASIC–Global Insight DRI Basic Economic Data Files

Table 13.13 FILETYPE=DRIBASIC–Global Insight DRI Basic Economic Data Files

Metadata Field Types	Metadata Fields Metadata Labels
Data Files	Database is stored in a single file.
INTERVAL=	YEAR (default), QUARTER, MONTH, WEEK, WEEK1.1, WEEK1.2, WEEK1.3, WEEK1.4, WEEK1.5, WEEK1.6, WEEK1.7, WEEKDAY
BY Variables	None
Series Variables	Variable names are taken from the series descriptor records in the data file. Note that series codes can be 20 bytes.
Missing Codes	MISSING=('1.000000E9'=. 'NA'-'ND'=.)

Note that when you specify the INTERVAL=WEEK option, all the weekly series will be aggregated, and the DATE variable in the OUT= data set will be set to the date of Sundays. The date of first observation for each series is the Sunday marking the beginning of the week that contains the starting date of that variable.

FILETYPE=DRIDDS–Global Insight DRI Data Delivery Service Data Files

Table 13.14 FILETYPE=DRIDDS–Global Insight DRI Data Delivery Service Data Files

Metadata Field Types	Metadata Fields Metadata Labels
Data Files	Database is stored in a single file.
INTERVAL=	YEAR (default), SEMIYEAR, QUARTER, MONTH, SEMIMONTH, TENDAY, WEEK, WEEK1.1, WEEK1.2, WEEK1.3, WEEK1.4, WEEK1.5, WEEK1.6, WEEK1.7, WEEKDAY, DAY
BY Variables	None
Series Variables	Variable names are taken from the series descriptor records in the data file. Note that series names can be 24 bytes.
Missing Codes	MISSING=('NA'-'ND'=.)

FILETYPE=CITIOLD–Old Format CITIBASE Data Files

This file type is used for CITIBASE data distributed prior to May 1987.

Table 13.15 FILETYPE=CITIOLD–Old Format CITIBASE Data Files

Metadata Field Types	Metadata Fields	Metadata Labels
Data Files	Database is stored in a single file.	
INTERVAL=	YEAR (default), QUARTER, MONTH	
BY Variables	None	
Series Variables	Variable names are taken from the series descriptor records in the data file and are the same as the series codes reported in the *CITIBASE Directory*.	
Missing Codes	1.0E9=.	

FILETYPE=CITIDISK–PC Format CITIBASE Databases

Table 13.16 FILETYPE=CITIDISK–PC Format CITIBASE Databases

Metadata Field Types	Metadata Fields	Metadata Labels
Data Files	Database is stored in groups of three associated files having the same filename but different extensions: KEY, IND, or DB. The INFILE= option should contain three filerefs in the following order: INFILE=(*keyfile indfile dbfile*).	
INTERVAL=	YEAR (default), QUARTER, MONTH	
BY Variables	None	
Series Variables	Series variable names are the same as series codes reported in the *CITIBASE Directory*.	
Missing Codes	1.0E9=.	

COMPUSTAT Data Files

COMPUSTAT data files, distributed by Standard & Poor's Compustat Services, Inc., consist of a collection of financial, statistical, and market information covering several thousand industrial and nonindustrial companies. Data are available in both an IBM 360/370 format and a "Universal Character" format, both of which further subdivide into annual and quarterly formats.

The BY variables are used to select individual companies or a group of companies. Individual companies can be selected by their unique six-digit CUSIP issuer code (CNUM). A number of specific groups of companies can be extracted by the following key fields:

FILE	specifies the file identification code used to group companies by files.
ZLIST	specifies the exchange listing code that can be used to group companies by exchange.
DNUM	is used to extract companies in a specific SIC industry group.

Series names are internally constructed from the data array names documented in the COMPUSTAT manual. Each column of data array is treated as a SAS variable. The names of these variables are generated by concatenating the corresponding column numbers to the array name.

Missing values use four codes. Missing code '.C' represents a combined figure where the data item has been combined into another data item, '.I' reports an insignificant figure, '.S' represents a semi-annual figure in the second and fourth quarters, '.A' represents an annual figure in the fourth quarter, and '.' indicates that the data item is not available. The missing codes '.C' and '.I' are not used for Aggregate or Prices, Dividends, and Earnings (PDE) files. The missing codes '.S' and '.A' are used only on the Industrial Quarterly File and not on the Aggregate Quarterly, Business Information, or PDE files.

FILETYPE=CSAIBM–COMPUSTAT Annual, IBM 360/370 Format
FILETYPE=CSAIY2–Four-Digit Year COMPUSTAT Annual, IBM 360/370 Format

Table 13.17 FILETYPE=CSAIBM,CSAIY2 –COMPUSTAT Annual,IBM 360/370 Format

Metadata Field Types	Metadata Fields	Metadata Labels
Data Files	Database is stored in a single file.	
INTERVAL=	YEAR (default)	
BY Variables	DNUM	Industry Classification Code (numeric)
	CNUM	CUSIP Issuer Code (character)
	CIC	CUSIP Issue Number and Check Digit (numeric)
	FILE	File Identification Code (numeric)
	ZLIST	Exchange Listing and S&P Index Code (numeric)
	CONAME	Company Name (character)
	INAME	Industry Name (character)
	SMBL	Stock Ticker Symbol (character)
	XREL	S&P Industry Index Relative Code (numeric)
	STK	Stock Ownership Code (numeric)
	STATE	Company Location Identification Code - State (numeric)
	COUNTY	Company Location Identification Code - County (numeric)
	FINC	Incorporation Code - Foreign (numeric)
	EIN	Employer Identification Number (character)
	CPSPIN	S&P Index Primary Marker (character)
	CSSPIN	S&P Index Secondary Identifier (character)
	CSSPII	S&P Index Subset Identifier (character)
	SDBT	S&P Senior Debt Rating - Current (character)
	SDBTIM	Footnote- S&P Senior Debt Rating- Current (character)
	SUBDBT	S&P Subordinated Debt Rating - Current (character)
	CPAPER	S&P Commercial Paper Rating - Current (character)
Sorting Order	BY DNUM CNUM CIC	
Series Variables	DATA1-DATA350 FYR UCODE SOURCE AFTNT1-AFTNT70	

Table 13.17 CSAIBM,CSAIY2 –COMPUSTAT Annual,IBM 360/370 Format continued)

Metadata Field Types	Metadata Fields	Metadata Labels
Default KEEP List	DROP DATA322-DATA326 DATA338 DATA345-DATA347 DATA350 AFTNT52-AFTNT70;	
Missing Codes	0.0001=. 0.0004=.C 0.0008=.I 0.0002=.S 0.0003=.A	

FILETYPE=CS48QIBM–COMPUSTAT 48-Quarter, IBM 360/370 Format
FILETYPE=CSQIY2–FOUR-DIGIT YEAR COMPUSTAT 48-Quarter, IBM 360/370 Format

Table 13.18 FILETYPE=CS48QIBM,CSQIY2 –COMPUSTAT 48-Quarter, IBM 360/370 Format

Metadata Field Types	Metadata Fields	Metadata Labels
Data Files	Database is stored in a single file.	
INTERVAL=	QUARTER (default)	
BY Variables	DNUM	Industry Classification Code (numeric)
	CNUM	CUSIP Issuer Code (character)
	CIC	CUSIP Issue Number and Check Digit (numeric)
	FILE	File Identification Code (numeric)
	CONAME	Company Name (character)
	INAME	Industry Name (character)
	EIN	Employer Identification Number (character)
	STK	Stock Ownership Code (numeric)
	SMBL	Stock Ticker Symbol (character)
	ZLIST	Exchange Listing and S&P Index Code (numeric)
	XREL	S&P Industry Index Relative Code (numeric)
	FIC	Incorporation Code - Foreign (numeric)
	INCORP	Incorporation Code - State (numeric)
	STATE	Company Location Identification Code - State (numeric)
	COUNTY	Company Location Identification Code - County (numeric)
	CANDX	Canadian Index Code - Current (character)
Sorting Order	BY DNUM CNUM CIC;	
Series Variables	DATA1-DATA232	Data Array
	QFTNT1-QFTNT60	Data Footnotes
	FYR	Fiscal Year-End Month of Data
	SPCSCYR	SPCS Calendar Year
	SPCSCQTR	SPCS Calendar Quarter
	UCODE	Update Code
	SOURCE	Source Document Code
	BONDRATE	S&P Bond Rating

Table 13.18 CS48QIBM,CSQIY2 –COMPUSTAT 48-Quarter, IBM 360/370 Format continued)

Metadata Field Types	Metadata Fields	Metadata Labels
	DEBTCL	S&P Class of Debt
	CPRATE	S&P Commercial Paper Rating
	STOCK	S&P Common Stock Ranking
	MIC	S&P Major Index Code
	IIC	S&P Industry Index Code
	REPORTDT	Report Date of Quarterly Earnings
	FORMAT	Flow of Funds Statement Format Code
	DEBTRT	S&P Subordinated Debt Rating
	CANIC	Canadian Index Code
	CS	Comparability Status
	CSA	Company Status Alert
	SENIOR	S&P Senior Debt Rating
Default KEEP List	DROP DATA122-DATA232 QFTNT24-QFTNT60;	
Missing Codes	0.0001=. 0.0004=.C 0.0008=.I 0.0002=.S 0.0003=.A	

FILETYPE=CSAUC–COMPUSTAT Annual, Universal Character Format
FILETYPE=CSAUCY2–Four-Digit Year COMPUSTAT Annual, Universal Character Format

Table 13.19 FILETYPE=CSAUC,CSAUCY2 –COMPUSTAT Annual, Universal Character Format

Metadata Field Types	Metadata Fields	Metadata Labels
Data Files	Database is stored in a single file.	
INTERVAL=	YEAR (default)	
BY Variables	DNUM	Industry Classification Code (numeric)
	CNUM	CUSIP Issuer Code (character)
	CIC	CUSIP Issue Number and Check Digit (character)
	FILE	File Identification Code (numeric)
	ZLIST	Exchange Listing and S&P Index Code (numeric)
	CONAME	Company Name (character)
	INAME	Industry Name (character)
	SMBL	Stock Ticker Symbol (character)
	XREL	S&P Industry Index Relative Code (numeric)
	STK	Stock Ownership Code (numeric)
	STATE	Company Location Identification Code - State (numeric)
	COUNTY	Company Location Identification Code - County (numeric)
	FINC	Incorporation Code - Foreign (numeric)
	EIN	Employer Identification Number (character)
	CPSPIN	S&P Index Primary Marker (character)

Table 13.19 CSAUC,CSAUCY2 –COMPUSTAT Annual, Universal Character Format continued)

Metadata Field Types	Metadata Fields	Metadata Labels
	CSSPIN	S&P Index Secondary Identifier (character)
	CSSPII	S&P Index Subset Identifier (character)
	SDBT	S&P Senior Debt Rating - Current (character)
	SDBTIM	Footnote- S&P Senior Debt Rating- Current (character)
	SUBDBT	S&P Subordinated Debt Rating - Current (character)
	CPAPER	S&P Commercial Paper Rating - Current (character)
Sorting Order	BY DNUM CNUM CIC	
Series Variables	DATA1-DATA350 FYR UCODE SOURCE AFTNT1-AFTNT70	
Default KEEP List	DROP DATA322-DATA326 DATA338 DATA345-DATA347 DATA350 AFTNT52-AFTNT70;	
Missing Codes	-0.001=. -0.004=.C -0.008=.I -0.002=.S -0.003=.A	

FILETYPE=CS48QUC–COMPUSTAT 48 Quarter, Universal Character Format
FILETYPE=CSQUCY2–Four-Digit Year COMPUSTAT 48 Quarter, Universal Character Format

Table 13.20 FILETYPE=CS48QUC,CSQUCY2 –COMPUSTAT 48 Quarter, Universal Character Format

Metadata Field Types	Metadata Fields	Metadata Labels
Data Files	Database is stored in a single file.	
INTERVAL=	QUARTER (default)	
BY Variables	DNUM	Industry Classification Code (numeric)
	CNUM	CUSIP Issuer Code (character)
	CIC	CUSIP Issue Number and Check Digit (character)
	FILE	File Identification Code (numeric)
	CONAME	Company Name (character)
	INAME	Industry Name (character)
	EIN	Employer Identification Number (character)
	STK	Stock Ownership Code (numeric)
	SMBL	Stock Ticker Symbol (character)
	ZLIST	Exchange Listing and S&P Index Code (numeric)
	XREL	S&P Industry Index Relative Code (numeric)
	FIC	Incorporation Code - Foreign (numeric)
	INCORP	Incorporation Code - State (numeric)
	STATE	Company Location Identification Code - State (numeric)
	COUNTY	Company Location Identification Code - County (numeric)
	CANDXC	Canadian Index Code - Current (numeric)
Sorting Order	BY DNUM CNUM CIC	

Table 13.20 CS48QUC,CSQUCY2 –COMPUSTAT 48 Quarter, Universal Character Format continued)

Metadata Field Types	Metadata Fields	Metadata Labels
Series Variables	DATA1-DATA232	Data Array
	QFTNT1-QFTNT60	Data Footnotes
	FYR	Fiscal Year-End Month of Data
	SPCSCYR	SPCS Calendar Year
	SPCSCQTR	SPCS Calendar Quarter
	UCODE	Update Code
	SOURCE	Source Document Code
	BONDRATE	S&P Bond Rating
	DEBTCL	S&P Class of Debt
	CPRATE	S&P Commercial Paper Rating
	STOCK	S&P Common Stock Ranking
	MIC	S&P Major Index Code
	IIC	S&P Industry Index Code
	REPORTDT	Report Date of Quarterly Earnings
	FORMAT	Flow of Funds Statement Format Code
	DEBTRT	S&P Subordinated Debt Rating
	CANIC	Canadian Index Code - Current
	CS	Comparability Status
	CSA	Company Status Alert
	SENIOR	S&P Senior Debt Rating
Default KEEP List	DROP DATA122-DATA232 QFTNT24-QFTNT60;	
Missing Codes	-0.001=. -0.004=.C -0.008=.I -0.002=.S -0.003=.A	

CRSP Stock Files

The Center for Research in Security Prices provides comprehensive security price data through two primary stock files, the NYSE/AMEX file and the NASDAQ file. These files contain master and return components, available separately or combined. CRSP stock files are further differentiated by the frequency at which prices and returns are reported, daily or monthly. Both daily and monthly files contain annual data fields.

CRSP data files are distributed in CRSPAccess format. See Chapter 46, "The SASECRSP Interface Engine," for more about accessing your CRSPAccess database. You can convert your CRSPAccess data to binary format (SFA format) by using the CRSP-supplied utility (STK_DUMP_BIN). Use the DATASOURCE procedure for SFA format access and use SASECRSP Interface for CRSPAccess.

CRSP stock data (in SFA format) are provided in two files, a main data file containing security information and a calendar/indices file containing a list of trading dates and market information associated with those trading dates.

The file types for CRSP stock files are constructed by concatenating CRSP with a D or M to indicate the frequency of data, followed by B, C, or I to indicate file formats. B is for host binary, C is for character, and I

is for IBM binary formats. The last character in the file type indicates if you are reading the Calendar/Indices file (I), or if you are extracting the security (S) or annual data (A). For example, the file type for the daily NYSE/AMEX combined data in IBM binary format is CRSPDIS. Its calendar/indices file can be read by CRSPDII, and its annual data can be extracted by CRSPDIA.

Starting in 1995, binary data used split records (RICFAC=2), so the 1995 filetypes (CR95*) should be used for 1995 and 1996 binary data. If you use utility routines supplied by CRSP to convert a character format file to a binary format file on a given host, then you need to use host binary file types (RIDFAC=1) to read those files in. Note that you cannot do the conversion on one host and transfer and read the file on another host.

If you are using the CRSPAccess Database, you will need to use the utility routine (stk_dump_bin) supplied by CRSP to generate the UNIX binary format of the data. You can access the UNIX (or SUN) binary data by using PROC DATASOURCE with the CRSPDUS for daily or CRSPMUS for monthly stock data.

For the four-digit year data, use the Y2K-compliant filetypes for that data type.

For CRSP file types, the INFILE= option must be of the form

```
INFILE=( calfile security1 < security2 ... > )
```

where *calfile* is the fileref assigned to the calendar/indices file, and *security1* < *security2* . . . > are the filerefs given to the security files, in the order in which they should be read.

CRSP Calendar/Indices Files

Table 13.21 CRSP Calendar/Indices Files Format

Metadata Field Types	Metadata Fields	Metadata Labels
Data Files	Database is stored in a single file.	
INTERVAL=	DAY	for products DA, DR, DX, EX, NX, and RA
	MONTH	for products MA, MX, and MZ
BY Variables	None	
Series Variables	VWRETD	Value-Weighted Return (including all distributions)
	VWRETX	Value-Weighted Return (excluding dividends)
	EWRETD	Equal-Weighted Return (including all distributions)
	EWRETX	Equal-Weighted Return (excluding dividends)
	TOTVAL	Total Market Value
	TOTCNT	Total Market Count
	USDVAL	Market Value of Securities Used
	USDCNT	Count of Securities Used
	SPINDX	Level of the Standard & Poor's Composite Index
	SPRTRN	Return on the Standard & Poor's Composite Index
	NCINDX	NASDAQ Composite Index
	NCRTRN	NASDAQ Composite Return
Default KEEP List	All variables will be kept.	

CRSP Daily Security Files

Table 13.22 CRSP Daily Security Files Format

Metadata Field Types	Metadata Fields	Metadata Labels	
Data Files	INFILE=(calfile securty1 < securty2 . . . >)		
INTERVAL=	DAY		
BY Variables	CUSIP	CUSIP Identifier (character)	
	PERMNO	CRSP Permanent Number (numeric)	
	COMPNO	NASDAQ Company Number (numeric)	
	ISSUNO	NASDAQ Issue Number (numeric)	
	HEXCD	Header Exchange Code (numeric)	
	HSICCD	Header SIC Code (numeric)	
Sorting Order	BY CUSIP		
Series Variables	BIDLO	Bid or Low	
	ASKHI	Ask or High	
	PRC	Closing Price of Bid/Ask Average	
	VOL	Share Volume	
	RET	Holding Period Return missing=(-66.0 = .p -77.0 = .t -88.0 = .r -99.0 = .b)	
	BXRET	Beta Excess Return missing=(-44.0 = .)	
	SXRET	Standard Deviation Excess Return missing=(-44.0 = .)	
Events	NAMES	NCUSIP	Name CUSIP
		TICKER	Exchange Ticker Symbol
		COMNAM	Company Name
		SHRCLS	Share Class
		SHRCD	Share Code
		EXCHCD	Exchange Code
		SICCD	Standard Industrial Classification Code
	DIST	DISTCD	Distribution Code
		DIVAMT	Dividend Cash Amount
		FACPR	Factor to Adjust Price
		FACSHR	Factor to Adjust Shares Outstanding
		DCLRDT	Declaration Date
		RCRDDT	Record Date
		PAYDT	Payment Date
	SHARES	SHROUT	Number of Shares Outstanding
		SHRFLG	Share Flag
	DELIST	DLSTCD	Delisting Code
		NWPERM	New CRSP Permanent Number
		NEXTDT	Date of Next Available Information
		DLBID	Delisting Bid

Table 13.22 CRSP Daily Security Files Format continued)

Metadata Field Types	Metadata Fields	Metadata Labels	
		DLASK	Delisting Ask
		DLPRC	Delisting Price
		DLVOL	Delisting Volume missing=(-99 = .)
		DLRET	Delisting Return missing=(-55.0=.s -66.0=.t -88.0=.a -99.0=.p);
	NASDIN	TRTSCD	Traits Code
		NMSIND	National Market System Indicator
		MMCNT	Market Maker Count
		NSDINX	NASD Index
Default KEEP Lists	All periodic series variables will be output to the OUT= data set and all event variables will be output to the OUTEVENT= data set.		

CRSP Monthly Security Files

Table 13.23 CRSP Monthly Security Files Format

Metadata Field Types	Metadata Fields	Metadata Labels	
Data Files	INFILE=(calfile security1 < security2 . . . >)		
INTERVAL=	MONTH		
BY Variables	CUSIP	CUSIP Identifier (character)	
	PERMNO	CRSP Permanent Number (numeric)	
	COMPNO	NASDAQ Company Number (numeric)	
	ISSUNO	NASDAQ Issue Number (numeric)	
	HEXCD	Header Exchange Code (numeric)	
	HSICCD	Header SIC Code (numeric)	
Sorting Order	BY CUSIP		
Series Variables	BIDLO	Bid or Low	
	ASKHI	Ask or High	
	PRC	Closing Price of Bid/Ask average	
	VOL	Share Volume	
	RET	Holding Period Return missing=(-66.0 = .p -77.0 = .t -88.0 = .r -99.0 = .b);	
	RETX	Return Without Dividends missing=(-44.0 = .)	
	PRC2	Secondary Price missing=(-44.0 = .)	
Events	NAMES	NCUSIP	Name CUSIP
		TICKER	Exchange Ticker Symbol
		COMNAM	Company Name
		SHRCLS	Share Class
		SHRCD	Share Code

Table 13.23 CRSP Monthly Security Files Format continued)

Metadata Field Types	Metadata Fields	Metadata Labels	
		EXCHCD	Exchange Code
		SICCD	Standard Industrial Classification Code
	DIST	DISTCD	Distribution Code
		DIVAMT	Dividend Cash Amount
		FACPR	Factor to Adjust Price
		FACSHR	Factor to Adjust Shares Outstanding
		EXDT	Ex-distribution Date
		RCRDDT	Record Date
		PAYDT	Payment Date
	SHARES	SHROUT	Number of Shares Outstanding
		SHRFLG	Share Flag
	DELIST	DLSTCD	Delisting Code
		NWPERM	New CRSP Permanent Number
		NEXTDT	Date of Next Available Information
		DLBID	Delisting Bid
		DLASK	Delisting Ask
		DLPRC	Delisting Price
		DLVOL	Delisting Volume
		DLRET	Delisting Return missing=(-55.0=.s -66.0=.t -88.0=.a -99.0=.p);
	NASDIN	TRTSCD	Traits Code
		NMSIND	National Market System Indicator
		MMCNT	Market Maker Count
		NSDINX	NASD Index
Default KEEP Lists	All periodic series variables will be output to the OUT= data set and all event variables will be output to the OUTEVENT= data set.		

CRSP Annual Data

Table 13.24 CRSP Annual Data Format

Metadata Field Types	Metadata Fields	Metadata Labels
Data Files	INFILE=(security1 < security2 . . . >)	
INTERVAL=	YEAR	
BY Variables	CUSIP	CUSIP Identifier (character)
	PERMNO	CRSP Permanent Number (numeric)
	COMPNO	NASDAQ Company Number (numeric)
	ISSUNO	NASDAQ Issue Number (numeric)
	HEXCD	Header Exchange Code (numeric)

Table 13.24 CRSP Annual Data Format continued)

Metadata Field Types	Metadata Fields	Metadata Labels
	HSICCD	Header SIC Code (numeric)
Sorting Order	BY CUSIP	
Series Variables	CAPV	Year End Capitalization
	SDEVV	Annual Standard Deviation missing=(-99.0 = .)
	BETAV	Annual Beta missing=(-99.0 = .)
	CAPN	Year End Capitalization Portfolio Assignment
	SDEVN	Standard Deviation Portfolio Assignment
	BETAN	Beta Portfolio Assignment
Default KEEP Lists	All variables will be kept.	

FAME Information Services Databases

The DATASOURCE procedure provides access to FAME Information Services databases for UNIX-based systems only. See "The SASEFAME Interface Engine" in Chapter 47, "The SASEFAME Interface Engine," for information about a more flexible FAME database access.

The DATASOURCE interface to FAME requires a component supplied by FAME Information Services, Inc. Once this FAME component is installed on your system, you can use the DATASOURCE procedure to extract data from your FAME databases by giving the following specifications.

Specify FILETYPE=FAME in the PROC DATASOURCE statement and give the FAME database name to access with a DBNAME='*fame-database* ' option. The character string you specify in the DBNAME= option is passed through to FAME; specify the value of this option as you would in accessing the database from within FAME software.

Specify the output SAS data set to be created, the frequency of the series to be extracted, and other usual DATASOURCE procedure options as appropriate.

Specify the time range to extract with a RANGE statement. The RANGE statement is required when extracting series from FAME databases.

Name the FAME series to be extracted with a KEEP statement. The items in the KEEP statement are passed through to FAME software; therefore, you can use any valid FAME expression to specify the series to be extracted. Enclose in quotes any FAME series name or expression that is not a valid SAS name.

Name the SAS variable names you want to use for the extracted series in a RENAME statement. Give the FAME series name or expression (in quotes if needed) followed by an equal sign and the SAS name. The RENAME statement is not required; however, if the FAME series name is not a valid SAS variable name, the DATASOURCE procedure will construct a SAS name by translating and truncating the FAME series name. This process might not produce the desired name for the variable in the output SAS data set, so a rename statement could be used to produce a more appropriate variable name. The VALIDVARNAME=ANY option in your SAS options statement can be used to allow special characters in the SAS variable name.

For an alternative solution to PROC DATASOURCE's access to FAME, see "The SASEFAME Interface Engine" in Chapter 47, "The SASEFAME Interface Engine."

FILETYPE=FAME–FAME Information Services Databases

Table 13.25 FILETYPE=FAME–FAME Information Services Database Format

Metadata Field Types	Metadata Fields	Metadata Labels
INTERVAL=	YEAR	correspond to FAME's ANNUAL(DECEMBER)
	YEAR.2	correspond to FAME's ANNUAL(JANUARY)
	YEAR.3	correspond to FAME's ANNUAL(FEBRUARY)
	YEAR.4	correspond to FAME's ANNUAL(MARCH)
	YEAR.5	correspond to FAME's ANNUAL(APRIL)
	YEAR.6	correspond to FAME's ANNUAL(MAY)
	YEAR.7	correspond to FAME's ANNUAL(JUNE)
	YEAR.8	correspond to FAME's ANNUAL(JULY)
	YEAR.9	correspond to FAME's ANNUAL(AUGUST)
	YEAR.10	correspond to FAME's ANNUAL(SEPTEMBER)
	YEAR.11	correspond to FAME's ANNUAL(OCTOBER)
	YEAR.12	correspond to FAME's ANNUAL(NOVEMBER)
	SEMIYEAR	correspond to FAME's SEMIYEAR
	QUARTER	correspond to FAME's QUARTER
	MONTH	correspond to FAME's MONTH
	SEMIMONTH	correspond to FAME's SEMIMONTH
	TENDAY	correspond to FAME's TENDAY
	WEEK	corresponds to FAME's WEEKLY(SATURDAY)
	WEEK.2	corresponds to FAME's WEEKLY(SUNDAY)
	WEEK.3	corresponds to FAME's WEEKLY(MONDAY)
	WEEK.4	corresponds to FAME's WEEKLY(TUESDAY)
	WEEK.5	corresponds to FAME's WEEKLY(WEDNESDAY)
	WEEK.6	corresponds to FAME's WEEKLY(THURSDAY)
	WEEK.7	corresponds to FAME's WEEKLY(FRIDAY)
	WEEK2	corresponds to FAME's BIWEEKLY(ASATURDAY)
	WEEK2.2	correspond to FAME's BIWEEKLY(ASUNDAY)
	WEEK2.3	correspond to FAME's BIWEEKLY(AMONDAY)
	WEEK2.4	correspond to FAME's BIWEEKLY(ATUESDAY)
	WEEK2.5	correspond to FAME's BIWEEKLY(AWEDNESDAY)
	WEEK2.6	correspond to FAME's BIWEEKLY(ATHURSDAY)
	WEEK2.7	correspond to FAME's BIWEEKLY(AFRIDAY)
	WEEK2.8	correspond to FAME's BIWEEKLY(BSATURDAY)
	WEEK2.9	correspond to FAME's BIWEEKLY(BSUNDAY)
	WEEK2.10	correspond to FAME's BIWEEKLY(BMONDAY)
	WEEK2.11	correspond to FAME's BIWEEKLY(BTUESDAY)
	WEEK2.12	correspond to FAME's BIWEEKLY(BWEDNESDAY)
	WEEK2.13	correspond to FAME's BIWEEKLY(BTHURSDAY)
	WEEK2.14	correspond to FAME's BIWEEKLY(BFRIDAY)
	WEEKDAY	correspond to FAME's WEEKDAY
	DAY	correspond to FAME's DAY
BY Variables	None	

Table 13.25 FILETYPE=FAME–FAME Information Services Database Format continued)

Metadata Field Types	Metadata Fields	Metadata Labels
Series Variables	Variable names are constructed from the FAME series codes. Note that series names are limited to 32 bytes.	

Haver Analytics Data Files

Haver Analytics offers a broad range of economic, financial, and industrial data for the United States and other countries. See "The SASEHAVR Interface Engine" in Chapter 49, "The SASEHAVR Interface Engine," for information about accessing your HAVER DLX database. SASEHAVR is supported on most Windows environments. Use the DATASOURCE procedure for serial access of your data. The format of Haver Analytics data files is similar to the CITIBASE/DRIBASIC formats.

FILETYPE=HAVER–Haver Analytics Data Files HAVERO–Old Format Haver Files

Table 13.26 FILETYPE=HAVER–Haver Analytics Data Files Format

Metadata Field Types	Metadata Fields	Metadata Labels
Data Files	Database is stored in a single file.	
INTERVAL=	YEAR (default), QUARTER, MONTH	
Series Variables	Variable names are taken from the series descriptor records in the data file. NOTE: HAVER filetype reports the UPDATE and SOURCE in the OUTCONT= data set, while HAVERO does not.	
Missing Codes	1.0E9=.	

IMF Data Files

The International Monetary Fund's Economic Information System (EIS) offers subscriptions for their International Financial Statistics (IFS), Direction of Trade Statistics (DOTS), Balance of Payment Statistics (BOPS), and Government Finance Statistics (GFS) databases. The first three contain annual, quarterly, and monthly data, while the GFS file has only annual data.

PROC DATASOURCE supports only the packed format IMF data.

FILETYPE=IMFIFSP–International Financial Statistics, Packed Format

The IFS data files contain over 23,000 time series including interest and exchange rates, national income and product accounts, price and production indexes, money and banking, export commodity prices, and balance of payments for nearly 200 countries and regional aggregates.

Table 13.27 FILETYPE=IMFIFSP–International Financial Statistics Format

Metadata Field Types	Metadata Fields	Metadata Labels
Data Files	Database is stored in a single file.	
INTERVAL=	YEAR (default), QUARTER, MONTH	
BY Variables	COUNTRY	Country Code (character, three digits)
	CSC	Control Source Code (character)
	PARTNER	Partner Country Code (character, three digits)
	VERSION	Version Code (character)
Sorting Order	BY COUNTRY CSC PARTNER VERSION	
Series Variables	Series variable names are the same as series codes reported in *IMF Documentation* prefixed by F for data and F_F for footnote indicators.	
Default KEEP List	By default all the footnote indicators will be dropped.	

FILETYPE=IMFDOTSP–Direction of Trade Statistics, Packed Format

The DOTS files contain time series on the distribution of exports and imports for about 160 countries and country groups by partner country and areas.

Table 13.28 FILETYPE=IMFDOTSP–Direction of Trade Statistics Format

Metadata Field Types	Metadata Fields	Metadata Labels
Data Files	Database is stored in a single file.	
INTERVAL=	YEAR (default), QUARTER, MONTH	
BY Variables	COUNTRY	Country Code (character, three digits)
	CSC	Control Source Code (character)
	PARTNER	Partner Country Code (character, three digits)
	VERSION	Version Code (character)
Sorting Order	BY COUNTRY CSC PARTNER VERSION	
Series Variables	Series variable names are the same as series codes reported in *IMF Documentation* prefixed by D for data and F_D for footnote indicators.	
Default KEEP List	By default all the footnote indicators will be dropped.	

FILETYPE=IMFBOPSP–Balance of Payment Statistics, Packed Format

The BOPS data files contain approximately 43,000 time series on balance of payments for about 120 countries.

Table 13.29 FILETYPE=IMFBOPSP–Balance of Payment Statistics Format

Metadata Field Types	Metadata Fields	Metadata Labels
Data Files	Database is stored in a single file.	
INTERVAL=	YEAR (default), QUARTER, MONTH	
BY Variables	COUNTRY	Country Code (character, three digits)
	CSC	Control Source Code (character)
	PARTNER	Partner Country Code (character, three digits)
	VERSION	Version Code (character)
Sorting Order	BY COUNTRY CSC PARTNER VERSION	
Series Variables	Series variable names are the same as series codes reported in *IMF Documentation* prefixed by B for data and F_B for footnote indicators.	
Default KEEP List	By default all the footnote indicators will be dropped.	

FILETYPE=IMFGFSP–Government Finance Statistics, Packed Format

The GFS data files encompass approximately 28,000 time series that give a detailed picture of federal government revenue, grants, expenditures, lending minus repayment financing and debt, and summary data of state and local governments, covering 128 countries.

Table 13.30 FILETYPE=IMFGFSP–Government Finance Statistics Format

Metadata Field Types	Metadata Fields	Metadata Labels
Data Files	Database is stored in a single file.	
INTERVAL=	YEAR (default), QUARTER, MONTH	
BY Variables	COUNTRY	Country Code (character, three digits)
	CSC	Control Source Code (character)
	PARTNER	Partner Country Code (character, three digits)
	VERSION	Version Code (character)
Sorting Order	BY COUNTRY CSC PARTNER VERSION	
Series Variables	Series variable names are the same as series codes reported in *IMF Documentation* prefixed by G for data and F_G for footnote indicators.	
Default KEEP List	By default all the footnote indicators will be dropped.	

OECD Data Files

The Organization for Economic Cooperation and Development compiles and distributes statistical data, including National Accounts and Main Economic Indicators.

FILETYPE=OECDANA–Annual National Accounts

The ANA data files contain both main national aggregates accounts (Volume I) and detailed tables for each OECD Member country (Volume II).

Table 13.31 FILETYPE=OECDANA–Annual National Accounts Format

Metadata Field Types	Metadata Fields	Metadata Labels
Data Files	Database is stored on a single file.	
INTERVAL=	YEAR (default), SEMIYR1.6, QUARTER, MONTH, WEEK, WEEKDAY	
BY Variables	PREFIX	Table number prefix (character)
	CNTRYZ	Country Code (character)
Series Variables	Series variable names are the same as the mnemonic name of the element given on the element 'E' record. They are taken from the 12 byte time series 'T' record time series indicative.	
Series Renamed	OLDNAME	NEWNAME
	p0discgdpe	p0digdpe
	doll2gdpe	dol2gdpe
	doll3gdpe	dol3gdpe
	doll1gdpe	dol1gdpe
	ppp1gdpd	pp1gdpd
	ppp1gdpd1	pp1gdpd1
	p0itxgdpc	p0itgdpc
	p0itxgdps	p0itgdps
	p0subgdpc	p0sugdpc
	p0subgdps	p0sugdps
	p0cfcgdpc	p0cfgdpc
	p0cfgddps	p0cfgdps
	p0discgdpc	p0dicgdc
	p0discgdps	p0dicgds
Missing Codes	A data value of * is interpreted as MISSING.	

FILETYPE=OECDQNA–Quarterly National Accounts

The QNA file contains the main aggregates of quarterly national accounts for 16 OECD Member Countries and on a selected number of aggregates for 4 groups of member countries: OECD-Total, OECD-Europe, EEC, and the 7 major countries.

Table 13.32 FILETYPE=OECDQNA–Quarterly National Accounts Format

Metadata Field Types	Metadata Fields	Metadata Labels
Data Files	Database is stored on a single file.	
INTERVAL=	QUARTER(default),YEAR	

Table 13.32 FILETYPE=OECDQNA–Quarterly National Accounts Format continued)

Metadata Field Types	Metadata Fields	Metadata Labels
BY Variables	COUNTRY	Country Code (character)
	SEASON	Seasonality S=seasonally adjusted 0=raw data, not seasonally adjusted
	PRICETAG	Prices C=data at current prices R,L,M=data at constant prices P,K,J,V=implicit price index or volume index
Series Variables	Subject code used to distinguish series within countries. Series variables are prefixed by _ for data, C for control codes, and D for relative date.	
Default DROP List	By default all the control codes and relative dates will be dropped.	
Missing Codes	A data value of + or - is interpreted as MISSING.	

FILETYPE=OECDMEI–Main Economic Indicators

The MEI file contains all series found in Parts 1 and 2 of the publication *Main Economic Indicators*.

Table 13.33 FILETYPE=OECDMEI–Main Economic Indicators Format

Metadata Field Types	Metadata Fields	Metadata Labels
Data Files	Database is stored on a single file.	
INTERVAL=	YEAR(default),QUARTER,MONTH	
BY Variables	COUNTRY	Country Code (character)
	CURRENCY	Unit of expression of the series.
	ADJUST	Adjustment 0,H,S,A,L=no adjustment 1,I=calendar or working day adjusted 2,B,J,M=seasonally adjusted by National Authorities 3,K,D=seasonally adjusted by OECD
Series Variables	Series variables are prefixed by _ for data, C for control codes, and D for relative date in weeks since last updated.	
Default DROP List	By default, all the control codes and relative dates will be dropped.	
Missing Codes	A data value of + or - is interpreted as MISSING.	

Examples: DATASOURCE Procedure

Example 13.1: BEA National Income and Product Accounts

In this example, exports and imports of goods and services are extracted to demonstrate how to work with a National Income and Product Accounts (NIPA) file.

From the "Statistical Tables" published by the United States Department of Commerce, Bureau of Economic Analysis, the relation of foreign transactions in the Balance of Payments Accounts (BPA) are given in the fifth table (TABNUM='05') of the "Foreign Transactions" section (PARTNO='4'). Moreover, the first line in the table gives BPAs, while the eighth gives exports of goods and services. The series names __00100 and __00800, are constructed by two underscores followed by three digits as the line numbers, and then two digits as the column numbers.

The following statements put this information together to extract quarterly BPAs and exports from a BEANIPA type file:

```
/*- assign fileref to the external file to be processed --------*/

filename ascifile "%sysget(DATASRC_DATA)beanipa.data" recfm=v lrecl=108;

title1 'Relation of Foreign Transactions to Balance of Payment Accounts';
title2 'Range from 1984 to 1989';

title3 'Annual';
proc datasource filetype=beanipa infile=ascifile
                interval=year
                outselect=off
                outkey=byfor4;

   range from 1984 to 1989;
   keep __00100 __00800;

   label __00100='Balance of Payment Accounts';
   label __00800='Exports of Goods and Services';

   rename __00100=BPAs __00800=exports;
run;

proc print data=byfor4;
run;

 /*- assign fileref to the external file to be processed --------*/

filename ascifile "%sysget(DATASRC_DATA)beanipa.data" recfm=v lrecl=108;

title1 'Relation of Foreign Transactions to Balance of Payment Accounts';
title2 'Range from 1984 to 1989';

title3 'Annual';
```

```
proc datasource filetype=beanipa infile=ascifile
                interval=year
                outselect=off
                outkey=byfor4
                out=foreign4;

   range from 1984 to 1989;
   keep __00100 __00800;

   label __00100='Balance of Payment Accounts';
   label __00800='Exports of Goods and Services';

   rename __00100=BPAs __00800=exports;

run;

proc contents data=foreign4;
run;
proc print data=foreign4;
run;
```

The results are shown in Output 13.1.1, Output 13.1.2, and Output 13.1.3.

Output 13.1.1 Listing of OUTBY=byfor4 of the BEANIPA Data

Relation of Foreign Transactions to Balance of Payment Accounts
Range from 1984 to 1989
Annual

Obs	PARTNO	TABNUM	ST_DATE	END_DATE	NTIME	NOBS	NINRANGE	NSERIES	NSELECT
1	1	07	1929	1989	61	0	6	2	0
2	1	14	1929	1989	61	0	6	1	0
3	1	15	1929	1989	61	0	6	1	0
4	1	20	1967	1989	23	23	6	2	1
5	1	23	1929	1989	61	0	6	2	0
6	2	04	1929	1989	61	0	6	1	0
7	2	05	1929	1989	61	0	6	2	0
8	3	05	1929	1989	61	0	6	1	0
9	3	14	1952	1989	38	0	6	2	0
10	3	15	1952	1989	38	0	6	7	0
11	3	16	1952	1989	38	0	6	1	0
12	4	05	1946	1989	44	44	6	1	1
13	5	07	1929	1989	61	0	6	1	0
14	5	09	1929	1989	61	0	6	1	0
15	6	04	1929	1989	61	0	6	3	0
16	6	05	1929	1948	20	0	0	2	0
17	6	07	1929	1948	20	0	0	1	0
18	6	08	1929	1989	61	0	6	3	0
19	6	09	1948	1989	42	0	6	1	0
20	6	10	1929	1948	20	0	0	1	0
21	6	14	1929	1948	20	0	0	1	0
22	6	19	1929	1948	20	0	0	1	0
23	6	20	1929	1989	61	0	6	2	0
24	6	22	1929	1989	61	0	6	2	0
25	6	23	1948	1989	42	0	6	1	0
26	6	24	1948	1989	42	0	6	1	0
27	7	09	1929	1989	61	0	6	1	0
28	7	10	1929	1989	61	0	6	2	0
29	7	13	1959	1989	31	0	6	1	0

Output 13.1.2 CONTENTS of OUT=foreign4 of the BEANIPA Data

Relation of Foreign Transactions to Balance of Payment Accounts
Range from 1984 to 1989
Annual

The CONTENTS Procedure

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Label
3	DATE	Num	4	YEAR4.	Date of Observation
1	PARTNO	Char	1		Part Number of Publication, IntegerPortion of the Table Number, 1-9
2	TABNUM	Char	2		Table Number Within Part, DecimalPortion of the Table Number, 1-24
4	exports	Num	5		Exports of Goods and Services

Output 13.1.3 Listing of OUT=foreign4 of the BEANIPA Data

Relation of Foreign Transactions to Balance of Payment Accounts
Range from 1984 to 1989
Annual

Obs	PARTNO	TABNUM	DATE	exports
1	1	20	1984	44
2	1	20	1985	53
3	1	20	1986	46
4	1	20	1987	40
5	1	20	1988	48
6	1	20	1989	47
7	4	05	1984	3835
8	4	05	1985	3709
9	4	05	1986	3965
10	4	05	1987	4496
11	4	05	1988	5520
12	4	05	1989	6262

This example illustrates the following features:

- You need to know the series variables names used by a particular vendor in order to construct the KEEP statement.
- You need to know the BY-variable names and their values for the required cross sections.
- You can use RENAME and LABEL statements to associate more meaningful names and labels with your selected series variables.

Example 13.2: BLS Consumer Price Index Surveys

This example compares changes of the prices in medical care services with respect to different regions for all urban consumers (SURVEY='CU') since May 1975. The source of the data is the Consumer Price Index Surveys distributed by the U.S. Department of Labor, Bureau of Labor Statistics.

An initial run of PROC DATASOURCE gives the descriptive information on different regions available (the OUTBY= data set), as well as the series variable name corresponding to medical care services (the OUTCONT= data set).

```
options yearcutoff = 1900;

filename datafile "%sysget(DATASRC_DATA)blscpi1.data" recfm=v lrecl=152;
proc datasource filetype=blscpi
   interval=mon
   outselect=off
   outby=cpikey(where=( upcase(areaname)
                  in ('NORTHEAST','NORTH CENTRAL','SOUTH','WEST')) )
   outcont=cpicont(where= ( index( upcase(label), 'MEDICAL CARE' )) );
   where survey='CU';
run;
```

```
title1 'OUTBY= Data Set, By AREANAME Selection';
proc print
   data=cpikey;
run;

title1 'OUTCONT= Data Set, By LABEL Selection';
proc print
   data=cpicont;
run;
```

The OUTBY= data set in Output 13.2.1 lists all cross sections available for the four geographical regions: Northeast (AREA='0100'), North Central (AREA='0200'), Southern (AREA='0300'), and Western (AREA='0400'). The OUTCONT= data set in Output 13.2.2 gives the variable names for medical care related series.

Output 13.2.1 Partial Listings of the OUTBY= Data Set

OUTBY= Data Set, By AREANAME Selection

Obs	SURVEY	SEASON	AREA	BASPTYPE	BASEPER	BYSELECT	ST_DATE
1	CU	U	0200	S	1982-84=100	1	DEC1977
2	CU	U	0100	S	1982-84=100	1	.
3	CW	U	0400	S	1982-84=100	0	DEC1977
4	CW	U	0100	S	1982-84=100	0	.
5	CW	U	0200	S	1982-84=100	0	.

Obs	END_DATE	NTIME	NOBS	NSERIES	NSELECT	SURTITLE	AREANAME
1	JUL1990	152	152	2	2	ALL URBAN CONSUM	NORTH CENTRAL
2	.	.	0	0	0	ALL URBAN CONSUM	NORTHEAST
3	JUL1990	152	0	1	0	URBAN WAGE EARN	WEST
4	.	.	0	0	0	URBAN WAGE EARN	NORTHEAST
5	.	.	0	0	0	URBAN WAGE EARN	NORTH CENTRAL

Output 13.2.2 Partial Listings of the OUTCONT= Data Set

OUTCONT= Data Set, By LABEL Selection

Obs	NAME	SELECTED	TYPE	LENGTH	VARNUM	LABEL	FORMAT	FORMATL	FORMATD
1	ASL5	1	1	5	.	SERVICES LESS MEDICAL CARE		0	0
2	A512	1	1	5	.	MEDICAL CARE SERVICES		0	0
3	A0L5	0	1	5	.	ALL ITEMS LESS MEDICAL CARE		0	0

The following statements make use of this information to extract the data for A512 and descriptive information on cross sections containing A512. Output 13.2.3 and Output 13.2.4 show these results.

```
options yearcutoff = 1900;

filename datafile "%sysget(DATASRC_DATA)blscpi1.data" recfm=v lrecl=152;

proc format;
   value $areafmt '0100' = 'Northeast Region'
                  '0200' = 'North Central Region'
                  '0300' = 'Southern Region'
                  '0400' = 'Western Region';
run;

proc datasource filetype=blscpi interval=month
             out=medical outall=medinfo;
   where survey='CU' and area in ( '0100','0200','0300','0400' );
   keep date a512;
   range  from 1988:9;
   format area $areafmt.;
   rename a512=medcare;
run;

title1 'Information on Medical Care Service, OUTALL= Data Set';
proc print
   data=medinfo;
run;

title1 'Medical Care Service By Region, OUT= Data Set';
title2 'Range from September, 1988';
proc print
   data=medical;
run;
```

Output 13.2.3 Printout of the OUTALL= Data Set

Information on Medical Care Service, OUTALL= Data Set

Obs	SURVEY	SEASON	AREA	BASPTYPE	BASEPER	BYSELECT	NAME	KEPT	SELECTED	TYPE
1	CU	U	North Central Region	S	1982-84=100	1	medcare	1	1	1

Obs	LENGTH	VARNUM	BLKNUM	LABEL	FORMAT	FORMATL	FORMATD	ST_DATE	END_DATE	NTIME
1	5	7	50	MEDICAL CARE SERVICES		0	0	DEC1977	JUL1990	152

Obs	NOBS	NINRANGE	SURTITLE	AREANAME	S_CODE	UNITS	NDEC
1	152	23	ALL URBAN CONSUM	NORTH CENTRAL	CUUR0200SA512		1

Output 13.2.4 Printout of the OUT= Data Set

Medical Care Service By Region, OUT= Data Set
Range from September, 1988

Obs	SURVEY	SEASON	AREA	BASPTYPE	BASEPER	DATE	medcare
1	CU	U	North Central Region	S	1982-84=100	SEP1988	1364
2	CU	U	North Central Region	S	1982-84=100	OCT1988	1365
3	CU	U	North Central Region	S	1982-84=100	NOV1988	1368
4	CU	U	North Central Region	S	1982-84=100	DEC1988	1372
5	CU	U	North Central Region	S	1982-84=100	JAN1989	1387
6	CU	U	North Central Region	S	1982-84=100	FEB1989	1399
7	CU	U	North Central Region	S	1982-84=100	MAR1989	1405
8	CU	U	North Central Region	S	1982-84=100	APR1989	1413
9	CU	U	North Central Region	S	1982-84=100	MAY1989	1416
10	CU	U	North Central Region	S	1982-84=100	JUN1989	1425
11	CU	U	North Central Region	S	1982-84=100	JUL1989	1439
12	CU	U	North Central Region	S	1982-84=100	AUG1989	1452
13	CU	U	North Central Region	S	1982-84=100	SEP1989	1460
14	CU	U	North Central Region	S	1982-84=100	OCT1989	1473
15	CU	U	North Central Region	S	1982-84=100	NOV1989	1481
16	CU	U	North Central Region	S	1982-84=100	DEC1989	1485
17	CU	U	North Central Region	S	1982-84=100	JAN1990	1500
18	CU	U	North Central Region	S	1982-84=100	FEB1990	1516
19	CU	U	North Central Region	S	1982-84=100	MAR1990	1528
20	CU	U	North Central Region	S	1982-84=100	APR1990	1538
21	CU	U	North Central Region	S	1982-84=100	MAY1990	1548
22	CU	U	North Central Region	S	1982-84=100	JUN1990	1557
23	CU	U	North Central Region	S	1982-84=100	JUL1990	1573

The OUTALL= data set in Output 13.2.3 indicates that data values are stored with one decimal place (see the NDEC variable). Therefore, they need to be rescaled, as follows:

```
data medical;
   set medical;
   medcare = medcare * 0.1;
run;
```

This example illustrates the following features:

- Descriptive information needed to write KEEP and WHERE statements can be obtained with an initial run of the DATASOURCE procedure.
- The OUTCONT= and OUTALL= data sets contain information on how data values are stored, such as the precision, the units, and so on.

- The OUTCONT= and OUTALL= data sets report the new series names assigned by the RENAME statement, not the old names (see the NAME variable in Output 13.2.3).
- You can use PROC FORMAT to define formats for series or BY variables to enhance your output. Note that PROC DATASOURCE associates a permanent format, $AREAFMT., with the BY variable AREA. As a result, the formatted values are displayed in the printout of the OUTALL=MEDINFO data set (see Output 13.2.3).

Example 13.3: BLS State and Area Employment, Hours, and Earnings Surveys

This example illustrates how to extract specific series from a State and Area Employment, Hours, and Earnings Survey. The series to be extracted is total employment in real estate and construction industries with respect to states from March 1989 to March 1990.

The State and Area, Employment, Hours and Earnings survey designates the totals for statewide figures by AREA='0000'.

The data type code for total employment is reported to be 1. Therefore, the series name for this variable is SA1, since series names are constructed by adding an SA prefix to the data type codes given by BLS.

Output 13.3.1 and Output 13.3.2 show statewide figures for total employment (SA1) in many industries from March 1989 through March 1990.

```
filename ascifile "%sysget(DATASRC_DATA)blseesa.dat" RECFM=F LRECL=152;
proc datasource  filetype=blseesa
                 infile=ascifile
                 outall=totkey
                 out=totemp;
   keep sa1;
   range from 1989:3 to 1990:3;
   rename sa1=totemp;
run;

title1 'Information on Total Employment, OUTALL= Data Set';
proc print data=totkey;
run;

title1 'Total Employment, OUT= Data Set';
proc print data=totemp;
run;
```

Output 13.3.1 Printout of the OUTALL= Data Set for All BY Groups

Information on Total Employment, OUTALL= Data Set

Obs	STATE	AREA	DIVISION	INDUSTRY	DETAIL	NAME	KEPT	SELECTED	TYPE	LENGTH	VARNUM	BLKNUM
1	5	2580	7	0000	1	totemp	1	1	1	5	7	3
2	6	0360	4	2039	6	totemp	1	1	1	5	7	6
3	6	6000	4	2300	2	totemp	1	1	1	5	7	7
4	6	7120	2	0000	1	totemp	1	1	1	5	7	8
5	10	0000	7	6102	6	totemp	1	1	1	5	7	10
6	11	8840	6	5600	2	totemp	1	1	1	5	7	11

Obs	LABEL	FORMAT	FORMATL	FORMATD	ST_DATE	END_DATE	NTIME	NOBS	NINRANGE	STATEABB
1	ALL EMP		0	0	JAN1970	JUN1990	246	246	13	AR
2	ALL EMP		0	0	JAN1972	JUN1990	222	222	13	CA
3	ALL EMP		0	0	JAN1972	JUN1990	222	222	13	CA
4	ALL EMP		0	0	JAN1957	DEC1987	372	372	0	CA
5	ALL EMP		0	0	JAN1984	DEC1987	48	48	0	DE
6	ALL EMP		0	0	JAN1972	JUN1990	222	222	13	DC

Obs	AREANAME	INDTITLE	S_CODE	SEASON	UNITS	NDEC
1	FAYETTEVILLE-SPRINGDALE	FINANCE, INSURANCE, AND REAL ESTATE	SAU0525807000011	U		1
2	ANAHEIM-SANTA ANA	CANNED, CURED, AND FROZEN FOODS	SAU0603604203961	U		1
3	OXNARD-VENTURA	APPAREL AND OTHER TEXTILE PRODUCTS	SAU0660004230021	U		1
4	SALINAS-SEASIDE-MONTEREY	CONSTRUCTION	SAU0671202000011	U		1
5	DELAWARE	NONDEPOS. INSTNS. & SEC. & COM. BRKRS.	SAU1000007610261	U		1
6	WASHINGTON MSA	APPAREL AND ACCESSORY STORES	SAU1188406560021	U		1

```
filename datafile "%sysget(DATASRC_DATA)blseesa.dat" RECFM=F LRECL=152;
proc datasource  filetype=blseesa
                 outall=totkey
                 out=totemp;
   where industry='0000';
   keep sa1;
   range from 1989:3 to 1990:3;
   rename sa1=totemp;
run;

title1 'Total Employment for Real Estate and Construction, OUT= Data Set';
proc print data=totemp;
run;
```

Output 13.3.2 Printout of the OUT= Data Set for INDUSTRY=0000

Total Employment for Real Estate and Construction, OUT= Data Set

Obs	STATE	AREA	DIVISION	INDUSTRY	DETAIL	DATE	totemp
1	5	2580	7	0000	1	MAR1989	16
2	5	2580	7	0000	1	APR1989	16
3	5	2580	7	0000	1	MAY1989	16
4	5	2580	7	0000	1	JUN1989	16
5	5	2580	7	0000	1	JUL1989	16
6	5	2580	7	0000	1	AUG1989	16
7	5	2580	7	0000	1	SEP1989	16
8	5	2580	7	0000	1	OCT1989	16
9	5	2580	7	0000	1	NOV1989	16
10	5	2580	7	0000	1	DEC1989	16
11	5	2580	7	0000	1	JAN1990	15
12	5	2580	7	0000	1	FEB1990	15
13	5	2580	7	0000	1	MAR1990	15

Note the following for this example:

- When the INFILE= option is omitted, the fileref assigned to the BLSEESA file is the default value DATAFILE.
- The FROM and TO values in the RANGE statement correspond to monthly data points since the INTERVAL= option defaults to MONTH for the BLSEESA filetype.

Example 13.4: DRI/McGraw-Hill Format CITIBASE Files

Output 13.4.1 and Output 13.4.2 illustrate how to extract weekly series from a sample CITIBASE file. They also demonstrate how the OUTSELECT= option affects the contents of the auxiliary data sets.

The weekly series contained in the sample data file CITIDEMO are listed by the following statements:

```
options yearcutoff=1920;

filename datafile "%sysget(DATASRC_DATA)citidem.dat" RECFM=D LRECL=80;

proc datasource filetype=citibase interval=week
                outall=citiall outby=citikey;
run;

title1 'Summary Information on Weekly Data for CITIDEMO File';
proc print data=citikey;
run;

title1 'Weekly Series Available in CITIDEMO File';
proc print data=citiall( drop=label );
run;
```

Output 13.4.1 Listing of the OUTBY= CITIKEY Data Set

Summary Information on Weekly Data for CITIDEMO File

Obs	ST_DATE	END_DATE	NTIME	NOBS	NSERIES	NSELECT
1	Sun, 29 Dec 1985	Sun, 3 Mar 1991	271	271	6	6

Output 13.4.2 Listing of the OUTALL= CITIALL Data Set

Weekly Series Available in CITIDEMO File

Obs	NAME	SELECTED	TYPE	LENGTH	VARNUM	BLKNUM	FORMAT	FORMATL
1	FF142B	1	1	5	.	36		0
2	WSPCA	1	1	5	.	37		0
3	WSPUA	1	1	5	.	38		0
4	WSPIA	1	1	5	.	39		0
5	WSPGLT	1	1	5	.	40		0
6	FCPOIL	1	1	5	.	41		0

Obs	FORMATD	ST_DATE	END_DATE	NTIME	NOBS	CODE	ATTRIBUT	NDEC
1	0	Sun, 29 Dec 1985	Sun, 3 Mar 1991	271	271	FF142B	1	2
2	0	Sun, 29 Dec 1985	Sun, 3 Mar 1991	271	271	WSPCA	1	2
3	0	Sun, 29 Dec 1985	Sun, 3 Mar 1991	271	271	WSPUA	1	2
4	0	Sun, 29 Dec 1985	Sun, 3 Mar 1991	271	271	WSPIA	1	2
5	0	Sun, 29 Dec 1985	Sun, 3 Mar 1991	271	271	WSPGLT	1	2
6	0	Sun, 29 Dec 1985	Sun, 3 Mar 1991	271	271	FCPOIL	1	4

Note the following from Output 13.4.2:

- The OUTALL= data set reports the time ranges of variables.
- There are six observations in the OUTALL= data set, the same number as reported by NSERIES and NSELECT variables in the OUTBY= data set.
- The VARNUM variable contains all MISSING values, since no OUT= data set is created.

Output 13.4.3 and Output 13.4.4 demonstrate how the OUTSELECT= option affects the contents of the OUTBY= and OUTALL= data sets when a KEEP statement is present. First, set the OUTSELECT= option to OFF.

```
filename citidemo "%sysget(DATASRC_DATA)citidem.dat" RECFM=D LRECL=80;

proc datasource filetype=citibase infile=citidemo interval=week
                outall=alloff outby=keyoff outselect=off;
   keep WSP:;
run;

title1 'Summary Information on Weekly Data for CITIDEMO File';
proc print data=keyoff;
run;

title1 'Weekly Series Available in CITIDEMO File';
proc print data=alloff( keep=name kept selected st_date
                            end_date ntime nobs );
run;
```

Output 13.4.3 Listing of the OUTBY= Data Set with OUTSELECT=OFF

Summary Information on Weekly Data for CITIDEMO File

Obs	ST_DATE	END_DATE	NTIME	NOBS	NSERIES	NSELECT
1	Sun, 29 Dec 1985	Sun, 3 Mar 1991	271	271	6	4

Output 13.4.4 Listing of the OUTALL= Data Set with OUTSELECT=OFF

Weekly Series Available in CITIDEMO File

Obs	NAME	KEPT	SELECTED	ST_DATE	END_DATE	NTIME	NOBS
1	FF142B	0	0	Sun, 29 Dec 1985	Sun, 3 Mar 1991	271	271
2	WSPCA	1	1	Sun, 29 Dec 1985	Sun, 3 Mar 1991	271	271
3	WSPUA	1	1	Sun, 29 Dec 1985	Sun, 3 Mar 1991	271	271
4	WSPIA	1	1	Sun, 29 Dec 1985	Sun, 3 Mar 1991	271	271
5	WSPGLT	1	1	Sun, 29 Dec 1985	Sun, 3 Mar 1991	271	271
6	FCPOIL	0	0	Sun, 29 Dec 1985	Sun, 3 Mar 1991	271	271

Setting the OUTSELECT= option ON gives results shown in Output 13.4.5 and Output 13.4.6.

```
filename citidemo "%sysget(DATASRC_DATA)citidem.dat" RECFM=D LRECL=80;
proc datasource filetype=citibase infile=citidemo
                interval=week
                outall=allon outby=keyon outselect=on;
   keep WSP:;
run;
```

```
title1 'Summary Information on Weekly Data for CITIDEMO File';
proc print data=keyon;
run;

title1 'Weekly Series Available in CITIDEMO File';
proc print data=allon( keep=name kept selected st_date
                           end_date ntime nobs );
run;
```

Output 13.4.5 Listing of the OUTBY= Data Set with OUTSELECT=ON

Summary Information on Weekly Data for CITIDEMO File

Obs	ST_DATE	END_DATE	NTIME	NOBS	NSERIES	NSELECT
1	Sun, 29 Dec 1985	Sun, 3 Mar 1991	271	271	6	4

Output 13.4.6 Listing of the OUTALL= Data Set with OUTSELECT=ON

Weekly Series Available in CITIDEMO File

Obs	NAME	KEPT	SELECTED	ST_DATE	END_DATE	NTIME	NOBS
1	WSPCA	1	1	Sun, 29 Dec 1985	Sun, 3 Mar 1991	271	271
2	WSPUA	1	1	Sun, 29 Dec 1985	Sun, 3 Mar 1991	271	271
3	WSPIA	1	1	Sun, 29 Dec 1985	Sun, 3 Mar 1991	271	271
4	WSPGLT	1	1	Sun, 29 Dec 1985	Sun, 3 Mar 1991	271	271

Comparison of Output 13.4.4 and Output 13.4.6 reveals the following:

- The OUTALL= data set contains six (NSERIES) observations when OUTSELECT=OFF, and four (NSELECT) observations when OUTSELECT=ON.
- The observations in OUTALL=ALLON are those for which SELECTED=1 in OUTALL=ALLOFF.
- The time ranges in the OUTBY= data set are computed over all the variables (selected or not) for OUTSELECT=OFF, but only computed over the selected variables for OUTSELECT=ON. This corresponds to computing time ranges over all the series reported in the OUTALL= data set.
- The variable NTIME is the number of time periods between ST_DATE and END_DATE, while NOBS is the number of observations the OUT= data set is to contain. Thus, NTIME is different depending on whether the OUTSELECT= option is set to ON or OFF, while NOBS stays the same.

The KEEP statement in the last two examples illustrates the use of an additional variable, KEPT, in the OUTALL= data sets of Output 13.4.4 and Output 13.4.6. KEPT, which reports the outcome of the KEEP statement, is only added to the OUTALL= data set when there is KEEP statement.

Adding the RANGE statement to the last example generates the data sets in Output 13.4.7 and Output 13.4.8:

```
filename citidemo "%sysget(DATASRC_DATA)citidem.dat" RECFM=D LRECL=80;
proc datasource filetype=citibase infile=citidemo interval=week
                outby=keyrange out=citiout outselect=on;
   keep WSP:;
   range from '01dec1990'd;
run;

title1 'Summary Information on Weekly Data for CITIDEMO File';
proc print data=keyrange;
run;

title1 'Weekly Data in CITIDEMO File';
proc print data=citiout;
run;
```

Output 13.4.7 Listing of the OUTBY=KEYRANGE Data Set for FILETYPE=CITIBASE

Summary Information on Weekly Data for CITIDEMO File

Obs	ST_DATE	END_DATE	NTIME	NOBS	NINRANGE	NSERIES	NSELECT
1	Sun, 29 Dec 1985	Sun, 3 Mar 1991	271	271	15	6	4

Output 13.4.8 Printout of the OUT=CITIOUT Data Set for FILETYPE=CITIBASE

Weekly Data in CITIDEMO File

Obs	DATE	WSPCA	WSPUA	WSPIA	WSPGLT
1	Sun, 25 Nov 1990	9.77000	9.66000	9.87000	8.62000
2	Sun, 2 Dec 1990	9.75000	9.64000	9.85000	8.47000
3	Sun, 9 Dec 1990	9.59000	9.48000	9.69000	8.22000
4	Sun, 16 Dec 1990	9.62000	9.51000	9.72000	8.35000
5	Sun, 23 Dec 1990	9.70000	9.60000	9.80000	8.48000
6	Sun, 30 Dec 1990	9.64000	9.53000	9.75000	8.31000
7	Sun, 6 Jan 1991	9.70000	9.59000	9.81000	8.62000
8	Sun, 13 Jan 1991	9.80000	9.70000	9.89000	8.58000
9	Sun, 20 Jan 1991	9.66000	9.57000	9.75000	8.36000
10	Sun, 27 Jan 1991	9.65000	9.56000	9.74000	8.38000
11	Sun, 3 Feb 1991	9.52000	9.43000	9.61000	8.16000
12	Sun, 10 Feb 1991	9.38000	9.29000	9.48000	8.14000
13	Sun, 17 Feb 1991	9.38000	9.29000	9.48000	8.21000
14	Sun, 24 Feb 1991	9.61000	9.53000	9.68000	8.50000
15	Sun, 3 Mar 1991	9.61000	9.53000	9.68000	8.50000

The OUTBY= data set in this last example contains an additional variable NINRANGE. This variable is added since there is a RANGE statement. Its value, 15, is the number of observations in the OUT= data set. In this case, NOBS gives the number of observations the OUT= data set would contain if there were not a RANGE statement.

Example 13.5: DRI Data Delivery Service Database

This example demonstrates the DRIDDS filetype for the daily Federal Reserve Series fxrates_dds. Use VALIDVARNAME=ANY in your SAS options statement to allow special characters such as @, $, and % to be in the series name. Note the use of long variable names in the OUT= data set in Output 13.5.2 and long labels in the OUTCONT= data set in Output 13.5.1.

The following statements extract daily series starting in January 1,1997:

```
options validvarname=any;
filename datafile "%sysget(DATASRC_DATA)drifxrat.dat" RECFM=F LRECL=80;
proc format;
   value distekfm 0 = 'Unspecified'
                  2 = 'Linear'
                  4 = 'Triag'
                  6 = 'Polynomial'
                  8 = 'Even'
                 10 = 'Step'
                 12 = 'Stocklast'
                 14 = 'LinearUnadjusted'
                 16 = 'PolyUnadjusted'
                 18 = 'StockWithNAS'
                 99 = 'None'
                255 = 'None';

   value convtkfm 0 = 'Unspecified'
                  1 = 'Average'
                  3 = 'AverageX'
                  5 = 'Sum'
                  7 = 'SumAnn'
                  9 = 'StockEnd'
                 11 = 'StockBegin'
                 13 = 'AvgNP'
                 15 = 'MaxNP'
                 17 = 'MinNP'
                 19 = 'StockEndNP'
                 21 = 'StockBeginNP'
                 23 = 'Max'
                 25 = 'Min'
                 27 = 'AvgXNP'
                 29 = 'SumNP'
                 31 = 'SumAnnNP'
                 99 = 'None'
                255 = 'None';

 /*-------------------------------------------------------*
  *                 process daily series                  *
  *-------------------------------------------------------*/
title3 'Reading DAILY Federal Reserve Series with fxrates_.dds';
proc datasource filetype=dridds
                infile=datafile
                interval=day
                out=fixr
```

```
                outcont=fixrcnt
                outall=fixrall;
   keep rx: ;
   range from '01jan97'd to '31dec99'd;
   format disttek distekfm.;
   format convtek convtkfm.;
run;

title1 'CONTENTS of FXRATES_.DDS File, KEEP RX:';
proc print
   data=fixrcnt;
run;

title1 'Daily Series Available in FXRATES_.DDS File, KEEP RX:';
proc print
   data=fixr;
run;
```

Output 13.5.1 Listing of the OUTCONT=FIXRCNT Data Set for FILETYPE=DRIDDS

CONTENTS of FXRATES_.DDS File, KEEP RX:

Obs	NAME	KEPT	SELECTED	TYPE	LENGTH	VARNUM	LABEL	FORMAT	FORMATL	FORMATD
1	RXA$%US$@AU	1	1	1	5	2	EXCHANGE RATE IN AUSTRALIAN DOLLAR PER US DOLLAR - AUSTRALIA		0	0
2	RXBF%US$@BE	1	1	1	5	3	EXCHANGE RATE IN BELGIAN FRANCS PER US DOLLAR - BELGIUM		0	0
3	RXDK%US$@DK	1	1	1	5	4	EXCHANGE RATE IN DANISH KRONE PER 100 US DOLLAR - DENMARK		0	0

Obs	SOURCEID	DISTTEK	CONVTEK	STATUS	UPDATE	UPTIME
1	@FACS/DATA.D	Unspecified	Unspecified	0	31JAN97	132605
2	@FACS/DATA.D	Unspecified	Unspecified	0	31JAN97	132544
3	@FACS/DATA.D	Unspecified	Unspecified	0	31JAN97	132544

Output 13.5.2 Printout of the OUT=FIXR Data Set for FILETYPE=DRIDDS

Daily Series Available in FXRATES_.DDS File, KEEP RX:

Obs	DATE	RXA$%US$@AU	RXBF%US$@BE	RXDK%US$@DK
1	01JAN1997	1.26133	31.9200	5.92877
2	02JAN1997	1.26133	31.9200	5.92877
3	03JAN1997	1.26133	31.9200	5.92877
4	04JAN1997	1.27708	32.4620	6.01098
5	05JAN1997	1.27708	32.4620	6.01098
6	06JAN1997	1.27708	32.4620	6.01098
7	07JAN1997	1.27708	32.4620	6.01098
8	08JAN1997	1.27708	32.4620	6.01098
9	09JAN1997	1.27708	32.4620	6.01098
10	10JAN1997	1.27708	32.4620	6.01098
11	11JAN1997	1.28443	32.9360	6.09112
12	12JAN1997	1.28443	32.9360	6.09112
13	13JAN1997	1.28443	32.9360	6.09112
14	14JAN1997	1.28443	32.9360	6.09112
15	15JAN1997	1.28443	32.9360	6.09112
16	16JAN1997	1.28443	32.9360	6.09112
17	17JAN1997	1.28443	32.9360	6.09112
18	18JAN1997	1.29195	33.7500	6.24658
19	19JAN1997	1.29195	33.7500	6.24658
20	20JAN1997	1.29195	33.7500	6.24658
21	21JAN1997	1.29195	33.7500	6.24658
22	22JAN1997	1.29195	33.7500	6.24658
23	23JAN1997	1.29195	33.7500	6.24658
24	24JAN1997	1.29195	33.7500	6.24658
25	25JAN1997	1.30133	33.8974	6.27520
26	26JAN1997	1.30133	33.8974	6.27520
27	27JAN1997	1.30133	33.8974	6.27520
28	28JAN1997	1.30133	33.8974	6.27520
29	29JAN1997	1.30133	33.8974	6.27520
30	30JAN1997	1.30133	33.8974	6.27520
31	31JAN1997	1.30133	33.8974	6.27520

Example 13.6: PC Format CITIBASE Database

This example uses a PC format CITIBASE database (FILETYPE=CITIDISK) to extract annual population estimates for females and males with respect to various age groups.

Population estimate series for all ages of females including those in the armed forces overseas are given by PANF, while PANM gives the population estimate for all ages of males including those in armed forces overseas. More population estimate time series are described in Output 13.6.1 and are output in Output 13.6.2.

The following statements extract the required population estimates series:

```
filename keyfile "%sysget(DATASRC_DATA)basekey.dat" RECFM=V LRECL=22;
filename indfile "%sysget(DATASRC_DATA)baseind.dat" RECFM=F LRECL=84;
filename dbfile "%sysget(DATASRC_DATA)basedb.dat" RECFM=F LRECL=4;

proc datasource filetype=citidisk  infile=( keyfile indfile dbfile )
                out=popest outall=popinfo;

run;

proc print data=popinfo;
run;
proc print data=popest;
run;
```

Output 13.6.1 Listing of the OUTALL=POPINFO Data Set for FILETYPE=CITIDISK

Daily Series Available in FXRATES_.DDS File, KEEP RX:

Obs	NAME	SELECTED	TYPE	LENGTH	VARNUM	BLKNUM	LABEL	FORMAT	FORMATL
1	PAN	1	1	5	2	1	POPULATION EST.: ALL AGES, INC.ARMED F. OVERSEAS(THOUS.,ANNUAL)		0
2	PAN17	1	1	5	3	2	POPULATION EST.: 16 YRS AND OVER,INC ARMED F.OVERSEAS(THOUS,ANNUAL)		0
3	PAN18	1	1	5	4	3	POPULATION EST.: 18-64 YRS,INC.ARMED F.OVERSEAS(THOUS,ANNUAL)		0
4	PANF	1	1	5	5	4	POPULATION EST.: FEMALES,ALL AGES,INC.ARMED F.O'SEAS(THOUS.,ANN)		0
5	PANM	1	1	5	6	5	POPULATION EST.: MALES, ALL AGES, INC.ARMED F.O'SEAS(THOUS.,ANN)		0

Obs	FORMATD	ST_DATE	END_DATE	NTIME	NOBS	DISKNUM	ATTRIBUT	NDEC	AGGREGAT
1	0	1980	1989	10	10	1	1	0	0
2	0	1980	1989	10	10	1	1	0	0
3	0	1980	1989	10	10	1	1	0	0
4	0	1980	1989	10	10	1	1	0	0
5	0	1980	1989	10	10	1	1	0	0

Output 13.6.2 Printout of the OUT=POPEST Data Set for FILETYPE=CITIDISK

Daily Series Available in FXRATES_.DDS File, KEEP RX:

Obs	DATE	PAN	PAN17	PAN18	PANF	PANM
1	1980	227757	172456	138358	116869	110888
2	1981	230138	175017	140618	118074	112064
3	1982	232520	177346	142740	119275	113245
4	1983	234799	179480	144591	120414	114385
5	1984	237001	181514	146257	121507	115494
6	1985	239279	183583	147759	122631	116648
7	1986	241625	185766	149149	123795	117830
8	1987	243942	187988	150542	124945	118997
9	1988	246307	189867	152113	126118	120189
10	1989	248762	191570	153695	127317	121445

This example demonstrates the following:

- The INFILE= options lists the filerefs of the key, index, and database files, in that order.
- The INTERVAL= option is omitted since the default interval for CITIDISK type files is YEAR.

Example 13.7: Quarterly COMPUSTAT Data Files

This example shows how to extract data from a 48-quarter Compustat Database File. For COMPUSTAT data files, the series variable names are constructed by concatenating the name of the data array DATA and the column number containing the required information. For example, for quarterly files the common stock data is in column 56. Therefore, the variable name for this series is DATA56. Similarly, the series variable names for quarterly footnotes are constructed by adding the column number to the array name, QFTNT. For example, the variable name for common stock footnotes is QFTNT14 since the 14th column of the QFTNT array contains this information.

The following example extracts common stock series (DATA56) and its footnote (QFTNT14) for companies whose stocks are traded over-the-counter and not in the S&P 500 Index (ZLIST=06) and whose data reside in the over-the-counter file (FILE=06).

```
filename compstat "%sysget(DATASRC_DATA)csqibm.dat" recfm=s370v
   lrecl=4820 blksize=14476;
proc datasource filetype=cs48qibm  infile=compstat
                out=stocks outby=company;
   keep data56 qftnt14;
   rename data56=comstock  qftnt14=ftcomstk;
   label  data56='Common Stock'
          qftnt14='Footnote for Common Stock';
   range  from 1990:4;

run;

/*- add company name to the out= data set    */
data stocks;
   merge stocks company( keep=dnum cnum cic coname );
   by dnum cnum cic;
run;

title1 'Common Stocks for Last Quarter of 1990';
proc print data=stocks ;
run;
```

Output 13.7.1 contains a listing of the STOCKS data set.

Output 13.7.1 Listing of the OUT=STOCKS Data Set

Common Stocks for Last Quarter of 1990

Obs	DNUM	CNUM	CIC	FILE	EIN	STK	SMBL	ZLIST	XREL	FIC
1	2670	293308	102	6	56-0481457	0	ENGH	6	0	0
2	2835	372917	104	6	06-1047163	0	GENZ	6	0	0
3	3564	896726	106	6	25-0922753	0	TRON	6	0	0
4	3576	172755	100	6	77-0024818	0	CRUS	6	0	0
5	3577	602191	108	6	11-2693062	0	MILT	6	0	0
6	3630	616350	104	6	34-0299600	0	MORF	6	0	0
7	3674	827079	203	6	94-1527868	0	SILI	6	0	0
8	3842	602720	104	6	25-0668780	0	MNES	6	0	0
9	5080	007698	103	6	59-1001822	0	AESM	6	0	0
10	5122	090324	104	6	84-0601662	0	BIND	6	0	0
11	5211	977865	104	6	38-1746752	0	WLHN	6	0	0
12	5600	299155	101	6	36-1050870	0	EVAN	6	0	0
13	5731	382091	106	6	94-2366177	0	GGUY	6	0	0
14	7372	45812M	104	6	94-2658153	0	INTS	6	0	0
15	7372	566140	109	6	04-2711580	0	MCAM	6	0	0
16	7373	913077	103	6	81-0422894	0	TOTE	6	0	0
17	7510	008450	108	6	34-1050582	0	AGNC	6	0	0
18	7819	026038	307	6	23-2359277	0	AFTI	6	0	0
19	8700	055383	103	6	59-1781257	0	BEIH	6	0	0
20	8731	759916	109	6	04-2729386	0	RGEN	6	0	0

Obs	INCORP	STATE	COUNTY	DATE	comstock	ftcomstk	CONAME
1	10	13	121	1990:4	16.2510		ENGRAPH INC
2	10	25	17	1990:4	0.1620		GENZYME CORP
3	42	37	105	1990:4	3.1380		TRION INC
4	6	6	85	1990:4	.		CIRRUS LOGIC INC
5	10	36	103	1990:4	.		MILTOPE GROUP INC
6	39	39	35	1990:4	.		MOR-FLO INDS
7	10	6	85	1990:4	.		SILICONIX INC
8	42	42	3	1990:4	6.7540		MINE SAFETY APPLIANCES CO
9	12	12	25	1990:4	.		AERO SYSTEMS INC
10	18	18	97	1990:4	3.2660		BINDLEY WESTERN INDS
11	26	26	145	1990:4	6.4800		WOLOHAN LUMBER CO
12	10	17	31	1990:4	.		EVANS INC
13	6	6	75	1990:4	0.0520		GOOD GUYS INC
14	6	6	85	1990:4	.		INTEGRATED SYSTEMS INC
15	25	25	17	1990:4	0.0770		MARCAM CORPORATION
16	10	30	111	1990:4	0.0570		UNITED TOTE INC
17	10	39	35	1990:4	.		AGENCY RENT-A-CAR INC
18	10	42	45	1990:4	0.0210		AMERICAN FILM TECHNOL
19	10	13	121	1990:4	0.5170		BEI HOLDINGS LTD
20	10	25	17	1990:4	.		REPLIGEN CORP

Note that quarterly Compustat data are also available in Universal Character format. If you have this type of file instead of IBM 360/370 General format, use the FILETYPE=CS48QUC option instead.

Example 13.8: Annual COMPUSTAT Data Files, V9.2 New Filetype CSAUC3

Annual COMPUSTAT data in Universal Character format is read for PRICES since the year 2002, so that the desired output show the PRICE (HIGH), PRICE (LOW), and PRICE (CLOSE) for each company.

```
filename datafile "%sysget(DATASRC_DATA)csaucy3.dat" RECFM=F LRECL=13612;
/*-------------------------------------------------------------*
 * create OUT=csauy3 data set with ASCII 2003 Industrial Data  *
 * compare it with the OUT=csauc data set created by DATA STEP *
 *-------------------------------------------------------------*/

proc datasource filetype=csaucy3 ascii
                infile=datafile
                interval=year
                outselect=on
                outkey=y3key
                out=csauy3;

     keep data197-data199 label;
     range from 2002;
run;

proc sort
   data=csauy3 out=csauy3;
   by dnum cnum cic file zlist smbl xrel stk;
run;

title1 'Price, High, Low and Close for Range from 2002';
proc contents data=csauy3;
run;

proc print data=csauy3;
run;
```

Output 13.8.1 shows information on the contents of the CSAUY3 data set while Output 13.8.2 shows a listing of the CSAUY3 data set.

Output 13.8.1 Listing of the CONTENTS of OUT=CSAUY3 Data Set

Price, High, Low and Close for Range from 2002

The CONTENTS Procedure

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Label
3	CIC	Char	3		
2	CNUM	Char	6		
11	COUNTY	Num	5		
13	CPSPIN	Char	1		
15	CSSPII	Char	1		
14	CSSPIN	Char	2		
18	DATA197	Num	5		Price - Fiscal Year - High ($&c,NA)
19	DATA198	Num	5		Price - Fiscal Year - Low ($&c,NA)
20	DATA199	Num	5		Price - Close - Fiscal Year-End ($&c,NA)
17	DATE	Num	4	YEAR4.	Date of Observation
1	DNUM	Num	5		
9	DUPFILE	Num	5		
16	EIN	Char	10		
4	FILE	Num	5		
12	FINC	Num	5		
6	SMBL	Char	8		
10	STATE	Num	5		
8	STK	Num	5		
7	XREL	Num	5		
5	ZLIST	Num	5		

Output 13.8.2 Listing of the OUT=CSAUY3 Data Set

Price, High, Low and Close for Range from 2002

Obs	DNUM	CNUM	CIC	FILE	ZLIST	SMBL	XREL	STK	DUPFILE	STATE	COUNTY	FINC	CPSPIN
1	3089	899896	104	11	1	TUP	444	0	0	12	95	0	1
2	3089	899896	104	11	1	TUP	444	0	0	12	95	0	1
3	3674	032654	105	11	1	ADI	928	0	0	25	21	0	1
4	3674	032654	105	11	1	ADI	928	0	0	25	21	0	1
5	3842	053801	106	1	5	AVR	0	0	0	25	21	0	
6	3842	053801	106	1	5	AVR	0	0	0	25	21	0	
7	6035	149547	101	3	25	CAVB	0	0	0	47	149	0	
8	6035	149547	101	3	25	CAVB	0	0	0	47	149	0	
9	6211	617446	448	11	1	MWD	725	0	0	36	61	0	1
10	6211	617446	448	11	1	MWD	725	0	0	36	61	0	1
11	6726	09247M	105	1	4	BMN	0	0	0	34	13	0	
12	6726	09247M	105	1	4	BMN	0	0	0	34	13	0	
13	7011	54021P	205	1	5	LGN	0	0	0	13	121	0	
14	7011	54021P	205	1	5	LGN	0	0	0	13	121	0	
15	7370	35921T	108	1	5	FNT	0	0	0	36	87	0	
16	7370	35921T	108	1	5	FNT	0	0	0	36	87	0	
17	7370	459200	101	11	1	IBM	903	0	0	36	119	0	1
18	7370	459200	101	11	1	IBM	903	0	0	36	119	0	1
19	7812	591610	100	1	4	MGM	0	0	0	6	37	0	
20	7812	591610	100	1	4	MGM	0	0	0	6	37	0	

Obs	CSSPIN	CSSPII	EIN	DATE	DATA197	DATA198	DATA199
1	10		36-4062333	2002	24.990	14.4000	15.0800
2	10		36-4062333	2003	.	.	.
3	10		04-2348234	2002	48.840	17.8800	26.8000
4	10		04-2348234	2003	.	.	.
5			06-1174053	2002	1.500	0.2200	0.2300
6			06-1174053	2003	.	.	.
7			62-1721072	2002	14.000	11.5810	13.3400
8			62-1721072	2003	.	.	.
9	10	1	36-3145972	2002	60.020	28.8010	45.2400
10	10	1	36-3145972	2003	.	.	.
11				2002	11.050	10.3700	11.0100
12				2003	.	.	.
13			52-2093696	2002	13.894	1.0084	13.8940
14			52-2093696	2003	.	.	.
15			13-3950283	2002	0.440	0.1200	0.2600
16			13-3950283	2003	.	.	.
17	10	1	13-0871985	2002	126.390	54.0100	77.5000
18	10	1	13-0871985	2003	.	.	.
19			95-4605850	2002	23.250	9.0000	13.0000
20			95-4605850	2003	.	.	.

Note that annual COMPUSTAT data are available in either IBM 360/370 General format or Universal Character format. The first example expects an IBM 360/370 General format file since the FILETYPE= is set to CSAIBM, while the second example uses a Universal Character format file (FILETYPE=CSAUC).

Example 13.9: CRSP Daily NYSE/AMEX Combined Stocks

This sample code reads all the data on a three-volume daily NYSE/AMEX combined character data set. Assume that the following filerefs are assigned to the calendar/indices file and security files that this database comprises:

Fileref	VOLSER	File Type
calfile	DXAA1	calendar/indices file on volume 1
secfile1	DXAA1	security file on volume 1
secfile2	DXAA2	security file on volume 2
secfile3	DXAA3	security file on volume 3

The data set CALDATA is created by the following statements to contain the calendar/indices file:

```
proc datasource filetype=crspdci infile=calfile out=caldata;
run;
```

Here the FILETYPE=CRSPDCI indicates that you are reading a character format (indicated by a C in the 6th position) daily (indicated by a D in the 5th position) calendar/indices file (indicated by an I in the 7th position).

The annual data in security files can be obtained by the following statements:

```
proc datasource filetype=crspdca
                infile=( secfile1 secfile2 secfile3 )
                out=annual;
run;
```

Similarly, the data sets to contain the daily security data (the OUT= data set) and the event data (the OUTEVENT= data set) are obtained by the following statements:

```
proc datasource filetype=crspdcs
                infile=( calfile secfile1 secfile2 secfile3 )
                out=periodic index outevent=events;
run;
```

Note that the FILETYPE= has an S in the 7th position, since you are reading the security files. Also, the INFILE= option first expects the fileref of the calendar/indices file since the dating variable (CALDT) is contained in that file. Following the fileref of calendar/indices file, you give the list of security files in the order in which you want to read them. When data span more than one physical volume, the filerefs of the security files residing on each volume must be given following the fileref of the calendar/indices file. The DATASOURCE procedure reads each of these files in the order in which they are specified. Therefore, you can request that all three volumes be mounted to the same drive, if you choose to do so.

This sample code illustrates the following points:

- The INDEX option in the second PROC DATASOURCE run creates an index file for the OUT=PERIODIC data set. This index file provides random access to the OUT= data set and may increase the efficiency of the subsequent PROC and DATA steps that use BY and WHERE statements. The index variables are CUSIP, CRSP permanent number (PERMNO), NASDAQ company number

(COMPNO), NASDAQ issue number (ISSUNO), header exchange code (HEXCD), and header SIC code (HSICCD). Each one of these variables forms a different key which is a single index. If you want to form keys from a combination of variables (composite indexes) or use some other variables as indexes, you should use the INDEX= data set option for the OUT= data set.

- The OUTEVENT=EVENTS data set is sparse. In fact, for each EVENT type, a unique set of event variables are defined. For example, for EVENT='SHARES', only the variables SHROUT and SHRFLG are defined, and they have missing values for all other EVENT types. Pictorially, this structure is similar to the data set shown in Figure 13.4. Because of this sparse representation, you should create the OUTEVENT= data set only when you need a subset of securities and events.

By default, the OUT= data set contains only the periodic data. However, you may also want to include the event-oriented data in the OUT= data set. This is accomplished by listing the event variables together with periodic variables in a KEEP statement. For example, if you want to extract the historical CUSIP (NCUSIP), number of shares outstanding (SHROUT), and dividend cash amount (DIVAMT) together with all the periodic series, use the following statements.

```
proc datasource filetype=crspdcs
                infile=( calfile secfile1 secfile2 secfile3 )
                out=both outevent=events;
   where cusip='09523220';
   keep  bidlo askhi prc vol ret sxret bxret ncusip shrout divamt;
run;
```

The KEEP statement has no effect on the event variables output to the OUTEVENT= data set. If you want to extract only a subset of event variables, you need to use the KEEPEVENT statement. For example, the following sample code outputs only NCUSIP and SHROUT to the OUTEVENT= data set for CUSIP='09523220':

```
proc datasource filetype=crspdxc
                infile=( calfile secfile)
                outevent=subevts;
   where cusip='09523220';
   keepevent  ncusip shrout;
run;
```

Output 13.9.1, Output 13.9.2, Output 13.9.3, and Output 13.9.4 show how to read the CRSP Daily NYSE/AMEX Combined ASCII Character Files.

```
filename dxci "%sysget(DATASRC_DATA)dxccal95.dat" RECFM=F LRECL=130;
filename dxc "%sysget(DATASRC_DATA)dxcsub95.dat" RECFM=F LRECL=400;

/*--- create output data sets from character format DX files ---*/
/*- create securities output data sets using DATASOURCE  -------*/
/*- statements                                                 -*/
proc datasource filetype=crspdcs  ascii
                infile=( dxci dxc )
                interval=day
                outcont=dxccont
                outkey=dxckey
                outall=dxcall
                out=dxc
```

```
                outevent=dxcevent
                outselect=off;
   range from '15aug95'd to '28aug95'd ;
   where cusip in ('12709510','35614220');
run;

title1 'Date Range 15aug95-28aug95 ';

title3 'DX Security File Outputs';
title4 'OUTKEY= Data Set';
proc print data=dxckey;
run;

title4 'OUTCONT= Data Set';
proc print data=dxccont;
run;

title4 "Listing of OUT= Data Set for cusip in ('12709510','35614220')";
proc print data=dxc;
run;

title4 "Listing of OUTEVENT= Data Set for cusip in ('12709510','35614220')";
proc print data=dxcevent;
run;
```

Output 13.9.1 Listing of the OUTBY= Data Set with OUTSELECT=OFF

Date Range 15aug95-28aug95

DX Security File Outputs
OUTKEY= Data Set

Obs	CUSIP	PERMNO	COMPNO	ISSUNO	HEXCD	HSICCD	BYSELECT	ST_DATE	END_DATE
1	68391610	10000	7952	9787	3	3990	0	07JAN1986	11JUN1987
2	12709510	10010	7967	9809	3	3840	1	17JAN1986	28AUG1995
3	49307510	10020	7972	9824	3	6710	0	27JAN1986	30APR1993
4	00338690	10030	22160	0	1	3310	0	02JUL1962	26DEC1968
5	41741F20	10040	7988	9846	3	6210	0	07FEB1986	15JUN1989
6	00074210	10050	13	11	3	3448	0	29DEC1972	16JUN1978
7	35614220	10060	8007	9876	3	1040	1	24FEB1986	29DEC1995

Obs	NTIME	NOBS	NINRANGE	NSERIES	NSELECT
1	521	0	0	35	7
2	3511	2431	10	35	7
3	2651	0	0	35	7
4	2370	0	0	35	7
5	1225	0	0	35	7
6	1996	0	0	35	7
7	3596	2492	10	35	7

Output 13.9.2 Listing of the OUTCONT= Data Set

Date Range 15aug95-28aug95

DX Security File Outputs
OUTCONT= Data Set

Obs	NAME	KEPT	SELECTED	TYPE	LENGTH	VARNUM	LABEL	FORMAT	FORMATL	FORMATD
1	BIDLO	1	1	1	6	8	Bid or Low		0	0
2	ASKHI	1	1	1	6	9	Ask or High		0	0
3	PRC	1	1	1	6	10	Closing Price of Bid/Ask average		0	0
4	VOL	1	1	1	6	11	Share Volume		0	0
5	RET	1	1	1	6	12	Holding Period Return		0	0
6	SXRET	1	1	1	6	13	Standard Deviation Excess Return		0	0
7	BXRET	1	1	1	6	14	Beta Excess Return		0	0
8	NCUSIP	0	0	2	8	.	Name CUSIP		0	0
9	TICKER	0	0	2	5	.	Exchange Ticker Symbol		0	0
10	COMNAM	0	0	2	32	.	Company Name		0	0
11	SHRCLS	0	0	2	1	.	Share Class		0	0
12	SHRCD	0	0	1	6	.	Share Code		0	0
13	EXCHCD	0	0	1	6	.	Exchange Code		0	0
14	SICCD	0	0	1	6	.	Standard Industrial Classification Code		0	0
15	DISTCD	0	0	1	6	.	Distribution Code		0	0
16	DIVAMT	0	0	1	6	.	Dividend Cash Amount		0	0
17	FACPR	0	0	1	6	.	Factor to adjust price		0	0
18	FACSHR	0	0	1	6	.	Factor to adjust shares outstanding		0	0
19	DCLRDT	0	0	1	6	.	Declaration date	DATE	7	0
20	RCRDDT	0	0	1	6	.	Record date	DATE	7	0
21	PAYDT	0	0	1	6	.	Payment date	DATE	7	0
22	SHROUT	0	0	1	6	.	Number of shares outstanding		0	0
23	SHRFLG	0	0	1	6	.	Share flag		0	0
24	DLSTCD	0	0	1	6	.	Delisting code		0	0
25	NWPERM	0	0	1	6	.	New CRSP permanent number		0	0
26	NEXTDT	0	0	1	6	.	Date of next available information	DATE	7	0
27	DLBID	0	0	1	6	.	Delisting bid		0	0
28	DLASK	0	0	1	6	.	Delisting ask		0	0
29	DLPRC	0	0	1	6	.	Delisting price		0	0
30	DLVOL	0	0	1	6	.	Delisting volume		0	0
31	DLRET	0	0	1	6	.	Delisting return		0	0
32	TRTSCD	0	0	1	6	.	Traits code		0	0
33	NMSIND	0	0	1	6	.	National Market System Indicator		0	0
34	MMCNT	0	0	1	6	.	Market maker count		0	0
35	NSDINX	0	0	1	6	.	NASD index		0	0

Output 13.9.3 Listing of the OUT= Data Set with OUTSELECT=OFF for CUSIPs 12709510 and 35614220

Date Range 15aug95-28aug95

DX Security File Outputs
Listing of OUT= Data Set for cusip in ('12709510','35614220')

Obs	CUSIP	PERMNO	COMPNO	ISSUNO	HEXCD	HSICCD	DATE	BIDLO	ASKHI
1	12709510	10010	7967	9809	3	3840	15AUG1995	7.500	7.8750
2	12709510	10010	7967	9809	3	3840	16AUG1995	7.500	7.8750
3	12709510	10010	7967	9809	3	3840	17AUG1995	7.500	7.8750
4	12709510	10010	7967	9809	3	3840	18AUG1995	7.375	7.5000
5	12709510	10010	7967	9809	3	3840	21AUG1995	7.375	7.3750
6	12709510	10010	7967	9809	3	3840	22AUG1995	7.250	7.3750
7	12709510	10010	7967	9809	3	3840	23AUG1995	7.250	7.3750
8	12709510	10010	7967	9809	3	3840	24AUG1995	7.125	7.5000
9	12709510	10010	7967	9809	3	3840	25AUG1995	6.875	7.3750
10	12709510	10010	7967	9809	3	3840	28AUG1995	7.000	7.1250
11	35614220	10060	8007	9876	3	1040	15AUG1995	12.375	12.6875
12	35614220	10060	8007	9876	3	1040	16AUG1995	12.125	12.3750
13	35614220	10060	8007	9876	3	1040	17AUG1995	12.250	12.3125
14	35614220	10060	8007	9876	3	1040	18AUG1995	12.250	12.6250
15	35614220	10060	8007	9876	3	1040	21AUG1995	12.375	12.6250
16	35614220	10060	8007	9876	3	1040	22AUG1995	12.250	12.3750
17	35614220	10060	8007	9876	3	1040	23AUG1995	12.125	12.2500
18	35614220	10060	8007	9876	3	1040	24AUG1995	12.125	12.3750
19	35614220	10060	8007	9876	3	1040	25AUG1995	12.000	12.2500
20	35614220	10060	8007	9876	3	1040	28AUG1995	12.000	12.0625

Obs	PRC	VOL	RET	SXRET	BXRET
1	7.5625	29200	-0.008197	.	.
2	7.5000	22365	-0.008264	.	.
3	7.5000	33416	0.000000	.	.
4	7.3750	16666	-0.016667	.	.
5	7.3750	9382	0.000000	.	.
6	7.2500	33674	-0.016949	.	.
7	7.3125	22371	0.008621	.	.
8	7.1250	38621	-0.025641	.	.
9	7.0000	29713	-0.017544	.	.
10	7.0000	38798	0.000000	.	.
11	12.3750	39136	0.000000	.	.
12	12.2031	45916	-0.013889	.	.
13	12.2500	43644	0.003841	.	.
14	12.3750	11027	0.010204	.	.
15	12.3750	7378	0.000000	.	.
16	12.2500	99655	-0.010101	.	.
17	12.1250	95148	-0.010204	.	.
18	12.3750	185572	0.020619	.	.
19	12.0000	9575	-0.030303	.	.
20	12.0625	12854	0.005208	.	.

Output 13.9.4 Listing of the OUTEVENT= Data Set in Range 15aug95-28aug95

Date Range 15aug95-28aug95

DX Security File Outputs
Listing of OUTEVENT= Data Set for cusip in ('12709510','35614220')

Obs	CUSIP	PERMNO	COMPNO	ISSUNO	HEXCD	HSICCD	EVENT	DATE	NCUSIP	TICKER	COMNAM	SHRCLS
1	12709510	10010	7967	9809	3	3840	DELIST	28AUG1995				
2	12709510	10010	7967	9809	3	3840	NASDIN	24AUG1995				

Obs	SHRCD	EXCHCD	SICCD	DISTCD	DIVAMT	FACPR	FACSHR	DCLRDT	RCRDDT	PAYDT	SHROUT	SHRFLG
1	.	.	.	.	.	.	.	.	.	.	.	.
2	.	.	.	.	.	.	.	.	.	.	.	.

Obs	DLSTCD	NWPERM	NEXTDT	DLBID	DLASK	DLPRC	DLVOL	DLRET	TRTSCD	NMSIND	MMCNT	NSDINX
1	203	23588	.	.	.	0	.	0.037500	.	.	.	.
2	.	.	.	.	.	.	.	.	1	2	17	2

Note in Output 13.9.4 that there were no events in range for cusip 35614220. See Chapter 46, "The SASECRSP Interface Engine," for more on CRSPAccess Data access.

References

Bureau of Economic Analysis (1986). *The National Income and Product Accounts of the United States, 1929–82*. Washington, DC: US Department of Commerce.

Bureau of Economic Analysis (1987). *Index of Items Appearing in the National Income and Product Accounts Tables*. Washington, DC: US Department of Commerce.

Bureau of Economic Analysis (1991). *Survey of Current Business*. Washington, DC: US Department of Commerce.

Bureau of Labor Statistics (2012). Washington, DC. `http://www.bls.gov/`.

Center for Research in Security Prices (2000). *CRSP SFA Guide*. Chicago: CRSP, Chicago Booth.

Center for Research in Security Prices (2006a). *CRSP Data Description Guide*. Chicago: CRSP, Chicago Booth.

Center for Research in Security Prices (2006b). *CRSP Fortran-77 to Fortran-95 Migration Guide*. Chicago: CRSP, Chicago Booth.

Center for Research in Security Prices (2006c). *CRSP Programmer's Guide*. Chicago: CRSP, Chicago Booth.

Center for Research in Security Prices (2006d). *CRSP Utilities Guide*. Chicago: CRSP, Chicago Booth.

Citibank (1990). *CITIBASE Directory*. New York: Citibank.

Citibank (1991a). *CITIBASE–Daily*. New York: Citibank.

Citibank (1991b). *CITIBASE–Weekly*. New York: Citibank.

DRI/McGraw-Hill (1997). *DataLink*. Lexington, MA: DRI/McGraw-Hill.

DRI/McGraw-Hill Data Search and Retrieval for Windows (1996). *DRIPRO User's Guide*. Lexington, MA: DRI/McGraw-Hill.

FAME Information Services (1995). *User's Guide to FAME*. Ann Arbor, MI: FAME Information Services.

Haver Analytics (2012). *DLX Database Profiles*. New York: Haver Analytics. Accessed November 27, 2012. http://www.haver.com/databaseprofiles.html.

International Monetary Fund (1984). *IMF Documentation on Computer Tape Subscription*. Washington, DC: IMF.

Organisation for Economic Co-operation and Development (1992a). *Annual National Accounts, Vol. 1: Main Aggregates Content Documentation for Magnetic Tape Subscription*. Paris: OECD.

Organisation for Economic Co-operation and Development (1992b). *Annual National Accounts, Vol. 2: Detailed Tables Technical Documentation for Magnetic Tape Subscription*. Paris: OECD.

Organisation for Economic Co-operation and Development (1992c). *Main Economic Indicators Database Note*. Paris: OECD.

Organisation for Economic Co-operation and Development (1992d). *OECD Statistical Information Research and Inquiry System Magnetic Tape Format Documentation*. Paris: OECD.

Organisation for Economic Co-operation and Development (1992e). *Quarterly National Accounts Inventory of Series Codes*. Paris: OECD.

Standard & Poor's Compustat Services (1991). *COMPUSTAT II Documentation*. Englewood, CO: Standard & Poor's.

Standard & Poor's Compustat Services (2003). *Compustat Technical Guide*. Englewood, CO: McGraw-Hill.

Chapter 14
The ENTROPY Procedure (Experimental)

Contents

Overview: ENTROPY Procedure **786**
Getting Started: ENTROPY Procedure **788**
- Simple Regression Analysis 788
- Using Prior Information 794
- Pure Inverse Problems 799
- Analyzing Multinomial Response Data 804

Syntax: ENTROPY Procedure **808**
- Functional Summary 808
- PROC ENTROPY Statement 810
- BOUNDS Statement 813
- BY Statement 815
- ID Statement 815
- MODEL Statement 815
- PRIORS Statement 816
- RESTRICT Statement 817
- TEST Statement 817
- WEIGHT Statement 819

Details: ENTROPY Procedure **819**
- Generalized Maximum Entropy 819
- Generalized Cross Entropy 820
- Moment Generalized Maximum Entropy 822
- Maximum Entropy-Based Seemingly Unrelated Regression 823
- Generalized Maximum Entropy for Multinomial Discrete Choice Models 825
- Censored or Truncated Dependent Variables 826
- Information Measures 827
- Parameter Covariance For GCE 828
- Parameter Covariance For GCE-M 828
- Statistical Tests 829
- Missing Values 829
- Input Data Sets 830
- Output Data Sets 831
- ODS Table Names 832
- ODS Graphics 832

Examples: ENTROPY Procedure **833**
- Example 14.1: Nonnormal Error Estimation 833

Example 14.2: Unreplicated Factorial Experiments 835
Example 14.3: Censored Data Models in PROC ENTROPY 838
Example 14.4: Use of the PDATA= Option 840
Example 14.5: Illustration of ODS Graphics 842
References **844**

Overview: ENTROPY Procedure

The ENTROPY procedure implements a parametric method of linear estimation based on generalized maximum entropy. The ENTROPY procedure is suitable when there are outliers in the data and robustness is required, when the model is ill-posed or under-determined for the observed data, or for regressions that involve small data sets.

The main features of the ENTROPY procedure are as follows:

- estimation of simultaneous systems of linear regression models
- estimation of Markov models
- estimation of seemingly unrelated regression (SUR) models
- estimation of unordered multinomial discrete Choice models
- solution of pure inverse problems
- allowance of bounds and restrictions on parameters
- performance of tests on parameters
- allowance of data and moment constrained generalized cross entropy

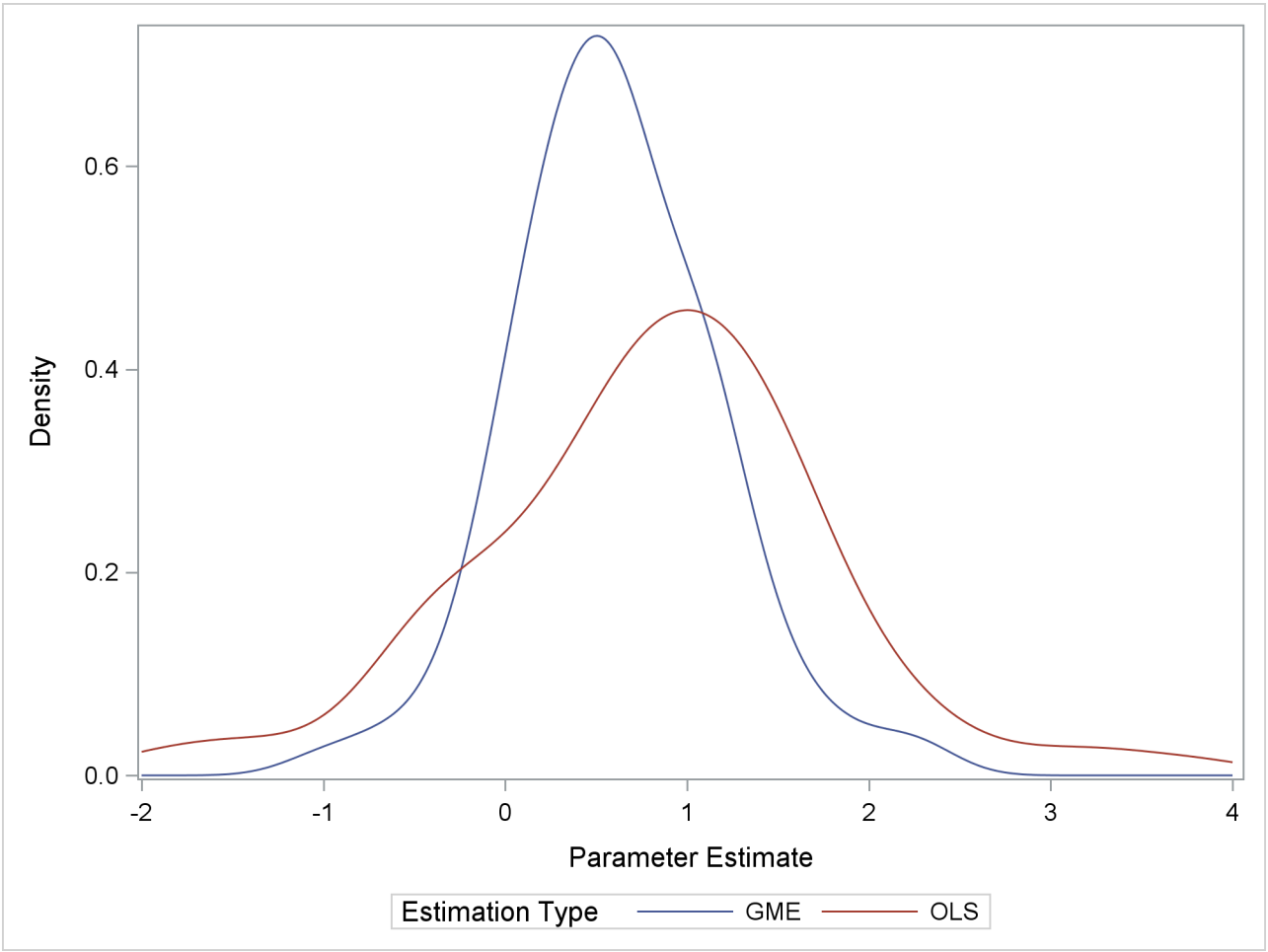
It is often the case that the statistical/economic model of interest is ill-posed or under-determined for the observed data. For the general linear model, this can imply that high degrees of collinearity exist among explanatory variables or that there are more parameters to estimate than observations available to estimate them. These conditions lead to high variances or non-estimability for traditional generalized least squares (GLS) estimates.

Under these situations it might be in the researcher's or practitioner's best interest to consider a nontraditional technique for model fitting. The principle of maximum entropy is the foundation for an estimation methodology that is characterized by its robustness to ill-conditioned designs and its ability to fit over-parameterized models. See Mittelhammer, Judge, and Miller (2000) and Golan, Judge, and Miller (1996) for a discussion of Shannon's maximum entropy measure and the related Kullback-Leibler information.

Generalized maximum entropy (GME) is a means of selecting among probability distributions to choose the distribution that maximizes uncertainty or uniformity remaining in the distribution, subject to information already known about the distribution. Information takes the form of data or moment constraints in the estimation procedure. PROC ENTROPY creates a GME distribution for each parameter in the linear model, based upon support points supplied by the user. The mean of each distribution is used as the estimate of the

parameter. Estimates tend to be biased, as they are a type of shrinkage estimate, but typically portray smaller variances than ordinary least squares (OLS) counterparts, making them more desirable from a mean squared error viewpoint (see Figure 14.1).

Figure 14.1 Distribution of Maximum Entropy Estimates versus OLS



Maximum entropy techniques are most widely used in the econometric and time series fields. Some important uses of maximum entropy include the following:

- size distribution of firms
- stationary Markov Process
- social accounting matrix (SAM)
- consumer brand preference
- exchange rate regimes
- wage-dependent firm relocation
- oil market dynamics

Getting Started: ENTROPY Procedure

This section introduces the ENTROPY procedure and shows how to use PROC ENTROPY for several kinds of statistical analyses.

Simple Regression Analysis

The ENTROPY procedure is similar in syntax to the other regression procedures in SAS. To demonstrate the similarity, suppose the endogenous/dependent variable is y, and x1 and x2 are two exogenous/independent variables of interest. To estimate the parameters in this single equation model using PROC ENTROPY, use the following SAS statements:

```
proc entropy;
   model y = x1 x2;
run;
```

Test Scores Data Set

Consider the following test score data compiled by Coleman et al. (1966):

```
title "Test Scores compiled by Coleman et al. (1966)";
data coleman;
   input test_score 6.2 teach_sal 6.2 prcnt_prof 8.2
            socio_stat 9.2 teach_score 8.2 mom_ed 7.2;
   label test_score="Average sixth grade test scores in observed district";
   label teach_sal="Average teacher salaries per student (1000s of dollars)";
   label prcnt_prof="Percent of students' fathers with professional employment";
   label socio_stat="Composite measure of socio-economic status in the district";
   label teach_score="Average verbal score for teachers";
   label mom_ed="Average level of education (years) of the students' mothers";
datalines;
37.01   3.83   28.87     7.20   26.60   6.19

   ... more lines ...
```

This data set contains outliers, and the condition number of the matrix of regressors, $\mathbf{X}$, is large, which indicates collinearity among the regressors. Since the maximum entropy estimates are both robust with respect to the outliers and also less sensitive to a high condition number of the $\mathbf{X}$ matrix, maximum entropy estimation is a good choice for this problem.

To fit a simple linear model to this data by using PROC ENTROPY, use the following statements:

```
proc entropy data=coleman;
   model test_score = teach_sal prcnt_prof socio_stat teach_score mom_ed;
run;
```

This requests the estimation of a linear model for TEST_SCORE with the following form:

$$\begin{aligned} test\_score &= intercept + a * teach\_sal + b * prcnt\_prof + c * socio\_stat \\ &\quad + d * teach\_score + e * mom\_ed + \epsilon; \end{aligned}$$

This estimation produces the "Model Summary" table in Figure 14.2, which shows the equation variables used in the estimation.

Figure 14.2 Model Summary Table

Test Scores compiled by Coleman et al. (1966)

The ENTROPY Procedure

Variables(Supports(Weights))	teach_sal prcnt_prof socio_stat teach_score mom_ed Intercept
Equations(Supports(Weights))	test_score

Since support points and prior weights are not specified in this example, they are not shown in the "Model Summary" table. The next four pieces of information displayed in Figure 14.3 are: the "Data Set Options," the "Minimization Summary," the "Final Information Measures," and the "Observations Processed."

Figure 14.3 Estimation Summary Tables

Test Scores compiled by Coleman et al. (1966)

The ENTROPY Procedure
GME Estimation Summary

Data Set Options
DATA= WORK.COLEMAN

Minimization Summary	
Parameters Estimated	6
Covariance Estimator	GME
Entropy Type	Shannon
Entropy Form	Dual
Numerical Optimizer	Quasi Newton

Final Information Measures	
Objective Function Value	9.553699
Signal Entropy	9.569484
Noise Entropy	-0.01578
Normed Entropy (Signal)	0.990976
Normed Entropy (Noise)	0.999786
Parameter Information Index	0.009024
Error Information Index	0.000214

Observations Processed	
Read	20
Used	20

The item labeled "Objective Function Value" is the value of the entropy estimation criterion for this estimation problem. This measure is analogous to the log-likelihood value in a maximum likelihood estimation. The "Parameter Information Index" and the "Error Information Index" are normalized entropy values that measure the proximity of the solution to the prior or target distributions.

The next table displayed is the ANOVA table, shown in Figure 14.4. This is in the same form as the ANOVA table for the MODEL procedure, since this is also a multivariate procedure.

Figure 14.4 Summary of Residual Errors

GME Summary of Residual Errors

Equation	DF Model	DF Error	SSE	MSE	Root MSE	R-Square	Adj RSq
test_score	6	14	175.8	8.7881	2.9645	0.7266	0.6290

The last table displayed is the "Parameter Estimates" table, shown in Figure 14.5. The difference between this parameter estimates table and the parameter estimates table produced by other regression procedures is that the standard error and the probabilities are labeled as approximate.

Figure 14.5 Parameter Estimates

GME Variable Estimates

Variable	Estimate	Approx Std Err	t Value	Approx Pr > |t|
teach_sal	0.287979	0.00551	52.26	<.0001
prcnt_prof	0.02266	0.00323	7.01	<.0001
socio_stat	0.199777	0.0308	6.48	<.0001
teach_score	0.497137	0.0180	27.61	<.0001
mom_ed	1.644472	0.0921	17.85	<.0001
Intercept	10.5021	0.3958	26.53	<.0001

The parameter estimates produced by the REG procedure for this same model are shown in Figure 14.6. Note that the parameters and standard errors from PROC REG are much different than estimates produced by PROC ENTROPY.

```
symbol v=dot h=1 c=green;

proc reg data=coleman;
   model test_score = teach_sal prcnt_prof socio_stat teach_score mom_ed;
   plot rstudent.*obs.
        / vref= -1.714 1.714 cvref=blue lvref=1
          HREF=0 to 30 by 5 cHREF=red  cframe=ligr;
run;
```

Figure 14.6 REG Procedure Parameter Estimates

Test Scores compiled by Coleman et al. (1966)

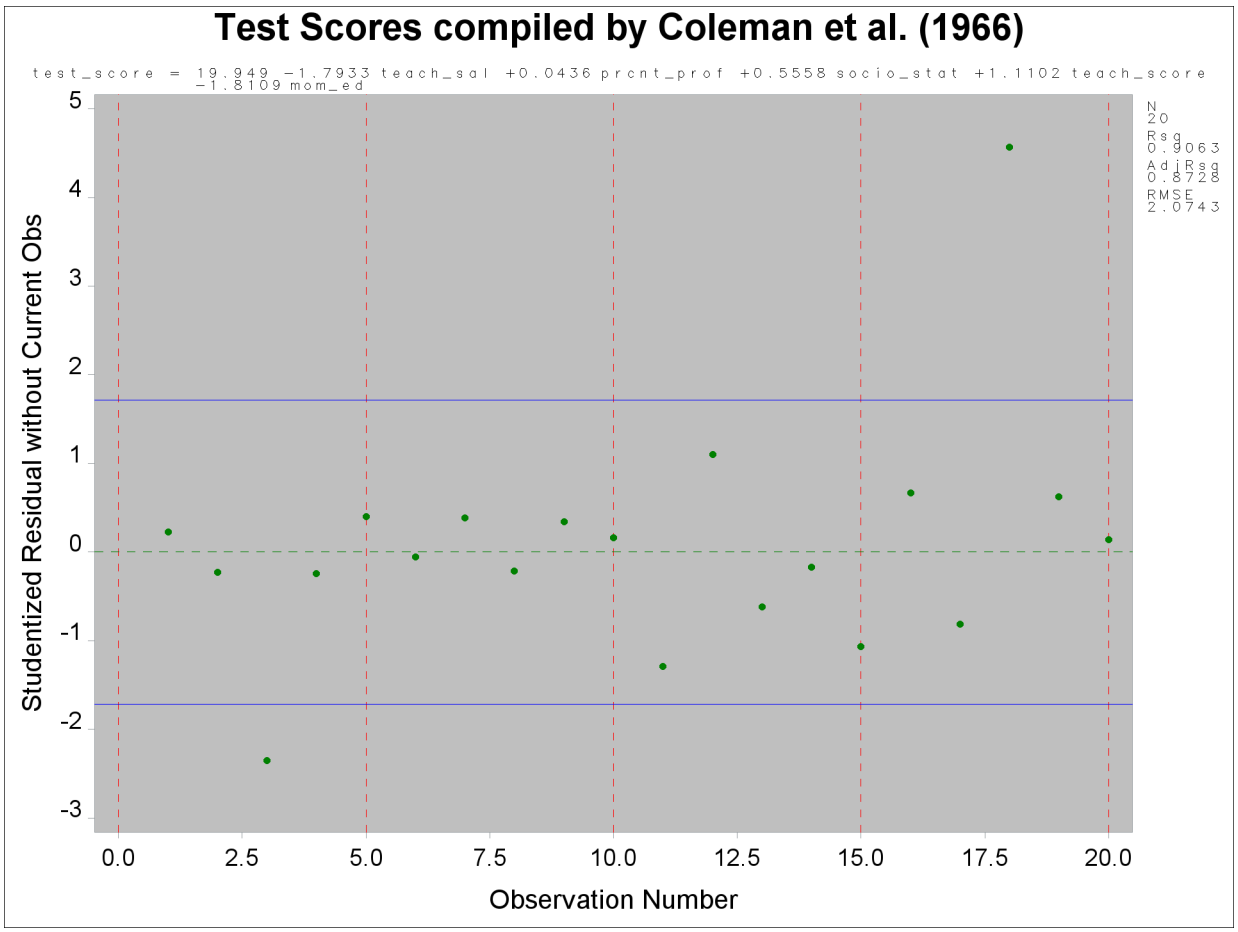
The REG Procedure
Model: MODEL1
Dependent Variable: test_score

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	19.94857	13.62755	1.46	0.1653
teach_sal	1	-1.79333	1.23340	-1.45	0.1680
prcnt_prof	1	0.04360	0.05326	0.82	0.4267
socio_stat	1	0.55576	0.09296	5.98	<.0001
teach_score	1	1.11017	0.43377	2.56	0.0227
mom_ed	1	-1.81092	2.02739	-0.89	0.3868

This data set contains two outliers, observations 3 and 18. These can be seen in a plot of the residuals shown in Figure 14.7

Figure 14.7 PROC REG Residuals with Outliers



The presence of outliers suggests that a robust estimator such as *M*-estimator in the ROBUSTREG procedure should be used. The following statements use the ROBUSTREG procedure to estimate the model.

```
proc robustreg data=coleman;
   model test_score = teach_sal prcnt_prof
                      socio_stat teach_score mom_ed;
run;
```

The results of the estimation are shown in Figure 14.8.

Figure 14.8 *M*-Estimation Results

Test Scores compiled by Coleman et al. (1966)

The ROBUSTREG Procedure

Parameter Estimates

Parameter	DF	Estimate	Standard Error	95% Confidence Limits		Chi-Square	Pr > ChiSq
Intercept	1	29.3416	6.0381	17.5072	41.1761	23.61	<.0001
teach_sal	1	-1.6329	0.5465	-2.7040	-0.5618	8.93	0.0028
prcnt_prof	1	0.0823	0.0236	0.0361	0.1286	12.17	0.0005
socio_stat	1	0.6653	0.0412	0.5846	0.7461	260.95	<.0001
teach_score	1	1.1744	0.1922	0.7977	1.5510	37.34	<.0001
mom_ed	1	-3.9706	0.8983	-5.7312	-2.2100	19.54	<.0001
Scale	1	0.6966					

Note that TEACH_SAL(VAR1) and MOM_ED(VAR5) change greatly when the robust estimation is used. Unfortunately, these two coefficients are negative, which implies that the test scores increase with decreasing teacher salaries and decreasing levels of the mother's education. Since ROBUSTREG is robust to outliers, they are not causing the counterintuitive parameter estimates.

The condition number of the regressor matrix $\mathbf{X}$ also plays a important role in parameter estimation. The condition number of the matrix can be obtained by specifying the COLLIN option in the PROC ENTROPY statement.

```
proc entropy data=coleman collin;
   model test_score = teach_sal prcnt_prof socio_stat teach_score mom_ed;
run;
```

The output produced by the COLLIN option is shown in Figure 14.9.

Figure 14.9 Collinearity Diagnostics

Test Scores compiled by Coleman et al. (1966)

The ENTROPY Procedure

Collinearity Diagnostics

Number	Eigenvalue	Condition Number	Proportion of Variation: teach_sal	prcnt_prof	socio_stat	teach_score	mom_ed	Intercept
1	4.978128	1.0000	0.0007	0.0012	0.0026	0.0001	0.0001	0.0000
2	0.937758	2.3040	0.0006	0.0028	0.2131	0.0001	0.0000	0.0001
3	0.066023	8.6833	0.0202	0.3529	0.6159	0.0011	0.0000	0.0003
4	0.016036	17.6191	0.7961	0.0317	0.0534	0.0059	0.0083	0.0099
5	0.001364	60.4112	0.1619	0.3242	0.0053	0.7987	0.3309	0.0282
6	0.000691	84.8501	0.0205	0.2874	0.1096	0.1942	0.6607	0.9614

The condition number of the $\mathbf{X}$ matrix is reported to be 84.85. This means that the condition number of $\mathbf{X}'\mathbf{X}$ is $84.85^2 = 7199.5$, which is very large.

Ridge regression can be used to offset some of the problems associated with ill-conditioned $\mathbf{X}$ matrices. Using the formula for the ridge value as

$$\lambda_R = \frac{kS^2}{\hat{\beta}'\hat{\beta}} \approx 0.9$$

where $\hat{\beta}$ and S^2 are the least squares estimators of β and σ^2 and $k = 6$. A ridge regression of the test score model was performed by using the data set with the outliers removed. The following PROC REG code performs the ridge regression:

```
data coleman;
   set coleman;
   if _n_ = 3 or _n_ = 18 then delete;
run;

proc reg data=coleman ridge=0.9 outest=t noprint;
   model test_score = teach_sal prcnt_prof socio_stat teach_score mom_ed;
run;

proc print data=t;
run;
```

The results of the estimation are shown in Figure 14.10.

Figure 14.10 Ridge Regression Estimates

Test Scores compiled by Coleman et al. (1966)

Obs	_MODEL_	_TYPE_	_DEPVAR_	_RIDGE_	_PCOMIT_	_RMSE_	Intercept	teach_sal
1	MODEL1	PARMS	test_score	.	.	0.78236	29.7577	-1.69854
2	MODEL1	RIDGE	test_score	0.9	.	3.19679	9.6698	-0.08892

Obs	prcnt_prof	socio_stat	teach_score	mom_ed	test_score
1	0.085118	0.66617	1.18400	-4.06675	-1
2	0.041889	0.23223	0.60041	1.32168	-1

Note that the ridge regression estimates are much closer to the estimates produced by the ENTROPY procedure that uses the original data set. Ridge regressions are not robust to outliers as maximum entropy estimates are. This might explain why the estimates still differ for TEACH_SAL.

Using Prior Information

You can use prior information about the parameters or the residuals to improve the efficiency of the estimates. Some authors prefer the terms *pre-sample* or *pre-data* over the term *prior* when used with maximum entropy to avoid confusion with Bayesian methods. The maximum entropy method described here does not use Bayes' rule when including prior information in the estimation.

To perform regression, the ENTROPY procedure uses a generalization of maximum entropy called *generalized maximum entropy*. In maximum entropy estimation, the unknowns are probabilities. Generalized maximum entropy expands the set of problems that can be solved by introducing the concept of *support points*.

Generalized maximum entropy still estimates probabilities, but these are the probabilities of a support point. Support points are used to map the $(0, 1)$ domain of the maximum entropy to the any finite range of values.

Prior information, such as expected ranges for the parameters or the residuals, is added by specifying support points for the parameters or the residuals. Support points are points in one dimension that specify the expected domain of the parameter or the residual. The wider the domain specified, the less efficient your parameter estimates are (the more variance they have). Specifying more support points in the same width interval also improves the efficiency of the parameter estimates at the cost of more computation. Golan, Judge, and Miller (1996) show that the gains in efficiency fall off for adding more than five support points. You can specify between 2 to 256 support points in the ENTROPY procedure.

If you have only a small amount of data, the estimates are very sensitive to your selection of support points and weights. For larger data sets, incorrect priors are discounted if they are not supported by the data.

Consider the data set generated by the following SAS statements:

```
data prior;
   do by = 1 to 100;
      do t = 1 to 10;
         y = 2*t + 5 * rannor(4);
         output;
      end;
   end;
run;
```

The PRIOR data set contains 100 samples of 10 observations each from the population

$$\begin{aligned} y &= 2 * t + \epsilon \\ \epsilon &\sim N(0, 5) \end{aligned}$$

You can estimate these samples using PROC ENTROPY as

```
proc entropy data=prior outest=parm1 noprint;
   model y = t ;
   by by;
run;
```

The 100 estimates are summarized by using the following SAS statements:

```
proc univariate data=parm1;
   var t;
run;
```

The summary statistics from PROC UNIVARIATE are shown in Output 14.11. The true value of the coefficient T is 2.0, demonstrating that maximum entropy estimates tend to be biased.

Figure 14.11 No Prior Information Monte Carlo Summary

Test Scores compiled by Coleman et al. (1966)

The UNIVARIATE Procedure
Variable: t

Basic Statistical Measures			
Location		**Variability**	
Mean	1.693608	**Std Deviation**	0.30199
Median	1.707653	**Variance**	0.09120
Mode	.	**Range**	1.46194
		Interquartile Range	0.32329

Now assume that you have prior information about the slope and the intercept for this model. You are reasonably confident that the slope is 2 and you are less confident that intercept is zero. To specify prior information about the parameters, use the PRIORS statement.

There are two parts to the prior information specified in the PRIORS statement. The first part is the support points for a parameter. The support points specify the domain of the parameter. For example, the following statement sets the support points -1000 and 1000 for the parameter associated with variable T:

```
priors t -1000 1000;
```

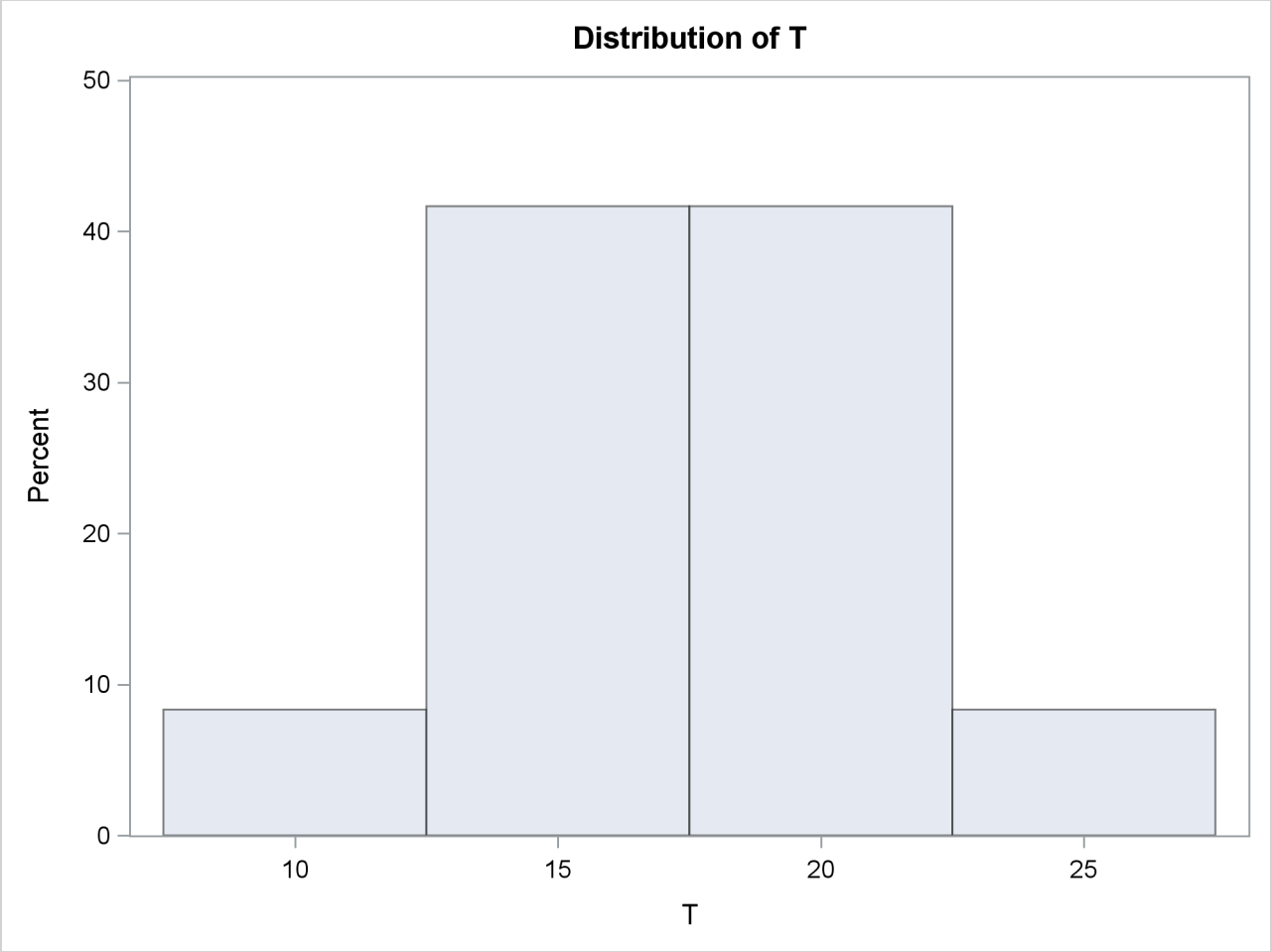
This means that the coefficient lies in the interval $[-1000, 1000]$. If the estimated value of the coefficient is actually outside of this interval, the estimation will not converge. In the previous PRIORS statement, no weights were specified for the support points, so uniform weights are assumed. This implies that the coefficient has a uniform probability of being in the interval $[-1000, 1000]$.

The second part of the prior information is the weights on the support points. For example, the following statements sets the support points 10, 15, 20, and 25 with weights 1, 5, 5, and 1 respectively for the coefficient of T:

```
priors t 10(1) 15(5) 20(5) 25(1);
```

This creates the prior distribution on the coefficient shown in Figure 14.12. The weights are automatically normalized so that they sum to one.

Figure 14.12 Prior Distribution of Parameter T



For the PRIOR data set created previously, the expected value of the coefficient of T is 2. The following SAS statements reestimate the parameters with a prior weight specified for each one.

```
proc entropy data=prior outest=parm2 noprint;
   priors t 0(1) 2(3) 4(1)
          intercept -100(.5) -10(1.5) 0(2) 10(1.5) 100(0.5);
   model y = t;
   by by;
run;
```

The priors on the coefficient of T express a confident view of the value of the coefficient. The priors on INTERCEPT express a more diffuse view on the value of the intercept. The following PROC UNIVARIATE statement computes summary statistics from the estimations:

```
proc univariate data=parm2;
   var t;
run;
```

The summary statistics for the distribution of the estimates of T are shown in Figure 14.13.

Figure 14.13 Prior Information Monte Carlo Summary

Prior Distribution of Parameter T

The UNIVARIATE Procedure
Variable: t

Basic Statistical Measures			
Location		**Variability**	
Mean	1.999953	**Std Deviation**	0.01436
Median	2.001423	**Variance**	0.0002061
Mode	.	**Range**	0.08525
		Interquartile Range	0.01855

The prior information improves the estimation of the coefficient of T dramatically. The downside of specifying priors comes when they are incorrect. For example, say the priors for this model were specified as

```
priors t -2(1) 0(3) 2(1);
```

to indicate a prior centered on zero instead of two.

The resulting summary statistics shown in Figure 14.14 indicate how the estimation is biased away from the solution.

Figure 14.14 Incorrect Prior Information Monte Carlo Summary

Prior Distribution of Parameter T

The UNIVARIATE Procedure
Variable: t

Basic Statistical Measures			
Location		**Variability**	
Mean	0.062550	**Std Deviation**	0.00920
Median	0.062527	**Variance**	0.0000847
Mode	.	**Range**	0.05442
		Interquartile Range	0.01112

The more data available for estimation, the less sensitive the parameters are to the priors. If the number of observations in each sample is 50 instead of 10, then the summary statistics shown in Figure 14.15 are produced. The prior information is not supported by the data, so it is discounted.

Figure 14.15 Incorrect Prior Information with More Data

Prior Distribution of Parameter T

The UNIVARIATE Procedure
Variable: t

Basic Statistical Measures			
Location		**Variability**	
Mean	0.652921	**Std Deviation**	0.00933
Median	0.653486	**Variance**	0.0000870
Mode	.	**Range**	0.04351
		Interquartile Range	0.01498

Pure Inverse Problems

A special case of systems of equations estimation is the pure inverse problem. A pure problem is one that contains an exact relationship between the dependent variable and the independent variables and does not have an error component. A pure inverse problem can be written as

$$\mathbf{y} = X\beta$$

where $\mathbf{y}$ is a n-dimensional vector of observations, $\mathbf{X}$ is a $n \times k$ matrix of regressors, and β is a k-dimensional vector of unknowns. Notice that there is no error term.

A classic example is a dice problem (Jaynes 1963). Given a six-sided die that can take on the values $x = 1, 2, 3, 4, 5, 6$ and the average outcome of the die $y = A$, compute the probabilities $\beta = (p_1, p_2, \cdots, p_6)'$ of rolling each number. This infers six values from two pieces of information. The data points are the expected value of y, and the sum of the probabilities is one. Given $E(y) = 4.0$, this problem is solved by using the following SAS code:

```
data one;
   array x[6] ( 1 2 3 4 5 6 );
   y=4.0;
run;

proc entropy data=one pure;
   priors x1 0 1 x2 0 1 x3 0 1 x4 0 1 x5 0 1 x6 0 1;
   model y = x1-x6/ noint;
   restrict x1 + x2 +x3 +x4 + x5 + x6 =1;
run;
```

The probabilities are given in Figure 14.16.

Figure 14.16 Jaynes' Dice Pure Inverse Problem

Prior Distribution of Parameter T

The ENTROPY Procedure

GME Variable Estimates

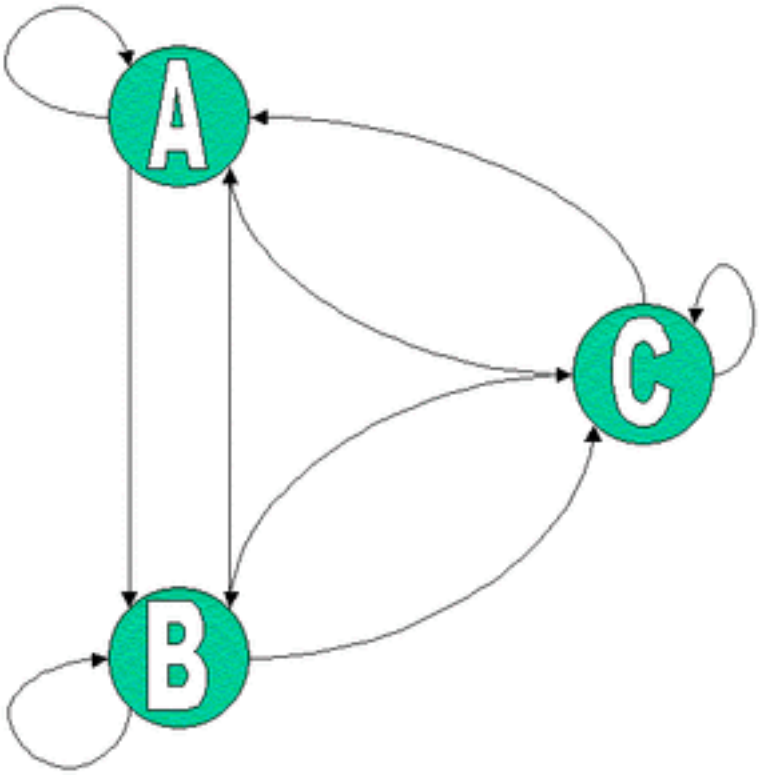
Variable	Estimate	Information Index	Label
x1	0.101763	0.5254	
x2	0.122658	0.4630	
x3	0.147141	0.3974	
x4	0.175533	0.3298	
x5	0.208066	0.2622	
x6	0.244839	0.1970	
Restrict0	2.388082	.	x1 + x2 + x3 + x4 + x5 + x6 = 1

Note how the probabilities are skewed to the higher values because of the high average roll provided in the input data.

First-Order Markov Process Estimation

A more useful inverse problem is the first-order markov process. Companies have a share of the marketplace where they do business. Generally, customers for a specific market space can move from company to company. The movement of customers can be visualized graphically as a flow diagram, as in Figure 14.17. The arrows represent movements of customers from one company to another.

Figure 14.17 Markov Transition Diagram



You can model the probability that a customer moves from one company to another using a first-order Markov model. Mathematically the model is:

$$y_t = P y_{t-1}$$

where y_t is a vector of k market shares at time t and P is a $k \times k$ matrix of unknown transition probabilities. The value p_{ij} represents the probability that a customer who is currently using company j at time $t-1$ moves to company i at time t. The diagonal elements then represent the probability that a customer stays with the current company. The columns in P sum to one.

Given market share information over time, you can estimate the transition probabilities P. In order to estimate P using traditional methods, you need at least k observations. If you have fewer than k transitions, you can use the ENTROPY procedure to estimate the probabilities.

Suppose you are studying the market share for four companies. If you want to estimate the transition probabilities for these four companies, you need a time series with four observations of the shares. Assume the current transition probability matrix is as follows:

$$\begin{bmatrix} 0.7 & 0.4 & 0.0 & 0.1 \\ 0.1 & 0.5 & 0.4 & 0.0 \\ 0.0 & 0.1 & 0.6 & 0.0 \\ 0.2 & 0.0 & 0.0 & 0.9 \end{bmatrix}$$

The following SAS DATA step statements generate a series of market shares from this probability matrix. A transition is represented as the current period shares, `y`, and the previous period shares, `x`.

```
data m;
            /* Known Transition matrix */
   array p[4,4] (0.7 .4 .0 .1
                 0.1 .5 .4 .0
                 0.0 .1 .6 .0
                 0.2 .0 .0 .9 ) ;
            /* Initial Market shares */
   array y[4] y1-y4 ( .4 .3 .2 .1 );
   array x[4] x1-x4;
   drop p1-p16 i;
   do i = 1 to 3;
       x[1] = y[1]; x[2] = y[2];
       x[3] = y[3]; x[4] = y[4];
       y[1] = p[1,1] * x1 + p[1,2] * x2 + p[1,3] * x3 + p[1,4] * x4;
       y[2] = p[2,1] * x1 + p[2,2] * x2 + p[2,3] * x3 + p[2,4] * x4;
       y[3] = p[3,1] * x1 + p[3,2] * x2 + p[3,3] * x3 + p[3,4] * x4;
       y[4] = p[4,1] * x1 + p[4,2] * x2 + p[4,3] * x3 + p[4,4] * x4;
       output;
   end;
run;
```

The following SAS statements estimate the transition matrix by using only the first transition.

```
proc entropy markov pure data=m(obs=1);
   model y1-y4 = x1-x4;
run;
```

The MARKOV option implies NOINT for each model, that the sum of the parameters in each column is one, and chooses support points of 0 and 1. This model can be expressed equivalently as

```
proc entropy pure data=m(obs=1) ;
   priors y1.x1 0 1 y1.x2 0 1 y1.x3 0 1 y1.x4 0 1;
   priors y2.x1 0 1 y2.x2 0 1 y2.x3 0 1 y2.x4 0 1;
   priors y3.x1 0 1 y3.x2 0 1 y3.x3 0 1 y3.x4 0 1;
   priors y4.x1 0 1 y4.x2 0 1 y4.x3 0 1 y4.x4 0 1;

   model y1 = x1-x4 / noint;
   model y2 = x1-x4 / noint;
   model y3 = x1-x4 / noint;
   model y4 = x1-x4 / noint;

   restrict y1.x1 + y2.x1 + y3.x1 + y4.x1 = 1;
   restrict y1.x2 + y2.x2 + y3.x2 + y4.x2 = 1;
   restrict y1.x3 + y2.x3 + y3.x3 + y4.x3 = 1;
   restrict y1.x4 + y2.x4 + y3.x4 + y4.x4 = 1;
run;
```

The transition matrix is given in Figure 14.18.

Figure 14.18 Estimate of *P* by Using One Transition

Prior Distribution of Parameter T

The ENTROPY Procedure

GME Variable Estimates

Variable	Estimate	Information Index
y1.x1	0.463407	0.0039
y1.x2	0.41055	0.0232
y1.x3	0.356272	0.0605
y1.x4	0.302163	0.1161
y2.x1	0.272755	0.1546
y2.x2	0.271459	0.1564
y2.x3	0.267252	0.1625
y2.x4	0.260084	0.1731
y3.x1	0.119926	0.4709
y3.x2	0.148481	0.3940
y3.x3	0.180224	0.3194
y3.x4	0.214394	0.2502
y4.x1	0.143903	0.4056
y4.x2	0.169504	0.3434
y4.x3	0.196252	0.2856
y4.x4	0.223364	0.2337

Note that *P* varies greatly from the true solution.

If two transitions are used instead (OBS=2), the resulting transition matrix is shown in Figure 14.19.

```
proc entropy markov pure data=m(obs=2);
   model y1-y4 = x1-x4;
run;
```

Figure 14.19 Estimate of *P* by Using Two Transitions

Prior Distribution of Parameter T

The ENTROPY Procedure

GME Variable Estimates

Variable	Estimate	Information Index
y1.x1	0.721012	0.1459
y1.x2	0.355703	0.0609
y1.x3	0.026095	0.8256
y1.x4	0.096654	0.5417
y2.x1	0.083987	0.5839
y2.x2	0.53886	0.0044
y2.x3	0.373668	0.0466
y2.x4	0.000133	0.9981
y3.x1	0.000062	0.9990
y3.x2	0.099848	0.5315
y3.x3	0.600104	0.0291
y3.x4	7.871E-8	1.0000
y4.x1	0.194938	0.2883
y4.x2	0.00559	0.9501
y4.x3	0.000133	0.9981
y4.x4	0.903214	0.5413

This transition matrix is much closer to the actual transition matrix.

If, in addition to the transitions, you had other information about the transition matrix, such as your own company's transition values, that information can be added as restrictions to the parameter estimates. For noisy data, the PURE option should be dropped. Note that this example has six zero probabilities in the transition matrix; the accurate estimation of transition matrices with fewer zero probabilities generally requires more transition observations.

Analyzing Multinomial Response Data

Multinomial discrete choice models suffer the same problems with collinearity of the regressors and small sample sizes as linear models. Unordered multinomial discrete choice models can be estimated using a variant of GME for discrete models called GME-D.

Consider the model shown in Golan, Judge, and Perloff (1996). In this model, there are five occupational categories, and the categories are considered a function of four individual characteristics. The sample contains 337 individuals.

```
data kpdata;
   input job x1 x2 x3 x4;
datalines;
   0 1 3 11 1

   ... more lines ...
```

The dependent variable in this data, job, takes on values 0 through 4. Support points are used only for the error terms; so error supports are specified on the MODEL statement.

```
proc entropy data=kpdata gmed tech=nra;
   model job = x1 x2 x3 x4 / noint
         esupports=( -.1 -0.0666 -0.0333 0 0.0333 0.0666 .1 );
run;
```

Figure 14.20 Estimate of Jobs Model by Using GME-D

Prior Distribution of Parameter T

The ENTROPY Procedure

GME-D Variable Estimates

Variable	Estimate	Approx Std Err	t Value	Approx Pr > |t|
x1_1	1.802572	1.3610	1.32	0.1863
x2_1	-0.00251	0.0154	-0.16	0.8705
x3_1	-0.17282	0.0885	-1.95	0.0517
x4_1	1.054659	0.6986	1.51	0.1321
x1_2	0.089156	1.2764	0.07	0.9444
x2_2	0.019947	0.0146	1.37	0.1718
x3_2	0.010716	0.0830	0.13	0.8974
x4_2	0.288629	0.5775	0.50	0.6176
x1_3	-4.62047	1.6476	-2.80	0.0053
x2_3	0.026175	0.0166	1.58	0.1157
x3_3	0.245198	0.0986	2.49	0.0134
x4_3	1.285466	0.8367	1.54	0.1254
x1_4	-9.72734	1.5813	-6.15	<.0001
x2_4	0.027382	0.0156	1.75	0.0805
x3_4	0.660836	0.0947	6.98	<.0001
x4_4	1.47479	0.6970	2.12	0.0351

Note there are five estimates of the parameters produced for each regressor, one for each choice. The first choice is restricted to zero for normalization purposes. PROC ENTROPY drops the zeroed regressors. PROC ENTROPY also generates tables of marginal effects for each regressor. The following statements generate the marginal effects table for the previous analysis at the means of the variables.

```
proc entropy data=kpdata gmed tech=nra;
   model job = x1 x2 x3 x4 / noint
         esupports=( -.1 -0.0666 -0.0333 0 0.0333 0.0666 .1 )
         marginals;
run;
```

Figure 14.21 Estimate of Jobs Model by Using GME-D (Marginals)

Prior Distribution of Parameter T

The ENTROPY Procedure

GME-D Variable Marginal Effects Table

Variable	Marginal Effect	Mean
x1_0	0.338758	1
x2_0	-0.0019	20.50148
x3_0	-0.02129	13.09496
x4_0	-0.09917	0.916914
x1_1	0.859883	1
x2_1	-0.00345	20.50148
x3_1	-0.0648	13.09496
x4_1	0.034396	0.916914
x1_2	0.86101	1
x2_2	0.000963	20.50148
x3_2	-0.04948	13.09496
x4_2	-0.16297	0.916914
x1_3	-0.25969	1
x2_3	0.0015	20.50148
x3_3	0.009289	13.09496
x4_3	0.065569	0.916914
x1_4	-1.79996	1
x2_4	0.00288	20.50148
x3_4	0.126283	13.09496
x4_4	0.162172	0.916914

The marginals are derivatives of the probabilities with respect to each variable and so summarize how a small change in each variable affects the overall probability.

PROC ENTROPY also enables the user to specify where the derivative is evaluated, as shown below:

```
proc entropy data=kpdata gmed tech=nra;
   model job = x1 x2 x3 x4 / noint
         esupports=( -.1 -0.0666 -0.0333 0 0.0333 0.0666 .1 )
         marginals=( x2=.4 x3=10 x4=0);
run;
```

Figure 14.22 Estimate of Jobs Model by Using GME-D (Marginals)

Prior Distribution of Parameter T

The ENTROPY Procedure

GME-D Variable Marginal Effects Table

Variable	Marginal Effect	Mean	Marginal Effect at User Supplied Values	User Supplied Values
x1_0	0.338758	1	-0.0901	1
x2_0	-0.0019	20.50148	-0.00217	0.4
x3_0	-0.02129	13.09496	0.009586	10
x4_0	-0.09917	0.916914	-0.14204	0
x1_1	0.859883	1	0.463181	1
x2_1	-0.00345	20.50148	-0.00311	0.4
x3_1	-0.0648	13.09496	-0.04339	10
x4_1	0.034396	0.916914	0.174876	0
x1_2	0.86101	1	-0.07894	1
x2_2	0.000963	20.50148	0.004405	0.4
x3_2	-0.04948	13.09496	0.015555	10
x4_2	-0.16297	0.916914	-0.072	0
x1_3	-0.25969	1	-0.16459	1
x2_3	0.0015	20.50148	0.000623	0.4
x3_3	0.009289	13.09496	0.00929	10
x4_3	0.065569	0.916914	0.02648	0
x1_4	-1.79996	1	-0.12955	1
x2_4	0.00288	20.50148	0.000256	0.4
x3_4	0.126283	13.09496	0.008956	10
x4_4	0.162172	0.916914	0.012684	0

In this example, you evaluate the derivative when x1=1, x2=0.4, x3=10, and x4=0. If the user neglects a variable, PROC ENTROPY uses its mean value.

Syntax: ENTROPY Procedure

The following statements can be used with the ENTROPY procedure:

```
PROC ENTROPY options ;
  BOUNDS bound1 < , bound2, . . . > ;
  BY variable < variable . . . > ;
  ID variable < variable . . . > ;
  MODEL variable = variable < variable > . . . < / options > ;
  PRIORS variable < support points > variable < value > . . . ;
  RESTRICT restriction1 < , restriction2 . . . > ;
  TEST < "name" > test1 < , test2 . . . > < / options > ;
  WEIGHT variable ;
```

Functional Summary

The statements and options in the ENTROPY procedure are summarized in the following table.

Description	Statement	Option
Data Set Options		
specify the input data set for the variables	ENTROPY	DATA=
specify the input data set for support points and priors	ENTROPY	PDATA=
specify the output data set for residual, predicted, and actual values	ENTROPY	OUT=
specify the output data set for the support points and priors	ENTROPY	OUTP=
write the covariance matrix of the estimates to OUTEST= data set	ENTROPY	OUTCOV
write the parameter estimates to a data set	ENTROPY	OUTEST=
write the Lagrange multiplier estimates to a data set	ENTROPY	OUTL=
write the covariance matrix of the equation errors to a data set	ENTROPY	OUTS=
write the **S** matrix used in the objective function definition to a data set	ENTROPY	OUTSUSED=
read the covariance matrix of the equation errors	ENTROPY	SDATA=
Printing Options		
request that the procedure produce graphics via the Output Delivery System	ENTROPY	PLOTS=
print collinearity diagnostics	ENTROPY	COLLIN
suppress the normal printed output	ENTROPY	NOPRINT

Description	Statement	Option
Options to Control Iteration Output		
print a summary iteration listing	ENTROPY	ITPRINT
Options to Control the Minimization Process		
specify the convergence criteria	ENTROPY	CONVERGE=
specify the maximum number of iterations allowed	ENTROPY	MAXITER=
specify the maximum number of subiterations allowed	ENTROPY	MAXSUBITER=
select the iterative minimization method to use	ENTROPY	METHOD=
Statements That Declare Variables		
specify BY-group processing	BY	
specify a weight variable	WEIGHT	
specify identifying variables	ID	
General PROC ENTROPY Statement Options		
specify seemingly unrelated regression	ENTROPY	SUR
specify iterated seemingly unrelated regression	ENTROPY	ITSUR
specify data-constrained generalized maximum entropy	ENTROPY	GME
specify moment generalized maximum entropy	ENTROPY	GMEM
specify the denominator for computing variances and covariances	ENTROPY	VARDEF=
General TEST Statement Options		
specify that a Wald test be computed	TEST	WALD
specify that a Lagrange multiplier test be computed	TEST	LM
specify that a likelihood ratio test be computed	TEST	LR
request all three types of tests	TEST	ALL

PROC ENTROPY Statement

PROC ENTROPY *options* **;**

The following options can be specified in the PROC ENTROPY statement.

General Options

COLLIN
requests that the collinearity diagnostics of the $X'X$ matrix be printed.

COVBEST=CROSS | GME | GMEM
specifies the method for producing the covariance matrix of parameters for output and for standard error calculations. GMEM and GME are aliases and are the default.

GME | GCE
requests generalized maximum entropy or generalized cross entropy. This is the default estimation method.

GMEM | GCEM
requests moment maximum entropy or the moment cross entropy.

GMED
requests a variant of GME suitable for multinomial discrete choice models.

MARKOV
specifies that the model is a first-order Markov model.

PURE
specifies a regression without an error term.

SUR | ITSUR
specifies seemingly unrelated regression or iterated seemingly unrelated regression.

VARDEF=N | WGT | DF | WDF
specifies the denominator to be used in computing variances and covariances. VARDEF=N specifies that the number of nonmissing observations be used. VARDEF=WGT specifies that the sum of the weights be used. VARDEF=DF specifies that the number of nonmissing observations minus the model degrees of freedom (number of parameters) be used. VARDEF=WDF specifies that the sum of the weights minus the model degrees of freedom be used. The default is VARDEF=DF.

Data Set Options

DATA=*SAS-data-set*
specifies the input data set. Values for the variables in the model are read from this data set.

PDATA=*SAS-data-set*
names the SAS data set that contains the data about priors and supports.

OUT=*SAS-data-set*
names the SAS data set to contain the residuals from each estimation.

OUTCOV
COVOUT
writes the covariance matrix of the estimates to the OUTEST= data set in addition to the parameter estimates. The OUTCOV option is applicable only if the OUTEST= option is also specified.

OUTEST=*SAS-data-set*
names the SAS data set to contain the parameter estimates and optionally the covariance of the estimates.

OUTL=*SAS-data-set*
names the SAS data set to contain the estimated Lagrange multipliers for the models.

OUTP=*SAS-data-set*
names the SAS data set to contain the support points and estimated probabilities.

OUTS=*SAS-data-set*
names the SAS data set to contain the estimated covariance matrix of the equation errors. This is the covariance of the residuals computed from the parameter estimates.

OUTSUSED=*SAS-data-set*
names the SAS data set to contain the **S** matrix used in the objective function definition. The OUTSUSED= data set is the same as the OUTS= data set for the methods that iterate the **S** matrix.

SDATA=*SAS-data-set*
specifies a data set that provides the covariance matrix of the equation errors. The matrix read from the SDATA= data set is used for the equation error covariance matrix (**S** matrix) in the estimation. The SDATA= matrix is used to provide only the initial estimate of **S** for the methods that iterate the S matrix.

Printing Options

ITPRINT
prints the parameter estimates, objective function value, and convergence criteria at each iteration.

NOPRINT
suppresses the normal printed output but does not suppress error listings. Using any other print option turns the NOPRINT option off.

PLOTS=*global-plot-options* | *plot-request*
controls the plots that the ENTROPY procedure produces. (For general information about ODS Graphics, see Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*).) The *global-plot-options* apply to all relevant plots generated by the ENTROPY procedure.

The *global-plot-options* supported by the ENTROPY procedure are as follows:

ONLY suppresses the default plots. Only the plots specifically requested are produced.

UNPACKPANEL displays each graph separately. (By default, some graphs can appear together in a single panel.)

The specific *plot-request* values supported by the ENTROPY procedure are as follows:

ALL	requests that all plots appropriate for the particular analysis be produced. ALL is equivalent to specifying FITPLOT, COOKSD, QQ, RESIDUALHISTOGRAM, and STUDENTRESIDUAL.
FITPLOT	plots the predicted and actual values.
COOKSD	produces the Cook's *D* plot.
QQ	produces a Q-Q plot of residuals.
RESIDUALHISTOGRAM	plots the histogram of residuals.
STUDENTRESIDUAL	plots the studentized residuals.
NONE	suppresses all plots.

The default behavior is to plot all plots appropriate for the particular analysis (ALL) in a panel.

Options to Control the Minimization Process

The following options can be helpful if a convergence problem occurs for a given model and set of data. The ENTROPY procedure uses the nonlinear optimization subsystem (NLO) to perform the model optimizations. In addition to the options listed below, all options supported in the NLO subsystem can be specified on the ENTROPY procedure statement. See Chapter 7, "Nonlinear Optimization Methods," for more details.

CONVERGE=*value*

GCONV=*value*

specifies the convergence criteria for *S*-iterated methods. The convergence measure computed during model estimation must be less than *value* before convergence is assumed. The default value is CONVERGE=0.001.

DUAL | PRIMAL

specifies whether the optimization problem is solved using the dual or primal form. The dual form is the default.

MAXITER=*n*

specifies the maximum number of iterations allowed. The default is MAXITER=100.

MAXSUBITER=*n*

specifies the maximum number of subiterations allowed for an iteration. The MAXSUBITER= option limits the number of step halvings. The default is MAXSUBITER=30.

METHOD=TR | NEWRAP | NRR | QN | CONGR | NSIMP | DBLDOG | LEVMAR

TECHNIQUE=TR | NEWRAP | NRR | QN | CONGR | NSIMP | DBLDOG | LEVMAR

TECH=TR | NEWRAP | NRR | QN | CONGR | NSIMP | DBLDOG | LEVMAR

specifies the iterative minimization method to use. METHOD=TR specifies the trust region method, METHOD=NEWRAP specifies the Newton-Raphson method, METHOD=NRR specifies the Newton-Raphson ridge method, and METHOD=QN specifies the quasi-Newton method. See Chapter 7, "Nonlinear Optimization Methods," for more details about optimization methods. The default is METHOD=QN for the dual form and METHOD=NEWRAP for the primal form.

BOUNDS Statement

BOUNDS *bound1* < , *bound2* . . . > ;

The BOUNDS statement imposes simple boundary constraints on the parameter estimates. BOUNDS statement constraints refer to the parameters estimated by the ENTROPY procedure. You can specify any number of BOUNDS statements.

Each *boundary constraint* is composed of variables, constants, and inequality operators in the following form:

```
item operator item <,operator item <,operator item ...> >
```

Each *item* is a constant, the name of a regressor variable, or a list of regressor names. Each *operator* is <, >, <=, or >=.

You can use either the BOUNDS statement or the RESTRICT statement to impose boundary constraints; the BOUNDS statement provides a simpler syntax for specifying inequality constraints. See section "RESTRICT Statement" on page 817 for more information about the computational details of estimation with inequality restrictions.

Lagrange multipliers are reported for all the active boundary constraints. In the printed output and in the OUTEST= data set, the Lagrange multiplier estimates are identified with the names BOUND1, BOUND2, and so forth. The probability of the Lagrange multipliers are computed using a beta distribution (LaMotte 1994). Nonactive or nonbinding bounds have no effect on the estimation results and are not noted in the output. To give the constraints more descriptive names, use the RESTRICT statement instead of the BOUNDS statement.

The following BOUNDS statement constrains the estimates of the coefficients of WAGE and TARGET and the 10 coefficients of x1 through x10 to be between zero and one. This example illustrates the use of parameter lists to specify boundary constraints.

```
bounds 0 < wage target x1-x10 < 1;
```

The following is an example of the use of the BOUNDS statement to impose boundary constraints on the variables X1, X2, and X3:

```
proc entropy data=zero;
   bounds .1 <= x1 <= 100,
          0 <= x2 <=  25.6,
          0 <= x3 <=   5;

   model y = x1 x2 x3;
run;
```

The parameter estimates from this run are shown in Figure 14.23.

Figure 14.23 Output from Bounded Estimation

Prior Distribution of Parameter T

The ENTROPY Procedure

Variables(Supports(Weights))	x1 x2 x3 Intercept
Equations(Supports(Weights))	y

Prior Distribution of Parameter T

The ENTROPY Procedure
GME Estimation Summary

Data Set Options
DATA= WORK.ZERO

Minimization Summary	
Parameters Estimated	4
Covariance Estimator	GME
Entropy Type	Shannon
Entropy Form	Dual
Numerical Optimizer	Newton-Raphson

Final Information Measures	
Objective Function Value	6.292861
Signal Entropy	6.375715
Noise Entropy	-0.08285
Normed Entropy (Signal)	0.990364
Normed Entropy (Noise)	1.004172
Parameter Information Index	0.009636
Error Information Index	-0.00417

Observations Processed	
Read	20
Used	20

NOTE: At GME Iteration 20 convergence criteria met.

GME Summary of Residual Errors

Equation	DF Model	DF Error	SSE	MSE	Root MSE	R-Square	Adj RSq
y	4	16	1665620	83281.0	288.6	-0.0013	-0.1891

Figure 14.23 *continued*

GME Variable Estimates

Variable	Estimate	Approx Std Err	t Value	Approx Pr > |t|	Label
x1	0.1	0	.	.	
x2	0	0	.	.	
x3	3.33E-16	0	.	.	
Intercept	-0.00432	3.406E-6	-1269.3	<.0001	
	1.25731	9130.3	0.00	0.9999	0.1 <= x1
	0.009384	0	.	.	0 <= x2
	0.000025	0	.	.	0 <= x3

BY Statement

BY *variables* **;**

A BY statement is used to obtain separate estimates for observations in groups defined by the BY variables. To save parameter estimates for each BY group, use the OUTEST= option.

ID Statement

ID *variables* **;**

The ID statement specifies variables to identify observations in error messages or other listings and in the OUT= data set. The ID variables are normally SAS date or datetime variables. If more than one ID variable is used, the first variable is used to identify the observations and the remaining variables are added to the OUT= data set.

MODEL Statement

MODEL *dependent = regressors* < */ options* > **;**

The MODEL statement specifies the dependent variable and independent regressor variables for the regression model. If no independent variables are specified in the MODEL statement, only the mean (intercept) is estimated. To model a system of equations, specify more than one MODEL statement.

The following options can be used in the MODEL statement after a slash (/).

ESUPPORTS=(*support (prior) . . .* **)**

specifies the support points and prior weights on the residuals for the specified equation. The default is the following five support values:

$$-10 * value, -value, 0, value, 10 * value$$

where *value* is computed as

$$value = (max(y) - \bar{y}) * multiplier$$

for GME, where y is the dependent variable, and

$$value = (max(y) - \bar{y}) * multiplier * nobs * max(X) * 0.1$$

for generalized maximum entropy—moments (GME-M), where $\mathbf{X}$ is the information matrix, and *nobs* is the number of observations. The *multiplier* depends on the MULTIPLIER= option. The MULTIPLIER= option defaults to 2 for unrestricted models and to 4 for restricted models. The prior probabilities default to the following:

$$0.0005, 0.333, 0.333, 0.333, 0.0005$$

The support points and prior weights are selected so that hypothesis tests can be performed without adding significant bias to the estimation. These prior probability values are ad hoc.

NOINT
suppresses the intercept parameter.

MARGINALS = *(variable = value, . . . , variable = value)*
requests that the marginal effects of each variable be calculated for GME-D. Specifying the MARGINALS option with an optional list of values calculates the marginals at that vector of values. For example, if x1–x4 are explanatory variables, then including

MARGINALS = (x1 = 2, x2 = 4, x3 = –1, x4 = 5)

calculates the marginal effects at that vector. A skipped variable implies that its mean value is to be used.

CENSORED *((UB | LB) = (variable | value)*, **ESUPPORTS** *=(support (prior) . . .))*
specifies that the dependent variable be observed with censoring and specifies the censoring thresholds and the supports of the censored observations.

CATEGORY= *variable*
specifies the variable that keeps track of the categories the dependent variable is in when there is range censoring. When the actual value is observed, this variable should be set to MISSING.

RANGE (ID = (QS | INT) L = (NUMBER) R = (NUMBER) , ESUPPORTS=(*support < (prior) > . . .* **))**
specifies that the dependent variable be range bound. The RANGE option defines the range and the key (RANGE) that is used to identify the observation as being range bound. The RANGE = value should be some value in the CATEGORY= variable. The L and R define, respectively, the left endpoint of the range and the right endpoint of the range. ESUPPORTS sets the error supports on the variable.

PRIORS Statement

PRIORS *variable < support points < (priors) > > variable < support points < (priors) > > . . .* **;**

The PRIORS statement specifies the support points and prior weights for the coefficients on the variables.

Support points for coefficients default to five points, determined as follows:

$$-2 * value, -value, 0, value, 2 * value$$

where *value* is computed as

$$value = (\|mean\| + 3 * stderr) * multiplier$$

where the *mean* and the *stderr* are obtained from OLS and the *multiplier* depends on the MULTIPLIER= option. The MULTIPLIER= option defaults to 2 for unrestricted models and to 4 for restricted models. The prior probabilities for each support point default to the uniform distribution.

The number of support points must be at least two. If priors are specified, they must be positive and there must be the same number of priors as there are support points. Priors and support points can also be specified through the PDATA= data set.

RESTRICT Statement

RESTRICT *restriction1* < *, restriction2 . . .* > **;**

The RESTRICT statement is used to impose linear restrictions on the parameter estimates. You can specify any number of RESTRICT statements.

Each *restriction* is written as an optional name, followed by an expression, followed by an equality operator (=) or an inequality operator (<, >, <=, >=), followed by a second expression:

```
<"name" > expression operator expression
```

The optional *"name"* is a string used to identify the restriction in the printed output and in the OUTEST= data set. The *operator* can be =, <, >, <= , or >=. The operator and second expression are optional, as in the TEST statement, where they default to = 0.

Restriction expressions can be composed of variable names, multiplication (∗), and addition (+) operators, and constants. Variable names in restriction expressions must be among the variables whose coefficients are estimated by the model. The restriction expressions must be a linear function of the variables.

The following is an example of the use of the RESTRICT statement:

```
proc entropy data=one;
   restrict y1.x1*2 <= x2 + y2.x1;
   model y1 = x1 x2;
   model y2 = x1 x3;
run;
```

This example illustrates the use of compound names, `y1.x1`, to specify coefficients of specific equations.

TEST Statement

TEST < *"name"*> *test1* < *, test2 . . .* > < *,/ options* > **;**

The TEST statement performs tests of linear hypotheses on the model parameters.

The TEST statement applies only to parameters estimated in the model. You can specify any number of TEST statements.

Each *test* is written as an expression optionally followed by an equal sign (=) and a second expression:

```
expression <= expression>
```

Test expressions can be composed of variable names, multiplication (∗), addition (+), and subtraction (−) operators, and constants. Variables named in test expressions must be among the variables estimated by the model.

If you specify only one expression in a TEST statement, that expression is tested against zero. For example, the following two TEST statements are equivalent:

```
test a + b;

test a + b = 0;
```

When you specify multiple tests on the same TEST statement, a joint test is performed. For example, the following TEST statement tests the joint hypothesis that both of the coefficients on a and b are equal to zero:

```
test a, b;
```

To perform separate tests rather than a joint test, use separate TEST statements. For example, the following TEST statements test the two separate hypotheses that a is equal to zero and that b is equal to zero:

```
test a;
test b;
```

You can use the following options in the TEST statement:

WALD
specifies that a Wald test be computed. WALD is the default.

LM

RAO

LAGRANGE
specifies that a Lagrange multiplier test be computed.

LR

LIKE
specifies that a pseudo-likelihood ratio test be computed.

ALL
requests all three types of tests.

OUT=
specifies the name of an output SAS data set that contains the test results. The format of the OUT= data set produced by the TEST statement is similar to that of the OUTEST= data set.

WEIGHT Statement

WEIGHT *variable* **;**

The WEIGHT statement specifies a variable to supply weighting values to use for each observation in estimating parameters.

If the weight of an observation is nonpositive, that observation is not used for the estimation. *Variable* must be a numeric variable in the input data set. The regressors and the dependent variables are multiplied by the square root of the weight variable to form the weighted **X** matrix and the weighted dependent variable. The same weight is used for all MODEL statements.

Details: ENTROPY Procedure

Shannon's measure of entropy for a distribution is given by

$$\begin{aligned} &\text{maximize} && -\sum_{i=1}^{n} p_i \ln(p_i) \\ &\text{subject to} && \sum_{i=1}^{n} p_i = 1 \end{aligned}$$

where p_i is the probability associated with the ith support point. Properties that characterize the entropy measure are set forth by Kapur and Kesavan (1992).

The objective is to maximize the entropy of the distribution with respect to the probabilities p_i and subject to constraints that reflect any other known information about the distribution (Jaynes 1957). This measure, in the absence of additional information, reaches a maximum when the probabilities are uniform. A distribution other than the uniform distribution arises from information already known.

Generalized Maximum Entropy

Reparameterization of the errors in a regression equation is the process of specifying a support for the errors, observation by observation. If a two-point support is used, the error for the tth observation is reparameterized by setting $e_t = w_{t1} v_{t1} + w_{t2} v_{t2}$, where v_{t1} and v_{t2} are the upper and lower bounds for the tth error e_t, and w_{t1} and w_{t2} represent the weight associated with the point v_{t1} and v_{t2}. The error distribution is usually chosen to be symmetric, centered around zero, and the same across observations so that $v_{t1} = -v_{t2} = R$, where R is the support value chosen for the problem (Golan, Judge, and Miller 1996).

The generalized maximum entropy (GME) formulation was proposed for the ill-posed or underdetermined case where there is insufficient data to estimate the model with traditional methods. β is reparameterized by defining a support for β (and a set of weights in the cross entropy case), which defines a prior distribution for β.

In the simplest case, each β_k is reparameterized as $\beta_k = p_{k1} z_{k1} + p_{k2} z_{k2}$, where p_{k1} and p_{k2} represent the probabilities ranging from [0,1] for each β, and z_{k1} and z_{k2} represent the lower and upper bounds placed

on β_k. The support points, z_{k1} and z_{k2}, are usually distributed symmetrically around the most likely value for β_k based on some prior knowledge.

With these reparameterizations, the GME estimation problem is

$$\begin{aligned} \text{maximize} \quad & H(p,w) = -p' \ln(p) - w' \ln(w) \\ \text{subject to} \quad & y = X Z p + V w \\ & 1_K = (I_K \otimes 1_L') p \\ & 1_T = (I_T \otimes 1_L') w \end{aligned}$$

where y denotes the column vector of length T of the dependent variable; $\mathbf{X}$ denotes the $(T \times K)$ matrix of observations of the independent variables; p denotes the LK column vector of weights associated with the points in Z; w denotes the LT column vector of weights associated with the points in V; 1_K, 1_L, and 1_T are K-, L-, and T-dimensional column vectors, respectively, of ones; and I_K and I_T are $(K \times K)$ and $(T \times T)$ dimensional identity matrices.

These equations can be rewritten using set notation as follows:

$$\text{maximize} \quad H(p,w) = -\sum_{l=1}^{L}\sum_{k=1}^{K} p_{kl} \ln(p_{kl}) - \sum_{l=1}^{L}\sum_{t=1}^{T} w_{tl} \ln(w_{tl})$$

$$\text{subject to} \quad y_t = \sum_{l=1}^{L}\left[\sum_{k=1}^{K} (X_{kt} Z_{kl} p_{kl}) + V_{tl} w_{tl}\right]$$

$$\sum_{l=1}^{L} p_{kl} = 1 \text{ and } \sum_{l=1}^{L} w_{tl} = 1$$

The subscript l denotes the support point (l=1, 2, ..., L), k denotes the parameter (k=1, 2, ..., K), and t denotes the observation (t=1, 2, ..., T).

The GME objective is strictly concave; therefore, a unique solution exists. The optimal estimated probabilities, p and w, and the prior supports, Z and V, can be used to form the point estimates of the unknown parameters, β, and the unknown errors, e.

Generalized Cross Entropy

Kullback and Leibler (1951) cross entropy measures the "discrepancy" between one distribution and another. Cross entropy is called a measure of discrepancy rather than distance because it does not satisfy some of the properties one would expect of a distance measure. (See Kapur and Kesavan (1992) for a discussion of cross entropy as a measure of discrepancy.) Mathematically, cross entropy is written as

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} p_i \ln(p_i / q_i) \\ \text{subject to} \quad & \sum_{i=1}^{n} p_i = 1, \end{aligned}$$

where q_i is the probability associated with the ith point in the distribution from which the discrepancy is measured. The q_i (in conjunction with the support) are often referred to as the prior distribution. The measure is nonnegative and is equal to zero when p_i equals q_i. The properties of the cross entropy measure are examined by Kapur and Kesavan (1992).

The principle of minimum cross entropy (Kullback 1959; Good 1963) states that one should choose probabilities that are as close as possible to the prior probabilities. That is, out of all probability distributions that satisfy a given set of constraints which reflect known information about the distribution, choose the distribution that is closest (as measured by $p(\ln(p) - \ln(q))$) to the prior distribution. When the prior distribution is uniform, maximum entropy and minimum cross entropy produce the same results (Kapur and Kesavan 1992), where the higher values for entropy correspond exactly with the lower values for cross entropy.

If the prior distributions are nonuniform, the problem can be stated as a generalized cross entropy (GCE) formulation. The cross entropy terminology specifies weights, q_i and u_i, for the points Z and V, respectively. Given informative prior distributions on Z and V, the GCE problem is

$$\begin{aligned} \text{minimize} \quad & I(p,q,w,u) = p' \ln(p/q) + w' \ln(w/u) \\ \text{subject to} \quad & y = X\,Z\,p + V\,w \\ & 1_K = (I_K \otimes 1_L')\,p \\ & 1_T = (I_T \otimes 1_L')\,w \end{aligned}$$

where y denotes the T column vector of observations of the dependent variables; $\mathbf{X}$ denotes the $(T \times K)$ matrix of observations of the independent variables; q and p denote LK column vectors of prior and posterior weights, respectively, associated with the points in Z; u and w denote the LT column vectors of prior and posterior weights, respectively, associated with the points in V; 1_K, 1_L, and 1_T are K-, L-, and T-dimensional column vectors, respectively, of ones; and $\mathbf{I_K}$ and $\mathbf{I_T}$ are $(K \times K)$ and $(T \times T)$ dimensional identity matrices.

The optimization problem can be rewritten using set notation as follows

$$\text{minimize} \quad I(p,q,w,u) = \sum_{l=1}^{L}\sum_{k=1}^{K} p_{kl} \ln(p_{kl}/q_{kl}) + \sum_{l=1}^{L}\sum_{t=1}^{T} w_{tl} \ln(w_{tl}/u_{tl})$$

$$\text{subject to} \quad y_t = \sum_{l=1}^{L}\left[\sum_{k=1}^{K} (X_{kt}\,Z_{kl}\,p_{kl}) + V_{tl}\,w_{tl}\right]$$

$$\sum_{l=1}^{L} p_{kl} = 1 \text{ and } \sum_{l=1}^{L} w_{tl} = 1$$

The subscript l denotes the support point (l=1, 2, ..., L), k denotes the parameter (k=1, 2, ..., K), and t denotes the observation (t=1, 2, ..., T).

The objective function is strictly convex; therefore, there is a unique global minimum for the problem (Golan, Judge, and Miller 1996). The optimal estimated weights, p and w, and the prior supports, Z and V, can be used to form the point estimates of the unknown parameters, β, and the unknown errors, e, by using

$$\beta = Z\,p = \begin{bmatrix} z_{11} & \cdots & z_{L1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & z_{12} & \cdots & z_{L2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ddots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & z_{1K} & \cdots & z_{LK} \end{bmatrix} \begin{bmatrix} p_{11} \\ \vdots \\ p_{L1} \\ p_{12} \\ \vdots \\ p_{L2} \\ \vdots \\ p_{1K} \\ \vdots \\ p_{LK} \end{bmatrix}$$

$$e = V\,w = \begin{bmatrix} v_{11} & \cdots & v_{L1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & v_{12} & \cdots & v_{L2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ddots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & v_{1T} & \cdots & v_{LT} \end{bmatrix} \begin{bmatrix} w_{11} \\ \vdots \\ w_{L1} \\ w_{12} \\ \vdots \\ w_{L2} \\ \vdots \\ w_{1T} \\ \vdots \\ w_{LT} \end{bmatrix}$$

Computational Details

This constrained estimation problem can be solved either directly (primal) or by using the dual form. Either way, it is prudent to factor out one probability for each parameter and each observation as the sum of the other probabilities. This factoring reduces the computational complexity significantly. If the primal formalization is used and two support points are used for the parameters and the errors, the resulting GME problem is $O((nparms + nobs)^3)$. For the dual form, the problem is $O((nobs)^3)$. Therefore for large data sets, GME-M should be used instead of GME.

Moment Generalized Maximum Entropy

The default estimation technique is moment generalized maximum entropy (GME-M). This is simply GME with the data constraints modified by multiplying both sides by X'. GME-M then becomes

$$\begin{aligned} \text{maximize} \quad & H(p,w) = -p' \ln(p) - w' \ln(w) \\ \text{subject to} \quad & X'y = X'X\,Z\,p + X'V\,w \\ & 1_K = (I_K \otimes 1_L')\,p \\ & 1_T = (I_T \otimes 1_L')\,w \end{aligned}$$

There is also the cross entropy version of GME-M, which has the same form as GCE but with the moment constraints.

GME versus GME-M

GME-M is more computationally attractive than GME for large data sets because the computational complexity of the estimation problem depends primarily on the number of parameters and not on the number of observations. GME-M is based on the first moment of the data, whereas GME is based on the data itself. If the distribution of the residuals is well defined by its first moment, then GME-M is a good choice. So if the residuals are normally distributed or exponentially distributed, then GME-M should be used. On the other hand if the distribution is Cauchy, lognormal, or some other distribution where the first moment does not describe the distribution, then use GME. See Example 14.1 for an illustration of this point.

Maximum Entropy-Based Seemingly Unrelated Regression

In a multivariate regression model, the errors in different equations might be correlated. In this case, the efficiency of the estimation can be improved by taking these cross-equation correlations into account. Seemingly unrelated regression (SUR), also called joint generalized least squares (JGLS) or Zellner estimation, is a generalization of OLS for multi-equation systems.

Like SUR in the least squares setting, the generalized maximum entropy SUR (GME-SUR) method assumes that all the regressors are independent variables and uses the correlations among the errors in different equations to improve the regression estimates. The GME-SUR method requires an initial entropy regression to compute residuals. The entropy residuals are used to estimate the cross-equation covariance matrix.

In the iterative GME-SUR (ITGME-SUR) case, the preceding process is repeated by using the residuals from the GME-SUR estimation to estimate a new cross-equation covariance matrix. ITGME-SUR method alternates between estimating the system coefficients and estimating the cross-equation covariance matrix until the estimated coefficients and covariance matrix converge.

The estimation problem becomes the generalized maximum entropy system adapted for multi-equations as follows:

$$
\begin{aligned}
\text{maximize} \quad & H(p,w) = -p' \ln(p) - w' \ln(w) \\
\text{subject to} \quad & y = X Z p + V w \\
& 1_{KM} = (I_{KM} \otimes 1_L') p \\
& 1_{MT} = (I_{MT} \otimes 1_L') w
\end{aligned}
$$

where

$$\beta = Z p$$

$$Z = \begin{bmatrix} z^1_{11} & \cdots & z^1_{L1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & z^K_{11} & \cdots & z^K_{L1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ddots & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & z^1_{1M} & \cdots & z^1_{LM} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ddots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & z^K_{1M} & \cdots & z^K_{LM} \end{bmatrix}$$

$$p = \begin{bmatrix} p^1_{11} & \cdot & p^1_{L1} & \cdot & p^K_{11} & \cdot & p^K_{L1} & \cdot & p^1_{1M} & \cdot & p^1_{LM} & \cdot & p^K_{1M} & \cdot & p^K_{LM} \end{bmatrix}'$$

$$e = V\,w$$

$$V = \begin{bmatrix} v^1_{11} & \cdots & v^L_{11} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & v^1_{1T} & \cdots & v^L_{1T} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ddots & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & v^1_{M1} & \cdots & v^L_{M1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ddots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & v^1_{MT} & \cdots & v^L_{MT} \end{bmatrix}$$

$$w = \begin{bmatrix} w^1_{11} & \cdot & w^L_{11} & \cdot & w^1_{1T} & \cdot & w^L_{1T} & \cdot & w^1_{M1} & \cdot & w^L_{M1} & \cdot & w^1_{MT} & \cdot & w^L_{MT} \end{bmatrix}'$$

y denotes the MT column vector of observations of the dependent variables; $\mathbf{X}$ denotes the (MT x KM) matrix of observations for the independent variables; p denotes the LKM column vector of weights associated with the points in Z; w denotes the LMT column vector of weights associated with the points in V; 1_L, 1_{KM}, and 1_{MT} are L-, KM-, and MT-dimensional column vectors, respectively, of ones; and $\mathbf{I_{KM}}$ and $\mathbf{I_{MT}}$ are (KM x KM) and (MT x MT) dimensional identity matrices. The subscript l denotes the support point ($l = 1, 2, \ldots, L$), k denotes the parameter ($k = 1, 2, \ldots, K$), m denotes the equation ($m = 1, 2, \ldots, M$), and t denotes the observation ($t = 1, 2, \ldots, T$).

Using this notation, the maximum entropy problem that is analogous to the OLS problem used as the initial step of the traditional SUR approach is

$$\begin{aligned} \text{maximize} \quad & H(p, w) = -p' \ln(p) - w' \ln(w) \\ \text{subject to} \quad & (y - X\,Z\,p) = \sqrt{\Sigma}\,V\,w \\ & 1_{KM} = (I_{KM} \otimes 1'_L)\,p \\ & 1_{MT} = (I_{MT} \otimes 1'_L)\,w \end{aligned}$$

The results are GME-SUR estimates with independent errors, the analog of OLS. The covariance matrix $\hat{\Sigma}$ is computed based on the residual of the equations, $Vw = e$. An $L'L$ factorization of the $\hat{\Sigma}$ is used to compute the square root of the matrix.

After solving this problem, these entropy-based estimates are analogous to the Aitken two-step estimator. For iterative GME-SUR, the covariance matrix of the errors is recomputed, and a new $\hat{\Sigma}$ is computed and factored. As in traditional ITSUR, this process repeats until the covariance matrix and the parameter estimates converge.

The estimation of the parameters for the normed-moment version of SUR (GME-SUR-NM) uses an identical process. The constraints for GME-SUR-NM is defined as:

$$X'y \;=\; X'(\mathbf{S}^{-1}\otimes\mathbf{I})X\;Z\;p \;+\; X'(\mathbf{S}^{-1}\otimes\mathbf{I})V\;w$$

The estimation of the parameters for GME-SUR-NM uses an identical process as outlined previously for GME-SUR.

Generalized Maximum Entropy for Multinomial Discrete Choice Models

Multinomial discrete choice models take the form of an experiment that consists of n trials. On each trial, one of k alternatives is observed. If y_{ij} is the random variable that takes on the value 1 when alternative j is selected for the ith trial and 0 otherwise, then the probability that y_{ij} is 1, conditional on a vector of regressors X_i and unknown parameter vector β_j, is

$$\Pr(y_{ij} = 1|X_i, \beta_j) = G(X_i'\beta_j)$$

where $G()$ is a link function. For noisy data the model becomes:

$$y_{ij} = G(X_i'\beta_j) + \epsilon_{ij} = p_{ij} + \epsilon_{ij}$$

The standard maximum likelihood approach for multinomial logit is equivalent to the maximum entropy solution for discrete choice models. The generalized maximum entropy approach avoids an assumption of the form of the link function $G()$.

The generalized maximum entropy for discrete choice models (GME-D) is written in primal form as

$$\begin{array}{lrl}
\text{maximize} & H(p,w) & = -p'\,\ln(p) \;-\; w'\,\ln(w) \\
\text{subject to} & (I_j \otimes X'y) & = (I_j \otimes X')p \;+\; (I_j \otimes X')V\,w \\
& \sum_j^k p_{ij} & = 1 \quad \text{for } i = 1 \text{ to } N \\
& \sum_m^L w_{ijm} & = 1 \quad \text{for } i = 1 \text{ to } N \text{ and } j = 1 \text{ to } k
\end{array}$$

Golan, Judge, and Miller (1996) have shown that the dual unconstrained formulation of the GME-D can be viewed as a general class of logit models. Additionally, as the sample size increases, the solution of the dual problem approaches the maximum likelihood solution. Because of these characteristics, only the dual approach is available for the GME-D estimation method.

The parameters β_j are the Lagrange multipliers of the constraints. The covariance matrix of the parameter estimates is computed as the inverse of the Hessian of the dual form of the objective function.

Censored or Truncated Dependent Variables

In practice, you might find that variables are not always measured throughout their natural ranges. A given variable might be recorded continuously in a range, but, outside of that range, only the endpoint is denoted. In other words, say that the data generating process is:

$$y_i = \mathbf{x}_{i\bullet} + \epsilon.$$

However, you observe the following:

$$y_i^\star = \begin{cases} ub & : \quad y_i \geq ub \\ \mathbf{x}_{i\bullet} + \epsilon & : \quad lb < y_i < ub \\ lb & : \quad y_i \leq lb \end{cases}$$

The primal problem is simply a slight modification of the primal formulation for GME-GCE. You specify different supports for the errors in the truncated or censored region, perhaps reflecting some nonsample information. Then the data constraints are modified. The constraints that arise in the censored areas are changed to inequality constraints (Golan, Judge, and Perloff 1997). Let the variable $\mathbf{X}^u$ denote the observations of the explanatory variable where censoring occurs from the top, $\mathbf{X}^l$ from the bottom, and $\mathbf{X}^a$ in the middle region (no censoring). Let, $\mathbf{V}^u$ be the supports for the observations at the upper bound, $\mathbf{V}^l$ lower bound, and $\mathbf{V}^a$ in the middle.

You have:

$$\begin{bmatrix} \mathbf{y}^u \geq ub \\ \mathbf{y}^a \\ \mathbf{y}^l \leq lb \end{bmatrix} = \begin{bmatrix} \mathbf{X}^u \\ \mathbf{X}^a \\ \mathbf{X}^l \end{bmatrix} \mathbf{Zp} + \begin{bmatrix} \mathbf{V}^u \mathbf{w}^u \\ \mathbf{V}^a \mathbf{w}^a \\ \mathbf{V}^l \mathbf{w}^l \end{bmatrix}$$

The primal problem then becomes

$$\begin{aligned} \text{maximize} \quad & H(p, w) = -p' \ln(p) - w' \ln(w) \\ \text{subject to} \quad & \mathbf{y}^a = \mathbf{X}^a \mathbf{V}^a p + \mathbf{V}^a \mathbf{w}^a \\ & \mathbf{y}^u \geq \mathbf{X}^u \mathbf{V}^u p + \mathbf{V}^u \mathbf{w}^u \\ & \mathbf{y}^l \leq \mathbf{X}^l \mathbf{V}^l p + \mathbf{V}^l \mathbf{w}^l \\ & 1_K = (I_K \otimes 1_L') p \\ & 1_T = (I_T \otimes 1_L') w \end{aligned}$$

PROC ENTROPY requires that the number of supports be identical for all three regions.

Alternatively, you can think of cases where the dependent variable is observed continuously for most of its range. However, the variable's range is reported for some observations. Such data is often found in highly disaggregated state level employment measures.

$$y_i^\star = \begin{cases} missing & : \quad l_1 \leq y \leq r_1 \\ \vdots & : \quad \vdots \\ missing & : \quad l_k \leq y \leq r_k \\ \mathbf{x}_{i\bullet} + \epsilon & : \quad otherwise \end{cases}$$

Just as in the censored case, each range yields two inequality constraints for each observation in that range.

Information Measures

PROC ENTROPY returns several measures of fit. First, the value of the objective function is returned. Next, the signal entropy is provided followed by the noise entropy. The sum of the noise and signal entropies should equal the value of the objective function. The next two metrics that follow are the normed entropies of both the signal and the noise.

Normalized entropy (NE) measures the relative informational content of both the signal and noise components through p and w, respectively (Golan, Judge, and Miller 1996). Let S denote the normalized entropy of the signal, $X\beta$, defined as:

$$S(\tilde{p}) = \frac{-\tilde{p}' \ln(\tilde{p})}{-q' \ln(q)}$$

where $S(\tilde{p}) \epsilon [0, 1]$. In the case of GME, where uniform priors are assumed, S can be written as:

$$S(\tilde{p}) = \frac{-\tilde{p}' \ln(\tilde{p})}{\sum_i \ln(M_i)}$$

where M_i is the number of support points for parameter i. A value of 0 for S implies that there is no uncertainty regarding the parameters; hence, it is a degenerate situation. However, a value of 1 implies that the posterior distributions equal the priors, which indicates total uncertainty if the priors are uniform.

Because NE is relative, it can be used for comparing various situations. Consider adding a data point to the model. If $S_{T+1} = S_T$, then there is no additional information contained within that data constraint. However, if $S_{T+1} < S_T$, then the data point gives a more informed set of parameter estimates.

NE can be used for determining the importance of particular variables with regard to the reduction of the uncertainty they bring to the model. Each of the k parameters that is estimated has an associated NE defined as

$$S(\tilde{p_k}) = \frac{-\tilde{p}_k' \ln(\tilde{p}_k)}{-\ln(q_k)}$$

or, in the GME case,

$$S(\tilde{p_k}) = \frac{-\tilde{p}_k' \ln(\tilde{p}_k)}{\ln(M)}$$

where $\tilde{p_k}$ is the vector of supports for parameter β_k and M is the corresponding number of support points. Since a value of 1 implies no relative information for that particular sample, Golan, Judge, and Miller (1996) suggest an exclusion criteria of $S(\tilde{p_k}) > 0.99$ as an acceptable means of selecting noninformative variables. See Golan, Judge, and Miller (1996) for some simulation results.

The final set of measures of fit are the parameter information index and error information index. These measures can be best summarized as 1 – the appropriate normed entropy.

Parameter Covariance For GCE

For the cross-entropy problem, the estimate of the asymptotic variance of the signal parameter is given by:

$$\hat{Var}(\hat{\beta}) = \frac{\hat{\sigma_\gamma^2}(\hat{\beta})}{\hat{\psi}^2(\hat{\beta})}(X'X)^{-1}$$

where

$$\hat{\sigma_\gamma^2}(\hat{\beta}) = \frac{1}{N}\sum_{i=1}^{N}\gamma_i^2$$

and γ_i is the Lagrange multiplier associated with the i th row of the Vw constraint matrix. Also,

$$\hat{\psi}^2(\hat{\beta}) = \left[\frac{1}{N}\sum_{i=1}^{N}\left(\sum_{j=1}^{J}v_{ij}^2 w_{ij} - (\sum_{j=1}^{J}v_{ij}w_{ij})^2\right)^{-1}\right]^2$$

Parameter Covariance For GCE-M

Golan, Judge, and Miller (1996) give the finite approximation to the asymptotic variance matrix of the moment formulation as:

$$\hat{Var}(\hat{\beta}) = \Sigma_z X'XC^{-1}DC^{-1}X'X\Sigma_z$$

where

$$C = X'X\Sigma_z X'X + \Sigma_v$$

and

$$D = X'\Sigma_e X$$

Recall that in the moment formulation, V is the support of $\frac{X'e}{T}$, which implies that Σ_v is a k-dimensional variance matrix. Σ_z and Σ_v are both diagonal matrices with the form

$$\Sigma_z = \begin{bmatrix} \sum_{l=1}^{L}z_{1l}^2 p_{1l} - (\sum_{l=1}^{L}z_{1l}p_{1l})^2 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sum_{l=1}^{L}z_{Kl}^2 p_{Kl} - (\sum_{l=1}^{L}z_{Kl}p_{Kl})^2 \end{bmatrix}$$

and

$$\Sigma_v = \begin{bmatrix} \sum_{j=1}^{J}v_{1j}^2 w_{jl} - (\sum_{j=1}^{J}v_{1j}w_{1j})^2 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sum_{j=1}^{J}v_{Kl}^2 w_{Kl} - (\sum_{j=1}^{J}v_{Kl}w_{Kl})^2 \end{bmatrix}$$

Statistical Tests

Since the GME estimates have been shown to be asymptotically normally distributed, the classical Wald, Lagrange multiplier, and likelihood ratio statistics can be used for testing linear restrictions on the parameters.

Wald Tests

Let $H_0 : L\beta = m$, where L is a set of linearly independent combinations of the elements of β. Then under the null hypothesis, the Wald test statistic,

$$T_W = (L\beta - m)' \left(L(\hat{Var}(\hat{\beta}))L' \right)^{-1} (L\beta - m)$$

has a central χ^2 limiting distribution with degrees of freedom equal to the rank of L.

Pseudo-Likelihood Ratio Tests

Using the conditionally maximized entropy function as a pseudo-likelihood, F, Mittelhammer and Cardell (2000) state that:

$$\frac{2\hat{\psi}(\hat{\beta})}{\hat{\sigma}_\gamma^2(\hat{\beta})} \left(F(\hat{\beta}) - F(\tilde{\beta}) \right)$$

has the limiting distribution of the Wald statistic when testing the same hypothesis. Note that $F(\hat{\beta})$ and $F(\tilde{\beta})$ are the maximum values of the entropy objective function over the full and restricted parameter spaces, respectively.

Lagrange Multiplier Tests

Again using the GME function as a pseudo-likelihood, Mittelhammer and Cardell (2000) define the Lagrange multiplier statistic as:

$$\frac{1}{\hat{\sigma}_\gamma^2(\tilde{\beta})} G(\tilde{\beta})'(X'X)^{-1} G(\tilde{\beta})$$

where G is the gradient of F, which is being evaluated at the optimum point for the restricted parameters. This test statistic shares the same limiting distribution as the Wald and pseudo-likelihood ratio tests.

Missing Values

If an observation in the input data set contains a missing value for any of the regressors or dependent values, that observation is dropped from the analysis.

Input Data Sets

DATA= Data Set

The DATA= data set specified in the PROC ENTROPY statement is the data set that contains the data to be analyzed.

PDATA= Data Set

The PDATA= data set specified in the PROC ENTROPY statement specifies the support points and prior probabilities to be used in the estimation. The PDATA= can be used in lieu of a PRIORS statement, but is intended for use in conjunction with the OUTP= option. Once priors are entered through a PRIORS statement, they can be reused in subsequent estimations by specifying the PDATA= option.

The variables in the data set are as follows:

- BY variables (if any)
- _TYPE_, a character variable of length 8 that identifies the estimation method: GME or GMEM. This is an optional column.
- variable, a character variable of length 32 that indicates the name of the regressor. The regressor name and the equation name identify a unique coefficient. This is required.
- _OBS_, a numeric variable that is either missing when the probabilities are for coefficients or the observation number when the probabilities are for the residual terms. The _OBS_ and the equation name identify which residual the probability is associated with. This an optional column.
- equation, a character variable of length 32 indicating the name of the dependent variable. This is a required column.
- NSupport, a numeric variable that indicates the number of support points for each basis. This variable is required.
- support, a numeric variable that is the support value the probability is associated with. This is a required column.
- prior, a numeric variable that is the prior probability associated with the probability. This is a required column.
- Prb, a numeric variable that is the estimated probability. This is optional.

SDATA= Data Set

The SDATA= data set specifies a data set that provides the covariance matrix of the equation errors. The matrix read from the SDATA= data set is used for the equation covariance matrix (**S** matrix) in the estimation. (The SDATA= **S** matrix is used to provide only the initial estimate of **S** for the methods that iterate the **S** matrix.)

Output Data Sets

OUT= Data Set

The OUT= data set specified in the PROC ENTROPY statement contains residuals of the dependent variables computed from the parameter estimates. The ID and BY variables are also added to this data set.

OUTEST= Data Set

The OUTEST= data set contains parameter estimates and, if requested via the COVOUT option, estimates of the covariance of the parameter estimates.

The variables in the data set are as follows:

- BY variables
- _NAME_, a character variable of length 32, blank for observations that contain parameter estimates or a parameter name for observations that contain covariances
- _TYPE_, a character variable of length 8 that identifies the estimation method: GME or GMEM
- the parameters estimated

If the COVOUT option is specified, an additional observation is written for each row of the estimate of the covariance matrix of parameter estimates, with the _NAME_ values containing the parameter names for the rows.

OUTP= Data Set

The OUTP= data set specified in the PROC ENTROPY statement contains the probabilities estimated for each support point, as well as the support points and prior probabilities used in the estimation.

The variables in the data set are as follows:

- BY variables (if any)
- _TYPE_, a character variable of length 8 that identifies the estimation method: GME or GMEM.
- variable, a character variable of length 32 that indicates the name of the regressor. The regressor name and the equation name identify a unique coefficient.
- _OBS_, a numeric variable that is either missing when the probabilities are for coefficients or the observation number when the probabilities are for the residual terms. The _OBS_ and the equation name identify which residual the probability is associated with.
- equation, a character variable of length 32 that indicates the name of the dependent variable
- NSupport, a numeric variable that indicates the number of support points for each basis
- support, a numeric variable that is the support value the probability is associated with
- prior, a numeric variable that is the prior probability associated with the probability
- Prb, a numeric variable that is the estimated probability

OUTL= Data Set

The OUTL= data set specified in the PROC ENTROPY statement contains the Lagrange multiplier values for the underlying maximum entropy problem.

The variables in the data set are as follows:

- BY variables
- equation, a character variable of length 32 that indicates the name of the dependent variable
- variable, a character variable of length 32 that indicates the name of the regressor. The regressor name and the equation name identify a unique coefficient.
- _OBS_, a numeric variable that is either missing when the probabilities are for coefficients or the observation number when the probabilities are for the residual terms. The _OBS_ and the equation name identify which residual the Lagrange multiplier is associated with
- LagrangeMult, a numeric variable that contains the Lagrange multipliers

ODS Table Names

PROC ENTROPY assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in the following table.

Table 14.2 ODS Tables Produced in PROC ENTROPY

ODS Table Name	Description	Option
ConvCrit	Convergence criteria for estimation	default
ConvergenceStatus	Convergence status	default
DatasetOptions	Data sets used	default
MinSummary	Number of parameters, estimation kind	default
ObsUsed	Observations read, used, and missing	default
ParameterEstimates	Parameter estimates	default
ResidSummary	Summary of the SSE, MSE for the equations	default
TestResults	Test statement table	TEST statement

ODS Graphics

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in Chapter 21, “Statistical Graphics Using ODS” (*SAS/STAT User’s Guide*).

Before you create graphs, ODS Graphics must be enabled (for example, with the ODS GRAPHICS ON statement). For more information about enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" in that chapter.

The overall appearance of graphs is controlled by ODS styles. Styles and other aspects of using ODS Graphics are discussed in the section "A Primer on ODS Statistical Graphics" in that chapter.

This section describes the use of ODS for creating graphics with the ENTROPY procedure.

ODS Graph Names

PROC ENTROPY assigns a name to each graph it creates using ODS. You can use these names to reference the graphs when using ODS. The names are listed in Table 14.3.

To request these graphs, you must specify the ODS GRAPHICS statement.

Table 14.3 ODS Graphics Produced by PROC ENTROPY

ODS Graph Name	Plot Description
DiagnosticsPanel	Includes all the plots listed below
FitPlot	Predicted versus actual plot
CooksD	Cook's *D* plot
QQPlot	Q-Q plot of residuals
StudentResidualPlot	Studentized residual plot
ResidualHistogram	Histogram of the residuals

Examples: ENTROPY Procedure

Example 14.1: Nonnormal Error Estimation

This example illustrates the difference between GME-M and GME. One of the basic assumptions of OLS estimation is that the errors in the estimation are normally distributed. If this assumption is violated, the estimated parameters are biased. For GME-M, the story is similar. If the first moment of the distribution of the errors and a scale factor cannot be used to describe the distribution, then the parameter estimates from GME-MN are more biased. GME is much less sensitive to the underlying distribution of the errors than GME-M.

To illustrate this, data for the following model is simulated with three different error distributions:

$$y = a * x_1 + b * x_2 + \epsilon.$$

For the first simulation, ϵ is distributed normally, then a chi-squared distribution with six degrees of freedom is assumed for the second simulation, and finally ϵ is assumed to have a Cauchy distribution in the third simulation.

In each of the three simulations, 100 samples of 10 observations each were simulated. The data for the model with the Cauchy error distribution is generated using the following DATA step code:

```
data one;
   call streaminit(156789);
   do by = 1 to 100;
      do x2 = 1 to 10;
         x1 = 10 * ranuni( 512);
         y = x1 + 2*x2 + rand('cauchy');
         output;
      end;
   end;
run;
```

The statements for the other distributions are identical except for the argument to the RAND() function.

The parameters to the model were estimated by using maximum entropy with the following programming statements:

```
proc entropy data=one gme outest=parm1;
   model y = x1 x2;
   by by;
run;
```

The estimation by using moment-constrained maximum entropy was performed by changing the GME option to GMEM. For comparison, the same model was estimated by using OLS with the following PROC REG statements:

```
proc reg data=one outest=parm3;
   model y = x1 x2;
   by by;
run;
```

The 100 estimations of the coefficient on variable x1 are then summarized for each of the three error distributions by using PROC UNIVARIATE, as follows:

```
proc univariate data=parm1;
   var x1;
run;
```

The following table summarizes the results from the estimations. The true value for the coefficient on x1 is 1.0.

Estimation	Normal		Chi-Squared		Cauchy	
Method	Mean	Std Deviation	Mean	Std Deviation	Mean	Std Deviation
GME	0.418	0.117	0.626	.330	0.818	3.36
GME-M	0.878	0.116	0.948	0.427	3.03	13.62
OLS	0.973	0.142	1.023	0.467	5.54	26.83

For normally distributed or nearly normally distributed data, moment-constrained maximum entropy is a good choice. For distributions not well described by a normal distribution, data-constrained maximum entropy is a good choice.

Example 14.2: Unreplicated Factorial Experiments

Factorial experiments are useful for studying the effects of various factors on a response. For the practitioner constrained to the use of OLS regression, there must be replication to estimate all of the possible main and interaction effects in a factorial experiment. Using OLS regression to analyze unreplicated experimental data results in zero degrees of freedom for error in the ANOVA table, since there are as many parameters as observations. This situation leaves the experimenter unable to compute confidence intervals or perform hypothesis testing on the parameter estimates.

Several options are available when replication is impossible. The higher-order interactions can be assumed to have negligible effects, and their degrees of freedom can be pooled to create the error degrees of freedom used to perform inference on the lower-order estimates. Or, if a preliminary experiment is being run, a normal probability plot of all effects can provide insight as to which effects are significant, and therefore focused, in a later, more complete experiment.

The following example illustrates the probability plot methodology and the alternative by using PROC ENTROPY. Consider a 2^4 factorial model with no replication. The data are taken from Myers and Montgomery (1995).

```
data rate;
   do a=-1,1; do b=-1,1; do c=-1,1; do d=-1,1;
      input y @@;
      ab=a*b; ac=a*c; ad=a*d; bc=b*c; bd=b*d; cd=c*d;
      abc=a*b*c; abd=a*b*d; acd=a*c*d; bcd=b*c*d;
      abcd=a*b*c*d;
      output;
   end; end; end; end;
   datalines;
   45 71 48 65 68 60 80 65 43 100 45 104 75 86 70 96
   ;
run;
```

Analyze the data by using PROC REG, then output the resulting estimates.

```
proc reg data=rate outest=regout;
   model y=a b c d ab ac ad bc bd cd abc abd acd bcd abcd;
run;

proc transpose data=regout out=ploteff name=effect prefix=est;
   var a b c d ab ac ad bc bd cd abc abd acd bcd abcd;
run;
```

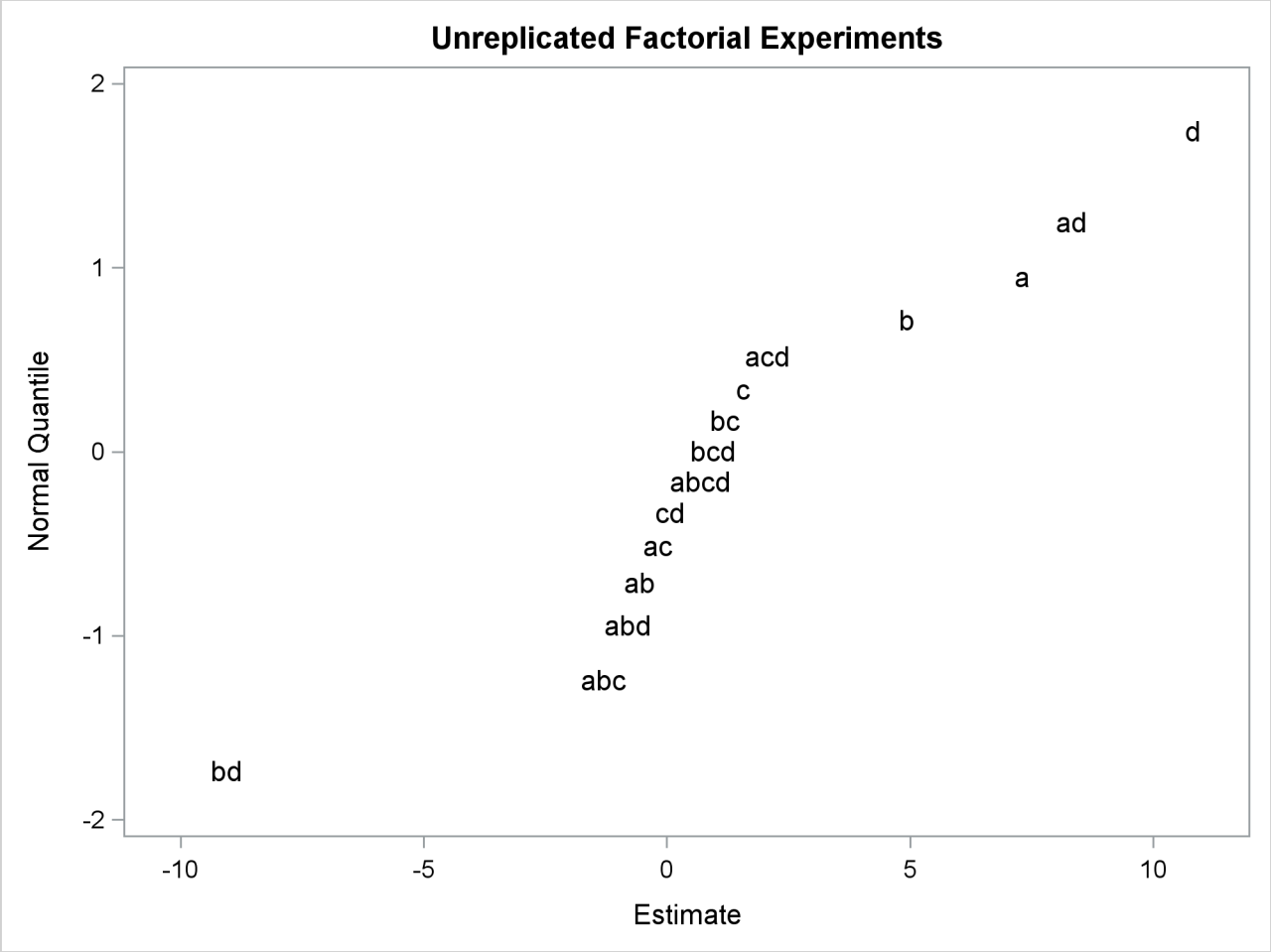
Now the normal scores for the estimates can be computed with the rank procedure as follows:

```
proc rank data=ploteff normal=blom out=qqplot;
   var est1;
   ranks normalq;
run;
```

To create the probability plot, simply plot the estimates versus their normal scores by using PROC SGPLOT as follows:

```
title "Unreplicated Factorial Experiments";
proc sgplot data=qqplot;
   scatter x=est1 y=normalq / markerchar=effect
                              markercharattrs=(size=10pt);
   xaxis label="Estimate";
   yaxis label="Normal Quantile";
run;
```

Output 14.2.1 Normal Probability Plot of Effects



The plot shown in Output 14.2.1 displays evidence that the a, b, d, ad, and bd estimates do not fit into the purely random normal model, which suggests that they may have some significant effect on the response variable. To verify this, fit a reduced model that contains only these effects.

```
proc reg data=rate;
   model y=a b d ad bd;
run;
```

The estimates for the reduced model are shown in Output 14.2.2.

Output 14.2.2 Reduced Model OLS Estimates

Unreplicated Factorial Experiments

The REG Procedure
Model: MODEL1
Dependent Variable: y

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	70.06250	1.10432	63.44	<.0001
a	1	7.31250	1.10432	6.62	<.0001
b	1	4.93750	1.10432	4.47	0.0012
d	1	10.81250	1.10432	9.79	<.0001
ad	1	8.31250	1.10432	7.53	<.0001
bd	1	-9.06250	1.10432	-8.21	<.0001

These results support the probability plot methodology.

PROC ENTROPY can directly estimate the full model without having to rely upon the probability plot for insight into which effects can be significant. To illustrate this, PROC ENTROPY is run by using default parameter and error supports in the following statements:

```
proc entropy data=rate;
   model y=a b c d ab ac ad bc bd cd abc abd acd bcd abcd;
run;
```

The resulting GME estimates are shown in Output 14.2.3. Note that the parameter estimates associated with the a, b, d, ad, and bd effects are all significant.

Output 14.2.3 Full Model Entropy Results

Unreplicated Factorial Experiments

The ENTROPY Procedure

GME Variable Estimates

Variable	Estimate	Approx Std Err	t Value	Approx Pr > |t|
a	5.688414	0.7911	7.19	<.0001
b	2.988032	0.5464	5.47	<.0001
c	0.234331	0.1379	1.70	0.1086
d	9.627308	0.9765	9.86	<.0001
ab	-0.01386	0.0270	-0.51	0.6149
ac	-0.00054	0.00325	-0.16	0.8712
ad	6.833076	0.8627	7.92	<.0001
bc	0.113908	0.0941	1.21	0.2435
bd	-7.68105	0.9053	-8.48	<.0001
cd	0.00002	0.000364	0.05	0.9569
abc	-0.14876	0.1087	-1.37	0.1900
abd	-0.0399	0.0516	-0.77	0.4509
acd	0.466938	0.1961	2.38	0.0300
bcd	0.059581	0.0654	0.91	0.3756
abcd	0.024785	0.0387	0.64	0.5312
Intercept	69.87294	1.1403	61.28	<.0001

Example 14.3: Censored Data Models in PROC ENTROPY

Data available to an analyst might sometimes be censored, where only part of the actual series is observed. Consider the case in which only observations greater than some lower bound are recorded, as defined by the following process:

$$\mathbf{y} = max\left(\mathbf{X}\boldsymbol{\beta} + \epsilon, lb\right).$$

Running ordinary least squares estimation on data generated by the preceding process is not optimal because the estimates are likely to be biased and inefficient. One alternative to estimating models with censored data is the tobit estimator. This model is supported in the QLIM procedure in SAS/ETS and in the LIFEREG procedure in SAS/STAT. PROC ENTROPY provides another alternative which can make it very easy to estimate such a model correctly.

The following DATA step generates censored data in which any negative values of the dependent variable, y, are set to a lower bound of 0.

```
data cens;
  do t = 1 to 100;
     x1 = 5 * ranuni(456);
     x2 = 10 * ranuni(456);
     y = 4.5*x1 + 2*x2 + 15 * rannor(456);
     if( y<0 ) then y = 0;
     output;
   end;
run;
```

To illustrate the effect of the censored option in PROC ENTROPY, the model is initially estimated without accounting for censoring in the following statements:

```
title "Censored Data Estimation";
proc entropy data = cens gme primal;
   priors intercept -32 32
          x1        -15 15
          x2        -15 15;
    model y = x1 x2 /
          esupports = (-25 1 25);
run;
```

Output 14.3.1 GME Estimates

Censored Data Estimation

The ENTROPY Procedure

GME Variable Estimates

Variable	Estimate	Approx Std Err	t Value	Approx Pr > |t|
x1	2.377609	0.000503	4725.98	<.0001
x2	2.353014	0.000255	9244.87	<.0001
intercept	5.478121	0.00188	2906.41	<.0001

The previous model is reestimated by using the CENSORED option in the following statements:

```
proc entropy data = cens gme primal;
   priors intercept -32 32
          x1        -15 15
          x2        -15 15;
   model y = x1 x2 /
         esupports = (-25 1 25)
         censored(lb = 0, esupports=(-15 1 15) );
run;
```

Output 14.3.2 Entropy Estimates

Censored Data Estimation

The ENTROPY Procedure

GME Variable Estimates

Variable	Estimate	Approx Std Err	t Value	Approx Pr > |t|
x1	4.429697	0.00690	641.85	<.0001
x2	1.46858	0.00349	420.61	<.0001
intercept	8.261412	0.0259	319.51	<.0001

The second set of entropy estimates are much closer to the true parameter estimates of 4.5 and 2. Since another alternative available for fitting a model of censored data is a tobit model, PROC QLIM is used in the following statements to fit a tobit model to the data:

```
proc qlim data=cens;
   model y = x1 x2;
   endogenous y ~ censored(lb=0);
run;
```

Output 14.3.3 QLIM Estimates

Censored Data Estimation

The QLIM Procedure

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	2.979455	3.824252	0.78	0.4359
x1	1	4.882284	1.019913	4.79	<.0001
x2	1	1.374006	0.513000	2.68	0.0074
_Sigma	1	13.723213	1.032911	13.29	<.0001

For this data and code, PROC ENTROPY produces estimates that are closer to the true parameter values than those computed by PROC QLIM.

Example 14.4: Use of the PDATA= Option

It is sometimes useful to specify priors and supports by using the PDATA= option. This example illustrates how to create a PDATA= data set which contains the priors and support points for use in a subsequent PROC ENTROPY step. In order to have a model to estimate in PROC ENTROPY, you must first have data to analyze. The following DATA step generates the data used in this analysis:

```
title "Using a PDATA= data set";
data a;
   array x[4];
   do t = 1 to 100;
      ys = -5;
      do k = 1 to 4;
         x[k] = rannor( 55372 )  ;
         ys = ys + x[k] * k;
      end;
      ys = ys + rannor( 55372 );
      output;
    end;
run;
```

Next you fit this data with some arbitrary parameter support points and priors by using the following PROC ENTROPY statements:

```
proc entropy data = a gme primal;
   priors          x1  -10(2) 30(1)
                   x2  -20(3) 30(2)
                   x3  -15(4) 30(4)
                   x4  -25(3) 30(2)
             intercept  -13(4) 30(2) ;
   model ys = x1 x2 x3 x4 / esupports=(-25 0 25);
run;
```

These statements produce the output shown in Output 14.4.1.

Output 14.4.1 Output From PROC ENTROPY

Using a PDATA= data set

The ENTROPY Procedure

GME Variable Estimates

Variable	Estimate	Approx Std Err	t Value	Approx Pr > |t|
x1	1.195688	0.1078	11.09	<.0001
x2	1.844903	0.1018	18.12	<.0001
x3	3.268396	0.1136	28.77	<.0001
x4	3.908194	0.0934	41.83	<.0001
intercept	-4.94319	0.1005	-49.21	<.0001

You can estimate the same model by first creating a PDATA= data set, which includes the same information as the PRIORS statement in the preceding PROC ENTROPY step.

A data set that defines the supports and priors for the model parameters is shown in the following statements:

```
data test;
   length Variable $ 12 Equation $ 12;
   input Variable $ Equation $ Nsupport Support Prior ;
datalines;
     Intercept    .  2 -13 0.66667
     Intercept    .  2  30 0.33333
            x1    .  2 -10 0.66667
            x1    .  2  30 0.33333
            x2    .  2 -20 0.60000
            x2    .  2  30 0.40000
            x3    .  2 -15 0.50000
            x3    .  2  30 0.50000
            x4    .  2 -25 0.60000
            x4    .  2  30 0.40000
;
```

The following statements reestimate the model by using these support points.

```
proc entropy data=a gme primal pdata=test;
   model ys = x1 x2 x3 x4 / esupports=(-25 0 25);
run;
```

These statements produce the output shown in Output 14.4.2.

Output 14.4.2 Output From PROC ENTROPY with PDATA= option

Using a PDATA= data set

The ENTROPY Procedure

GME Variable Estimates

Variable	Estimate	Approx Std Err	t Value	Approx Pr > |t|
x1	1.195686	0.1078	11.09	<.0001
x2	1.844902	0.1018	18.12	<.0001
x3	3.268395	0.1136	28.77	<.0001
x4	3.908194	0.0934	41.83	<.0001
Intercept	-4.94319	0.1005	-49.21	<.0001

These results are identical to the ones produced by the previous PROC ENTROPY step.

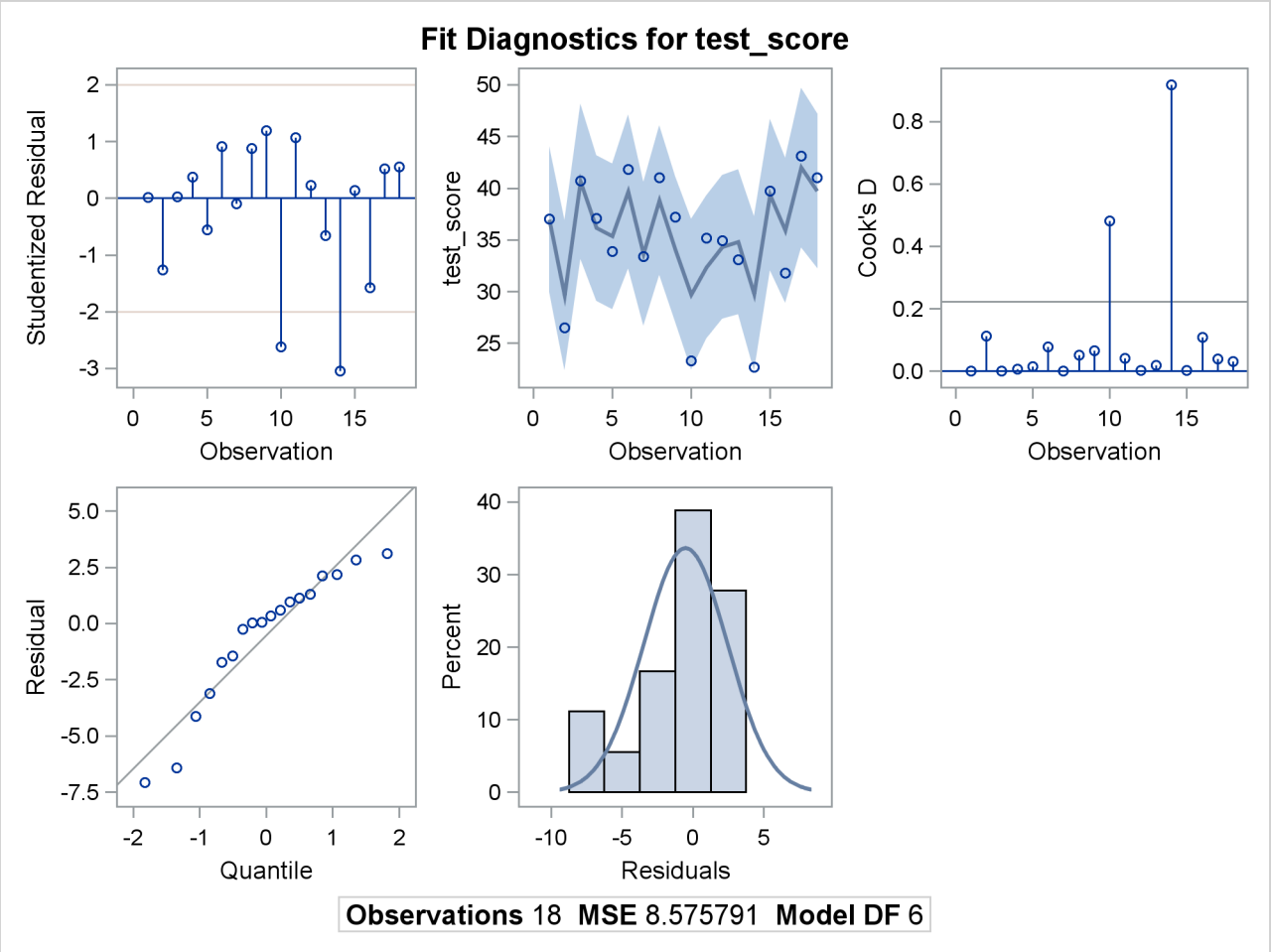
Example 14.5: Illustration of ODS Graphics

This example illustrates how to use ODS graphics in the ENTROPY procedure. This example is a continuation of the example in the section "Simple Regression Analysis" on page 788. Graphical displays are requested by specifying the ODS GRAPHICS statement. For information about the graphics available in the ENTROPY procedure, see the section "ODS Graphics" on page 832.

The following statements show how to generate ODS graphics plots with the ENTROPY procedure. The plots are displayed in Output 14.5.1.

```
proc entropy data=coleman;
   model test_score = teach_sal prcnt_prof socio_stat
                      teach_score mom_ed;
run;
```

Output 14.5.1 Model Diagnostics Plots



References

Coleman, J. S., Campbell, E. Q., Hobson, C. J., McPartland, J., Mood, A. M., Weinfeld, F. D., and York, R. L. (1966). *Equality of Educational Opportunity*. Washington, DC: US Government Printing Office.

Deaton, A., and Muellbauer, J. (1980). "An Almost Ideal Demand System." *American Economic Review* 70:312–326.

Golan, A., Judge, G. G., and Miller, D. J. (1996). *Maximum Entropy Econometrics: Robust Estimation with Limited Data*. Chichester, UK: John Wiley & Sons.

Golan, A., Judge, G. G., and Perloff, J. (1996). "A Generalized Maximum Entropy Approach to Recovering Information from Multinomial Response Data." *Journal of the American Statistical Association* 91:841–853.

Golan, A., Judge, G. G., and Perloff, J. (1997). "Estimation and Inference with Censored and Ordered Multinomial Response Data." *Journal of Econometrics* 79:23–51.

Golan, A., Judge, G. G., and Perloff, J. (2002). "Comparison of Maximum Entropy and Higher-Order Entropy Estimators." *Journal of Econometrics* 107:195–211.

Good, I. J. (1963). "Maximum Entropy for Hypothesis Formulation, Especially for Multidimensional Contingency Tables." *Annals of Mathematical Statistics* 34:911–934.

Harmon, A. M., Preckel, P., and Eales, J. (1998). "Maximum Entropy-Based Seemingly Unrelated Regression." Staff Paper 98-8, Department of Agricultural Economics, Purdue University. http://ageconsearch.umn.edu/bitstream/28682/1/sp98-08.pdf.

Jaynes, E. T. (1957). "Information of Theory and Statistical Mechanics." *Physics Review* 106:620–630.

Jaynes, E. T. (1963). "Information Theory and Statistical Mechanics." In *Statistical Physics*, edited by K. W. Ford, 181–218. Vol. 3 of Brandeis University Summer Institute/Lectures in Theoretical Physics. New York: W. A. Benjamin.

Kapur, J. N., and Kesavan, H. K. (1992). *Entropy Optimization Principles with Applications*. Boston: Academic Press.

Kullback, J. (1959). *Information Theory and Statistics*. New York: John Wiley & Sons.

Kullback, J., and Leibler, R. A. (1951). "On Information and Sufficiency." *Annals of Mathematical Statistics* 22:79–86.

LaMotte, L. R. (1994). "A Note on the Role of Independence in *t* Statistics Constructed from Linear Statistics in Regression Models." *American Statistician* 48:238–240.

Miller, D., Eales, J., and Preckel, P. (2003). "Quasi-maximum Likelihood Estimation with Bounded Symmetric Errors." In *Advances in Econometrics*, vol. 17, edited by T. B. Fomby, and R. C. Hill, 133–148. Amsterdam: Elsevier Science.

Mittelhammer, R. C., and Cardell, S. (2000). "The Data-Constrained GME Estimator of the GLM: Asymptotic Theory and Inference." Working paper, Department of Statistics, Washington State University, Pullman.

Mittelhammer, R. C., Judge, G. G., and Miller, D. J. (2000). *Econometric Foundations*. Cambridge: Cambridge University Press.

Myers, R. H., and Montgomery, D. C. (1995). *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*. New York: John Wiley & Sons.

Shannon, C. E. (1948). "A Mathematical Theory of Communication." *Bell System Technical Journal* 27:379–423, 623–656.

Chapter 15
The ESM Procedure

Contents

Overview: ESM Procedure **848**
Getting Started: ESM Procedure **848**
Syntax: ESM Procedure **850**
- Functional Summary 850
- PROC ESM Statement 852
- BY Statement 855
- FORECAST Statement 855
- ID Statement 857

Details: ESM Procedure **860**
- Accumulation 860
- Missing Value Interpretation 862
- Transformations 862
- Parameter Estimation 862
- Missing Value Modeling Issues 863
- Forecasting 863
- Inverse Transformations 863
- Statistics of Fit 863
- Forecast Summation 863
- Data Set Output 864
- Printed Output 868
- ODS Table Names 869
- ODS Graphics 870

Examples: ESM Procedure **871**
- Example 15.1: Forecasting of Time Series Data 871
- Example 15.2: Forecasting of Transactional Data 873
- Example 15.3: Specifying the Forecasting Model 875
- Example 15.4: Extending the Independent Variables for Multivariate Forecasts 875
- Example 15.5: Illustration of ODS Graphics 877

Overview: ESM Procedure

The ESM procedure generates forecasts by using exponential smoothing models with optimized smoothing weights for many time series or transactional data.

- For typical time series, you can use the following smoothing models:
  - simple
  - double
  - linear
  - damped trend
  - seasonal
  - Winters method (additive and multiplicative)
- Additionally, transformed versions of these models are provided:
  - log
  - square root
  - logistic
  - Box-Cox

Graphics are available with the ESM procedure. For more information, see the section "ODS Graphics" on page 870.

The exponential smoothing models supported in PROC ESM differ from those supported in PROC FORECAST since all parameters associated with the forecasting model are optimized by PROC ESM based on the data.

The ESM procedure writes the time series extrapolated by the forecasts, the series summary statistics, the forecasts and confidence limits, the parameter estimates, and the fit statistics to output data sets. The ESM procedure optionally produces printed output for these results by using the Output Delivery System (ODS).

The ESM procedure can forecast both time series data, whose observations are equally spaced by a specific time interval (for example, monthly, weekly), or transactional data, whose observations are not spaced with respect to any particular time interval. Internet, inventory, sales, and similar data are typical examples of transactional data. For transactional data, the data are accumulated based on a specified time interval to form a time series prior to modeling and forecasting.

Getting Started: ESM Procedure

The ESM procedure is simple to use and does not require in-depth knowledge of forecasting methods. It can provide results in output data sets or in other output formats by using the Output Delivery System (ODS). The following examples are more fully illustrated in "Example 15.2: Forecasting of Transactional Data" on page 873.

Given an input data set that contains numerous time series variables recorded at a specific frequency, the ESM procedure can forecast the series as follows:

```
proc esm data=<input-data-set> out=<output-data-set>;
   id <time-ID-variable> interval=<frequency>;
   forecast <time-series-variables>;
run;
```

For example, suppose that the input data set SALES contains sales data recorded monthly, the variable that represents time is DATE, and the forecasts are to be recorded in the output data set NEXTYEAR. The ESM procedure could be used as follows:

```
proc esm data=sales out=nextyear;
   id date interval=month;
   forecast _numeric_;
run;
```

The preceding statements generate forecasts for every numeric variable in the input data set SALES for the next twelve months and store these forecasts in the output data set NEXTYEAR. Other output data sets can be specified to store the parameter estimates, forecasts, statistics of fit, and summary data.

By default, PROC ESM generates no printed output. If you want to print the forecasts by using the Output Delivery System (ODS), then you need to add the PRINT=FORECASTS option to the PROC ESM statement, as shown in the following example:

```
proc esm data=sales out=nextyear print=forecasts;
   id date interval=month;
   forecast _numeric_;
run;
```

Other PRINT= options can be specified to print the parameter estimates, statistics of fit, and summary data.

The ESM procedure can forecast both time series data, whose observations are equally spaced by a specific time interval (for example, monthly, weekly), or transactional data, whose observations are not spaced with respect to any particular time interval.

Given an input data set that contains transactional variables not recorded at any specific frequency, the ESM procedure accumulates the data to a specific time interval and forecasts the accumulated series as follows:

```
proc esm data=<input-data-set> out=<output-data-set>;
   id <time-ID-variable> interval=<frequency>
                         accumulate=<accumulation>;
   forecast <time-series-variables> / model=<esm>;
run;
```

For example, suppose that the input data set WEBSITES contains three variables (BOATS, CARS, PLANES) that are Internet data recorded on no particular time interval, and the variable that represents time is TIME, which records the time of the Web hit. The forecasts for the total daily values are to be recorded in the output data set NEXTWEEK. The ESM procedure could be used as follows:

```
proc esm data=websites out=nextweek lead=7;
   id time interval=dtday accumulate=total;
   forecast boats cars planes;
run;
```

The preceding statements accumulate the data into a daily time series, generate forecasts for the BOATS, CARS, and PLANES variables in the input data set (WEBSITES) for the next seven days, and store the forecasts in the output data set (NEXTWEEK). Because the MODEL= option is not specified in the FORECAST statement, a simple exponential smoothing model is fit to each series.

Syntax: ESM Procedure

The following statements are used with the ESM procedure:

PROC ESM *options* **;**
BY *variables* **;**
ID *variable INTERVAL= interval options* **;**
FORECAST *variable-list / options* **;**

Functional Summary

The statements and options that control the ESM procedure are summarized in the following table.

Table 15.1 Syntax Summary

Description	Statement	Option
Statements		
specify data sets and options	PROC ESM	
specify BY-group processing	BY	
specify variables to forecast	FORECAST	
specify the time ID variable	ID	
Data Set Options		
specify the input data set	PROC ESM	DATA=
specify to output forecasts only	PROC ESM	NOOUTALL
specify the output data set	PROC ESM	OUT=
specify parameter output data set	PROC ESM	OUTEST=
specify forecast output data set	PROC ESM	OUTFOR=
specify the forecast procedure information output data set	PROC ESM	OUTPROCINFO=

Description	Statement	Option
specify statistics output data set	PROC ESM	OUTSTAT=
specify summary output data set	PROC ESM	OUTSUM=
replace actual values held back	FORECAST	REPLACEBACK
replace missing values	FORECAST	REPLACEMISSING
use forecast value to append	FORECAST	USE=
Accumulation and Seasonality Options		
specify accumulation frequency	ID	INTERVAL=
specify length of seasonal cycle	PROC ESM	SEASONALITY=
specify interval alignment	ID	ALIGN=
specify that time ID variable values are not sorted	ID	NOTSORTED
specify starting time ID value	ID	START=
specify ending time ID value	ID	END=
specify accumulation statistic	ID, FORECAST	ACCUMULATE=
specify missing value interpretation	ID, FORECAST	SETMISSING=
specify zero value interpretation	ID, FORECAST	ZEROMISS=
Forecasting Horizon, Holdback Options		
specify data to hold back	PROC ESM	BACK=
specify forecast horizon or lead	PROC ESM	LEAD=
specify horizon to start summation	PROC ESM	STARTSUM=
Forecasting Model Options		
specify confidence limit width	FORECAST	ALPHA=
specify forecast model	FORECAST	MODEL=
specify median forecasts	FORECAST	MEDIAN
specify backcast initialization	FORECAST	NBACKCAST=
specify model transformation	FORECAST	TRANSFORM=
Printing and Plotting Control Options		
specify time ID format	ID	FORMAT=
specify graphical output	PROC ESM	PLOT=
specify printed output	PROC ESM	PRINT=
specify detailed printed output	PROC ESM	PRINTDETAILS
Miscellaneous Options		
specify that analysis variables are processed in sorted order	PROC ESM	SORTNAMES
limit error and warning messages	PROC ESM	MAXERROR=

PROC ESM Statement

PROC ESM *options* **;**

The following options can be used in the PROC ESM statement.

BACK=*n*
specifies the number of observations before the end of the data where the multistep forecasts are to begin. The default is BACK=0.

DATA=*SAS-data-set*
names the SAS data set that contains the input data for the procedure to forecast. If the DATA= option is not specified, the most recently created SAS data set is used.

LEAD=*n*
specifies the number of periods ahead to forecast (forecast lead or horizon). The default is LEAD=12.

The LEAD= value is relative to the BACK= option specification and to the last observation in the input data set or the accumulated series, and not to the last nonmissing observation of a particular series. Thus, if a series has missing values at the end, the actual number of forecasts computed for that series is greater than the LEAD= value.

MAXERROR=*number*
limits the number of warning and error messages produced during the execution of the procedure to the specified value. The default is MAXERRORS=50. This option is particularly useful in BY-group processing where it can be used to suppress the recurring messages.

NOOUTALL
specifies that only forecasts are written to the OUT= and OUTFOR= data sets. The NOOUTALL option includes only the final forecast observations in the output data sets; it does not include the one-step forecasts for the data before the forecast period.

The OUT= and OUTFOR= data set will only contain the forecast results starting at the next period following the last observation and ending with the forecast horizon specified by the LEAD= option.

OUT=*SAS-data-set*
names the output data set to contain the forecasts of the variables specified in the subsequent FORECAST statements. If an ID variable is specified, it is also included in the OUT= data set. The values are accumulated based on the ACCUMULATE= option, and forecasts are appended to these values based on the FORECAST statement USE= option. The OUT= data set is particularly useful in extending the independent variables. The OUT= data set can be used as the input data set in a subsequent PROC step to forecast a dependent series by using a regression modeling procedure. If the OUT= option is not specified, a default output data set is created by using the DATA*n* convention. If you do not want the OUT= data set created, use OUT=_NULL_.

OUTEST=*SAS-data-set*
names the output data set to contain the model parameter estimates and the associated test statistics and probability values. The OUTEST= data set is useful for evaluating the significance of the model parameters and understanding the model dynamics.

OUTFOR=*SAS-data-set*
names the output data set to contain the forecast time series components (actual, predicted, lower confidence limit, upper confidence limit, prediction error, prediction standard error). The OUTFOR= data set is useful for displaying the forecasts in tabular or graphical form.

OUTPROCINFO=*SAS-data-set*
names the output data set to contain information in the SAS log, specifically the number of notes, errors, and warnings and the number of series processed, forecasts requested, and forecasts failed.

OUTSTAT=*SAS-data-set*
names the output data set to contain the statistics of fit (or goodness-of-fit statistics). The OUTSTAT= data set is useful for evaluating how well the model fits the series.

OUTSUM=*SAS-data-set*
names the output data set to contain the summary statistics and the forecast summation. The summary statistics are based on the accumulated time series when the ACCUMULATE= or SETMISSING= options are specified. The forecast summations are based on the LEAD=, STARTSUM=, and USE= options. The OUTSUM= data set is useful when forecasting large numbers of series and a summary of the results are needed.

PLOT=*option* **|** **(** *options* **)**
specifies the graphical output desired. By default, the ESM procedure produces no graphical output. The following plotting options are available:

ACF plots prediction error autocorrelation function graphics.

ALL is the same as specifying all of the PLOT= options.

BASIC equivalent to specifying PLOT=(CORR ERRORS MODELFORECASTS).

CORR plots the prediction error series graphics panel containing the ACF, IACF, PACF, and white noise probability plots.

ERRORS plots prediction error time series graphics.

FORECASTS plots forecast graphics.

FORECASTSONLY plots the forecast in the forecast horizon only.

IACF plots prediction error inverse autocorrelation function graphics.

LEVELS plots smoothed level component graphics.

MODELFORECASTS plots the one-step ahead model forecast and its confidence bands in the historical period; the forecast and its confidence bands over the forecast horizon.

MODELS plots model graphics.

PACF plots prediction error partial autocorrelation function graphics.

PERIODOGRAM plots prediction error periodogram.

SEASONS plots smoothed seasonal component graphics.

SPECTRUM plots periodogram and smoothed periodogram of the prediction error series in a single graph.

TRENDS plots smoothed trend (slope) component graphics.

WN plots white noise graphics.

For example, PLOT=FORECASTS plots the forecasts for each series. The PLOT= option produces printed output for these results by using the Output Delivery System (ODS).

PRINT=*option* | **(** *options* **)**
specifies the printed output desired. By default, the ESM procedure produces no printed output. The following printing options are available:

ESTIMATES	prints the results of parameter estimation.
FORECASTS	prints the forecasts.
PERFORMANCE	prints the performance statistics for each forecast.
PERFORMANCESUMMARY	prints the performance summary for each BY group.
PERFORMANCEOVERALL	prints the performance summary for all of the BY groups.
STATISTICS	prints the statistics of fit.
STATES	prints the backcast, initial, and final states.
SUMMARY	prints the summary statistics for the accumulated time series.
ALL	Same as PRINT=(ESTIMATES FORECASTS STATISTICS SUMMARY).

For example, PRINT=FORECASTS prints the forecasts, PRINT=(ESTIMATES FORECASTS) prints the parameter estimates and the forecasts, and PRINT=ALL prints all of the output.

PRINTDETAILS
specifies that output requested with the PRINT= option be printed in greater detail.

SEASONALITY=*number*
specifies the length of the seasonal cycle. For example, SEASONALITY=3 means that every group of three observations forms a seasonal cycle. The SEASONALITY= option is applicable only for seasonal forecasting models. By default, the length of the seasonal cycle is one (no seasonality) or the length implied by the INTERVAL= option specified in the ID statement. For example, INTERVAL=MONTH implies that the length of the seasonal cycle is twelve.

SORTNAMES
specifies that the variables specified in the FORECAST statements are processed in sorted order.

STARTSUM=*n*
specifies the starting forecast lead (or horizon) for which to begin summation of the forecasts specified by the LEAD= option. The STARTSUM= value must be less than the LEAD= value. The default is STARTSUM=1; that is, the sum from the one-step ahead forecast (which is the first forecast in the forecast horizon) to the multistep forecast specified by the LEAD= option.

The prediction standard errors of the summation of forecasts take into account the correlation between the multistep forecasts. The section "Forecast Summation" on page 863 describes the STARTSUM= option in more detail.

BY Statement

BY *variables* **;**

A BY statement can be used with PROC ESM to obtain separate dummy variable definitions for groups of observations defined by the BY variables.

When a BY statement appears, the procedure expects the input data set to be sorted in order of the BY variables.

If your input data set is not sorted in ascending order, use one of the following alternatives:

- Sort the data by using the SORT procedure with a similar BY statement.
- Specify the option NOTSORTED or DESCENDING in the BY statement for the ESM procedure. The NOTSORTED option does not mean that the data are unsorted but rather that the data are arranged in groups (according to values of the BY variables) and that these groups are not necessarily in alphabetical or increasing numeric order.
- Create an index on the BY variables by using the DATASETS procedure.

For more information about the BY statement, see *SAS Language Reference: Concepts*. For more information about the DATASETS procedure, see the discussion in the *Base SAS Procedures Guide.*

FORECAST Statement

FORECAST *variable-list / options* **;**

The FORECAST statement lists the numeric variables in the DATA= data set whose accumulated values represent time series to be modeled and forecast. The options specify which forecast model is to be used.

A data set variable can be specified in only one FORECAST statement. Any number of FORECAST statements can be used. The following options can be used with the FORECAST statement.

ACCUMULATE=*option*

specifies how the data set observations are accumulated within each time period for the variables listed in the FORECAST statement. If the ACCUMULATE= option is not specified in the FORECAST statement, accumulation is determined by the ACCUMULATE= option of the ID statement. Use the ACCUMULATE= option with multiple FORECAST statements when you want different accumulation specifications for different variables. See the ID statement ACCUMULATE= option for more details.

ALPHA=*number*

specifies the significance level to use in computing the confidence limits of the forecast. The ALPHA= value must be between 0 and 1. The default is ALPHA=0.05, which produces 95% confidence intervals.

MEDIAN

specifies that the median forecast values are to be estimated. Forecasts can be based on the mean or median. By default, the mean value is provided. If no transformation is applied to the time series by using the TRANSFORM= option, the mean and median forecast values are identical.

MODEL=*model-name*

specifies the forecasting model to be used to forecast the time series. The default is MODEL=SIMPLE, which performs simple exponential smoothing. The following forecasting models are provided:

NONE	no forecast
SIMPLE	simple (single) exponential smoothing. This is the default.
DOUBLE	double (Brown) exponential smoothing
LINEAR	linear (Holt) exponential smoothing
DAMPTREND	damped trend exponential smoothing
ADDSEASONAL|SEASONAL	additive seasonal exponential smoothing
MULTSEASONAL	multiplicative seasonal exponential smoothing
WINTERS	Winters multiplicative method
ADDWINTERS	Winters additive method

When the option MODEL=NONE is specified, the time series is appended with missing values in the OUT= data set. This option is useful when the results stored in the OUT= data set are used in a subsequent analysis where forecasts of the independent variables are needed to forecast the dependent variable.

NBACKCAST=*n*

specifies the number of observations used to initialize the backcast states. The default is the entire series.

REPLACEBACK

specifies that actual values excluded by the BACK= option are replaced with one-step-ahead forecasts in the OUT= data set.

REPLACEMISSING

specifies that embedded missing values are replaced with one-step-ahead forecasts in the OUT= data set.

SETMISSING=*option* | *number*

specifies how missing values (either input or accumulated) are assigned in the accumulated time series for variables listed in the FORECAST statement. If the SETMISSING= option is not specified in the FORECAST statement, missing values are set based on the SETMISSING= option of the ID statement. See the ID statement SETMISSING= option for more details.

TRANSFORM=*option*

specifies the time series transformation to be applied to the input or accumulated time series. The following transformations are provided:

NONE	no transformation. This is the default.
LOG	logarithmic transformation
SQRT	square-root transformation
LOGISTIC	logistic transformation

BOXCOX(*n*) Box-Cox transformation with parameter number where number is between –5 and 5

When the TRANSFORM= option is specified, the time series must be strictly positive. After the time series is transformed, the model parameters are estimated by using the transformed series. The forecasts of the transformed series are then computed, and finally the transformed series forecasts are inverse transformed. The inverse transform produces either mean or median forecasts depending on whether the MEDIAN option is specified. The sections "Transformations" on page 862 and "Inverse Transformations" on page 863 describe this in more detail.

USE=*option*

specifies which forecast values are appended to the actual values in the OUT= and OUTSUM= data sets. The following USE= options are provided:

PREDICT	The predicted values are appended to the actual values. This option is the default.
LOWER	The lower confidence limit values are appended to the actual values.
UPPER	The upper confidence limit values are appended to the actual values.

Thus, the USE= option enables the OUT= and OUTSUM= data sets to be used for worst-case, best-case, average-case, and median-case decisions.

ZEROMISS=*option*

specifies how beginning or ending zero values (either input or accumulated) are interpreted in the accumulated time series for variables listed in the FORECAST statement. If the ZEROMISS= option is not specified in the FORECAST statement, beginning or ending zero values are set to missing values based on the ZEROMISS= option of the ID statement. See the ID statement ZEROMISS= option for more details.

ID Statement

ID *variable INTERVAL= interval* < *options* > **;**

The ID statement names a numeric variable that identifies observations in the input and output data sets. The ID variable's values are assumed to be SAS date or datetime values. In addition, the ID statement specifies the (desired) frequency associated with the time series. The ID statement options also specify how the observations are accumulated and how the time ID values are aligned to form the time series to be forecast. The information specified affects all variables specified in subsequent FORECAST statements. If the ID statement is specified, the INTERVAL= option must be specified. If an ID statement is not specified, the observation number, with respect to the BY group, is used as the time ID. The following options can be used with the ID statement.

ACCUMULATE=*option*

specifies how the data set observations are accumulated within each time period. The frequency (width of each time interval) is specified by the INTERVAL= option. The ID variable contains the time ID values. Each time ID variable value corresponds to a specific time period. The accumulated values form the time series, which is used in subsequent model fitting and forecasting.

The ACCUMULATE= option is particularly useful when there are gaps in the input data or when there are multiple input observations that coincide with a particular time period (for example, transactional

data). The EXPAND procedure offers additional frequency conversions and transformations that can also be useful in creating a time series.

The following options determine how the observations are accumulated within each time period based on the ID variable and the frequency specified by the INTERVAL= option:

NONE	No accumulation occurs; the ID variable values must be equally spaced with respect to the frequency. This is the default option.
TOTAL	Observations are accumulated based on the total sum of their values.
AVERAGE | AVG	Observations are accumulated based on the average of their values.
MINIMUM | MIN	Observations are accumulated based on the minimum of their values.
MEDIAN | MED	Observations are accumulated based on the median of their values.
MAXIMUM | MAX	Observations are accumulated based on the maximum of their values.
N	Observations are accumulated based on the number of nonmissing observations.
NMISS	Observations are accumulated based on the number of missing observations.
NOBS	Observations are accumulated based on the number of observations.
FIRST	Observations are accumulated based on the first of their values.
LAST	Observations are accumulated based on the last of their values.
STDDEV | STD	Observations are accumulated based on the standard deviation of their values.
CSS	Observations are accumulated based on the corrected sum of squares of their values.
USS	Observations are accumulated based on the uncorrected sum of squares of their values.

If the ACCUMULATE= option is specified, the SETMISSING= option is useful for specifying how accumulated missing values are treated. If missing values should be interpreted as zero, then SETMISSING=0 should be used. The section "Accumulation" on page 860 describes accumulation in greater detail.

ALIGN=*option*

controls the alignment of SAS dates used to identify output observations. The ALIGN= option accepts the following values: BEGINNING | BEG | B, MIDDLE | MID | M, and ENDING | END | E. BEGINNING is the default.

END=*date* | *datetime*

specifies a SAS date or datetime literal value that represents the end of the data. If the last time ID variable value is less than the END= value, the series is extended with missing values. If the last time ID variable value is greater than the END= value, the series is truncated. For example, `END='1jan2008'D` specifies that data for time periods after the first of January 2008 not be used. The option `END="&sysdate"D` uses the automatic macro variable SYSDATE to extend or truncate the series to the current date. This option and the START= option can be used to ensure that data associated with each BY group contains the same number of observations.

FORMAT=*format*

specifies the SAS format for the time ID values. If the FORMAT= option is not specified, the default format is implied from the INTERVAL= option.

INTERVAL=*interval*

specifies the frequency of the input time series or for the time series to be accumulated from the input data. For example, if the input data set consists of quarterly observations, then INTERVAL=QTR should be used. If the SEASONALITY= option is not specified, the length of the seasonal cycle is implied by the INTERVAL= option. For example, INTERVAL=QTR implies a seasonal cycle of length 4. If the ACCUMULATE= option is also specified, the INTERVAL= option determines the time periods for the accumulation of observations.

The basic intervals are YEAR, SEMIYEAR, QTR, MONTH, SEMIMONTH, TENDAY, WEEK, WEEKDAY, DAY, HOUR, MINUTE, SECOND. See Chapter 5, "Date Intervals, Formats, and Functions," for more information about the intervals that can be specified.

NOTSORTED

specifies that the time ID values are not in sorted order. The ESM procedure sorts the data with respect to the time ID prior to analysis.

SETMISSING=*option* | *number*

specifies how missing values (either input or accumulated) are assigned in the accumulated time series. If a number is specified, missing values are set to that number. If a missing value on the input data set indicates an unknown value, the SETMISSING= option should not be used. If a missing value indicates no value, SETMISSING=0 should be used. You typically use SETMISSING=0 for transactional data, because no recorded data usually implies no activity. The following options can also be used to determine how missing values are assigned:

MISSING	Missing values are set to missing. This is the default option.
AVERAGE | AVG	Missing values are set to the accumulated average value.
MINIMUM | MIN	Missing values are set to the accumulated minimum value.
MEDIAN | MED	Missing values are set to the accumulated median value.
MAXIMUM | MAX	Missing values are set to the accumulated maximum value.
FIRST	Missing values are set to the accumulated first nonmissing value.
LAST	Missing values are set to the accumulated last nonmissing value.
PREVIOUS | PREV	Missing values are set to the previous accumulated nonmissing value. Missing values at the beginning of the accumulated series remain missing.
NEXT	Missing values are set to the next accumulated nonmissing value. Missing values at the end of the accumulated series remain missing.

If SETMISSING=MISSING is specified, the missing observations are replaced with predicted values computed from the exponential smoothing model.

START=*date* | *datetime*

specifies a SAS date or datetime literal value that represents the beginning of the data. If the first time ID variable value is greater than the START= value, the series is prefixed with missing values. If the first time ID variable value is less than the START= value, the series is truncated. This option and the

END= option can be used to ensure that data associated with each BY group contains the same number of observations.

ZEROMISS=*option*
specifies how beginning and/or ending zero values (either input or accumulated) are interpreted in the accumulated time series. The following values can be specified for the ZEROMISS= option:

NONE	Beginning and/or ending zeros are unchanged. This is the default.
LEFT	Beginning zeros are set to missing.
RIGHT	Ending zeros are set to missing.
BOTH	Both beginning and ending zeros are set to missing.

If the accumulated series is all missing and/or zero the series is not changed.

Details: ESM Procedure

The ESM procedure can be used to forecast time series data as well as transactional data. If the data is transactional, then the procedure must first accumulate the data into a time series before it can be forecast. The procedure uses the following sequential steps to produce forecasts, with the options that control the step listed to the right:

Table 15.2 ESM Processing Steps and Control Options

Step	Operation	Option	Statement
1	accumulation	ACCUMULATE=	ID
2	missing value interpretation	SETMISSING=	ID, FORECAST
3	transformations	TRANSFORM=	FORECAST
4	parameter estimation	MODEL=	FORECAST
5	forecasting	MODEL=, LEAD=	FORECAST, PROC ESM
6	inverse transformation	TRANSFORM, MEDIAN	FORECAST
7	summation of forecasts	LEAD=, STARTSUM=	PROC ESM

Each of the steps shown in Table 15.2 is described in the following sections.

Accumulation

If the ACCUMULATE= option is specified in the ID statement, data set observations are accumulated within each time period. The frequency (width of each time interval) is specified by the INTERVAL= option, and the ID variable contains the time ID values. Each time ID value corresponds to a specific time period. Accumulation is particularly useful when the input data set contains transactional data, whose observations are not spaced with respect to any particular time interval. The accumulated values form the time series that is used in subsequent analyses by the ESM procedure.

For example, suppose a data set contains the following observations:

```
19MAR1999    10
19MAR1999    30
11MAY1999    50
12MAY1999    20
23MAY1999    20
```

If the INTERVAL=MONTH option is specified on the ID statement, all of the preceding observations fall within three time periods: March 1999, April 1999, and May 1999. The observations are accumulated within each time period as follows.

If the ACCUMULATE=NONE option is specified, an error is generated because the ID variable values are not equally spaced with respect to the specified frequency (MONTH).

If the ACCUMULATE=TOTAL option is specified, the resulting time series is:

```
O1MAR1999    40
O1APR1999    .
O1MAY1999    90
```

If the ACCUMULATE=AVERAGE option is specified, the resulting time series is:

```
O1MAR1999    20
O1APR1999    .
O1MAY1999    30
```

If the ACCUMULATE=MINIMUM option is specified, the resulting time series is:

```
O1MAR1999    10
O1APR1999    .
O1MAY1999    20
```

If the ACCUMULATE=MEDIAN option is specified, the resulting time series is:

```
O1MAR1999    20
01APR1999    .
O1MAY1999    20
```

If the ACCUMULATE=MAXIMUM option is specified, the resulting time series is:

```
O1MAR1999    30
O1APR1999    .
O1MAY1999    50
```

If the ACCUMULATE=FIRST option is specified, the resulting time series is:

```
O1MAR1999    10
O1APR1999    .
O1MAY1999    50
```

If the ACCUMULATE=LAST option is specified, the resulting time series is:

```
O1MAR1999    30
O1APR1999    .
O1MAY1999    20
```

If the ACCUMULATE=STDDEV option is specified, the resulting time series is:

```
O1MAR1999    14.14
O1APR1999    .
O1MAY1999    17.32
```

As can be seen from the preceding examples, even though the data set observations contained no missing values, the accumulated time series can have missing values.

Missing Value Interpretation

Sometimes missing values should be interpreted as truly unknown values and retained as missing values in the data set. The forecasting models used by the ESM procedure can effectively handle missing values (see the section "Missing Value Modeling Issues" on page 863). However, sometimes missing values are known, such as when missing values are created from accumulation and represent no observed values for the variable. In this case, the value for the period should be interpreted as zero (no values), and the SETMISSING=0 option should be used to cause PROC ESM to recode missing values as zero. In other cases, missing values should be interpreted as global values, such as minimum or maximum values of the accumulated series. The accumulated and missing-value-recoded time series is used in subsequent analyses in PROC ESM.

Transformations

If the TRANSFORM= option is specified in the FORECAST statement, the time series is transformed prior to model parameter estimation and forecasting. Only strictly positive series can be transformed. An error is generated when the TRANSFORM= option is used with a nonpositive series. (See Chapter 61, "Forecasting Process Details," for more details about forecasting transformed time series.)

Parameter Estimation

All the parameters (smoothing weights) associated with the exponential smoothing model used to forecast the time series (as specified by the MODEL= option) are optimized based on the data, with the default parameter restrictions imposed. If the TRANSFORM= option is specified, the transformed time series data are used to estimate the model parameters.

The techniques used in the ESM procedure are identical to those used for exponential smoothing models in the Time Series Forecasting System of SAS/ETS software. See Chapter 53, "Overview of the Time Series Forecasting System," for more information.

Missing Value Modeling Issues

The treatment of missing values varies with the forecasting model. Missing values after the start of the series are replaced with one-step-ahead predicted values, and the predicted values are used in the smoothing equations.

The treatment of missing values can also be specified with the SETMISSING= option, which changes the missing values prior to modeling.

NOTE: Even if all of the observed data are nonmissing, the ACCUMULATE= option can create missing values in the accumulated series (when the data contain no observations for some of the time periods specified by the INTERVAL= option).

Forecasting

Once the model parameters are estimated, one-step-ahead forecasts are generated for the full range of the accumulated and optionally transformed time series data, and multistep forecasts are generated from the end of the time series to the future time period specified by the LEAD= option. If there are missing values at the end of the time series, the forecast horizon will be greater than that specified by the LEAD= option.

Inverse Transformations

If the TRANSFORM= option is specified in the FORECAST statement, the forecasts of the transformed time series are inverse transformed. By default, forecasts of the mean (expected value) are generated. If the MEDIAN option is specified, median forecasts are generated. (See Chapter 61, "Forecasting Process Details," for more details about forecasting transformed time series.)

Statistics of Fit

The statistics of fit are computed by comparing the time series data (after accumulation and missing value recoding, if specified) with the generated forecasts. If the TRANSFORM= option is specified, the statistics of fit are based on the inverse transformed forecasts. (See Chapter 61, "Forecasting Process Details," for more details about statistics of fit for forecasting models.)

Forecast Summation

The multistep forecasts generated by the preceding steps can optionally be summed from the STARTSUM= value to the LEAD= value. For example, if the options STARTSUM=4 and LEAD=6 are specified on the PROC ESM statement, the four-step through six-step ahead forecasts are summed.

The forecasts are simply summed; however, the prediction error variance of this sum is computed by taking into account the correlation between the individual predictions. (These variance-related computations are performed only when no transformation is specified; that is, when TRANSFORM=NONE.) The upper and

lower confidence limits for the sum of the predictions is then computed based on the prediction error variance of the sum.

The forecast summation is particularly useful when it is desirable to model in one frequency but the forecast of interest is another frequency. For example, if a time series has a monthly frequency (INTERVAL=MONTH) and you want a forecast for the third and fourth future months, a forecast summation for the third and fourth month can be obtained by specifying STARTSUM=3 and LEAD=4.

Data Set Output

The ESM procedure can create the OUT=, OUTEST=, OUTFOR=, OUTSTAT=, and OUTSUM= data sets. These data sets contain the variables listed in the BY statement and statistics related to the variables listing in the FORECAST statement. In general, if a forecasting step related to an output data set fails, the values of this step are not recorded or are set to missing in the related output data set and appropriate error and/or warning messages are recorded in the log.

OUT= Data Set

The OUT= data set contains the variables specified in the BY, ID, and FORECAST statements. If the ID statement is specified, the ID variable values are aligned and extended based on the ALIGN= and INTERVAL= options. The values of the variables specified in the FORECAST statements are accumulated based on the ACCUMULATE= option, and missing values are interpreted based on the SETMISSING= option. If the REPLACEMISSING option is specified, embedded missing values are replaced by the one-step-ahead predicted values.

These FORECAST variables are then extrapolated based on the forecasts from the fitted models, or extended with missing values when the MODEL=NONE option is specified. If USE=LOWER is specified, the variable is extrapolated with the lower confidence limits; if USE=UPPER, the variable is extrapolated using the upper confidence limits; otherwise, the variable values are extrapolated with the predicted values. If the TRANSFORM= option is specified, the predicted values contain either mean or median forecasts depending on whether or not the MEDIAN option is specified.

If any of the forecasting steps fail for a particular variable, the variable is extended by missing values.

OUTEST= Data Set

The OUTEST= data set contains the variables specified in the BY statement as well as the variables listed below. For variables listed in FORECAST statements where the option MODEL=NONE is specified, no observations are recorded in the OUTEST= data set. For variables listed in FORECAST statements where the option MODEL=NONE is not specified, the following variables in the OUTEST= data set contain observations related to the parameter estimation step:

NAME	variable name
MODEL	forecasting model
TRANSFORM	transformation
PARM	parameter name
EST	parameter estimate

STDERR	standard errors
TVALUE	*t* values
PVALUE	probability values

If the parameter estimation step fails for a particular variable, no observations are output to the OUTEST= data set for that variable.

OUTFOR= Data Set

The OUTFOR= data set contains the variables specified in the BY statement as well as the variables listed below. For variables listed in FORECAST statements where the option MODEL=NONE is specified, no observations are recorded in the OUTFOR= data set for these variables. For variables listed in FORECAST statements where the option MODEL=NONE is not specified, the following variables in the OUTFOR= data set contain observations related to the forecasting step:

NAME	variable name
TIMEID	time ID values
ACTUAL	actual values
PREDICT	predicted values
STD	prediction standard errors
LOWER	lower confidence limits
UPPER	upper confidence limits
ERROR	prediction errors

If the forecasting step fails for a particular variable, no observations are recorded in the OUTFOR= data set for that variable. If the TRANSFORM= option is specified, the values in the preceding variables are the inverse transform forecasts. If the MEDIAN option is specified, the median forecasts are stored; otherwise, the mean forecasts are stored.

OUTPROCINFO= Data Set

The OUTPROCINFO= data set contains information about the run of the ESM procedure. The following variables are present:

SOURCE	set to the name of the procedure, in this case ESM
NAME	name of an item being reported; can be the number of errors, notes, or warnings, number of forecasts requested, and so on
LABEL	descriptive label for the item in _NAME_
STAGE	set to the current stage of the procedure, for ESM this is set to ALL
VALUE	value of the item specified in _NAME_

OUTSTAT= Data Set

The OUTSTAT= data set contains the variables specified in the BY statement as well as the variables listed below. For variables listed in FORECAST statements where the option MODEL=NONE is specified, no observations are recorded for these variables in the OUTSTAT= data set. For variables listed in FORECAST statements where the option MODEL=NONE is not specified, the following variables in the OUTSTAT= data set contain observations related to the statistics of fit:

NAME	variable name
REGION	the region in which the statistics are calculated. Statistics calculated in the fit region are indicated by FIT. Statistics calculated in the forecast region, which happens only if the BACK= option is greater than zero, are indicated by FORECAST.
DFE	degrees of freedom error
N	number of observations
NOBS	number of observations used
NMISSA	number of missing actuals
NMISSP	number of missing predicted values
NPARMS	number of parameters
TSS	total sum of squares
SST	corrected total sum of squares
SSE	sum of square error
MSE	mean square error
UMSE	unbiased mean square error
RMSE	root mean square error
URMSE	unbiased root mean square error
MAPE	mean absolute percent error
MAE	mean absolute error
MASE	mean absolute scaled error
RSQUARE	R square
ADJRSQ	adjusted R square
AADJRSQ	Amemiya's adjusted R square
RWRSQ	random walk R square
AIC	Akaike information criterion
AICC	finite sample corrected AIC
SBC	Schwarz Bayesian information criterion
APC	Amemiya's prediction criterion
MAXERR	maximum error
MINERR	minimum error

MINPE	minimum percent error
MAXPE	maximum percent error
ME	mean error
MPE	mean percent error
MDAPE	median absolute percent error
GMAPE	geometric mean absolute percent error
MINPPE	minimum predictive percent error
MAXPPE	maximum predictive percent error
MSPPE	mean predictive percent error
MAPPE	symmetric mean absolute predictive percent error
MDAPPE	median absolute predictive percent error
GMAPPE	geometric mean absolute predictive percent error
MINSPE	minimum symmetric percent error
MAXSPE	maximum symmetric percent error
MSPE	mean symmetric percent error
SMAPE	symmetric mean absolute percent error
MDASPE	median absolute symmetric percent error
GMASPE	geometric mean absolute symmetric percent error
MINRE	minimum relative error
MAXRE	maximum relative error
MRE	mean relative error
MRAE	mean relative absolute error
MDRAE	median relative absolute error
GMRAE	geometric mean relative absolute error
MINAPES	minimum absolute error percent of standard deviation
MAXAPES	maximum absolute error percent of standard deviation
MAPES	mean absolute error percent of standard deviation
MDAPES	median absolute error percent of standard deviation
GMAPES	geometric mean absolute error percent of standard deviation

If the statistics of fit cannot be computed for a particular variable, no observations are recorded in the OUTSTAT= data set for that variable. If the TRANSFORM= option is specified, the values in the preceding variables are computed based on the inverse transform forecasts. If the MEDIAN option is specified, the median forecasts are the basis; otherwise, the mean forecasts are the basis.

See Chapter 61, "Forecasting Process Details," for more information about the calculation of forecasting statistics of fit.

OUTSUM= Data Set

The OUTSUM= data set contains the variables specified in the BY statement as well as the variables listed below. The OUTSUM= data set records the summary statistics for each variable specified in a FORECAST statement. For variables listed in FORECAST statements where the option MODEL=NONE is specified, the values related to forecasts are set to missing for those variables in the OUTSUM= data set. For variables listed in FORECAST statements where the option MODEL=NONE is not specified, the forecast values are set based on the USE= option.

The following variables related to summary statistics are based on the ACCUMULATE= and SETMISSING= options:

NAME	variable name
STATUS	forecasting status. Nonzero values imply that no forecast was generated for the series.
NOBS	number of observations
N	number of nonmissing observations
NMISS	number of missing observations
MIN	minimum value
MAX	maximum value
MEAN	mean value
STDDEV	standard deviation

The following variables related to forecast summation are based on the LEAD= and STARTSUM= options:

PREDICT	forecast summation predicted values
STD	forecast summation prediction standard errors
LOWER	forecast summation lower confidence limits
UPPER	forecast summation upper confidence limits

Variance-related computations are computed only when no transformation is specified (TRANSFORM=NONE).

The following variables related to multistep forecast are based on the LEAD= and USE= options:

LEAD*n* multistep forecast (*n* ranges from one to the value of the LEAD= option). If USE=LOWER, this variable contains the lower confidence limits; if USE=UPPER, this variable contains the upper confidence limits; otherwise, this variable contains the predicted values.

If the forecast step fails for a particular variable, the variables that are related to forecasting are set to missing for that variable. The OUTSUM= data set contains both a summary of the (accumulated) time series and optionally its forecasts for all series.

Printed Output

The ESM procedure optionally produces printed output by using the Output Delivery System (ODS). By default, the procedure produces no printed output. All output is controlled by the PRINT= and PRINTDETAILS options in the PROC ESM statement. In general, if a forecasting step that is related to printed output

fails, the values of this step are not printed and appropriate error or warning messages are recorded in the log. The printed output is similar to the output data sets.

The printed output produced by the PRINT= option values is described as follows:

SUMMARY	prints the summary statistics and forecast summaries similar to the OUTSUM= data set.
ESTIMATES	prints the parameter estimates similar to the OUTEST= data set.
FORECASTS	prints the forecasts similar to the OUTFOR= data set.
PERFORMANCE	prints the performance statistics.
PERFORMANCESUMMARY	prints the performance summary for each BY group.
PERFORMANCEOVERALL	prints the performance summary for all BY groups.
STATES	prints the backcast, initial, and final smoothed states.
STATISTICS	prints the statistics of fit similar to the OUTSTAT= data set.

The PRINTDETAILS option is the opposite of the NOOUTALL option. Specifically, if PRINT=FORECASTS and the PRINTDETAILS options are specified in the PROC ESM statement, the one-step-ahead forecasts through the range of the data are printed in addition to the information related to a specific forecasting model, such as the smoothing states. If the PRINTDETAILS option is not specified, only the multistep forecasts are printed.

ODS Table Names

Table 15.3 relates the PRINT= options to ODS tables:

Table 15.3 ODS Tables Produced in PROC ESM

ODS Table Name	Description	PRINT= Option
DescStats	descriptive statistics	SUMMARY
ForecastSummary	forecast summary	SUMMARY
ForecastSummation	forecast summation	SUMMARY
ParameterEstimates	parameter estimates	ESTIMATES
Forecasts	forecasts	FORECASTS
Performance	performance statistics	PERFORMANCE
PerformanceSummary	performance summary	PERFORMANCESUMMARY
PerformanceOverall	performance overall	PERFORMANCEOVERALL
SmoothedStates	smoothed states	STATES
FitStatistics	evaluation statistics of fit	STATISTICS
PerformanceStatistics	performance (out-of-sample) statistics of fit	STATISTICS

The ODS table "ForecastSummary" is related to all time series within a BY group. The other tables are related to a single series within a BY group.

ODS Graphics

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*).

Before you create graphs, ODS Graphics must be enabled (for example, with the ODS GRAPHICS ON statement). For more information about enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" in that chapter.

The overall appearance of graphs is controlled by ODS styles. Styles and other aspects of using ODS Graphics are discussed in the section "A Primer on ODS Statistical Graphics" in that chapter.

This section describes the use of ODS for creating graphics with the ESM procedure. To request these graphs you must specify the PLOT= option in the PROC ESM statement.

ODS Graph Names

PROC ESM assigns a name to each graph it creates using ODS. You can use these names to reference the graphs when using ODS. The names are listed in Table 15.4.

Table 15.4 ODS Graphics Produced by the PLOT= Option in PROC ESM

ODS Graph Name	Plot Description	PLOT= Option
ErrorACFNORMPlot	standardized autocorrelation of prediction errors	ACF
ErrorACFPlot	autocorrelation of prediction errors	ACF
ErrorHistogram	prediction error histogram	ERRORS
ErrorCorrelationPlots	prediction error plot panel	CORR
ErrorIACFNORMPlot	standardized inverse autocorrelation of prediction errors	IACF
ErrorIACFPlot	inverse autocorrelation of prediction errors	IACF
ErrorPACFNORMPlot	standardized partial autocorrelation of prediction errors	PACF
ErrorPACFPlot	partial autocorrelation of prediction errors	PACF
ErrorPeriodogramPlot	periodogram of prediction errors	PERIODOGRAM
ErrorPlot	plot of prediction errors	ERRORS
ErrorSpectralDensityPlot	combined periodogram and spectral density estimate plot	SPECTRUM
ErrorWhiteNoiseLogProbPlot	white noise log probability plot of prediction errors	WN
ErrorWhiteNoiseProbPlot	white noise probability plot of prediction errors	WN

Table 15.4 (continued)

ODS Graph Name	Plot Description	PLOT= Option
ForecastsOnlyPlot	forecasts only plot	FORECASTSONLY
ForecastsPlot	forecasts plot	FORECASTS
LevelStatePlot	smoothed level state plot	LEVELS
ModelForecastsPlot	model and forecasts plot	MODELFORECASTS
ModelPlot	model plot	MODELS
SeasonStatePlot	smoothed season state plot	SEASONS
TrendStatePlot	smoothed trend state plot	TRENDS

Examples: ESM Procedure

Example 15.1: Forecasting of Time Series Data

This example uses retail sales data to illustrate how the ESM procedure can be used to forecast time series data.

The following DATA step creates a data set from data recorded monthly at numerous points of sale. The data set, SALES, contains a variable DATE that represents time and a variable for each sales item. Each value of the DATE variable is recorded in ascending order, and the values of each of the other variables represent a single time series:

```
data sales;
   format date date9.;
   input date : date9. shoes socks laces dresses
                       coats shirts ties belts hats blouses;
datalines;
01JAN1994  3557   3718   6368.80    575    987   10.8200   15.0000   102.600   12410   15013

   ... more lines ...
```

The following ESM procedure statements forecast each of the monthly time series:

```
proc esm data=sales out=nextyear;
   id date interval=month;
   forecast _numeric_;
run;
```

The preceding statements generate forecasts for every numeric variable in the input data set SALES for the next twelve months and store these forecasts in the output data set NEXTYEAR.

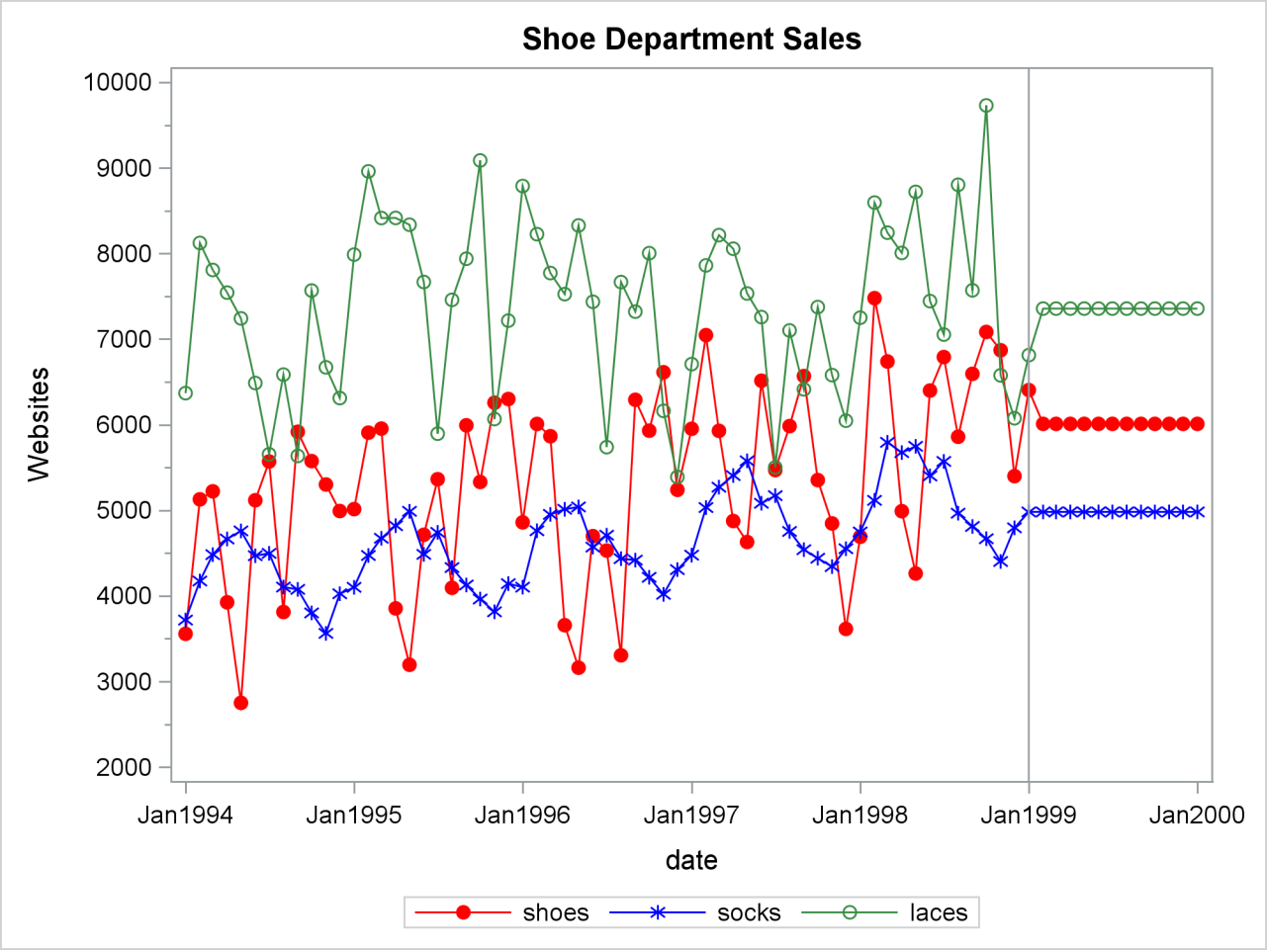
The following statements plot the forecasts:

```
title1 "Shoe Department Sales";
proc sgplot data=nextyear;
   series x=date y=shoes / markers
                           markerattrs=(symbol=circlefilled color=red)
                           lineattrs=(color=red);
```

```
    series x=date y=socks / markers
                            markerattrs=(symbol=asterisk color=blue)
                            lineattrs=(color=blue);
    series x=date y=laces / markers
                            markerattrs=(symbol=circle color=styg)
                           lineattrs=(color=styg);
    refline '01JAN1999'd / axis=x;
    xaxis values=('01JAN1994'd to '01DEC2000'd by year);
    yaxis values=(2000 to 10000 by 1000) minor label='Websites';
run;
```

The plots are shown in Output 15.1.1. The historical data is shown to the left of the reference line and the forecasts for the next twelve monthly periods is shown to the right.

Output 15.1.1 Retail Sales Forecast Plots



The default simple exponential smoothing model is used because the MODEL= option is omitted on the FORECAST statement. Note that for simple exponential smoothing the forecasts are constant.

The following ESM procedure statements are identical to the preceding statements except that the PRINT=FORECASTS option is specified:

```
proc esm data=sales out=nextyear print=forecasts;
   id date interval=month;
   forecast _numeric_;
run;
```

In addition to forecasting each of the monthly time series, the preceding statements print the forecasts by using the Output Delivery System (ODS); the forecasts are partially shown in Output 15.1.2. This output shows the predictions, prediction standard errors, and the upper and lower confidence limits for the next twelve monthly periods.

Output 15.1.2 Forecast Tables

Shoe Department Sales

The ESM Procedure

Forecasts for Variable shoes

Obs	Time	Forecasts	Standard Error	95% Confidence Limits	
62	FEB1999	6009.1986	1069.4059	3913.2016	8105.1956
63	MAR1999	6009.1986	1075.7846	3900.6996	8117.6976
64	APR1999	6009.1986	1082.1257	3888.2713	8130.1259
65	MAY1999	6009.1986	1088.4298	3875.9154	8142.4818
66	JUN1999	6009.1986	1094.6976	3863.6306	8154.7666
67	JUL1999	6009.1986	1100.9298	3851.4158	8166.9814
68	AUG1999	6009.1986	1107.1269	3839.2698	8179.1274
69	SEP1999	6009.1986	1113.2895	3827.1914	8191.2058
70	OCT1999	6009.1986	1119.4181	3815.1794	8203.2178
71	NOV1999	6009.1986	1125.5134	3803.2329	8215.1643
72	DEC1999	6009.1986	1131.5758	3791.3507	8227.0465
73	JAN2000	6009.1986	1137.6060	3779.5318	8238.8654

Example 15.2: Forecasting of Transactional Data

This example illustrates how the ESM procedure can be used to forecast transactional data.

The following DATA step creates a data set from data recorded at several Internet Web sites. The data set WEBSITES contains a variable TIME that represents time and the variables ENGINE, BOATS, CARS, and PLANES that represent Internet Web site data. Each value of the TIME variable is recorded in ascending order, and the values of each of the other variables represent a transactional data series.

The following ESM procedure statements forecast each of the transactional data series:

```
proc esm data=websites out=nextweek lead=7;
   id time interval=dtday accumulate=total;
   forecast boats cars planes;
run;
```

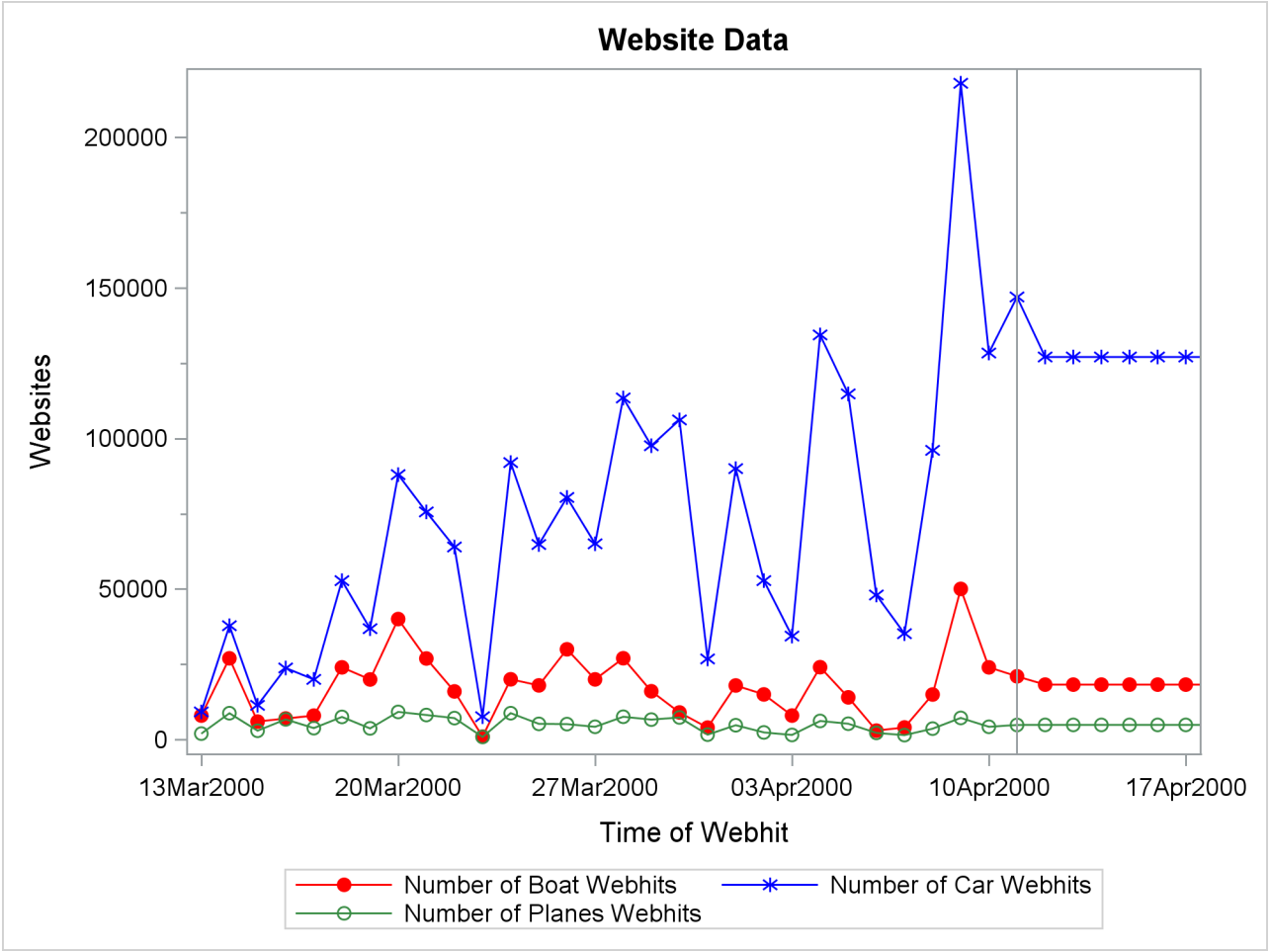
The preceding statements accumulate the data into a daily time series, generate forecasts for the BOATS, CARS, and PLANES variables in the input data set WEBSITES for the next week, and the forecasts are stored in the OUT= data set NEXTWEEK.

The following statements plot the forecasts related to the Internet data:

```
title1 "Website Data";
proc sgplot data=nextweek;
   series x=time y=boats / markers
                           markerattrs=(symbol=circlefilled color=red)
                           lineattrs=(color=red);
   series x=time y=cars / markers
                           markerattrs=(symbol=asterisk color=blue)
                           lineattrs=(color=blue);
   series x=time y=planes / markers
                           markerattrs=(symbol=circle color=styg)
                           lineattrs=(color=styg);
   refline '11APR2000:00:00:00'dt / axis=x;
   xaxis values=('13MAR2000:00:00:00'dt to '18APR2000:00:00:00'dt by dtweek);
   yaxis label='Websites' minor;
run;
```

The plots are shown in Output 15.2.1. The historical data is shown to the left of the reference line and the forecasts for the next seven days are shown to the right.

Output 15.2.1 Internet Data Forecast Plots



Example 15.3: Specifying the Forecasting Model

This example illustrates how the ESM procedure can be used to specify different models for different series. Internet data from the previous example are used for this illustration.

This example, forecasts the BOATS variable by using the seasonal exponential smoothing model (SEASONAL), the CARS variable by using the Winters (multiplicative) model (MULTWINTERS), and the PLANES variable by using the Log Winters (additive) model. The following ESM procedure statements forecast each of the transactional data series based on these requirements:

```
proc esm data=websites out=nextweek lead=7;
   id time interval=dtday accumulate=total;
   forecast boats   / model=seasonal;
   forecast cars    / model=multwinters;
   forecast planes  / model=addwinters transform=log;
run;
```

Example 15.4: Extending the Independent Variables for Multivariate Forecasts

In the previous example, the ESM procedure was used to forecast several transactional series variables by using univariate models. This example illustrates how the ESM procedure can be used to extend the independent variables that are associated with a multiple regression forecasting problem.

This example accumulates and forecasts the BOATS, CARS, and PLANES variables that were illustrated in the previous example. In addition, this example accumulates the ENGINES variable to form a time series that is then extended with missing values within the forecast horizon with the specification of MODEL=NONE.

```
proc esm data=websites out=nextweek lead=7;
   id time interval=dtday accumulate=total;
   forecast engines / model=none;
   forecast boats   / model=seasonal;
   forecast cars    / model=multwinters;
   forecast planes  / model=addwinters transform=log;
run;
```

The following AUTOREG procedure statements are used to forecast the ENGINES variable by regressing on the independent variables (BOATS, CARS, and PLANES).

```
proc autoreg data= nextweek;
   model engines = boats cars planes / noprint;
   output out=enginehits p=predicted;
run;
```

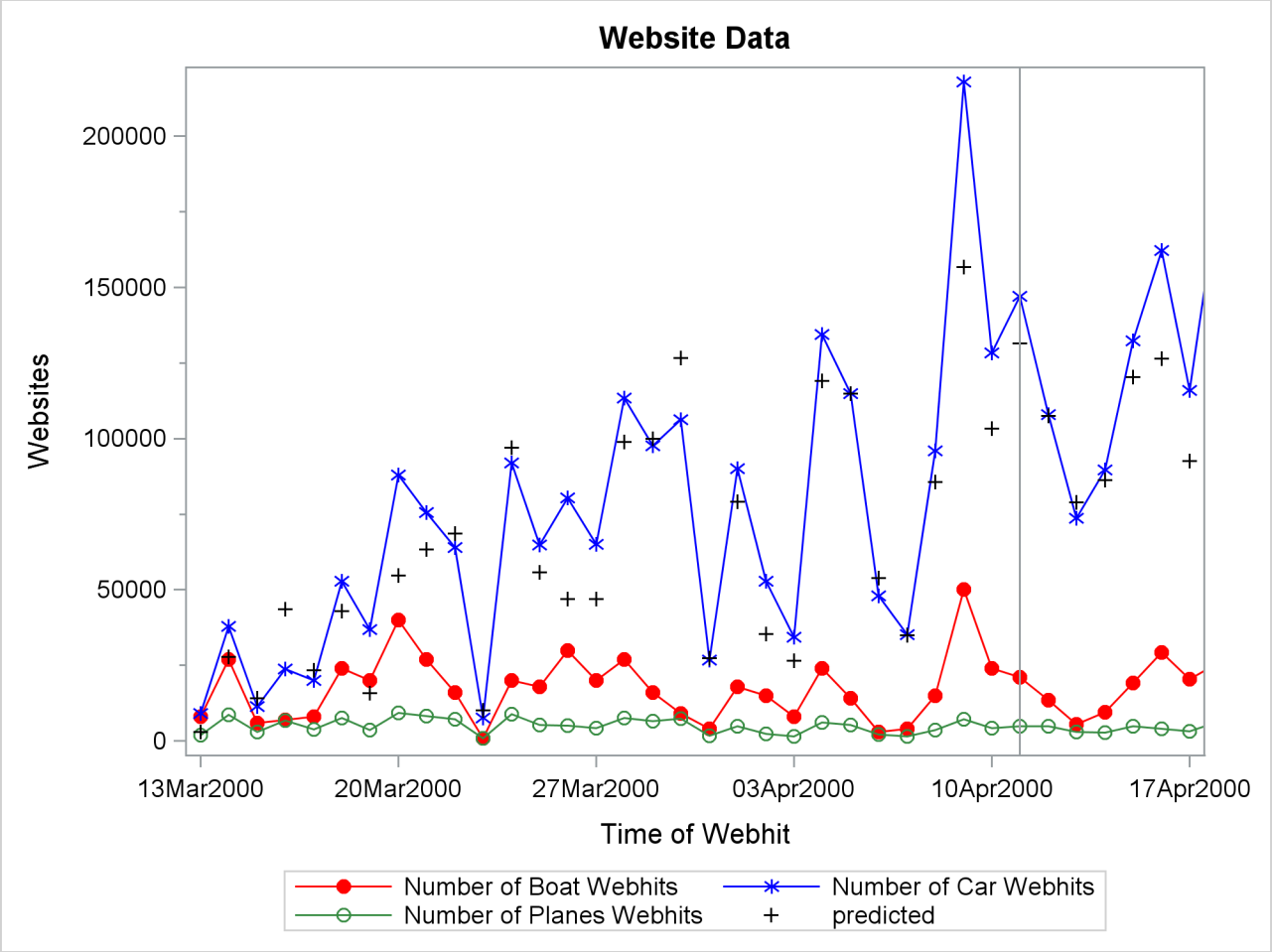
The NEXTWEEK data set created by PROC ESM is used as an input data set to PROC AUTOREG. The output data set from PROC AUTOREG contains the forecast of the variable ENGINES based on the regression model with the variables BOATS, CARS, and PLANES as regressors. See Chapter 9, "The AUTOREG Procedure," for details about autoregression models.

The following statements plot the forecasts related to the ENGINES variable:

```
title1 "Website Data";
proc sgplot data=enginehits;
   series x=time y=boats / markers
                           markerattrs=(symbol=circlefilled color=red)
                           lineattrs=(color=red);
   series x=time y=cars / markers
                           markerattrs=(symbol=asterisk color=blue)
                           lineattrs=(color=blue);
   series x=time y=planes / markers
                           markerattrs=(symbol=circle color=styg)
                           lineattrs=(color=styg);
   scatter x=time y=predicted / markerattrs=(symbol=plus color=black);
   refline '11APR2000:00:00:00'dt / axis=x;
   xaxis values=('13MAR2000:00:00:00'dt to '18APR2000:00:00:00'dt by dtweek);
   yaxis label='Websites' minor;
run;
```

The plots are shown in Output 15.4.1. The historical data is shown to the left of the reference line and the forecasts for the next seven daily periods are shown to the right.

Output 15.4.1 Internet Data Forecast Plots



Example 15.5: Illustration of ODS Graphics

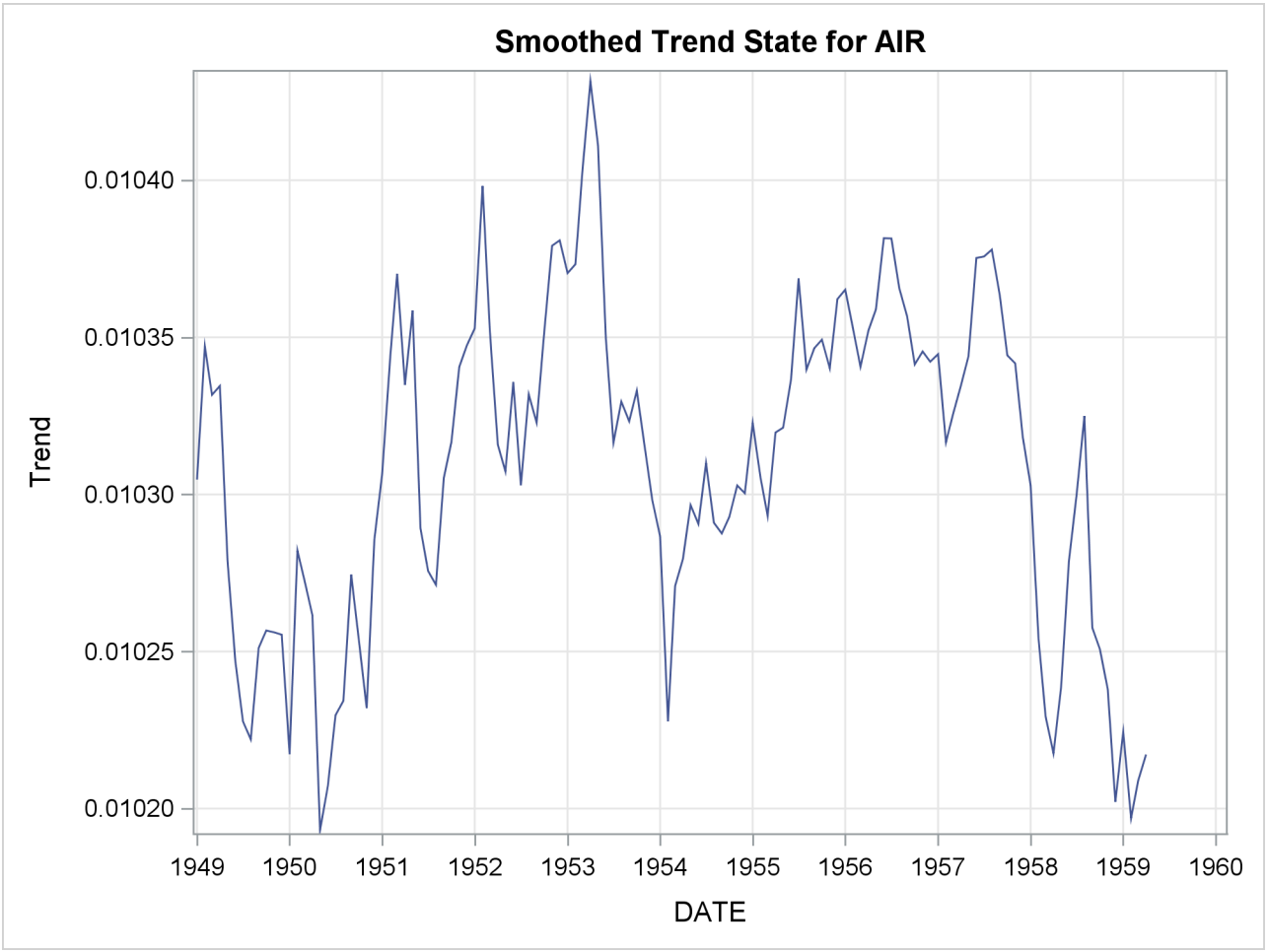
This example illustrates the use of ODS graphics in the ESM procedure and uses the SASHELP.AIR data set to forecast the time series of international airline travel.

The graphical displays are requested by specifying the PLOT= option in the PROC ESM statement. In this case, all plots are requested. Output 15.5.1 through Output 15.5.5 show a selection of the plots created.

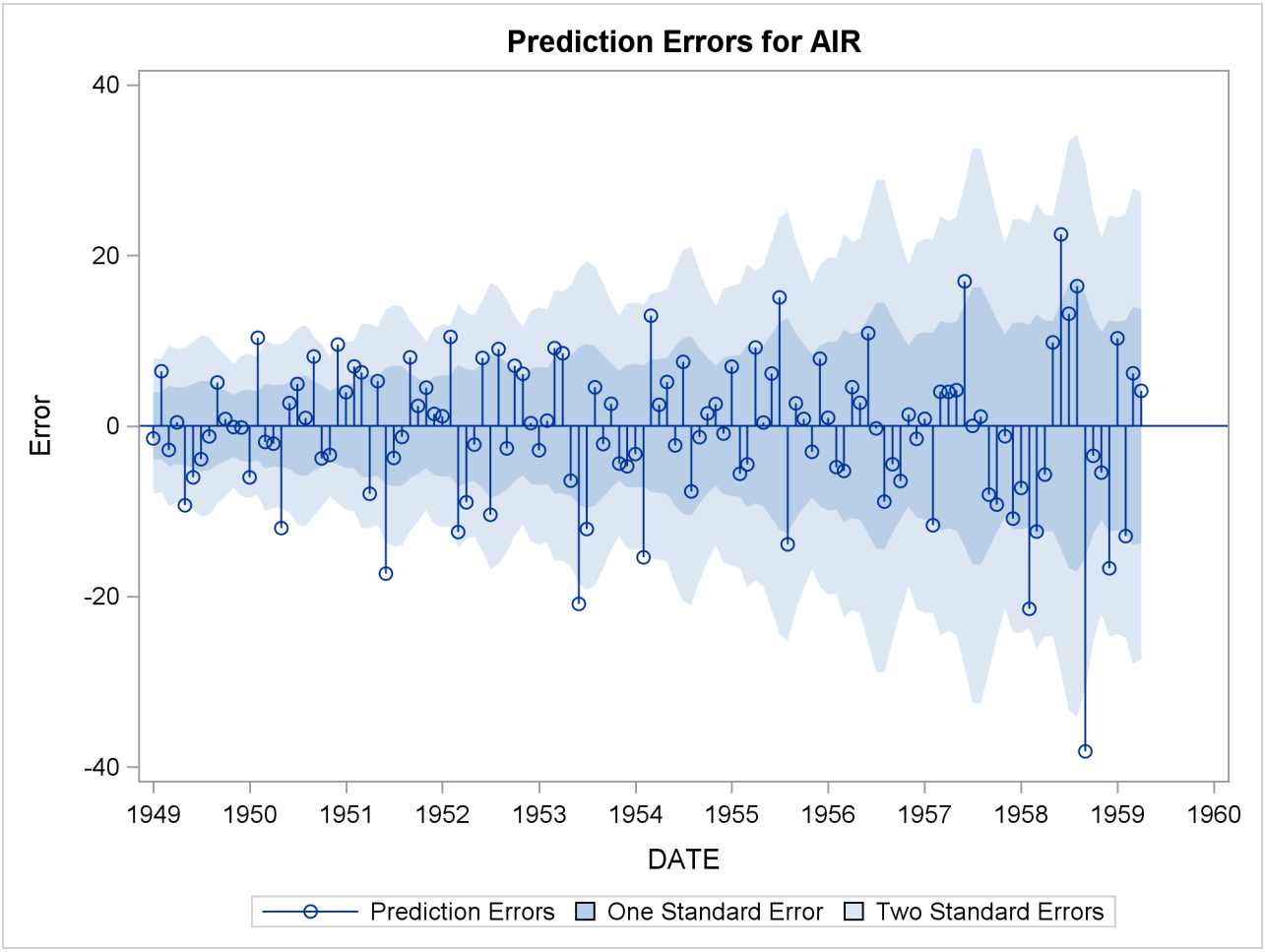
For information about the graphics available in the ESM procedure, see the section "ODS Graphics" on page 870.

```
proc esm data=sashelp.air out=_null_
         lead=20
         back=20
         print=all
         plot=all;
   id date interval=month;
   forecast air / model=addwinters transform=log;
run;
```

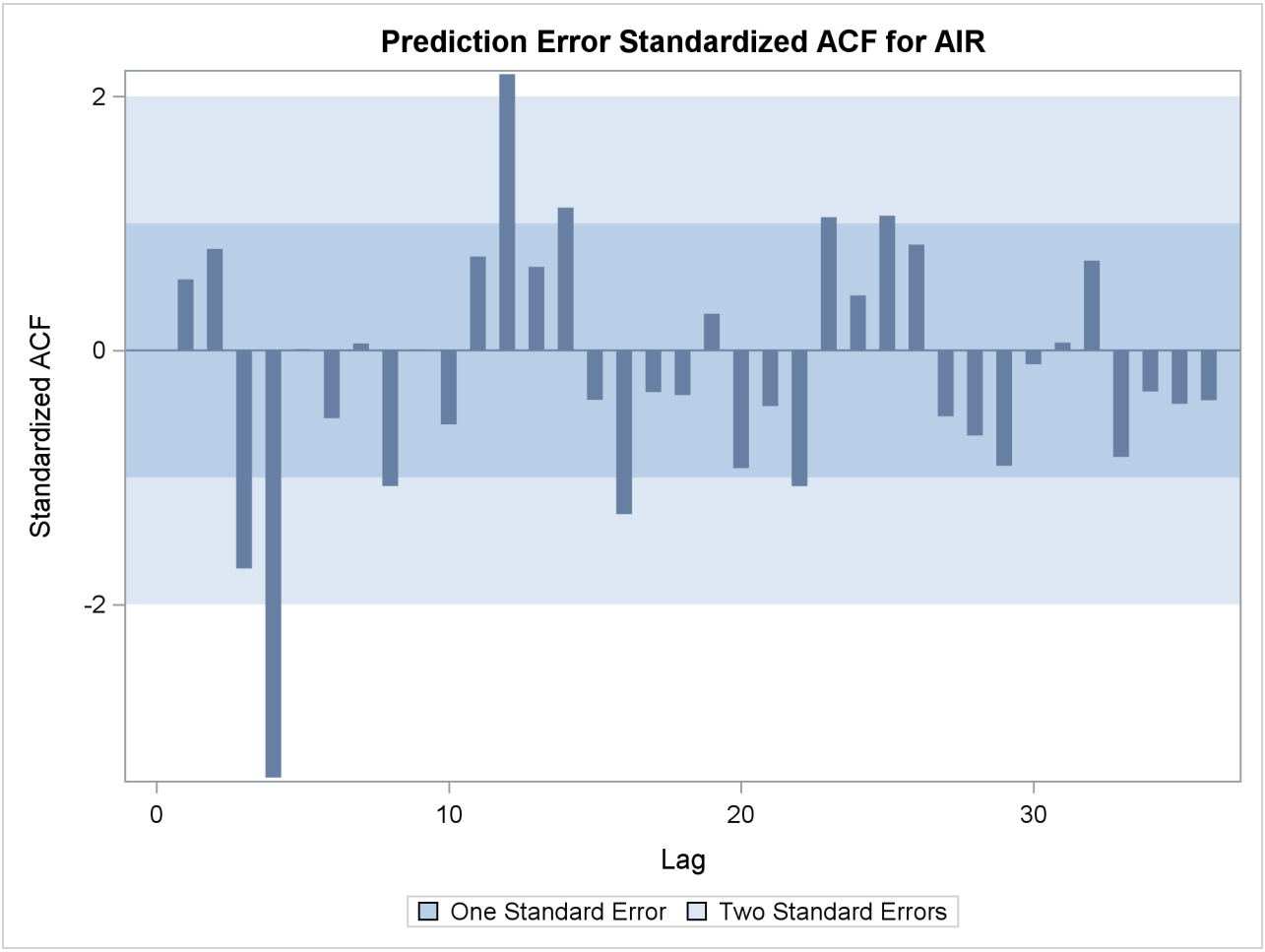
Output 15.5.1 Smoothed Trend Plot



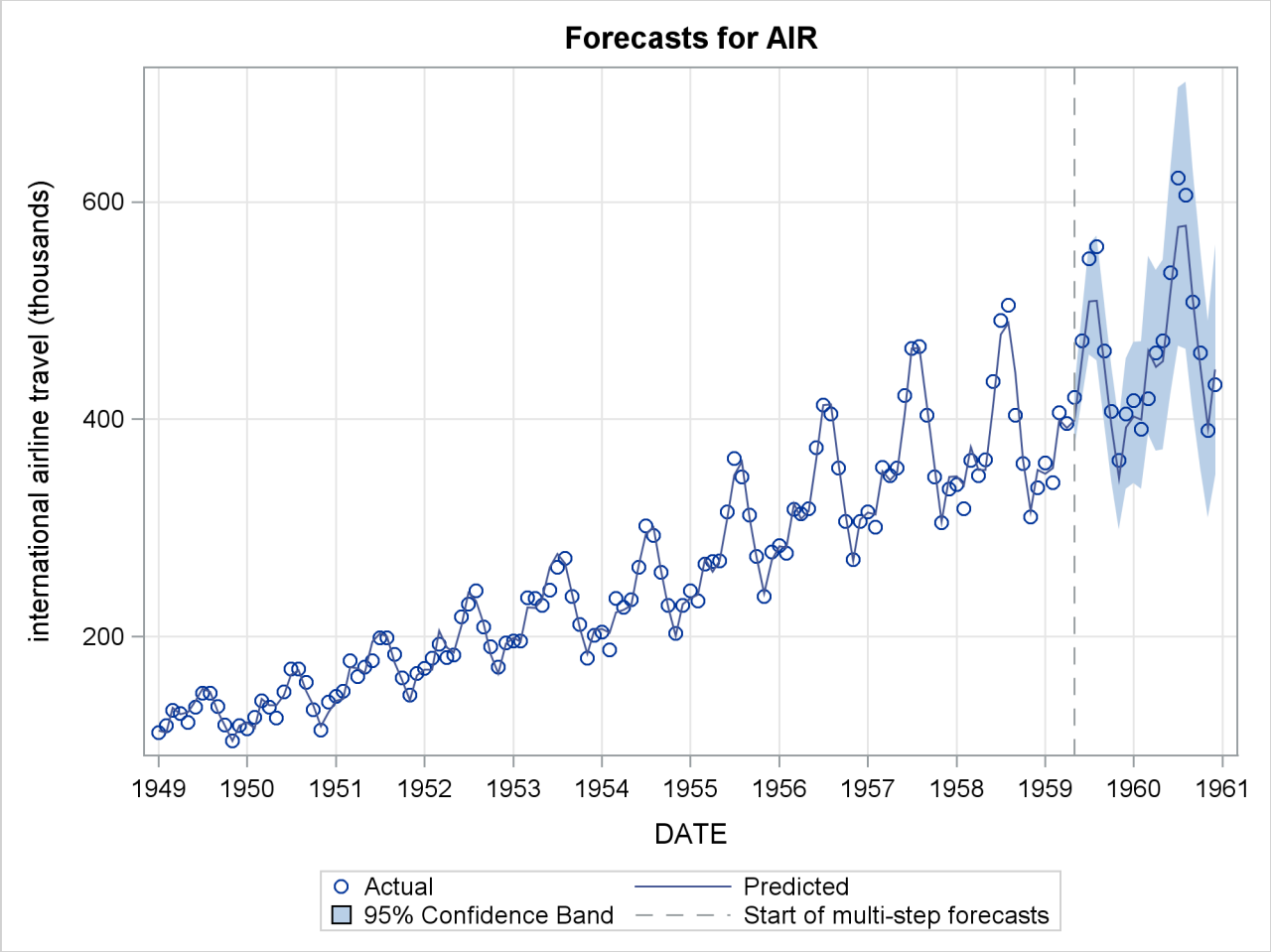
Output 15.5.2 Prediction Error Plot



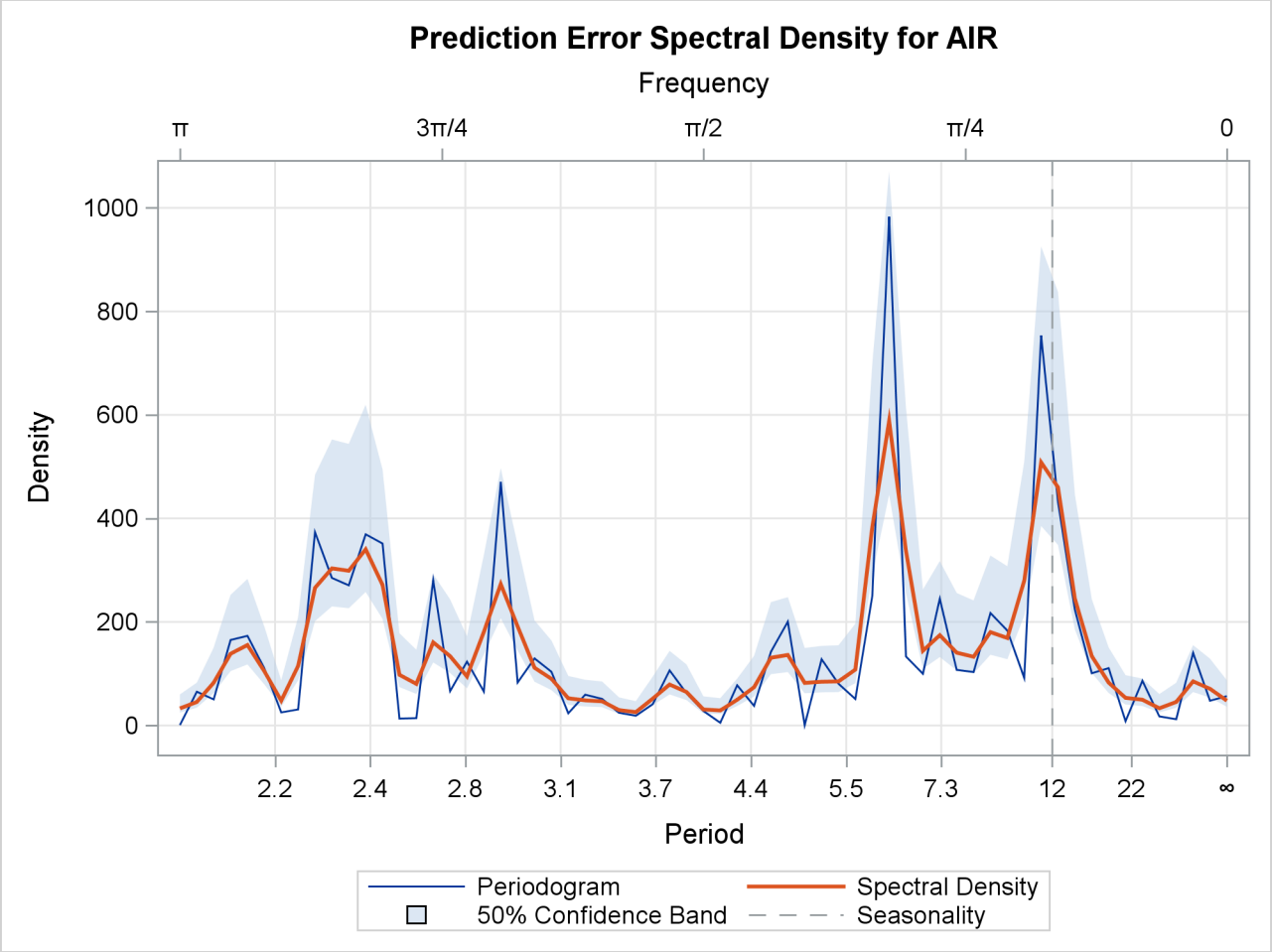
Output 15.5.3 Prediction Error Standardized ACF Plot



Output 15.5.4 Forecast Plot



Output 15.5.5 Prediction Error Spectral Density



Chapter 16
The EXPAND Procedure

Contents

Overview: EXPAND Procedure **884**
Getting Started: EXPAND Procedure **885**
Converting to Higher Frequency Series 885
Aggregating to Lower Frequency Series 885
Combining Time Series with Different Frequencies 886
Interpolating Missing Values 886
Requesting Different Interpolation Methods 887
Using the ID Statement 888
Specifying Observation Characteristics 888
Converting Observation Characteristics 889
Creating New Variables 889
Transforming Series 890
Syntax: EXPAND Procedure **891**
Functional Summary 891
PROC EXPAND Statement 892
BY Statement 895
CONVERT Statement 895
ID Statement 896
Details: EXPAND Procedure **897**
Frequency Conversion 897
Identifying Observations 898
Range of Output Observations 899
Extrapolation 899
OBSERVED= Option 900
Conversion Methods 902
Transformation Operations 904
OUT= Data Set 918
OUTEST= Data Set 919
ODS Graphics 920
Examples: EXPAND Procedure **922**
Example 16.1: Combining Monthly and Quarterly Data 922
Example 16.2: Illustration of ODS Graphics 925
Example 16.3: Interpolating Irregular Observations 929
Example 16.4: Using Transformations 932
References **934**

Overview: EXPAND Procedure

The EXPAND procedure converts time series from one sampling interval or frequency to another and interpolates missing values in time series. A wide array of data transformations is also supported. Using PROC EXPAND, you can collapse time series data from higher frequency intervals to lower frequency intervals, or expand data from lower frequency intervals to higher frequency intervals. For example, quarterly values can be aggregated to produce an annual series, or quarterly estimates can be interpolated from an annual series.

Time series frequency conversion is useful when you need to combine series with different sampling intervals into a single data set. For example, if you need as input to a monthly model a series that is only available quarterly, you might use PROC EXPAND to interpolate the needed monthly values.

You can also interpolate missing values in time series, either without changing series frequency or in conjunction with expanding or collapsing the series.

You can convert between any combination of input and output frequencies that can be specified by SAS time interval names. (See Chapter 5, "Date Intervals, Formats, and Functions," for a complete description of SAS interval names.) When the FROM= and TO= options are used to specify *from* and *to* intervals, PROC EXPAND automatically accounts for calendar effects such as the differing number of days in each month and leap years.

The EXPAND procedure also handles conversions of frequencies that cannot be defined by standard interval names. Using the FACTOR= option, you can interpolate any number of output observations for each group of a specified number of input observations. For example, if you specify the option FACTOR=(13:2), 13 equally spaced output observations are interpolated from each pair of input observations.

You can also convert aperiodic series, observed at arbitrary points in time, into periodic estimates. For example, a series of randomly timed quality control spot-check results might be interpolated to form estimates of monthly average defect rates.

The EXPAND procedure can also change the observation characteristics of time series. Time series observations can measure beginning-of-period values, end-of-period values, midpoint values, or period averages or totals. PROC EXPAND can convert between these cases. You can construct estimates of interval averages from end-of-period values of a variable, estimate beginning-of-period or midpoint values from interval averages, or compute averages from interval totals, and so forth.

By default, the EXPAND procedure fits cubic spline curves to the nonmissing values of variables to form continuous-time approximations of the input series. Output series are then generated from the spline approximations. Several alternate conversion methods are described in the section "Conversion Methods" on page 902. You can also interpolate estimates of the rate of change of time series by differentiating the interpolating spline curve.

Various transformations can be applied to the input series prior to interpolation and to the interpolated output series. For example, the interpolation process can be modified by transforming the input series, interpolating the transformed series, and applying the inverse of the input transformation to the output series. PROC EXPAND can also be used to apply transformations to time series without interpolation or frequency conversion.

The results of the EXPAND procedure are stored in a SAS data set. No printed output is produced.

Getting Started: EXPAND Procedure

Converting to Higher Frequency Series

To create higher frequency estimates, specify the input and output intervals with the FROM= and TO= options, and list the variables to be converted in a CONVERT statement. For example, suppose variables X, Y, and Z in the data set ANNUAL are annual time series, and you want monthly estimates. You can interpolate monthly estimates by using the following statements:

```
proc expand data=annual out=monthly from=year to=month;
   convert x y z;
run;
```

Note that interpolating values of a time series does not add any real information to the data as the interpolation process is not the same process that generated the other (nonmissing) values in the series. While time series interpolation can sometimes be useful, great care is needed in analyzing time series containing interpolated values.

Aggregating to Lower Frequency Series

PROC EXPAND provides two ways to convert from a higher frequency to a lower frequency. When a curve fitting method is used, converting to a lower frequency is no different than converting to a higher frequency–you just specify the desired output frequency with the TO= option. This provides for interpolation of missing values and allows conversion from non-nested intervals, such as converting from weekly to monthly values.

Alternatively, you can specify simple aggregation or selection without interpolation of missing values. This might be useful, for example, if you want to add up monthly values to produce annual totals, but want the annual output data set to contain values only for complete years.

To perform simple aggregation, use the METHOD=AGGREGATE option in the CONVERT statement. For example, the following statements aggregate monthly values to yearly values:

```
proc expand data=monthly out=annual
            from=month to=year;
   convert x y z / method=aggregate;
   convert a b c / method=aggregate observed=total;
   id date;
run;
```

This example assumes that the variables X, Y, and Z represent point-in-time values observed at the beginning of each month, and that the desired results are point-in-time values observed at the beginning of each year. (The default value of the OBSERVED= option is OBSERVED=(BEGINNING,BEGINNING).) The variables A, B, and C are assumed to represent monthly totals, and that the desired results are annual totals; therefor

the option OBSERVED=TOTAL is specified. See the section "Specifying Observation Characteristics" on page 888 for more information on the OBSERVED= option.

Note that the AGGREGATE method can be used only if the input intervals are nested within the output intervals, as when converting from daily to monthly or from monthly to yearly frequency.

Combining Time Series with Different Frequencies

One important use of PROC EXPAND is to combine time series measured at different sampling frequencies. For example, suppose you have data on monthly money stocks (M1), quarterly gross domestic product (GDP), and weekly interest rates (INTEREST), and you want to perform an analysis of a model that uses all these variables. To perform the analysis, you first need to convert the series to a common frequency and then combine the variables into one data set.

The following statements illustrate this process for the three data sets QUARTER, MONTHLY, and WEEKLY. The data sets QUARTER and WEEKLY are converted to monthly frequency using two PROC EXPAND steps, and the three data sets are then merged using a DATA step MERGE statement to produce the data set COMBINED. The quarterly GDP data are interpolated as the total GDP over each month (OBSERVED=TOTAL) while the weekly INTEREST data are converted to average rates over each month (OBSERVED=AVERAGE).

```
proc expand data=quarter out=temp1
            from=qtr to=month;
   id date;
   convert gdp / observed=total;
run;

proc expand data=weekly out=temp2
            from=week to=month;
   id date;
   convert interest / observed=average;
run;

data combined;
   merge monthly temp1 temp2;
   by date;
run;
```

See Chapter 4, "Working with Time Series Data," for further discussion of time series periodicity, time series dating, and time series interpolation. See the section "Specifying Observation Characteristics" on page 888 for more information on the OBSERVED= option.

Interpolating Missing Values

To interpolate missing values in time series without converting the observation frequency, leave off the TO= option on the PROC EXPAND statement. For example, the following statements interpolate any missing values in the time series in the data set ANNUAL.

```
proc expand data=annual out=new from=year;
   id date;
   convert x y z;
   convert a b c / observed=total;
run;
```

This example assumes that the variables X, Y, and Z represent point-in-time values observed at the beginning of each year. (The default value of the OBSERVED= option is OBSERVED=BEGINNING.) The variables A, B, and C are assumed to represent annual totals.

To interpolate missing values in variables observed at specific points in time, omit both the FROM= and TO= options and use the ID statement to supply time values for the observations. The observations do not need to be periodic or form regular time series, but the data set must be sorted by the ID variable. For example, the following statements interpolate any missing values in the numeric variables in the data set A.

```
proc expand data=a out=b;
   id date;
run;
```

If the observations are equally spaced in time, and all the series are observed as beginning-of-period values, only the input and output data sets need to be specified. For example, the following statements interpolate any missing values in the numeric variables in the data set A using a cubic spline function, assuming that the observations are at equally spaced points in time.

```
proc expand data=a out=b;
run;
```

Refer to the section "Missing Values" on page 911 for further information.

Requesting Different Interpolation Methods

By default, a cubic spline curve is fit to the input series, and the output is computed from this interpolating curve. Other interpolation methods can be specified with the METHOD= option on the CONVERT statement. The section "Conversion Methods" on page 902 explains the available methods.

For example, the following statements convert annual series to monthly series using linear interpolation instead of cubic spline interpolation.

```
proc expand data=annual out=monthly from=year to=month;
   id date;
   convert x y z / method=join;
run;
```

Using the ID Statement

An ID statement is normally used with PROC EXPAND to specify a SAS date or datetime variable to identify the time of each input observation. An ID variable allows PROC EXPAND to do the following:

- identify the observations in the output data set
- determine the time span between observations and detect gaps in the input series caused by omitted observations
- account for calendar effects such as the number of days in each month and leap years

If you do not specify an ID variable with SAS date or datetime values, PROC EXPAND makes default assumptions that may not be what you want. See the section "ID Statement" on page 896 for details.

Specifying Observation Characteristics

It is important to distinguish between variables that are measured at points in time and variables that represent totals or averages over an interval. Point-in-time values are often called *stocks* or *levels*. Variables that represent totals or averages over an interval are often called *flows* or *rates*.

For example, the annual series *U.S. Gross Domestic Product* represents the total value of production over the year and also the yearly average rate of production in dollars per year. However, a monthly variable *inventory* may represent the cost of a stock of goods as of the end of the month.

When the data represent periodic totals or averages, the process of interpolation to a higher frequency is sometimes called *distribution*, and the total values of the larger intervals are said to be *distributed* to the smaller intervals. The process of interpolating periodic total or average values to lower frequency estimates is sometimes called *aggregation*.

By default, PROC EXPAND assumes that all time series represent beginning-of-period point-in-time values. If a series does not measure beginning of period point-in-time values, interpolation of the data values using this assumption is not appropriate, and you should specify the correct observation characteristics of the series. The observation characteristics of the series are specified with the OBSERVED= option on the CONVERT statement.

For example, suppose that the data set ANNUAL contains variables A, B, and C that measure yearly totals, while the variables X, Y, and Z measure first-of-year values. The following statements estimate the contribution of each month to the annual totals in A, B, and C, and interpolate first-of-month estimates of X, Y, and Z.

```
proc expand data=annual out=monthly
            from=year to=month;
   id date;
   convert x y z;
   convert a b c / observed=total;
run;
```

The EXPAND procedure supports five different observation characteristics. The OBSERVED= options for these five observation characteristics are:

BEGINNING	beginning-of-period values
MIDDLE	period midpoint values
END	end-of-period values
TOTAL	period totals
AVERAGE	period averages

The interpolation of each series is adjusted appropriately for its observation characteristics. When OBSERVED=TOTAL or AVERAGE is specified, the interpolating curve is fit to the data values so that the area under the curve within each input interval equals the value of the series. For OBSERVED=MIDDLE or END, the curve is fit through the data points, with the time position of each data value placed at the specified offset from the start of the interval.

See the section "OBSERVED= Option" on page 900 for details.

Converting Observation Characteristics

The EXPAND procedure can be used to interpolate values for output series with different observation characteristics than the input series. To change observation characteristics, specify two values in the OBSERVED= option. The first value specifies the observation characteristics of the input series; the second value specifies the observation characteristics of the output series.

For example, the following statements convert the period total variable A in the data set ANNUAL to yearly midpoint estimates. This example does not change the series frequency, and the other variables in the data set are copied to the output data set unchanged.

```
proc expand data=annual out=new from=year;
   id date;
   convert a / observed=(total,middle);
run;
```

Creating New Variables

You can use the CONVERT statement to name a new variable to contain the results of the conversion. Using this feature, you can create several different versions of a series in a single PROC EXPAND step. Specify the new name after the input variable name and an equal sign:

```
convert variable=newname ... ;
```

For example, suppose you are converting quarterly data to monthly and you want both first-of-month and midmonth estimates for a beginning-of-period variable X. The following statements perform this task:

```
proc expand data=a out=b
            from=qtr to=month;
   id date;
   convert x=x_begin / observed=beginning;
   convert x=x_mid   / observed=(beginning,middle);
run;
```

Transforming Series

The interpolation methods used by PROC EXPAND assume that there are no restrictions on the range of values that series can have. This assumption can sometimes cause problems if the series must be within a certain range.

For example, suppose you are converting monthly sales figures to weekly estimates. Sales estimates should never be less than zero, but since the spline curve ignores this restriction some interpolated values may be negative. One way to deal with this problem is to transform the input series before fitting the interpolating spline and then reverse transform the output series.

You can apply various transformations to the input series using the TRANSFORMIN= option on the CONVERT statement. (The TRANSFORMIN= option can be abbreviated as TRANSFORM= or TIN=.) You can apply transformations to the output series using the TRANSFORMOUT= option. (The TRANSFORMOUT= option can be abbreviated as TOUT=.)

For example, you might use a logarithmic transformation of the input sales series and exponentiate the interpolated output series. The following statements fit a spline curve to the log of SALES and then exponentiate the output series.

```
proc expand data=a out=b from=month to=week;
   id date;
   convert sales / observed=total
                   transformin=(log)
                   transformout=(exp);
run;
```

Note that the transformations specified by the TRANSFORMIN= option are applied before the data are interpolated; the cubic spline curve or other interpolation method is fitted to transformed input data. The transformations specified by the TRANSFORMOUT= option are applied to interpolated values computed from the curves fit to the transformed input data.

As another example, suppose you are interpolating missing values in a series of market share estimates. Market shares must be between 0% and 100%, but applying a spline interpolation to the raw series can produce estimates outside of this range.

The following statements use the logistic transformation to transform proportions in the range 0 to 1 to values in the range $-\infty$ to $+\infty$. The TIN= option first divides the market shares by 100 to rescale percent values to proportions and then applies the LOGIT function. The TOUT= option applies the inverse logistic function ILOGIT to the interpolated values to convert back to proportions and then multiplies by 100 to rescale back to percentages.

```
proc expand data=a out=b;
   id date;
   convert mshare / tin=( / 100 logit )
                    tout=( ilogit * 100 );
run;
```

When more than one transformation is specified in the TRANSFORMIN= or TRANSFORMOUT= option, the transformations are applied in the order in which they are listed. Thus in the above example the complete

input transformation is *logit(mshare/100)* (and not *logit(mshare)/100*) because the division operation is listed first in the TIN= option.

You can also use the TRANSFORM= (or TRANSFORMOUT=) option as a convenient way to do calculations normally performed with the SAS DATA step. For example, the following statements add the lead of X to the data set A. The METHOD=NONE option is used to suppress interpolation.

```
proc expand data=a method=none;
   id date;
   convert x=xlead / transform=(lead);
run;
```

Any number of operations can be listed in the TRANSFORMIN= and TRANSFORMOUT= options. See Table 16.2 for a list of the operations supported.

Syntax: EXPAND Procedure

The EXPAND procedure uses the following statements:

PROC EXPAND *options* **;**
BY *variables* **;**
CONVERT *variables* / *options* **;**
ID *variable* **;**

Functional Summary

The statements and options controlling the EXPAND procedure are summarized in the following table.

Description	Statement	Option
Statements		
specify options	PROC EXPAND	
specify BY-group processing	BY	
specify conversion options	CONVERT	
specify the ID variable	ID	
Data Set Options		
specify the input data set	PROC EXPAND	DATA=
extrapolate values before or after input series	PROC EXPAND	EXTRAPOLATE
specify the output data set	PROC EXPAND	OUT=
write interpolating functions to a data set	PROC EXPAND	OUTEST=
Input and Output Frequencies		
control the alignment of SAS Date values	PROC EXPAND	ALIGN=
specify frequency conversion factor	PROC EXPAND	FACTOR=
specify input frequency	PROC EXPAND	FROM=
specify output frequency	PROC EXPAND	TO=

Description	Statement	Option
Interpolation Control Options		
specify interpolation method for all series	PROC EXPAND	METHOD=
specify interpolation method for series	CONVERT	METHOD=
specify observation characteristics for series	PROC EXPAND	OBSERVED=
specify observation characteristics for series	CONVERT	OBSERVED=
specify transformations of the input series	CONVERT	TRANSFORMIN=
specify transformations of the output series	CONVERT	TRANSFORMOUT=
Graphical Output Control Options		
specify graphical output	PROC EXPAND	PLOTS=

PROC EXPAND Statement

PROC EXPAND *options* **;**

The following options can be used with the PROC EXPAND statement:

Data Set Options

DATA= *SAS-data-set*
names the input data set. If the DATA= option is omitted, the most recently created SAS data set is used.

OUT= *SAS-data-set*
names the output data set containing the resulting time series. If OUT= is not specified, the data set is named using the DATA*n* convention. See the section "OUT= Data Set" on page 918 for details.

OUTEST= *SAS-data-set*
names an output data set containing the coefficients of the spline curves fit to the input series. If the OUTEST= option is not specified, the spline coefficients are not output. See the section "OUTEST= Data Set" on page 919 for details.

Options That Define Input and Output Frequencies

ALIGN= *option*
controls the alignment of SAS dates used to identify output observations. The ALIGN= option allows the following values: BEGINNING | BEG | B, MIDDLE | MID | M, and ENDING | END | E. BEGINNING is the default.

FACTOR= *n*

FACTOR=(*n* **:** *m* **)**

specifies the number of output observations to be created from the input observations. FACTOR=*n* specifies that *n* output observations are to be produced for each input observation. FACTOR=(*n* : *m*) specifies that *n* output observations are to be produced for each group of *m* input observations. FACTOR=*n* is the same as FACTOR=(*n* : 1).

In the FACTOR=() option, a comma can be used instead of a colon or the delimiter can be omitted. Thus FACTOR=(*n*, *m*) or FACTOR=(*n m*) is the same as FACTOR=(*n* : *m*).

The FACTOR= option cannot be used if the TO= option is used. The default value is FACTOR=(1:1). For more information, see the section "Frequency Conversion" on page 897.

FROM= *interval*

specifies the time interval between observations in the input data set. Examples of FROM= values are YEAR, QTR, MONTH, DAY, and HOUR. See Chapter 5, "Date Intervals, Formats, and Functions," for a complete description and examples of interval specifications.

TO= *interval*

specifies the time interval between observations in the output data set. By default, the TO= interval is generated from the combination of the FROM= and the FACTOR= values or is set to be the same as the FROM= value if FACTOR= is not specified. See Chapter 5, "Date Intervals, Formats, and Functions," for a description of interval specifications.

Options to Control the Interpolation

EXTRAPOLATE

specifies that missing values at the beginning or end of input series be replaced with values produced by a linear extrapolation of the interpolating curve fit to the input series. See the section "Extrapolation" on page 899 later in this chapter for details.

By default, PROC EXPAND avoids extrapolating values beyond the first or last input value for a series and only interpolates values within the range of the nonmissing input values. Note that the extrapolated values are often not very accurate and for the SPLINE method the EXTRAPOLATE option results may be very unreasonable. The EXTRAPOLATE option is rarely used.

METHOD= *option*

METHOD=SPLINE(*constraint* **<** **,** *constraint* **>** **)**

specifies the method used to convert the data series. The methods supported are SPLINE, JOIN, STEP, AGGREGATE, and NONE. The METHOD= option specified on the PROC EXPAND statement can be overridden for particular series by the METHOD= option on the CONVERT statement. The default is METHOD=SPLINE. The *constraint* specifications for METHOD=SPLINE can have the values NOTAKNOT (the default), NATURAL, SLOPE=*value*, and/or CURVATURE=*value*. See the section "Conversion Methods" on page 902 for more information about these methods.

OBSERVED= *value*

OBSERVED=(*from-value* **,** *to-value* **)**

indicates the observation characteristics of the input time series and of the output series. Specifying the OBSERVED= option on the PROC EXPAND statement sets the default OBSERVED= value for subsequent CONVERT statements. See the sections "CONVERT Statement" on page 895

and "OBSERVED= Option" on page 900 later in this chapter for details. The default is OBSERVED=BEGINNING.

Options to Control Graphical Output

PLOTS= *option* | **(** *options* **)**
specifies the graphical output desired. If the PLOTS= option is used, the specified graphical output is produced for each output variable that is specified by a CONVERT statement. By default, the EXPAND procedure produces no graphical output. The following PLOTS= options are available:

INPUT	plots the input series.
TRANSFORMIN	plots the transformed input series. The TRANSFORMIN= option must also be specified in the CONVERT statement.
CROSSINPUT	plots both the input series and the transformed input series on one plot with two Y axes. The input and transformed series are shown on separate scales. The TRANSFORMIN= option must also be specified in the CONVERT statement.
JOINTINPUT	plots both the input series and the transformed input series on one plot with one Y axis. The input and transformed series are shown on the same scale. The TRANSFORMIN= option must also be specified in the CONVERT statement.
CONVERTED	plots the converted series after input transformations and interpolation, but before any TRANSFORMOUT= transformations are applied. The METHOD= option must also be specified in the PROC EXPAND or CONVERT statements.
TRANSFORMOUT	plots the transformed output series. The TRANSFORMOUT= option must also be specified in the CONVERT statement.
CROSSOUTPUT	plots both the converted series and the transformed output series on one plot with two Y axes. The converted and transformed output series are shown on separate scales. The TRANSFORMOUT= option must also be specified in the CONVERT statement.
JOINTOUTPUT	plots both the converted series and the transformed output series on one plot with one Y axis. The converted and transformed output series are shown on the same scale. The TRANSFORMOUT= option must also be specified in the CONVERT statement.
OUTPUT	plots the series stored in the OUT= data set. The OUTPUT option does not require any options to be specified in the CONVERT statement.
ALL	produces all plots except the joint and cross plots. PLOTS=ALL is the same as PLOTS=(INPUT TRANFORMIN CONVERTED TRANSFORMOUT).

The PLOTS= option produces results associated with each CONVERT statement output variable and the options listed in the PLOTS= specification. See the section "PLOTS= Option Details" on page 921 for more information.

BY Statement

BY *variables* **;**

A BY statement can be used with PROC EXPAND to obtain separate analyses on observations in groups defined by the BY variables. The input data set must be sorted by the BY variables and be sorted by the ID variable within each BY group.

Use a BY statement when you want to interpolate or convert time series within levels of a cross-sectional variable. For example, suppose you have a data set STATE containing annual estimates of average disposable personal income per capita (DPI) by state and you want quarterly estimates by state. These statements convert the DPI series within each state:

```
proc sort data=state;
   by state date;
run;

proc expand data=state out=stateqtr from=year to=qtr;
   convert dpi;
   by state;
   id date;
run;
```

CONVERT Statement

CONVERT *variable = newname . . . < / options >* **;**

The CONVERT statement lists the variables to be processed. Only numeric variables can be processed.

For each of the variables listed, a new variable name can be specified after an equal sign to name the variable in the output data set that contains the converted values. If a name for the output series is not given, the variable in the output data set has the same name as the input variable. Variable lists may be used only when no name is given for the output series.

For example, variable lists can be specified as follows:

```
convert y1-y25 / observed=(beginning,end);
convert x--a / observed=average;
convert x-numeric-a / observed=average;
```

Any number of CONVERT statements can be used. If no CONVERT statement is used, all the numeric variables in the input data set except those appearing in the BY and ID statements are processed.

The following options can be used with the CONVERT statement.

METHOD= *option*

METHOD=SPLINE(*constraint* **<** **,** *constraint* **>** **)**

specifies the method used to convert the data series. (The method specified by the METHOD= option is applied to the input data series after applying any transformations specified by the TRANSFORMIN= option.) The methods supported are SPLINE, JOIN, STEP, AGGREGATE, and NONE. The METHOD= option specified on the PROC EXPAND statement can be overridden for particular series by the METHOD= option on the CONVERT statement. The default is METHOD=SPLINE. The *constraint* specifications for METHOD=SPLINE can have the values NOTAKNOT (the default), NATURAL, SLOPE=*value*, and/or CURVATURE=*value*. See the section "Conversion Methods" on page 902 section for more information about these methods.

OBSERVED= *value*

OBSERVED=(*from-value* **,** *to-value* **)**

indicates the observation characteristics of the input time series and of the output series. The values supported are TOTAL, AVERAGE, BEGINNING, MIDDLE, and END. In addition, DERIVATIVE can be specified as the *to-value* when the SPLINE method is used.

When only one value is specified, that value specifies both the *from-value* and the *to-value*. (That is, OBSERVED=*value* is equivalent to OBSERVED=(*value*, *value*).) If the OBSERVED= option is omitted from both the PROC EXPAND and the CONVERT statements, the default is OBSERVED=(BEGINNING, BEGINNING). See the section "OBSERVED= Option" on page 900 for details.

TRANSFORMIN=(*operation . . .* **)**

specifies a list of transformations to be applied to the input series before the interpolating function is fit. The operations are applied in the order listed. See the section "Transformation Operations" on page 904 later in this chapter for the operations that can be specified. The TRANSFORMIN= option can be abbreviated as TRANSIN=, TIN=, or TRANSFORM=.

TRANSFORMOUT=(*operation . . .* **)**

specifies a list of transformations to be applied to the output series. The operations are applied in the order listed. See the section "Transformation Operations" on page 904 later in this chapter for the operations that can be specified. The TRANSFORMOUT= option can be abbreviated as TRANSOUT=, or TOUT=.

ID Statement

ID *variable* **;**

The ID statement names a numeric variable that identifies observations in the input and output data sets. The ID variable's values are assumed to be SAS date or datetime values.

The input data must form time series. This means that the observations in the input data set must be sorted by the ID variable (within the BY variables, if any). Moreover, there should be no duplicate observations, and no two observations should have ID values within the same time interval as defined by the FROM= option.

If the ID statement is omitted, SAS date or datetime values are generated to label the input observations. These ID values are generated by assuming that the input data set starts at a SAS date value of 0, that is, 1 January 1960. This default starting date is then incremented for each observation by the FROM= interval

(using the same logic as DATA step INTNX function). If the FROM= option is not specified, the ID values are generated as the observation count minus 1. When the ID statement is not used, an ID variable is added to the output data set named either DATE or DATETIME, depending on the value specified in the TO= option. If neither the TO= option nor the FROM= option is given, the ID variable in the output data set is named TIME.

Details: EXPAND Procedure

Frequency Conversion

Frequency conversion is controlled by the FROM=, TO=, and FACTOR= options. The possible combinations of these options are explained in the following:

None Used
If FROM=, TO=, and FACTOR= are not specified, no frequency conversion is done. The data are processed to interpolate any missing values and perform any specified transformations. Each input observation produces one output observation.

FACTOR=(n:m)
FACTOR=(*n* :*m*) specifies that *n* output observations are produced for each group of *m* input observations. The fraction *m* /*n* is reduced first: thus FACTOR=(10:6) is equivalent to FACTOR=(5:3). Note that if *m* /*n* =1, the result is the same as the case given previously under “None Used”.

FROM=interval
The FROM= option used alone establishes the frequency and interval widths of the input observations. Missing values are interpolated, and any specified transformations are performed, but no frequency conversion is done.

TO=interval
When the TO= option is used without the FROM= option, output observations with the TO= frequency are generated over the range of input ID values. The first output observation is for the TO= interval containing the ID value of the first input observation; the last output observation is for the TO= interval containing the ID value of the last input observation. The input observations are not assumed to form regular time series and may represent aperiodic points in time. An ID variable is required to give the date or datetime of the input observations.

FROM=interval TO=interval
When both the FROM= and TO= options are used, the input observations have the frequency given by the FROM= interval, and the output observations have the frequency given by the TO= interval.

FROM=interval FACTOR=(n:m)
When both the FROM= and FACTOR= options are used, a TO= interval is inferred from the combination of the FROM=*interval* and the FACTOR=(*n:m*) values specified. For example, FROM=YEAR FACTOR=4 is the same as FROM=YEAR TO=QTR. Also, FROM=YEAR FACTOR=(3:2) is the same as FROM=YEAR used with TO=MONTH8. Once the implied TO= interval is determined, this combination operates the same as if FROM= and TO= had been specified. If no valid TO= interval can be constructed from the combination of the FROM= and FACTOR= options, an error is produced.

TO=interval FACTOR=(n:m)
The combination of the TO= option and the FACTOR= option is not allowed and produces an error.

ALIGN= option
Controls the alignment of SAS dates used to identify output observations. The ALIGN= option allows the following values: BEGINNING | BEG | B, MIDDLE | MID | M, and ENDING | END | E. BEGINNING is the default.

Converting to a Lower Frequency

When converting to a lower frequency, the results are either exact or approximate, depending on whether or not the input interval nests within the output interval and depending on the need to interpolate missing values within the series. If the TO= interval is nested within the FROM= interval (as when converting from monthly to yearly), and if there are no missing input values or partial periods, the results are exact.

When values are missing or the FROM= interval is not nested within the TO= interval (as when aggregating from weekly to monthly), the results depend on an interpolation. The METHOD=AGGREGATE option always produces exact results, never an interpolation. However, this method can only be used if the FROM= interval is nested within the TO= interval.

Identifying Observations

The variable specified in the ID statement is used to identify the observations. Usually, SAS date or datetime values are used for this variable. PROC EXPAND uses the ID variable to do the following:

- identify the time interval of the input values
- validate the input data set observations
- compute the ID values for the observations in the output data set

Identifying the Input Time Intervals

When the FROM= option is specified, observations are understood to refer to the whole time interval and not to a single time point. The ID values are interpreted as identifying the FROM= time interval containing the value. In addition, the widths of these input intervals are used by the OBSERVED= values TOTAL, AVERAGE, MIDDLE, and END.

For example, if FROM=MONTH is specified, then each observation is for the whole calendar month containing the ID value for the observation, and the width of the time interval covered by the observation is the number of days in that month. Therefore, if FROM=MONTH, the ID value '31MAR92'D is equivalent to the ID value '1MAR92'D–both of these ID values identify the same interval, March of 1992.

Widths of Input Time Intervals

When the FROM= option is not specified, the ID variable values are usually interpreted as referring to points in time. However, if an OBSERVED= option value is specified that assumes the observations refer to whole intervals and also requires interval widths (TOTAL or AVERAGE), then, in the absence of the FROM=

specification, interval widths are assumed to be the time span between ID values. For the last observation, the interval width is assumed to be the same as for the next to last observation. (If neither the FROM= option nor the ID statement are specified, interval widths are assumed to be 1.0.) A note is printed in the SAS log warning that this assumption is made.

Validating the Input Data Set Observations

The ID variable is used to verify that successive observations read from the input data set correspond to sequential FROM= intervals. When the FROM= option is not used, PROC EXPAND verifies that the ID values are nonmissing and in ascending order. An error message is produced and the observation is ignored when an invalid ID value is found in the input data set.

ID values for Observations in the Output Data Set

The time unit used for the ID variable in the output data set is controlled by the interval value specified by the TO= option. If you specify a date interval for the TO= value, the ID variable values in the output data set are SAS date values. If you specify a datetime interval for the TO= value, the ID variable values in the output data set are SAS datetime values.

The date or datetime values for the ID variable for output observations is the first date or datetime of the TO= interval, unless the ALIGN= option is used to specify a different alignment. (For example, if TO=WEEK is specified, then the output dates are Sundays. If TO=WEEK.2 is specified, then the output date are Mondays.) See Chapter 5, "Date Intervals, Formats, and Functions," for more information on interval specifications.

Range of Output Observations

If no frequency conversion is done, the range of output observations is the same as in the input data set.

When frequency conversion is done, the observations in the output data set range from the earliest start of any result series to the latest end of any result series. Observations at the beginning or end of the input range for which all result values are missing are not written to the OUT= data set.

When the EXTRAPOLATE option is not used, the range of the nonmissing output results for each series is as follows. The first result value is for the TO= interval that contains the ID value of the start of the FROM= interval containing the ID value of the first nonmissing input observation for the series. The last result value is for the TO= interval that contains the end of the FROM= interval containing the ID value of the last nonmissing input observation for the series.

When the EXTRAPOLATE option is used, result values for all series are computed for the full time range covered by the input data set.

Extrapolation

The spline functions fit by the EXPAND procedure are very good at approximating continuous curves within the time range of the input data but poor at extrapolating beyond the range of the data. The accuracy of the results produced by PROC EXPAND may be somewhat less at the ends of the output series than at time periods for which there are several input values at both earlier and later times. The curves fit by PROC EXPAND should not be used for forecasting.

PROC EXPAND normally avoids extrapolation of values beyond the time range of the nonmissing input data for a series, unless the EXTRAPOLATE option is used. However, if the start or end of the input series does not correspond to the start or end of an output interval, some output values may depend in part on an extrapolation.

For example, if FROM=YEAR, TO=WEEK, and OBSERVED=BEGINNING are specified, then the first observation output for a series is for the week of 1 January of the first nonmissing input year. If 1 January of that year is not a Sunday, the beginning of this week falls before the date of the first input value, and therefore a beginning-of-period output value for this week is extrapolated.

This extrapolation is made only to the extent needed to complete the terminal output intervals that overlap the endpoints of the input series and is limited to no more than the width of one FROM= interval or one TO= interval, whichever is less. This restriction of the extrapolation to complete terminal output intervals is applied to each series separately, and it takes into account the OBSERVED= option for the input and output series.

When the EXTRAPOLATE option is used, the normal restriction on extrapolation is overridden. Output values are computed for the full time range covered by the input data set.

For the SPLINE method, extrapolation is performed by a linear projection of the trend of the cubic spline curve fit to the input data, not by extrapolation of the first and last cubic segments.

The EXTRAPLOTE option should be used with caution.

OBSERVED= Option

The values of the CONVERT statement OBSERVED= option are as follows:

BEGINNING	indicates that the data are beginning-of-period values. OBSERVED=BEGINNING is the default.
MIDDLE	indicates that the data are period midpoint values.
ENDING	indicates that the data represent end-of-period values.
TOTAL	indicates that the data values represent period totals for the time interval corresponding to the observation.
AVERAGE	indicates that the data values represent period averages.
DERIVATIVE	requests that the output series be the derivatives of the cubic spline curve fit to the input data by the SPLINE method.

If only one value is specified in the OBSERVED= option, that value applies to both the input and the output series. For example, OBSERVED=TOTAL is the same as OBSERVED=(TOTAL,TOTAL), which indicates that the input values represent totals over the time intervals corresponding to the input observations, and the converted output values also represent period totals. The value DERIVATIVE can be used only as the second OBSERVED= option value, and it can be used only when METHOD=SPLINE is specified or is the default method.

Since the TOTAL, AVERAGE, MIDDLE, and END cases require that the width of each input interval be known, both the FROM= option and an ID statement are normally required if one of these observation characteristics is specified for any series. However, if the FROM= option is not specified, each input interval

is assumed to extend from the ID value for the observation to the ID value of the next observation, and the width of the interval for the last observation is assumed to be the same as the width for the next to last observation.

Scale of OBSERVED=AVERAGE Values

The average values are assumed to be expressed in the time units defined by the FROM= or TO= option. That is, the product of the average value for an interval and the width of the interval is assumed to equal the total value for the interval. For purposes of interpolation, OBSERVED=AVERAGE values are first converted to OBSERVED=TOTAL values using this assumption, and then the interpolated totals are converted back to averages by dividing by the widths of the output intervals.

For example, suppose the options FROM=MONTH, TO=HOUR, and OBSERVED=AVERAGE are specified. Since FROM=MONTH is specified, each input value is assumed to represent an average rate per day such that the product of the value and the number of days in the month is equal to the total for the month. The input values are assumed to represent a per-day rate because FROM=MONTH implies SAS date ID values that measure time in days, and therefore the widths of MONTH intervals are measured in days. If FROM=DTMONTH is used instead, the values are assumed to represent a per-second rate, because the widths of DTMONTH intervals are measured in seconds.

Since TO=HOUR is specified, the output values are scaled as an average rate per second such that the product of each output value and the number of seconds in an hour (3600) is equal to the interpolated hourly total. A per-second rate is used because TO=HOUR implies SAS datetime ID values that measure time in seconds, and therefore the widths of HOUR intervals are measured in seconds.

Note that the scale assumed for OBSERVED=AVERAGE data is important only when converting between AVERAGE and another OBSERVED= option, or when converting between SAS date and SAS datetime ID values. When both the input and the output series are AVERAGE values, and the units for the ID values are not changed, the scale assumed does not matter.

For example, suppose you are converting gross domestic product (GDP) from quarterly to monthly. The GDP values are quarterly averages measured at annual rates. If you want the interpolated monthly values to also be measured at annual rates, then the option OBSERVED=AVERAGE works fine. Since there is no change of scale involved in this problem, it makes no difference that PROC EXPAND assumes daily rates instead of annual rates.

However, suppose you want to convert GDP from quarterly to monthly and also convert from annual rates to monthly rates, so that the result is total gross domestic product for the month. Using the option OBSERVED=(AVERAGE,TOTAL) would fail, because PROC EXPAND assumes the average is scaled to daily, not annual, rates.

One solution is to rescale to quarterly totals and treat the data as totals. You could use the options TRANSFORMIN=(/ 4) OBSERVED=TOTAL. Alternatively, you could treat the data as averages but first convert to daily rates. In this case you would use the options TRANSFORMIN=(/ 365.25) OBSERVED=AVERAGE.

Results of the OBSERVED=DERIVATIVE Option

If the first value of the OBSERVED= option is BEGINNING, TOTAL, or AVERAGE, the result is the derivative of the spline curve evaluated at first-of-period ID values for the output observation. For OBSERVED=(MIDDLE,DERIVATIVE), the derivative of the function is evaluated at output interval midpoints. For OBSERVED=(END,DERIVATIVE), the derivative is evaluated at end-of-period ID values.

Conversion Methods

The SPLINE Method

The SPLINE method fits a cubic spline curve to the input values. A cubic spline is a segmented function consisting of third-degree (cubic) polynomial functions joined together so that the whole curve and its first and second derivatives are continuous.

For point-in-time input data, the spline curve is constrained to pass through the given data points. For interval total or average data, the definite integrals of the spline over the input intervals are constrained to equal the given interval totals.

For boundary constraints, the *not-a-knot* condition is used by default. This means that the first two spline pieces are constrained to be part of the same cubic curve, as are the last two pieces. Thus the spline used by PROC EXPAND by default is not the same as the commonly used natural spline, which uses zero second-derivative endpoint constraints. While De Boor (1978) recommends the *not-a-knot* constraint for cubic spline interpolation, using this constraint can sometimes produce anomalous results at the ends of the interpolated series. PROC EXPAND provides options to specify other endpoint constraints for spline curves.

To specify endpoint constraints, use the following form of the METHOD= option.

METHOD=SPLINE(*constraint* **<** **,** *constraint* **>** **)**

The first constraint specification applies to the lower endpoint, and the second constraint specification applies to the upper endpoint. If only one constraint is specified, it applies to both the lower and upper endpoints.

The *constraint* specifications can have the following values:

NOTAKNOT
specifies the not-a-knot constraint. This is the default.

NATURAL
specifies the *natural spline* constraint. The second derivative of the spline curve is constrained to be zero at the endpoint.

SLOPE= value
specifies the first derivative of the spline curve at the endpoint. The value specified can be any positive or negative number, but extreme values may produce unreasonable results.

CURVATURE= value
specifies the second derivative of the spline curve at the endpoint. The value specified can be any positive or negative number, but extreme values may produce unreasonable results. Specifying CURVATURE=0 is equivalent to specifying the NATURAL option.

For example, to specify natural spline interpolation, use the following option in the CONVERT or PROC EXPAND statement:

```
method=spline(natural)
```

For OBSERVED=BEGINNING, MIDDLE, and END series, the spline knots are placed at the beginning, middle, and end of each input interval, respectively. For total or averaged series, the spline knots are set at the start of the first interval, at the end of the last interval, and at the interval midpoints, except that there are no knots for the first two and last two midpoints.

Once the cubic spline curve is fit to the data, the spline is extended by adding linear segments at the beginning and end. These linear segments are used for extrapolating values beyond the range of the input data.

For point-in-time output series, the spline function is evaluated at the appropriate points. For interval total or average output series, the spline function is integrated over the output intervals.

The JOIN Method

The JOIN method fits a continuous curve to the data by connecting successive straight line segments. For point-in-time data, the JOIN method connects successive nonmissing input values with straight lines. For interval total or average data, interval midpoints are used as the break points, and ordinates are chosen so that the integrals of the piecewise linear curve agree with the input totals.

For point-in-time output series, the JOIN function is evaluated at the appropriate points. For interval total or average output series, the JOIN function is integrated over the output intervals.

The STEP Method

The STEP method fits a discontinuous piecewise constant curve. For point-in-time input data, the resulting step function is equal to the most recent input value. For interval total or average data, the step function is equal to the average value for the interval.

For point-in-time output series, the step function is evaluated at the appropriate points. For interval total or average output series, the step function is integrated over the output intervals.

The AGGREGATE Method

The AGGREGATE method performs simple aggregation of time series without interpolation of missing values.

If the input data are totals or averages, the results are the sums or averages, respectively, of the input values for observations corresponding to the output observations. That is, if either TOTAL or AVERAGE is specified for the OBSERVED= option, the METHOD=AGGREGATE result is the sum or mean of the input values corresponding to the output observation. For example, suppose METHOD=AGGREGATE, FROM=MONTH, and TO=YEAR are specified. For OBSERVED=TOTAL series, the result for each output year is the sum of the input values over the months of that year. If any input value is missing, the corresponding sum or mean is also a missing value.

If the input data are point-in-time values, the result value of each output observation equals the input value for a selected input observation determined by the OBSERVED= attribute. For example, suppose METHOD=AGGREGATE, FROM=MONTH, and TO=YEAR are specified. For OBSERVED=BEGINNING series, January observations are selected as the annual values. For OBSERVED=MIDDLE series, July observations are selected as the annual values. For OBSERVED=END series, December observations are selected as the annual values. If the selected value is missing, the output annual value is missing.

The AGGREGATE method can be used only when the FROM= intervals are nested within the TO= intervals. For example, you can use METHOD=AGGREGATE when FROM=MONTH and TO=QTR because months are nested within quarters. You cannot use METHOD=AGGREGATE when FROM=WEEK and TO=QTR because weeks are not nested within quarters.

In addition, the AGGREGATE method cannot convert between point-in-time data and interval total or average data. Conversions between TOTAL and AVERAGE data are allowed, but conversions between BEGINNING, MIDDLE, and END are not.

Missing input values produce missing result values for METHOD=AGGREGATE. However, gaps in the sequence of input observations are not allowed. For example, if FROM=MONTH, you may have a missing value for a variable in an observation for a given February. But if an observation for January is followed by an observation for March, there is a gap in the data, and METHOD=AGGREGATE cannot be used.

When the AGGREGATE method is used, there is no interpolating curve, and therefore the EXTRAPOLATE option is not allowed.

Alternate methods for aggregating or accumulating time series data are supported by the TIMESERIES procedure. See Chapter 39, "The TIMESERIES Procedure," for more information.

METHOD=NONE

The option METHOD=NONE specifies that no interpolation be performed. This option is normally used in conjunction with the TRANSFORMIN= or TRANSFORMOUT= option.

When METHOD=NONE is specified, there is no difference between the TRANSFORMIN= and TRANSFORMOUT= options; if both are specified, the TRANSFORMIN= operations are performed first, followed by the TRANSFORMOUT= operations. TRANSFORM= can be used as an abbreviation for TRANSFORMIN=. METHOD=NONE cannot be used when frequency conversion is specified.

Transformation Operations

The operations that can be used in the TRANSFORMIN= and TRANSFORMOUT= options are shown in Table 16.2. Operations are applied to each value of the series. Each value of the series is replaced by the result of the operation.

In Table 16.2, x_t or x represents the value of the series at a particular time period t before the transformation is applied, y_t represents the value of the result series, and N represents the total number of observations.

The notation n_{optional} indicates that the argument n_{optional} is an optional integer; the default is 1. The notation *window* is used as the argument for the moving statistics operators, and it indicates that you can specify either a number of periods *n* (where *n* is an integer) or a list of *n* weights in parentheses. The internal maximum value of the number of periods *n* is clipped at the number of observations in the series. The notation *sequence* is used as the argument for the sequence operators, and it indicates that you must specify a sequence of numbers. The notation *s* indicates the length of seasonality, and it is a required argument.

Table 16.2 Transformation Operations

Syntax	Result
+ *number*	Adds the specified *number* : $x + number$
− *number*	Subtracts the specified *number* : $x - number$
* *number*	Multiplies by the specified *number* : $x * number$
/ *number*	Divides by the specified *number* : $x / number$
ABS	Absolute value: $\|x\|$
ADJUST	Indicates that the following moving window summation or product operator should be adjusted for window width

Table 16.2 *continued*

Syntax	Result
CD_I *s*	Classical decomposition irregular component
CD_S *s*	Classical decomposition seasonal component
CD_SA *s*	Classical decomposition seasonally adjusted series
CD_TC *s*	Classical decomposition trend-cycle component
CDA_I *s*	Classical decomposition (additive) irregular component
CDA_S *s*	Classical decomposition (additive) seasonal component
CDA_SA *s*	Classical decomposition (additive) seasonally adjusted series
CEIL	Smallest integer greater than or equal to x : $\mathrm{ceil}(x)$
CMOVAVE *window*	Centered moving average
CMOVCSS *window*	Centered moving corrected sum of squares
CMOVGMEAN *window*	Centered moving geometric mean for *window* = number of periods, *n*: $(\prod_{j=j_{\min}}^{j_{\max}} x_{t+j})^{1/n}$ $j_{\min} = -(n + n \mod 2)/2 + 1$ $j_{\max} = (n - n \mod 2)/2$ for *window* = weight list, *w*: $(\prod_{j=j_{\min}}^{j_{\max}} x_{t+j}^{w_{j-j_{\min}}})^{1/\sum_{j=0}^{n-1} w_j}$
CMOVMAX *n*	Centered moving maximum
CMOVMED *n*	Centered moving median
CMOVMIN *n*	Centered moving minimum
CMOVPROD *window*	Centered moving product for *window* = number of periods, *n*: $\prod_{j=j_{\min}}^{j_{\max}} x_{t+j}$ for *window* = weight list, *w*: $(\prod_{j=j_{\min}}^{j_{\max}} x_{t+j}^{w_{j-j_{\min}}})^{1/\sum_{j=0}^{n-1} w_j}$
CMOVRANGE *n*	Centered moving range
CMOVRANK *n*	Centered moving rank
CMOVSTD *window*	Centered moving standard deviation
CMOVSUM *n*	Centered moving sum
CMOVTVALUE *window*	Centered moving *t* value
CMOVUSS *window*	Centered moving uncorrected sum of squares
CMOVVAR *window*	Centered moving variance
CUAVE n_{optional}	Cumulative average
CUCSS n_{optional}	Cumulative corrected sum of squares
CUGMEAN n_{optional}	Cumulative geometric mean
CUMAX n_{optional}	Cumulative maximum
CUMED n_{optional}	Cumulative median
CUMIN n_{optional}	Cumulative minimum
CUPROD n_{optional}	Cumulative product
CURANK n_{optional}	Cumulative rank
CURANGE n_{optional}	Cumulative range
CUSTD n_{optional}	Cumulative standard deviation
CUSUM n_{optional}	Cumulative sum
CUTVALUE n_{optional}	Cumulative *t* value

Table 16.2 *continued*

Syntax	Result
CUUSS n_{optional}	Cumulative uncorrected sum of squares
CUVAR n_{optional}	Cumulative variance
DIF n_{optional}	Span n difference: $x_t - x_{t-n}$
EWMA *number*	Exponentially weighted moving average of x with smoothing weight *number*, where $0 < number < 1$: $y_t = number\ \ x_t + (1 - number)y_{t-1}$. This operation is also called simple exponential smoothing.
EXP	Exponential function: $\exp(x)$
FDIF *d*	Fractional difference with difference order d where $0 < d < 0.5$
FLOOR	Largest integer less than or equal to x : $\text{floor}(x)$
FSUM *d*	Fractional summation with summation order d where $0 < d < 0.5$
HP_T *lambda*	Hodrick-Prescott Filter trend component where *lambda* is the nonnegative filter parameter
HP_C *lambda*	Hodrick-Prescott Filter cycle component where *lambda* is the nonnegative filter parameter
ILOGIT	Inverse logistic function: $\frac{\exp(x)}{1+\exp(x)}$
LAG n_{optional}	Value of the series n periods earlier: x_{t-n}
LEAD n_{optional}	Value of the series n periods later: x_{t+n}
LOG	Natural logarithm: $\log(x)$
LOGIT	Logistic function: $\log(\frac{x}{1-x})$
MAX *number*	Maximum of x and *number* : $\max(x, number)$
MIN *number*	Minimum of x and *number* : $\min(x, number)$
> *number*	Missing value if $x <= number$, else x
>= *number*	Missing value if $x < number$, else x
= *number*	Missing value if $x \neq number$, else x
∧= *number*	Missing value if $x = number$, else x
< *number*	Missing value if $x >= number$, else x
<= *number*	Missing value if $x > number$, else x
MOVAVE *n*	Backward moving average of n neighboring values: $\frac{1}{n}\sum_{j=0}^{n-1} x_{t-j}$
MOVAVE *window*	Backward weighted moving average of neighboring values: $(\sum_{j=1}^{n} w_j x_{t-n+j})/(\sum_{j=1}^{n} w_j)$
MOVCSS *window*	Backward moving corrected sum of squares
MOVGMEAN *window*	Backward moving geometric mean for *window* = number of periods, n: $(\prod_{j=1}^{n} x_{t-n+j})^{1/n}$ for *window* = weight list, w: $(\prod_{j=1}^{n} x_{t-n+j}^{w_j})^{1/\sum_{j=1}^{n} w_j}$
MOVMAX *n*	Backward moving maximum
MOVMED *n*	Backward moving median
MOVMIN *n*	Backward moving minimum

Table 16.2 *continued*

Syntax	Result
MOVPROD *window*	Backward moving product for *window* = number of periods, n: $\prod_{j=1}^{n} x_{t-n+j}$ for *window* = weight list, w: $(\prod_{j=1}^{n} x_{t-n+j}^{w_j})^{1/\sum_{j=1}^{n} w_j}$
MOVRANGE n	Backward moving range
MOVRANK n	Backward moving rank
MOVSTD *window*	Backward moving weighted standard deviation: $\sqrt{\frac{1}{n-1}\sum_{j=1}^{n} w_i (x_j - \overline{x}_w)^2}$
MOVSUM n	Backward moving sum
MOVTVALUE *window*	Backward moving t value
MOVUSS *window*	Backward moving uncorrected sum of squares
MOVVAR *window*	Backward moving variance
MISSONLY <MEAN>	Indicates that the following moving time window statistic operator should replace only missing values with the moving statistic and should leave nonmissing values unchanged. If the option MEAN is specified, then missing values are replaced by the overall mean of the series.
NEG	Changes the sign: $-x$
NOMISS	Indicates that the following moving time window statistic operator should not allow missing values
PCTDIF n	Percent difference of the current value and lag n
PCTSUM n	Percent summation of the current value and cumulative sum n-lag periods
RATIO n	Ratio of current value to lag n
RECIPROCAL	Reciprocal: $1/x$
REVERSE	Reverses the series: x_{N-t}
SCALE n_1 n_2	Scales the series between n_1 and n_2
SEQADD *sequence*	Adds sequence values to series
SEQDIV *sequence*	Divides the series by sequence values
SEQMINUS *sequence*	Subtracts sequence values to series
SEQMULT *sequence*	Multiplies the series by sequence values
SET (n_1 n_2)	Sets all values of n_1 to n_2
SETEMBEDDED (n_1 n_2)	Sets embedded values of n_1 to n_2
SETLEFT (n_1 n_2)	Sets beginning values of n_1 to n_2
SETMISS *number*	Replaces missing values in the series with the number specified
SETRIGHT (n_1 n_2)	Sets ending values of n_1 to n_2
SIGN	-1, 0, or 1 as x is < 0, equals 0, or > 0, respectively
SQRT	Square root: $\sqrt{x}$
SQUARE	Square: x^2
SUM	Cumulative sum: $\sum_{j=1}^{t} x_j$
SUM n	Cumulative sum of multiples of n-period lags: $x_t + x_{t-n} + x_{t-2n} + \ldots$

Table 16.2 *continued*

Syntax	Result
TRIM n	Sets x_t to missing a value if $t \leq n$ or $t \geq N - n + 1$
TRIMLEFT n	Sets x_t to missing a value if $t \leq n$
TRIMRIGHT n	Sets x_t to missing a value if $t \geq N - n + 1$

Moving Time Window Operators

Some operators compute statistics for a set of values within a moving time window; these are called *moving time window operators*. There are centered and backward versions of these operators.

The centered moving time window operators are CMOVAVE, CMOVCSS, CMOVGMEAN, CMOVMAX, CMOVMED, CMOVMIN, CMOVPROD, CMOVRANGE, CMOVRANK, CMOVSTD, CMOVSUM, CMOVTVALUE, CMOVUSS, and CMOVVAR. These operators compute statistics of the n values x_i for observations $t - (n + n \mod 2)/2 + 1 \leq i \leq t + (n - n \mod 2)/2$

The backward moving time window operators are MOVAVE, MOVCSS, MOVGMEAN, MOVMAX, MOVMED, MOVMIN, MOVPROD, MOVRANGE, MOVRANK, MOVSTD, MOVSUM, MOVTVALUE, MOVUSS, and MOVVAR. These operators compute statistics of the n values $x_t, x_{t-1}, \ldots, x_{t-n+1}$.

All the moving time window operators accept an argument n specifying the number of periods to include in the time window. For example, the following statement computes a five-period backward moving average of *X*.

```
convert x=y / transformout=( movave 5 );
```

In this example, the resulting transformation is

$$y_t = (x_t + x_{t-1} + x_{t-2} + x_{t-3} + x_{t-4})/5$$

The following statement computes a five-period centered moving average of *X*.

```
convert x=y / transformout=( cmovave 5 );
```

In this example, the resulting transformation is

$$y_t = (x_{t-2} + x_{t-1} + x_t + x_{t+1} + x_{t+2})/5$$

If the window with a centered moving time window operator is not an odd number, one more lead value than lag value is included in the time window. For example, the result of the CMOVAVE 4 operator is

$$y_t = (x_{t-1} + x_t + x_{t+1} + x_{t+2})/4$$

You can compute a forward moving time window operation by combining a backward moving time window operator with the REVERSE operator. For example, the following statement computes a five-period forward moving average of *X*.

```
convert x=y / transformout=( reverse movave 5 reverse );
```

In this example, the resulting transformation is

$$y_t = (x_t + x_{t+1} + x_{t+2} + x_{t+3} + x_{t+4})/5$$

Some of the moving time window operators enable you to specify a list of weight values to compute weighted statistics. These are CMOVAVE, CMOVCSS, CMOVGMEAN, CMOVPROD, CMOVSTD, CMOVTVALUE, CMOVUSS, CMOVVAR, MOVAVE, MOVCSS, MOVGMEAN, MOVPROD, MOVSTD, MOVTVALUE, MOVUSS, and MOVVAR.

To specify a weighted moving time window operator, enter the weight values in parentheses after the operator name. The window width n is equal to the number of weights that you specify; do not specify n.

For example, the following statement computes a weighted five-period centered moving average of *X*.

```
convert x=y / transformout=( cmovave( .1 .2 .4 .2 .1 ) );
```

In this example, the resulting transformation is

$$y_t = .1x_{t-2} + .2x_{t-1} + .4x_t + .2x_{t+1} + .1x_{t+2}$$

The weight values must be greater than zero. If the weights do not sum to 1, the weights specified are divided by their sum to produce the weights used to compute the statistic.

A complete time window is not available at the beginning of the series. For the centered operators a complete window is also not available at the end of the series. The computation of the moving time window operators is adjusted for these boundary conditions as follows.

For backward moving window operators, the width of the time window is shortened at the beginning of the series. For example, the results of the MOVSUM 3 operator are

$$\begin{aligned} y_1 &= x_1 \\ y_2 &= x_1 + x_2 \\ y_3 &= x_1 + x_2 + x_3 \\ y_4 &= x_2 + x_3 + x_4 \\ y_5 &= x_3 + x_4 + x_5 \\ &\dots \end{aligned}$$

For centered moving window operators, the width of the time window is shortened at the beginning and the end of the series due to unavailable observations. For example, the results of the CMOVSUM 5 operator are

$$
\begin{aligned}
y_1 &= x_1 + x_2 + x_3 \\
y_2 &= x_1 + x_2 + x_3 + x_4 \\
y_3 &= x_1 + x_2 + x_3 + x_4 + x_5 \\
y_4 &= x_2 + x_3 + x_4 + x_5 + x_6 \\
&\dots \\
y_{N-2} &= x_{N-4} + x_{N-3} + x_{N-2} + x_{N-1} + x_N \\
y_{N-1} &= x_{N-3} + x_{N-2} + x_{N-1} + x_N \\
y_N &= x_{N-2} + x_{N-1} + x_N
\end{aligned}
$$

For weighted moving time window operators, the weights for the unavailable or unused observations are ignored and the remaining weights renormalized to sum to 1.

Cumulative Statistics Operators

Some operators compute cumulative statistics for a set of current and previous values of the series. The cumulative statistics operators are CUAVE, CUCSS, CUMAX, CUMED, CUMIN, CURANGE, CUSTD, CUSUM, CUUSS, and CUVAR.

By default, the cumulative statistics operators compute the statistics from all previous values of the series, so that y_t is based on the set of values $x_t, x_{t-1}, \dots, x_1$. For example, the following statement computes y_t as the cumulative sum of nonmissing x_i values for $i \le t$.

```
convert x=y / transformout=( cusum );
```

You can specify a lag increment argument n for the cumulative statistics operators. In this case, the statistic is computed from the current and every nth previous value. When n is specified these operators compute statistics of the values $x_t, x_{t-n}, x_{t-2n}, \dots, x_{t-in}$ for $t - in > 0$.

For example, the following statement computes y_t as the cumulative sum of nonmissing x_i values for odd i when t is odd and for even i when t is even.

```
convert x=y / transformout=( cusum 2 );
```

The results of this example are

$$
\begin{aligned}
y_1 &= x_1 \\
y_2 &= x_2 \\
y_3 &= x_1 + x_3 \\
y_4 &= x_2 + x_4 \\
y_5 &= x_1 + x_3 + x_5 \\
y_6 &= x_2 + x_4 + x_6 \\
&\dots
\end{aligned}
$$

Missing Values

You can truncate the length of the result series by using the TRIM, TRIMLEFT, and TRIMRIGHT operators to set values to missing at the beginning or end of the series.

You can use these functions to trim the results of moving time window operators so that the result series contains only values computed from a full width time window. For example, the following statements compute a centered five-period moving average of *X*, and they set to missing values at the ends of the series that are averages of fewer than five values.

```
convert x=y / transformout=( cmovave 5 trim 2 );
```

Normally, the moving time window and cumulative statistics operators ignore missing values and compute their results for the nonmissing values. When preceded by the NOMISS operator, these functions produce a missing result if any value within the time window is missing.

The NOMISS operator does not perform any calculations, but serves to modify the operation of the moving time window operator that follows it. The NOMISS operator has no effect unless it is followed by a moving time window operator.

For example, the following statement computes a five-period moving average of the variable *X* but produces a missing value when any of the five values are missing.

```
convert x=y / transformout=( nomiss movave 5 );
```

The following statement computes the cumulative sum of the variable *X* but produces a missing value for all periods after the first missing X value.

```
convert x=y / transformout=( nomiss cusum );
```

Similar to the NOMISS operator, the MISSONLY operator does not perform any calculations (unless followed by the MEAN option), but it serves to modify the operation of the moving time window operator that follows it. When preceded by the MISSONLY operator, these moving time window operators replace any missing values with the moving statistic and leave nonmissing values unchanged.

For example, the following statement replaces any missing values of the variable *X* with an exponentially weighted moving average of the past values of *X* and leaves nonmissing values unchanged. The missing values are interpolated using the specified exponentially weighted moving average. (This is also called simple exponential smoothing.)

```
convert x=y / transformout=( missonly ewma 0.3 );
```

The following statement replaces any missing values of the variable *X* with the overall mean of *X*.

```
convert x=y / transformout=( missonly mean );
```

You can use the SETMISS operator to replace missing values with a specified number. For example, the following statement replaces any missing values of the variable *X* with the number 8.77.

```
convert x=y / transformout=( setmiss 8.77 );
```

Classical Decomposition Operators

If x_t is a seasonal time series with s observations per season, *classical decomposition* methods "break down" the time series into four components: trend, cycle, seasonal, and irregular components. The trend and cycle components are often combined to form the trend-cycle component. There are two basic forms of classical decomposition: multiplicative and additive, which are show below.

$$\begin{aligned} x_t &= TC_t S_t I_t \\ x_t &= TC_t + S_t + I_t \end{aligned}$$

where

TC_t	is the trend-cycle component
S_t	is the seasonal component or seasonal factors that are periodic with period s and with mean one (multiplicative) or zero (additive)
I_t	is the irregular or random component that is assumed to have mean one (multiplicative) or zero (additive)

For multiplicative decomposition, all of the x_t values should be positive.

The CD_TC operator computes the trend-cycle component for both the multiplicative and additive models. When s is odd, this operator computes an s-period centered moving average as follows:

$$TC_t = \sum_{k=-\lfloor s/2 \rfloor}^{\lfloor s/2 \rfloor} x_{t+k}/s$$

For example, in the case where s=5, the CD_TC s operator

```
convert x=tc / transformout=( cd_tc 5 );
```

is equivalent to the following CMOVAVE operator:

```
convert x=tc / transformout=( cmovave 5 trim 2 );
```

When s is even, the CD_TC s operator computes the average of two adjacent s-period centered moving averages as follows:

$$TC_t = \sum_{k=-\lfloor s/2 \rfloor}^{\lfloor s/2 \rfloor - 1} (x_{t+k} + x_{t+1+k})/2s$$

For example, in the case where s=12, the CD_TC s operator

```
convert x=tc / transformout=( cd_tc 12 );
```

is equivalent to the following CMOVAVE operator:

```
convert x=tc / transformout=(cmovave 12 movave 2 trim 6);
```

The CD_S and CDA_S operators compute the seasonal components for the multiplicative and additive models, respectively. First, the trend-cycle component is computed as shown previously. Second, the seasonal-irregular component is computed by $SI_t = x_t / TC_t$ for the multiplicative model and by $SI_t = x_t - TC_t$ for the additive model. The seasonal component is obtained by averaging the seasonal-irregular component for each season.

$$S_{k+js} = \sum_{t=k \bmod s} \frac{SI_t}{n/s}$$

where $0 \leq j \leq n/s$ and $1 \leq k \leq s$. The seasonal components are normalized to sum to one (multiplicative) or zero (additive).

The CD_I and CDA_I operators compute the irregular component for the multiplicative and additive models respectively. First, the seasonal component is computed as shown previously. Next, the irregular component is determined from the seasonal-irregular and seasonal components as appropriate.

$$\begin{aligned} I_t &= SI_t / S_t \\ I_t &= SI_t - S_t \end{aligned}$$

The CD_SA and CDA_SA operators compute the seasonally adjusted time series for the multiplicative and additive models, respectively. After decomposition, the original time series can be seasonally adjusted as appropriate.

$$\begin{aligned} \tilde{x}_t &= x_t / S_t = TC_t I_t \\ \tilde{x}_t &= x_t - S_t = TC_t + I_t \end{aligned}$$

The following statements compute all the multiplicative classical decomposition components for the variable X for s=12.

```
convert x=tc / transformout=( cd_tc 12 );
convert x=s  / transformout=( cd_s  12 );
convert x=i  / transformout=( cd_i  12 );
convert x=sa / transformout=( cd_sa 12 );
```

The following statements compute all the additive classical decomposition components for the variable X for s=4.

```
convert x=tc / transformout=( cd_tc  4 );
convert x=s  / transformout=( cda_s  4 );
convert x=i  / transformout=( cda_i  4 );
convert x=sa / transformout=( cda_sa 4 );
```

The X12 and X11 procedures provides other methods for seasonal decomposition. See Chapter 44, "The X12 Procedure," and Chapter 43, "The X11 Procedure."

Fractional Operators

For fractional operators, the parameter, *d*, represents the order of fractional differencing. Fractional summation is the inverse operation of fractional differencing.

Examples of Usage

Suppose that *X* is a fractionally integrated time series variable of order d=0.25. Fractionally differencing *X* forms a time series variable *Y* which is not integrated.

```
convert x=y / transformout=(fdif 0.25);
```

Suppose that *Z* is a non-integrated time series variable. Fractionally summing *Z* forms a time series *W* which is fractionally integrated of order $d = 0.25$.

```
convert z=w / transformout=(fsum 0.25);
```

Moving Rank Operators

For the rank operators, the ranks are computed based on the current value with respect to the cumulative, centered, or moving window values. If the current value is missing, the transformed current value is set to missing. If the NOMISS option was previously specified and if any missing values are present in the moving window, the transformed current value is set to missing. Otherwise, redundant values from the moving window are removed and the rank of the current value is computed among the unique values of the moving window.

Examples of Usage

The trades of a particular security are recorded for each weekday in a variable named PRICE. Given the historical daily trades, the ranking of the price of this security for each trading day, considering its entire past history, can be computed as follows:

```
convert price=history / transformout=( curank );
```

The ranking of the price of this security for each trading day considering the previous week's history can be computed as follows:

```
convert price=lastweek / transformout=( movrank 5 );
```

The ranking of the price of this security for each trading day considering the previous two week's history can be computed as follows:

```
convert price=twoweek / transformout=( movrank 10 );
```

Moving Product and Geometric Mean Operators

For the product and geometric mean operators, the current transformed value is computed based on the (weighted) product of the cumulative, centered, or moving window values. If missing values are present in the moving window and the NOMISS operator is previously specified, the current transformed value is set to missing. Otherwise, the current transformed value is set to the product of the nonmissing values within the moving window. If a geometric mean operator is specified for a window of size *n*, the *n*th root of the product is taken. In cases where weights are specified explicitly, both the product and geometric mean operators normalize these exponents so that they sum to one.

Examples of Usage

The interest rates for a savings account are recorded for each month in the data set variable RATES. The cumulative interest rate for each month considering the entire account past history can be computed as follows:

```
convert rates=history / transformout=( + 1 cuprod - 1);
```

The interest rate for each quarter considering the previous quarter's history can be computed as follows:

```
convert rates=lastqtr / transformout=( + 1 movprod 3 - 1);
```

The average interest rate for the previous quarter's history can be computed as follows:

```
convert rates=lastqtr / transformout=( + 1 movprod (1 1 1) - 1);
```

Sequence Operators

For the sequence operators, the sequence values are used to compute the transformed values from the original values in a sequential fashion. You can add to or subtract from the original series or you can multiply or divide by the sequence values. The first sequence value is applied to the first observation of the series, the second sequence value is applied to the second observation of the series, and so on until the end of the sequence is reached. At this point, the first sequence value is applied to the next observation of the series and the second sequence value on the next observation and so on.

Let $v_1, \ldots, v_m$ be the sequence values and let $x_t, t = 1, \ldots N$, be the original time series. The transformed series, y_t, is computed as follows:

$$
\begin{aligned}
y_1 &= x_1 \; op \; v_1 \\
y_2 &= x_2 \; op \; v_2 \\
&\ldots \\
y_m &= x_m \; op \; v_m \\
y_{m+1} &= x_{m+1} \; op \; v_1 \\
y_{m+2} &= x_{m+2} \; op \; v_2 \\
&\ldots \\
y_{2m} &= x_{2m} \; op \; v_m \\
y_{2m+1} &= x_{2m+1} \; op \; v1 \\
y_{2m+2} &= x_{2m+2} \; op \; v2 \\
&\ldots
\end{aligned}
$$

where $op = +, -, *,$ or $/$.

Examples of Usage

The multiplicative seasonal indices are 0.9, 1.2. 0.8, and 1.1 for the four quarters. Let SEASADJ be a quarterly time series variable that has been seasonally adjusted in a multiplicative fashion. To restore the seasonality to SEASADJ use the following transformation:

```
convert seasadj=seasonal /
        transformout=(seqmult (0.9 1.2 0.8 1.1));
```

The additive seasonal indices are 4.4, -1.1, -2.1, and -1.2 for the four quarters. Let SEASADJ be a quarterly time series variable that has been seasonally adjusted in additive fashion. To restore the seasonality to SEASADJ use the following transformation:

```
convert seasadj=seasonal /
        transformout=(seqadd (4.4 -1.1 -2.1 -1.2));
```

Set Operators

For the set operators, the first parameter, n_1, represents the value to be replaced and the second parameter, n_2, represents the replacement value. The replacement can be localized to the beginning, middle, or end of the series.

Examples of Usage

Suppose that a store opened recently and that the sales history is stored in a database that does not recognize missing values. Even though demand may have existed prior to the stores opening, this database assigns the value of zero. Modeling the sales history may be problematic because the sales history is mostly zero. To compensate for this deficiency, the leading zero values should be set to missing with the remaining zero values unchanged (representing no demand).

```
convert sales=demand / transformout=(setleft (0 .));
```

Likewise, suppose a store is closed recently. The demand might still be present; hence, a recorded value of zero does not accurately reflect actual demand.

```
convert sales=demand / transformout=(setright (0 .));
```

Scale Operator

For the scale operator, the first parameter, n_1, represents the value associated with the minimum value (x_{min}) and the second parameter, n_2, represents the value associated with the maximum value (x_{max}) of the original series (x_t). The scale operator rescales the original data to be between the parameters n_1 and n_2 as follows:

$$y_t = ((n_2 - n_1)/(x_{max} - x_{min}))(x_t - x_{min}) + n_1$$

Examples of Usage

Suppose that two new product sales histories are stored in variables *X* and *Y* and you wish to determine their adoption rates. In order to compare their adoption histories the variables must be scaled for comparison.

```
convert x=w / transformout=(scale 0 1);
convert y=z / transformout=(scale 0 1);
```

Adjust Operator

For the moving summation and product window operators, the window widths at the beginning and end of the series are smaller than those in the middle of the series. Likewise, if there are embedded missing values, the window width is smaller than specified. When preceded by the ADJUST operator, the moving summation (MOVSUM CMOVSUM) and moving product operators (MOVPROD CMOVPROD) are adjusted by the window width.

For example, suppose the variable *X* has 10 values and the moving summation operator of width 3 is applied to *X* to create the variable *Y* with window width adjustment and the variable *Z* without adjustment.

```
convert x=y / transformout=(adjust movsum 3);
convert x=z / transformout=(movsum 3);
```

The above transformations result in the following relationship between *Y* and *Z*: $y_1 = 3z_1$, $y_2 = \frac{3}{2}z_2$, $y_t = z_t$ for $t > 2$ because the first two window widths are smaller than 3.

For example, suppose the variable *X* has 10 values and the moving multiplicative operator of width 3 is applied to *X* to create the variable *Y* with window width adjustment and the variable *Z* without adjustment.

```
convert x=y / transformout=(adjust movprod 3);
convert x=z / transformout=(movprod 3);
```

The above transformation result in the following: $y_1 = z_1^3$, $y_2 = z_2^{3/2}$, $y_t = z_t$ for $t > 2$ because the first two window widths are smaller than 3.

Moving T-Value Operators

The moving *t*-value operators (CUTVALUE, MOVTVALUE, CMOVTVALUE) compute the *t*-value of the cumulative series or moving window. They can be viewed as combinations of the moving average (CUAVE, MOVAVE, CMOVAVE) and the moving standard deviation (CUSTD, MOVSTD, CMOVSTD), respectively.

Percent Operators

The percentage operators compute the percent summation and the percent difference of the current value and the $\mathrm{lag}(n)$. The percent summation operator (PCTSUM) computes $y_t = 100x_t/\mathrm{cusum}(x_{t-n})$. If any of the values of the preceding equation are missing or the cumulative summation is zero, the result is set to missing. The percent difference operator (PCTDIF) computes $y_t = 100(x_t - x_{t-n})/x_{t-n}$. If any of the values of the preceding equation are missing or the lag value is zero, the result is set to missing.

For example, suppose variable *X* contains the series. The percent summation of lag 4 is applied to *X* to create the variable *Y*. The percent difference of lag 4 is applied to *X* to create the variable *Z*.

```
convert x=y / transformout=(pctsum 4);
convert x=z / transformout=(pctdif 4);
```

Ratio Operators

The ratio operator computes the ratio of the current value and the $\mathrm{lag}(n)$ value. The ratio operator (RATIO) computes $y_t = x_t/x_{t-n}$. If any of the values of the preceding equation are missing or the lag value is zero, the result is set to missing.

For example, suppose variable *X* contains the series. The ratio of the current value and the lag 4 value of *X* assigned to the variable *Y*. The percent ratio of the current value and lag 4 value of *X* is assigned to the variable *Z*.

```
convert x=y / transformout=(ratio 4);
convert x=z / transformout=(ratio 4 * 100);
```

OUT= Data Set

The OUT= output data set contains the following variables:

- the BY variables, if any
- an ID variable that identifies the time period for each output observation
- the result variables
- if no frequency conversion is performed (so that there is one output observation corresponding to each input observation), all the other variables in the input data set are copied to the output data set

The ID variable in the output data set is named as follows:

- If an ID statement is used, the new ID variable has the same name as the variable used in the ID statement.
- If no ID statement is used, but the FROM= option is used, then the name of the ID variable is either DATE or DATETIME, depending on whether the TO= option indicates SAS date or SAS datetime values.
- If neither an ID statement nor the TO= option is used, the ID variable is named TIME.

OUTEST= Data Set

The OUTEST= data set contains the coefficients of the spline curves fit to the input series. The OUTEST= data set is of interest if you want to verify the interpolating curve PROC EXPAND uses, or if you want to use this function in another context, (for example, in a SAS/IML program).

The OUTEST= data set contains the following variables:

- the BY variables, if any
- VARNAME, a character variable containing the name of the input variable to which the coefficients apply
- METHOD, a character variable containing the value of the METHOD= option used to fit the series
- OBSERVED, a character variable containing the first letter of the OBSERVED= option name for the input series
- the ID variable that contains the lower breakpoint (or "knot") of the spline segment to which the coefficients apply. The ID variable has the same name as the variable used in the ID statement. If an ID statement is not used, but the FROM= option is used, then the name of the ID variable is DATE or DATETIME, depending on whether the FROM= option indicates SAS date or SAS datetime values. If neither an ID statement nor the FROM= option is used, the ID variable is named TIME.
- CONSTANT, the constant coefficient for the spline segment
- LINEAR, the linear coefficient for the spline segment
- QUAD, the quadratic coefficient for the spline segment
- CUBIC, the cubic coefficient for the spline segment

For each BY group, the OUTEST= data set contains observations for each polynomial segment of the spline curve fit to each input series. To obtain the observations defining the spline curve used for a series, select the observations where the value of VARNAME equals the name of the series.

The observations for a series in the OUTEST= data set encode the spline function fit to the series as follows. Let a_i, b_i, c_i, and d_i be the values of the variables CUBIC, QUAD, LINEAR, and CONSTANT, respectively, for the ith observation for the series. Let x_i be the value of the ID variable for the ith observation for the

series. Let n be the number of observations in the OUTEST= data set for the series. The value of the spline function evaluated at a point x is

$$f(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i$$

where the segment number i is selected as follows:

$$i = \begin{cases} i & x_i \le x < x_{i+1}, 1 \le i < n \\ 1 & x < x_1 \\ n & x \ge x_n \end{cases}$$

In other words, if x is between the first and last ID values ($x_1 \le x < x_n$), use the observation from the OUTEST= data set with the largest ID value less than or equal to x. If x is less than the first ID value x_1, then $i = 1$. If x is greater than or equal to the last ID value ($x \ge x_n$), then $i = n$.

For METHOD=JOIN, the curve is a linear spline, and the values of CUBIC and QUAD are 0. For METHOD=STEP, the curve is a constant spline, and the values of CUBIC, QUAD, and LINEAR are 0. For METHOD=AGGREGATE, no coefficients are output.

ODS Graphics

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*).

Before you create graphs, ODS Graphics must be enabled (for example, with the ODS GRAPHICS ON statement). For more information about enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" in that chapter.

The overall appearance of graphs is controlled by ODS styles. Styles and other aspects of using ODS Graphics are discussed in the section "A Primer on ODS Statistical Graphics" in that chapter.

This section describes the use of ODS for creating graphics with the EXPAND procedure. To request these graphs, you must specify the PLOTS= option in the PROC EXPAND statement.

ODS Graph Names

PROC EXPAND assigns a name to each graph it creates using ODS. You can use these names to reference the graphs when using ODS. The names are listed in Table 16.3.

Table 16.3 ODS Graphics Produced by PROC EXPAND

ODS Graph Name	Plot Description	PLOTS= Options
ConvertedSeriesPlot	Converted Series Plot	CONVERTED OUTPUT ALL
CrossInputSeriesPlot	Cross Input Series Plot	CROSSINPUT
CrossOutputSeriesPlot	Cross Output Series Plot	CROSSOUTPUT
InputSeriesPlot	Input Series Plot	INPUT JOINTINPUT ALL
JointInputSeriesPlot	Joint Input Series Plot	JOINTINPUT
JointOutputSeriesPlot	Joint Output Series Plot	JOINTOUTPUT

Table 16.3 (continued)

ODS Graph Name	Plot Description	PLOTS= Options
OutputSeriesPlot	Output Series Plot	SERIES|OUTPUT
TransformedInputSeriesPlot	Transformed Input Series Plot	TRANSFORMIN OUTPUT ALL
TransformedOutputSeriesPlot	Transformed Output Series Plot	TRANSFORMOUT OUTPUT ALL

PLOTS= Option Details

Some plots are produced for a series only if the relevant options are also specified. For example, if PLOTS=TRANSFORMIN is specified, then the TRANSFORMIN plot is not produced for a variable unless the TRANSFORMIN= option is specified in a CONVERT statement for that variable. The PLOTS=TRANSFORMIN option plots the series after the input transformation (TRANSFORMIN= option) is applied.

The PLOTS=CONVERTED option plots the series after the input transformation (TRANSFORMIN= option) is applied and after frequency conversion (METHOD= option). If there is no frequency conversion for an output variable, the converted series plot is not produced.

The PLOTS=TRANSFORMOUT option plots the series after the output transformation (TRANSFORMOUT= option) is applied. If the TRANFORMOUT= option is not specified in the CONVERT statement for an output variable, the output transformation plot is not produced.

The PLOTS=OUTPUT option plots the series after it has undergone input transformation (TRANSFORMIN= option), frequency conversion (METHOD= option), and output transformation (TRANSFORMOUT= option) if these CONVERT statement options were specified.

Cross and Joint Plots

The PLOTS= option values CROSSINPUT and CROSSOUTPUT produce graphs that overlay plots of two series by using two Y axes and with each of the two plots shown at a separate scale. These plots are called cross plots.

The PLOTS= option values JOINTINPUT and JOINTOUTPUT produce graphs that overlay plots of two series by using a single Y axis and with both of the plots shown on the same scale. These plots are called joint plots. The joint graphics options (PLOTS=JOINTINPUT or PLOTS=JOINTOUTPUT) plot the (input or converted) series and the transformed series on the same scale; therefore if the transformation changes, the range of the series these plots might be hard to visualize.

The PLOTS=CROSSINPUT option plots both the input series and the series after the input transformation (TRANSFORMIN= option) is applied. The left vertical axis refers to the input series, while the right vertical axis refers to the series after the transformation. If the TRANFORMIN= option is not specified in the CONVERT statement for an output variable, then the cross input plot is not produced for that variable.

The PLOTS=JOINTINPUT option jointly plots both the input series and the series after the input transformation (TRANSFORMIN= option) is applied. If the TRANSFORMIN= option is not specified in the CONVERT statement for an output variable, then the joint input plot is not produced for that variable.

The PLOTS=CROSSOUTPUT option plots both the converted series and the converted series after the output transformation (TRANSFORMOUT= option) is applied. The left vertical axis refers to the input series,

while the right vertical axis refers to the series after the transformation. If the TRANSFORMOUT= option is not specified in the CONVERT statement for an output variable, then the cross output plot is not produced for that variable.

The PLOTS=JOINTOUTPUT option jointly plots both the converted series and the converted series after the output transformation (TRANSFORMOUT= option) is applied. If the TRANSFORMOUT= option is not specified in the CONVERT statement for an output variable, then the joint output plot is not produced for that variable.

Requesting All Plots

The PLOTS=ALL option is a convenient way to specify all the plots except the OUTPUT plots and the joint and cross plots. The option PLOTS=(ALL OUTPUT JOINTINPUT JOINTOUTPUT CROSSINPUT CROSSOUTPUT) requests that all possible plots be produced.

Examples: EXPAND Procedure

Example 16.1: Combining Monthly and Quarterly Data

This example combines monthly and quarterly data sets by interpolating monthly values for the quarterly series. The series are extracted from two small sample data sets stored in the SASHELP library. These data sets were contributed by Citicorp Data Base services and contain selected U.S. macro economic series.

The quarterly series gross domestic product (GDP) and implicit price deflator (GD) are extracted from SASHELP.CITIQTR. The monthly series industrial production index (IP) and unemployment rate (LHUR) are extracted from SASHELP.CITIMON. Only observations for the years 1990 and 1991 are selected. PROC EXPAND is then used to interpolate monthly estimates for the quarterly series, and the interpolated series are merged with the monthly data.

The following statements extract and print the quarterly data, shown in Output 16.1.1.

```
data qtrly;
   set sashelp.citiqtr;
   where date >= '1jan1990'd &
         date <  '1jan1992'd ;
   keep date gdp gd;
run;

title "Quarterly Data";
proc print data=qtrly;
run;
```

Output 16.1.1 Quarterly Data Set

Quarterly Data

Obs	DATE	GD	GDP
1	1990:1	111.100	5422.40
2	1990:2	112.300	5504.70
3	1990:3	113.600	5570.50
4	1990:4	114.500	5557.50
5	1991:1	115.900	5589.00
6	1991:2	116.800	5652.60
7	1991:3	117.400	5709.20
8	1991:4	.	5736.60

The following statements extract and print the monthly data, shown in Output 16.1.2.

```
data monthly;
   set sashelp.citimon;
   where date >= '1jan1990'd &
         date <  '1jan1992'd ;
   keep date ip lhur;
run;

title "Monthly Data";
proc print data=monthly;
run;
```

Output 16.1.2 Monthly Data Set

Monthly Data

Obs	DATE	IP	LHUR
1	JAN1990	107.500	5.30000
2	FEB1990	108.500	5.30000
3	MAR1990	108.900	5.20000
4	APR1990	108.800	5.40000
5	MAY1990	109.400	5.30000
6	JUN1990	110.100	5.20000
7	JUL1990	110.400	5.40000
8	AUG1990	110.500	5.60000
9	SEP1990	110.600	5.70000
10	OCT1990	109.900	5.80000
11	NOV1990	108.300	6.00000
12	DEC1990	107.200	6.10000
13	JAN1991	106.600	6.20000
14	FEB1991	105.700	6.50000
15	MAR1991	105.000	6.70000
16	APR1991	105.500	6.60000
17	MAY1991	106.400	6.80000
18	JUN1991	107.300	6.90000
19	JUL1991	108.100	6.80000
20	AUG1991	108.000	6.80000
21	SEP1991	108.400	6.80000
22	OCT1991	108.200	6.90000
23	NOV1991	108.000	6.90000
24	DEC1991	107.800	7.10000

The following statements interpolate monthly estimates for the quarterly series and merge the interpolated series with the monthly data. The resulting combined data set is then printed, as shown in Output 16.1.3.

```
proc expand data=qtrly out=temp from=qtr to=month;
   convert gdp gd / observed=average;
   id date;
run;

data combined;
   merge monthly temp;
   by date;
run;

title "Combined Data Set";
proc print data=combined;
run;
```

Output 16.1.3 Combined Data Set

Combined Data Set

Obs	DATE	IP	LHUR	GDP	GD
1	JAN1990	107.500	5.30000	5409.69	110.879
2	FEB1990	108.500	5.30000	5417.67	111.048
3	MAR1990	108.900	5.20000	5439.39	111.367
4	APR1990	108.800	5.40000	5470.58	111.802
5	MAY1990	109.400	5.30000	5505.35	112.297
6	JUN1990	110.100	5.20000	5538.14	112.801
7	JUL1990	110.400	5.40000	5563.38	113.264
8	AUG1990	110.500	5.60000	5575.69	113.641
9	SEP1990	110.600	5.70000	5572.49	113.905
10	OCT1990	109.900	5.80000	5561.64	114.139
11	NOV1990	108.300	6.00000	5553.83	114.451
12	DEC1990	107.200	6.10000	5556.92	114.909
13	JAN1991	106.600	6.20000	5570.06	115.452
14	FEB1991	105.700	6.50000	5588.18	115.937
15	MAR1991	105.000	6.70000	5608.68	116.314
16	APR1991	105.500	6.60000	5630.81	116.600
17	MAY1991	106.400	6.80000	5652.92	116.812
18	JUN1991	107.300	6.90000	5674.06	116.988
19	JUL1991	108.100	6.80000	5693.43	117.164
20	AUG1991	108.000	6.80000	5710.54	117.380
21	SEP1991	108.400	6.80000	5724.11	117.665
22	OCT1991	108.200	6.90000	5733.65	.
23	NOV1991	108.000	6.90000	5738.46	.
24	DEC1991	107.800	7.10000	5737.75	.

Example 16.2: Illustration of ODS Graphics

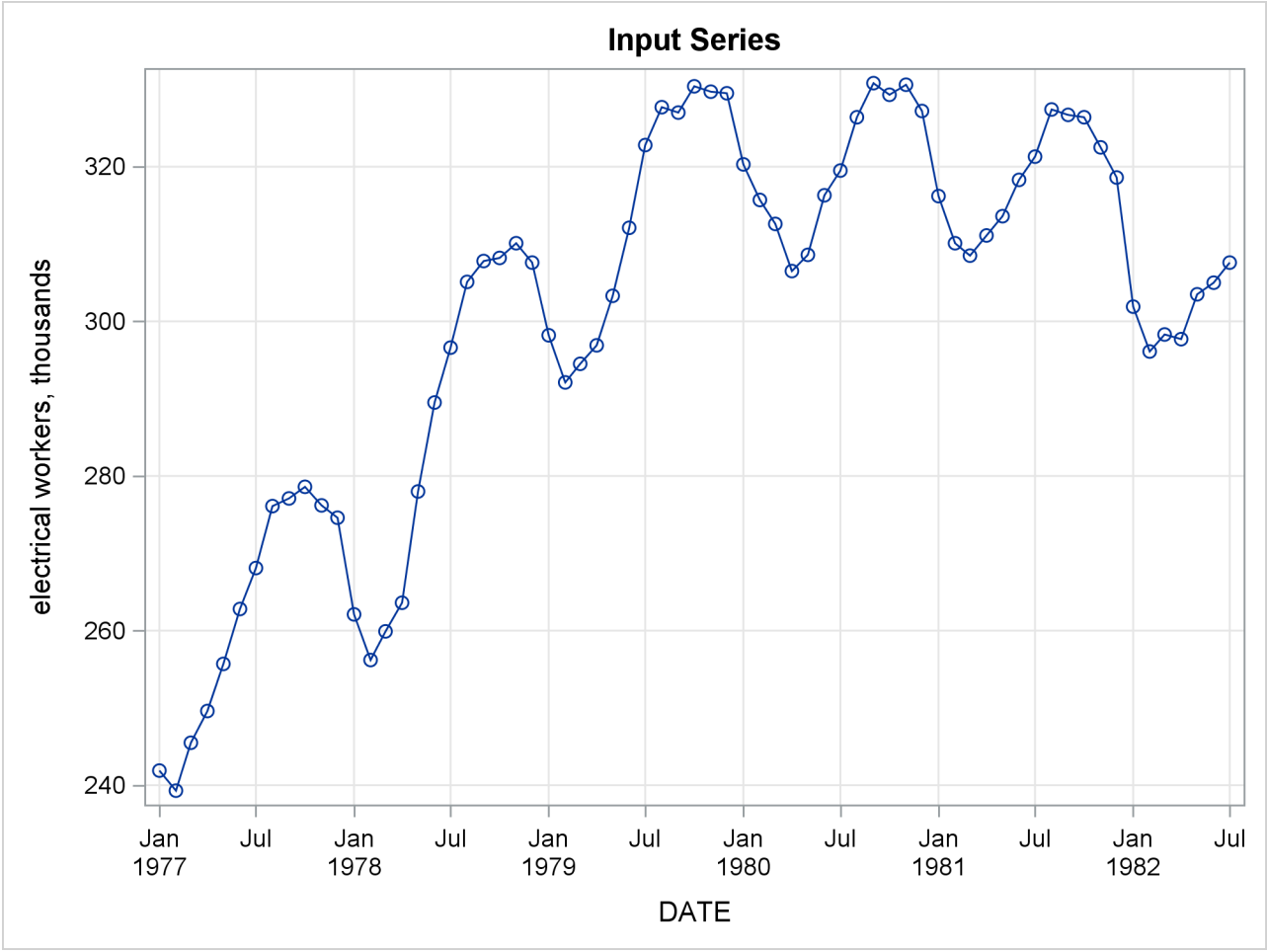
This example illustrates the use of ODS graphics with PROC EXPAND.

The graphical displays are requested by specifying the PLOTS= option in the PROC EXPAND statement. For information about the graphics available in the EXPAND procedure, see the section "ODS Graphics" on page 920.

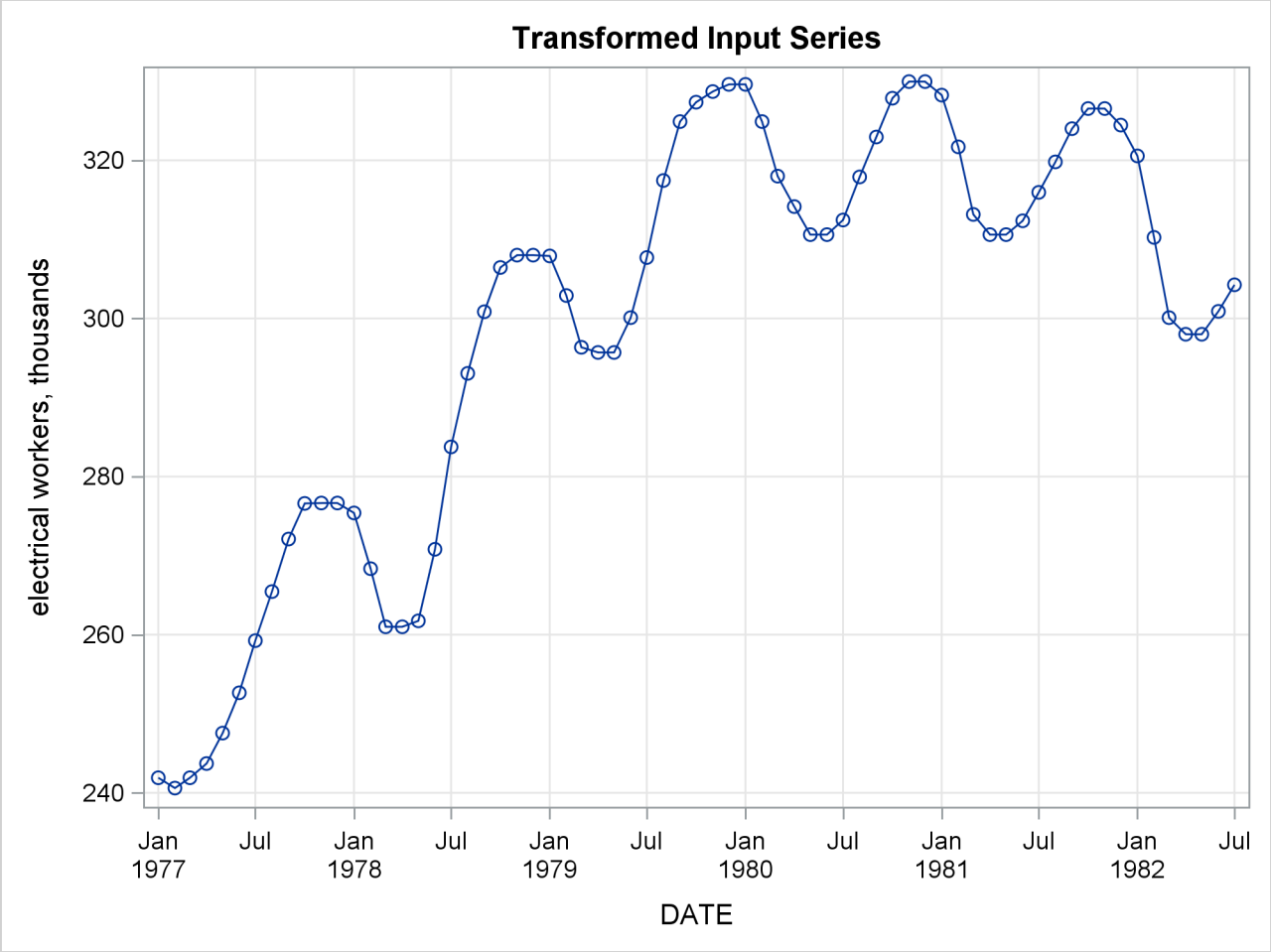
The following statements utilize the SASHELP.WORKERS data set to convert the time series of electrical workers from monthly to quarterly frequency and display ODS graphics plots. The PLOTS=ALL option is specified to request the plots of the input series, the transformed input series, the converted series, and the transformed output series. Figure 16.2.1 through Figure 16.2.4 show these plots.

```
proc expand data=sashelp.workers out=out
            from=month to=qtr
            plots=all;
   id date;
   convert electric=eout / method=spline
                           transformin=(movmed 4)
                           transformout=(movave 3);
run;
```

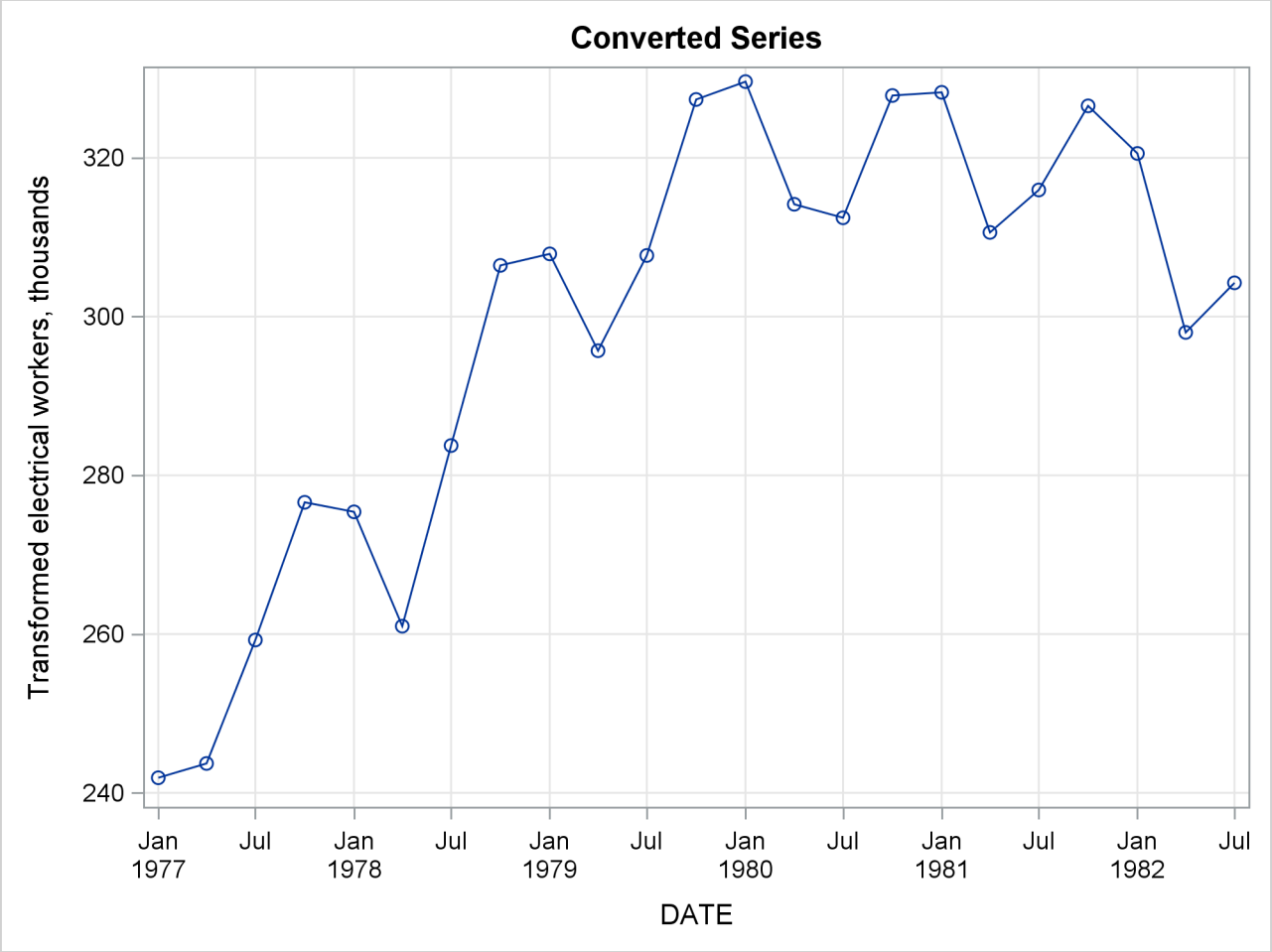
Output 16.2.1 Input Series Plot



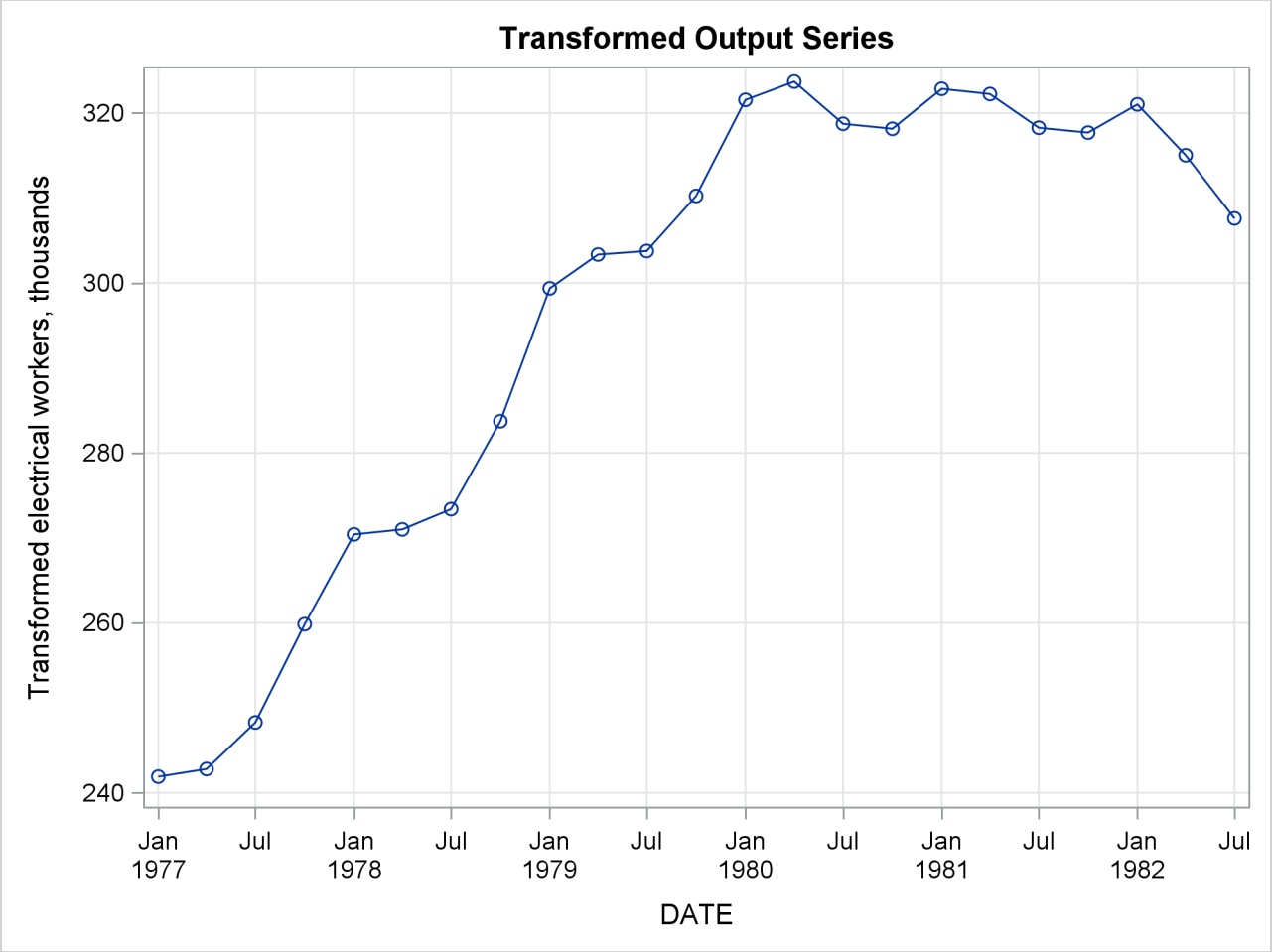
Output 16.2.2 Transformed Input Series Plot—Four-Period Moving Median



Output 16.2.3 Converted Plot of Transformed Input Series



Output 16.2.4 Transformed Output Series Plot—Three-Period Moving Average



Example 16.3: Interpolating Irregular Observations

This example shows the interpolation of a series of values measured at irregular points in time. The data are hypothetical. Assume that a series of randomly timed quality control inspections are made and defect rates for a process are measured. The problem is to produce two reports: estimates of monthly average defect rates for the months within the period covered by the samples, and a plot of the interpolated defect rate curve over time.

The following statements read and print the input data, as shown in Output 16.3.1.

```
data samples;
  input date : date9. defects @@;
  label defects = "Defects per 1000 Units";
  format date date9.;
datalines;
13jan1992    55    27jan1992    73    19feb1992    84    8mar1992    69

   ... more lines ...


title "Sampled Defect Rates";
proc print data=samples;
run;
```

Output 16.3.1 Measured Defect Rates

Sampled Defect Rates

Obs	date	defects
1	13JAN1992	55
2	27JAN1992	73
3	19FEB1992	84
4	08MAR1992	69
5	27MAR1992	66
6	05APR1992	77
7	29APR1992	63
8	11MAY1992	81
9	25MAY1992	89
10	07JUN1992	94
11	23JUN1992	105
12	11JUL1992	97
13	15AUG1992	112
14	29AUG1992	89
15	10SEP1992	77
16	27SEP1992	82

To compute the monthly estimates, use PROC EXPAND with the TO=MONTH option and specify OBSERVED=(BEGINNING,AVERAGE). The following statements interpolate the monthly estimates.

```
proc expand data=samples
            out=monthly
            to=month
            plots=(input output);
  id date;
  convert defects / observed=(beginning,average);
run;
```

The following PROC PRINT step prints the results, as shown in Output 16.3.2.

```
title "Estimated Monthly Average Defect Rates";
proc print data=monthly;
run;
```

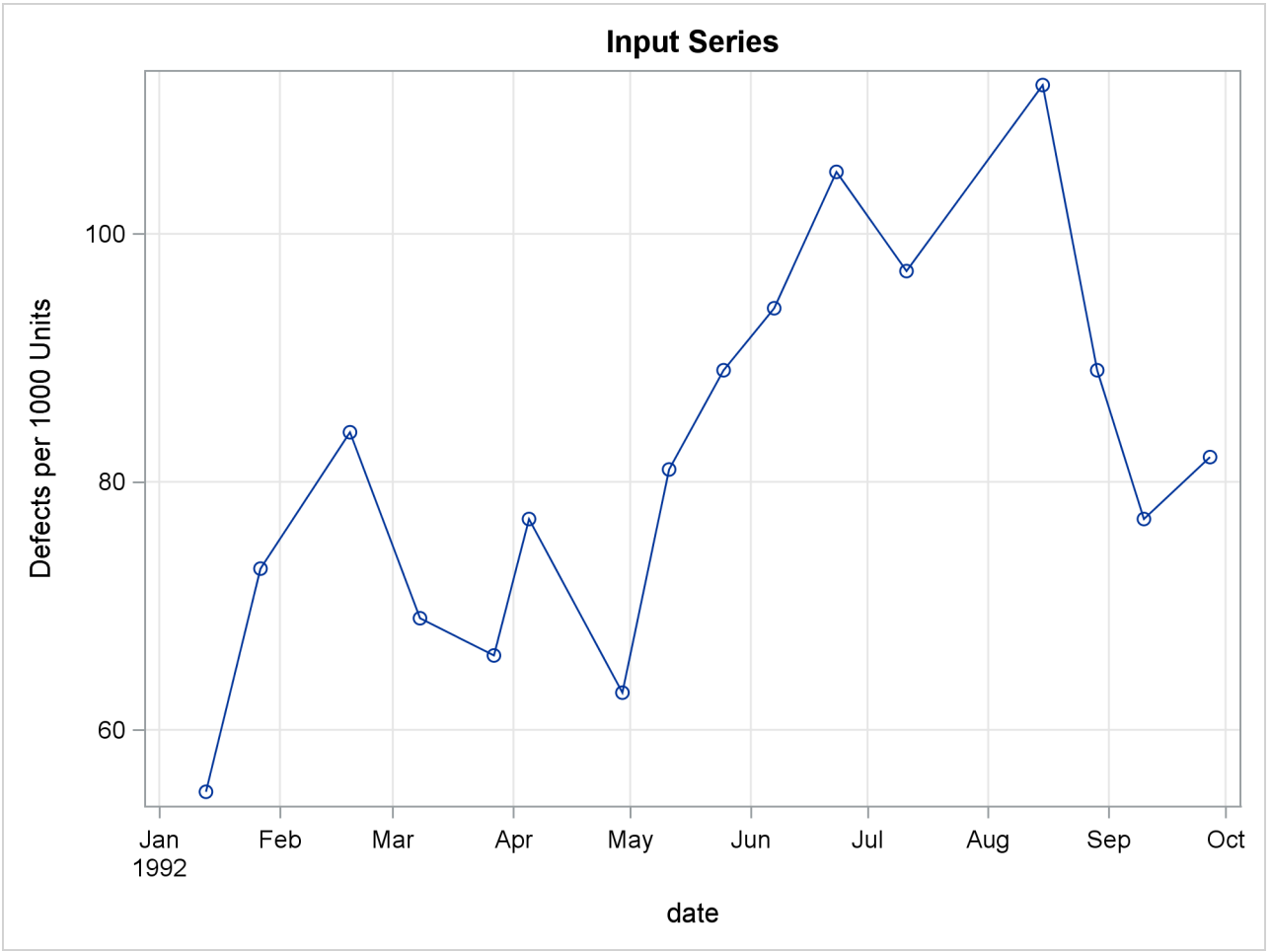
Output 16.3.2 Monthly Average Estimates

Estimated Monthly Average Defect Rates

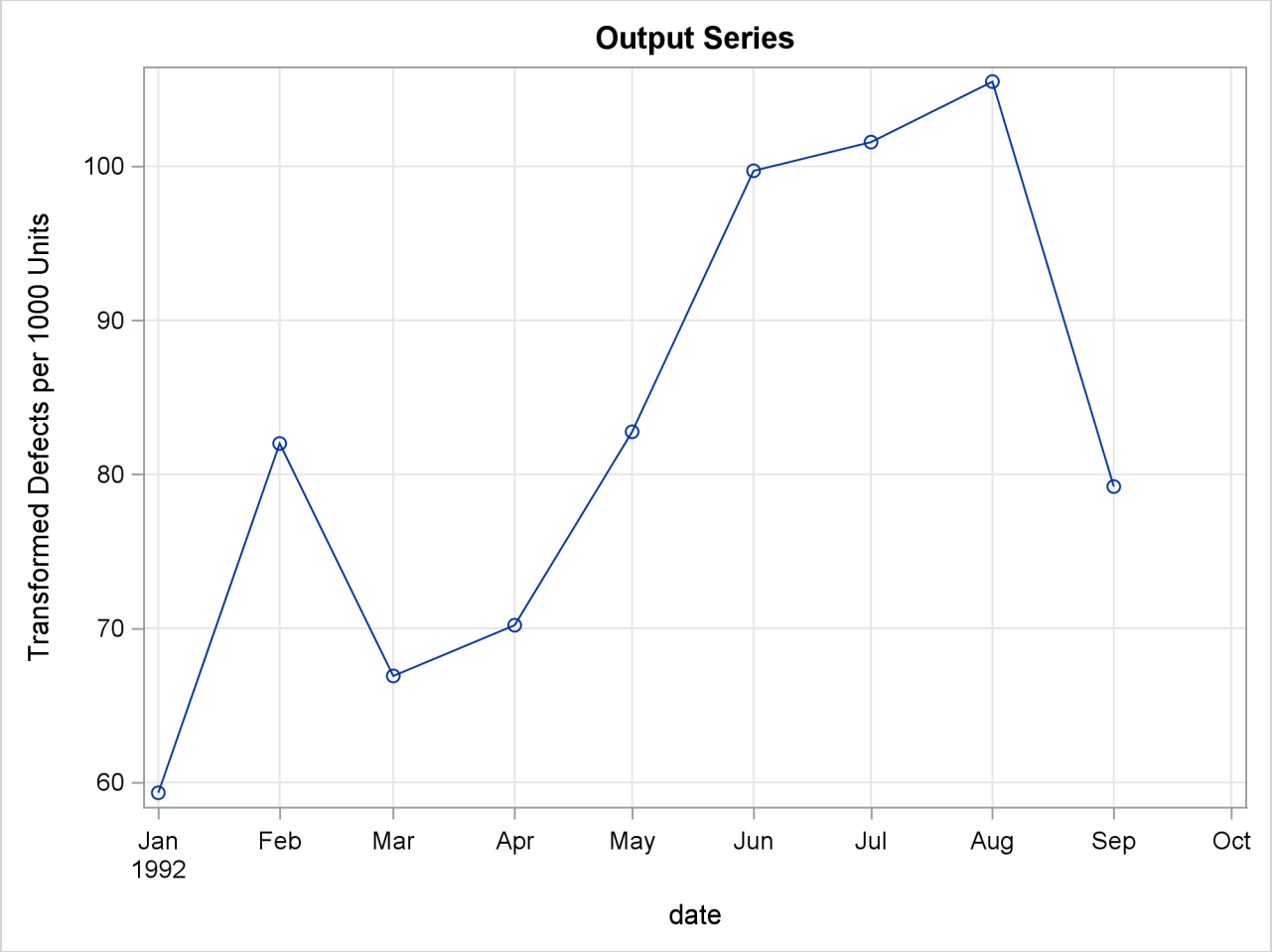
Obs	date	defects
1	JAN1992	59.323
2	FEB1992	82.000
3	MAR1992	66.909
4	APR1992	70.205
5	MAY1992	82.762
6	JUN1992	99.701
7	JUL1992	101.564
8	AUG1992	105.491
9	SEP1992	79.206

The plots produced by PROC EXPAND are shown in Output 16.3.3.

Output 16.3.3 Interpolated Defects Rate Curve



Output 16.3.3 *continued*



Example 16.4: Using Transformations

This example shows the use of PROC EXPAND to perform various transformations of time series. The following statements read in monthly values for a variable X.

```
data test;
   input year qtr x;
   date = yyq( year, qtr );
   format date yyqc.;
datalines;
1989 3 5238
1989 4 5289
1990 1 5375
1990 2 5443
1990 3 5514
1990 4 5527
1991 1 5557
1991 2 5615
;
```

The following statements use PROC EXPAND to compute lags and leads and a 3-period moving average of the X series.

```
proc expand data=test out=out method=none;
   id date;
   convert x = x_lag2   / transformout=(lag 2);
   convert x = x_lag1   / transformout=(lag 1);
   convert x;
   convert x = x_lead1  / transformout=(lead 1);
   convert x = x_lead2  / transformout=(lead 2);
   convert x = x_movave / transformout=(movave 3);
run;

title "Transformed Series";
proc print data=out;
run;
```

Because there are no missing values to interpolate and no frequency conversion, the METHOD=NONE option is used to prevent PROC EXPAND from performing unnecessary computations. Because no frequency conversion is done, all variables in the input data set are copied to the output data set. The CONVERT X; statement is included to control the position of X in the output data set. This statement can be omitted, in which case X is copied to the output data set following the new variables computed by PROC EXPAND.

The results are shown in Output 16.4.1.

Output 16.4.1 Output Data Set with Transformed Variables

Transformed Series

Obs	date	x_lag2	x_lag1	x	x_lead1	x_lead2	x_movave	year	qtr
1	1989:3	.	.	5238	5289	5375	5238.00	1989	3
2	1989:4	.	5238	5289	5375	5443	5263.50	1989	4
3	1990:1	5238	5289	5375	5443	5514	5300.67	1990	1
4	1990:2	5289	5375	5443	5514	5527	5369.00	1990	2
5	1990:3	5375	5443	5514	5527	5557	5444.00	1990	3
6	1990:4	5443	5514	5527	5557	5615	5494.67	1990	4
7	1991:1	5514	5527	5557	5615	.	5532.67	1991	1
8	1991:2	5527	5557	5615	.	.	5566.33	1991	2

References

De Boor, C. (1978). *A Practical Guide to Splines*. New York: Springer-Verlag.

Hodrick, R. J., and Prescott, E. C. (1980). "Post-war U.S. Business Cycles: An Empirical Investigation." Discussion Paper 451, Carnegie Mellon University.

Levenbach, H., and Cleary, J. P. (1984). *The Modern Forecaster*. Belmont, CA: Lifetime Learning Publications.

Makridakis, S. G., and Wheelwright, S. C. (1978). *Interactive Forecasting: Univariate and Multivariate Methods*. 2nd ed. San Francisco: Holden-Day.

Wheelwright, S. C., and Makridakis, S. G. (1973). *Forecasting Methods for Management*. 3rd ed. New York: Wiley-Interscience.

Chapter 17
The FORECAST Procedure

Contents

Overview: FORECAST Procedure **936**
Getting Started: FORECAST Procedure **938**
Giving Dates to Forecast Values 939
Computing Confidence Limits 939
Form of the OUT= Data Set 941
Plotting Forecasts 941
Plotting Residuals 942
Model Parameters and Goodness-of-Fit Statistics 943
Controlling the Forecasting Method 945
Introduction to Forecasting Methods 946
Time Trend Models 947
Time Series Methods 949
Combining Time Trend with Autoregressive Models 950
Syntax: FORECAST Procedure **951**
Functional Summary 951
PROC FORECAST Statement 953
BY Statement 957
ID Statement 957
VAR Statement 958
Details: FORECAST Procedure **958**
Missing Values 958
Data Periodicity and Time Intervals 958
Forecasting Methods 959
Specifying Seasonality 966
Data Requirements 968
OUT= Data Set 968
OUTEST= Data Set 969
Examples: FORECAST Procedure **972**
Example 17.1: Forecasting Auto Sales 972
Example 17.2: Forecasting Retail Sales 977
Example 17.3: Forecasting Petroleum Sales 982
References **985**

Overview: FORECAST Procedure

The FORECAST procedure is obsolete and has been superseded by newer SAS/ETS procedures. These newer procedures provide more powerful and flexible versions of the forecasting methods that PROC FORECAST uses, and they also provide additional forecasting methods that are not available in PROC FORECAST.

The FORECAST procedure is still available for use. However, before choosing to use PROC FORECAST, consider the following alternatives:

- For forecasting by using exponential smoothing methods or Winters method, consider using the ESM procedure. The models that can be selected by the PROC FORECAST options METHOD=EXPO, METHOD=WINTERS, and METHOD=ADDWINTERS are provided by PROC ESM, which also provides additional forecasting methods that PROC FORECAST does not support. Unlike PROC FORECAST, the ESM procedure optimizes the smoothing weights for the forecasting model based on the data. Also unlike PROC FORECAST, the ESM procedure can automatically select the form of exponential smoothing model that is most appropriate for your data. For information about forecasting with PROC ESM, see Chapter 15, "The ESM Procedure."

- For forecasting by using time trend models with autoregressive errors, consider using the AUTOREG procedure. The models that can be selected by the PROC FORECAST options METHOD=STEPAR and TREND= can be fit and forecast using PROC AUTOREG, which also allows the inclusion of additional predictor variables in the forecasting model. For information about PROC AUTOREG, see Chapter 9, "The AUTOREG Procedure."

- For forecasting by using more general and sophisticated time series models, consider using the UCM procedure, which fits and forecasts unobserved components models that are not available in PROC FORECAST. Using UCM models, you can fit and forecast much more complex data patterns than you can by using the simple methods that PROC FORECAST provides. Unlike PROC FORECAST, the UCM procedure can also model and forecast the effect of independent predictor variables. For information about PROC UCM, see Chapter 41, "The UCM Procedure."

- For forecasting by using ARIMA models and the Box-Jenkins methodology, consider using the ARIMA procedure. PROC ARIMA identifies, fits, and forecasts general autoregressive integrated moving average models, optionally incorporating transfer function models for the effects of independent predictor variables. (As a special case, you can use seasonal ARMA models for forecasting seasonal series for which the Winters and additive Winters methods might be used.) PROC ARIMA also provides features for automatically identifying the specific ARIMA model that is most appropriate for the data. ARIMA and ARIMAX models are not available in PROC FORECAST. For information about PROC ARIMA, see Chapter 8, "The ARIMA Procedure."

- For forecasting multivariate time series, where two or more related variables need to be forecast jointly, consider using the VARMAX procedure or the SSM procedure. PROC VARMAX fits and forecasts vector autoregressive moving average models, optionally incorporating multivariate transfer function models for the effects of independent predictor variables. For information about PROC VARMAX, see Chapter 42, "The VARMAX Procedure." PROC SSM fits and forecasts general linear state space models. The general state space model encompasses most of the other forecasting models that are mentioned in this section, and it enables generalizations that can model time series data patterns of

almost any type and complexity. For information about PROC SSM, see Chapter 34, "The SSM Procedure."

- For forecasting both the future expectation and future volatility or risk, consider using the AUTOREG procedure or the VARMAX procedure. PROC AUTOREG can fit and forecast many types of GARCH models of time-varying volatility, while also fitting and forecasting future expected values of the dependent variable. PROC VARMAX supports multivariate GARCH models. For information about PROC AUTOREG, see Chapter 9, "The AUTOREG Procedure." For information about PROC VARMAX, see Chapter 42, "The VARMAX Procedure."

If you decide to use PROC FORECAST instead of these newer alternatives, this chapter explains the features of the FORECAST procedure.

The FORECAST procedure provides a quick and automatic way to generate forecasts for many time series in one step. The procedure can forecast hundreds of series at a time, with the series organized into separate variables or across BY groups. PROC FORECAST uses extrapolative forecasting methods where the forecasts for a series are functions only of time and past values of the series, not of other variables.

You can use the following forecasting methods. For each of these methods, you can specify linear, quadratic, or no trend.

- The stepwise autoregressive method is used by default. This method combines time trend regression with an autoregressive model and uses a stepwise method to select the lags to use for the autoregressive process.
- The exponential smoothing method produces a time trend forecast. However, in fitting the trend, the parameters are allowed to change gradually over time, and earlier observations are given exponentially declining weights. Single, double, and triple exponential smoothing are supported, depending on whether no trend, linear trend, or quadratic trend, respectively, is specified. Holt two-parameter linear exponential smoothing is supported as a special case of the Holt-Winters method without seasons.
- The Winters method (also called Holt-Winters) combines a time trend with multiplicative seasonal factors to account for regular seasonal fluctuations in a series. Like the exponential smoothing method, the Winters method allows the parameters to change gradually over time, with earlier observations given exponentially declining weights. You can also specify the additive version of the Winters method, which uses additive instead of multiplicative seasonal factors. When seasonal factors are omitted, the Winters method reduces to the Holt two-parameter version of double exponential smoothing.

The FORECAST procedure writes the forecasts and confidence limits to an output data set. It can also write parameter estimates and fit statistics to an output data set. The FORECAST procedure does not produce printed output.

PROC FORECAST is an extrapolation procedure useful for producing practical results efficiently. However, in the interest of speed, PROC FORECAST uses some shortcuts that cause some statistical results (such as confidence limits) to be only approximate. For many time series, the FORECAST procedure, with appropriately chosen methods and weights, can yield satisfactory results. Other SAS/ETS procedures can produce better forecasts.

Getting Started: FORECAST Procedure

To use PROC FORECAST, specify the input and output data sets and the number of periods to forecast in the PROC FORECAST statement, and then list the variables to forecast in a VAR statement.

For example, suppose you have monthly data on the sales of some product in a data set named PAST, as shown in Figure 17.1, and you want to forecast sales for the next 10 months.

Figure 17.1 Example Data Set PAST

Obs	date	sales
1	JUL89	9.5161
2	AUG89	9.6994
3	SEP89	9.2644
4	OCT89	9.6837
5	NOV89	10.0784
6	DEC89	9.9005
7	JAN90	10.2375
8	FEB90	10.6940
9	MAR90	10.6290
10	APR90	11.0332
11	MAY90	11.0270
12	JUN90	11.4165
13	JUL90	11.2918
14	AUG90	11.3475
15	SEP90	11.2913
16	OCT90	11.3771
17	NOV90	11.5457
18	DEC90	11.6433
19	JAN91	11.9293
20	FEB91	11.9752
21	MAR91	11.9283
22	APR91	11.8985
23	MAY91	12.0419
24	JUN91	12.3537
25	JUL91	12.4546

The following statements forecast 10 observations for the variable SALES by using the default STEPAR method and write the results to the output data set PRED:

```
proc forecast data=past lead=10 out=pred;
   var sales;
run;
```

The following statements use the PRINT procedure to print the data set PRED:

```
proc print data=pred;
run;
```

The PROC PRINT listing of the forecast data set PRED is shown in Figure 17.2.

Figure 17.2 Forecast Data Set PRED

Obs	_TYPE_	_LEAD_	sales
1	FORECAST	1	12.6205
2	FORECAST	2	12.7665
3	FORECAST	3	12.9020
4	FORECAST	4	13.0322
5	FORECAST	5	13.1595
6	FORECAST	6	13.2854
7	FORECAST	7	13.4105
8	FORECAST	8	13.5351
9	FORECAST	9	13.6596
10	FORECAST	10	13.7840

Giving Dates to Forecast Values

Normally, your input data set has an ID variable that gives dates to the observations, and you want the forecast observations to have dates also. Usually, the ID variable has SAS date values. (See Chapter 4, "Working with Time Series Data," for information about using SAS date and datetime values.) The ID statement specifies the identifying variable.

If the ID variable contains SAS date or datetime values, the INTERVAL= option should be used on the PROC FORECAST statement to specify the time interval between observations. (See Chapter 5, "Date Intervals, Formats, and Functions," for more information about time intervals.) The FORECAST procedure uses the INTERVAL= option to generate correct dates for forecast observations.

The data set PAST, shown in Figure 17.1, has monthly observations and contains an ID variable DATE with SAS date values identifying each observation. The following statements produce the same forecast as the preceding example and also include the ID variable DATE in the output data set. Monthly SAS date values are extrapolated for the forecast observations.

```
proc forecast data=past interval=month lead=10 out=pred;
   var sales;
   id date;
run;
```

Computing Confidence Limits

Depending on the output options specified, multiple observations are written to the OUT= data set for each time period. The different parts of the results are contained in the VAR statement variables in observations identified by the character variable _TYPE_ and by the ID variable.

For example, the following statements use the OUTLIMIT option to write forecasts and 95% confidence limits for the variable SALES to the output data set PRED. This data set is printed with the PRINT procedure.

```
proc forecast data=past interval=month lead=10
              out=pred outlimit;
   var sales;
   id date;
run;

proc print data=pred;
run;
```

The output data set PRED is shown in Figure 17.3.

Figure 17.3 Output Data Set

Obs	date	_TYPE_	_LEAD_	sales
1	AUG91	FORECAST	1	12.6205
2	AUG91	L95	1	12.1848
3	AUG91	U95	1	13.0562
4	SEP91	FORECAST	2	12.7665
5	SEP91	L95	2	12.2808
6	SEP91	U95	2	13.2522
7	OCT91	FORECAST	3	12.9020
8	OCT91	L95	3	12.4001
9	OCT91	U95	3	13.4039
10	NOV91	FORECAST	4	13.0322
11	NOV91	L95	4	12.5223
12	NOV91	U95	4	13.5421
13	DEC91	FORECAST	5	13.1595
14	DEC91	L95	5	12.6435
15	DEC91	U95	5	13.6755
16	JAN92	FORECAST	6	13.2854
17	JAN92	L95	6	12.7637
18	JAN92	U95	6	13.8070
19	FEB92	FORECAST	7	13.4105
20	FEB92	L95	7	12.8830
21	FEB92	U95	7	13.9379
22	MAR92	FORECAST	8	13.5351
23	MAR92	L95	8	13.0017
24	MAR92	U95	8	14.0686
25	APR92	FORECAST	9	13.6596
26	APR92	L95	9	13.1200
27	APR92	U95	9	14.1993
28	MAY92	FORECAST	10	13.7840
29	MAY92	L95	10	13.2380
30	MAY92	U95	10	14.3301

Form of the OUT= Data Set

The OUT= data set PRED, shown in Figure 17.3, contains three observations for each of the 10 forecast periods. Each of these three observations has the same value of the ID variable DATE, the SAS date value for the month and year of the forecast.

The three observations for each forecast period have different values of the variable _TYPE_. For the _TYPE_=FORECAST observation, the value of the variable SALES is the forecast value for the period indicated by the DATE value. For the _TYPE_=L95 observation, the value of the variable SALES is the lower limit of the 95% confidence interval for the forecast. For the _TYPE_=U95 observation, the value of the variable SALES is the upper limit of the 95% confidence interval.

You can control the types of observations written to the OUT= data set with the PROC FORECAST statement options OUTLIMIT, OUTRESID, OUTACTUAL, OUT1STEP, OUTSTD, OUTFULL, and OUTALL. For example, the OUTFULL option outputs the confidence limit values, the one-step-ahead predictions, and the actual data, in addition to the forecast values. See the sections "Syntax: FORECAST Procedure" on page 951 and "OUTEST= Data Set" on page 969 for more information.

Plotting Forecasts

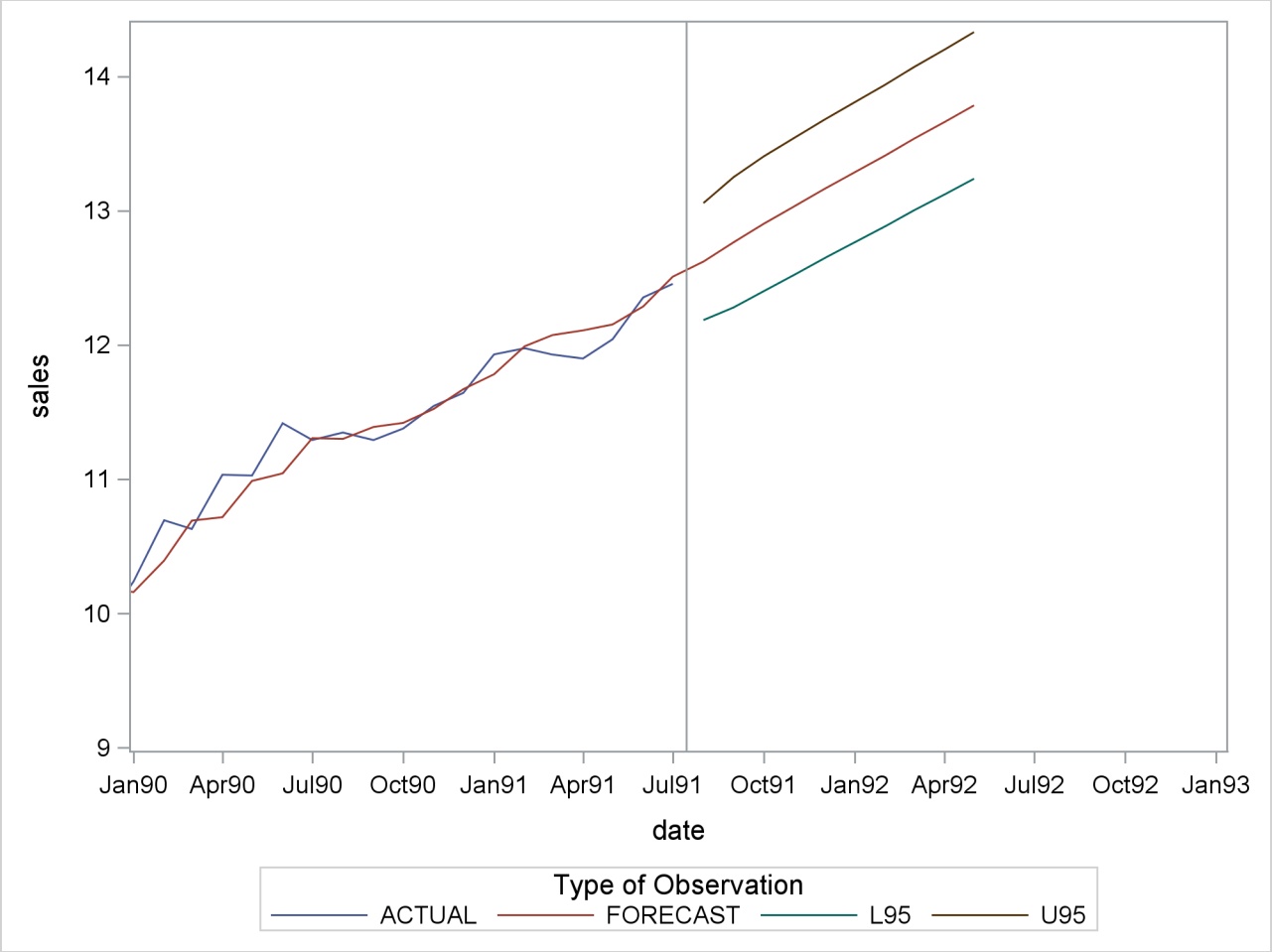
The forecasts, confidence limits, and actual values can be plotted on the same graph with the SGPLOT procedure. Use the appropriate output control options in the PROC FORECAST statement to include in the OUT= data set the series you want to plot. Use the _TYPE_ variable in the SGPLOT procedure GROUP option to separate the observations for the different plots.

The OUTFULL option is used in the following statements. The resulting output data set contains the actual and predicted values, as well as the upper and lower 95% confidence limits.

```
proc forecast data=past interval=month lead=10
              out=pred outfull;
   id date;
   var sales;
run;

proc sgplot data=pred;
   series x=date y=sales / group=_type_ lineattrs=(pattern=1);
   xaxis values=('1jan90'd to '1jan93'd by qtr);
   refline '15jul91'd / axis=x;
run;
```

The _TYPE_ variable is used in the SGPLOT procedure's PLOT statement to make separate plots over time for each type of value. A reference line marks the start of the forecast period. (See *SAS/GRAPH: Reference* for more information about using PROC SGPLOT.) The WHERE statement restricts the range of the actual data shown in the plot. In this example, the variable SALES has monthly data from July 1989 through July 1991, but only the data for 1990 and 1991 are shown in Figure 17.4.

Figure 17.4 Plot of Forecast with Confidence Limits



Plotting Residuals

You can plot the residuals from the forecasting model by using PROC SGPLOT and a WHERE statement.

1. Use the OUTRESID option or the OUTALL option in the PROC FORECAST statement to include the residuals in the output data set.
2. Use a WHERE statement to specify the observation type of 'RESIDUAL' in the PROC GPLOT code.

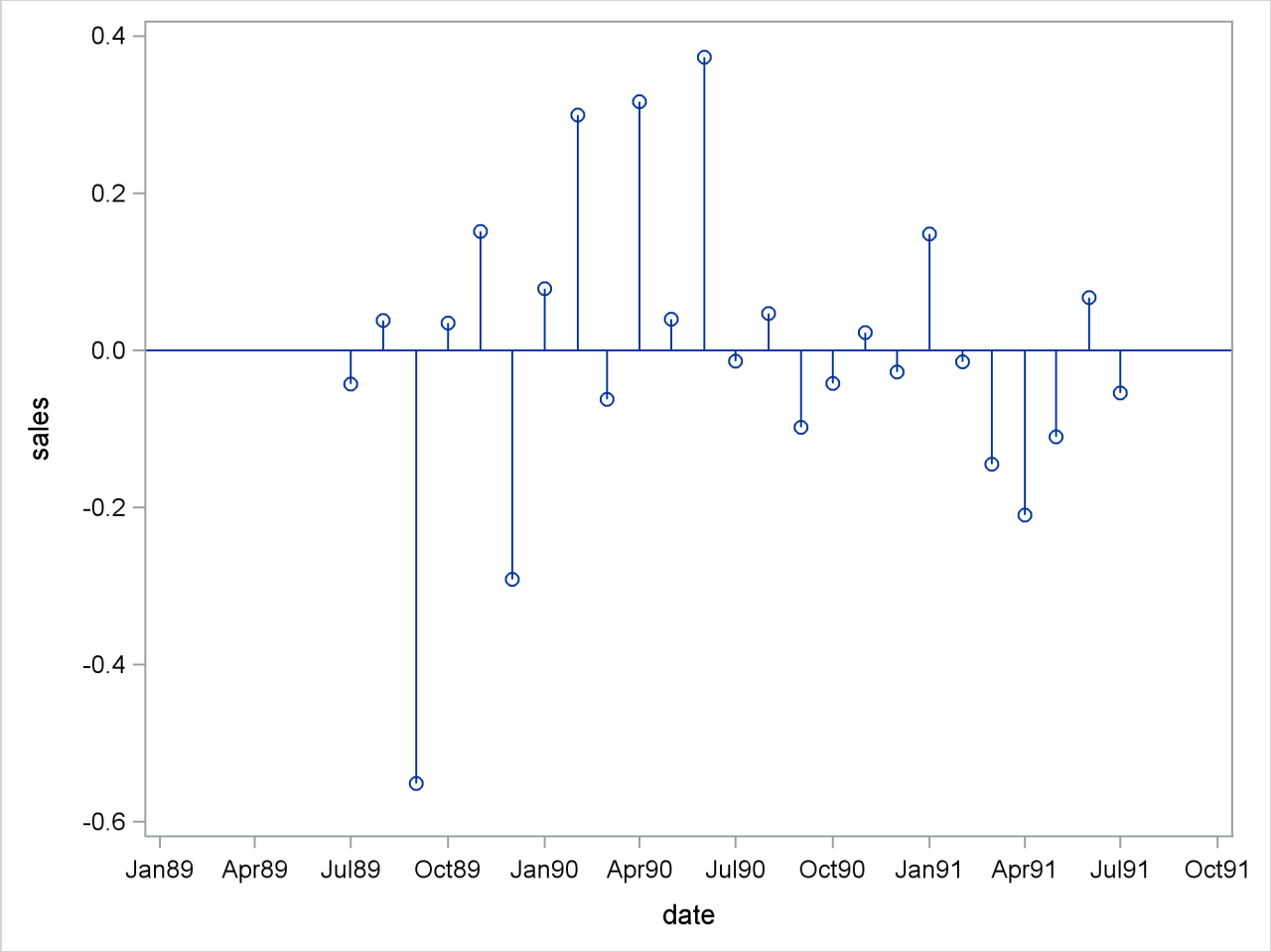
The following statements add the OUTRESID option to the preceding example and plot the residuals:

```
proc forecast data=past interval=month lead=10
              out=pred outfull outresid;
   id date;
   var sales;
run;
```

```
proc sgplot data=pred;
   where _type_='RESIDUAL';
   needle x=date y=sales / markers;
   xaxis values=('1jan89'd to '1oct91'd by qtr);
run;
```

The plot of residuals is shown in Figure 17.5.

Figure 17.5 Plot of Residuals



Model Parameters and Goodness-of-Fit Statistics

You can write the parameters of the forecasting models used, as well as statistics that measure how well the forecasting models fit the data, to an output SAS data set by using the OUTEST= option. The options OUTFITSTATS, OUTESTTHEIL, and OUTESTALL control what goodness-of-fit statistics are added to the OUTEST= data set.

For example, the following statements add the OUTEST= and OUTFITSTATS options to the previous example to create the output statistics data set EST for the results of the default stepwise autoregressive forecasting method:

```
proc forecast data=past interval=month lead=10
              out=pred outfull outresid
              outest=est outfitstats;
   id date;
   var sales;
run;

proc print data=est;
run;
```

The PRINT procedure prints the OUTEST= data set, as shown in Figure 17.6.

Figure 17.6 The OUTEST= Data Set for STEPAR Method

Obs	_TYPE_	date	sales
1	N	JUL91	25
2	NRESID	JUL91	25
3	DF	JUL91	22
4	SIGMA	JUL91	0.2001613
5	CONSTANT	JUL91	9.4348822
6	LINEAR	JUL91	0.1242648
7	AR1	JUL91	0.5206294
8	AR2	JUL91	.
9	AR3	JUL91	.
10	AR4	JUL91	.
11	AR5	JUL91	.
12	AR6	JUL91	.
13	AR7	JUL91	.
14	AR8	JUL91	.
15	SST	JUL91	21.28342
16	SSE	JUL91	0.8793714
17	MSE	JUL91	0.0399714
18	RMSE	JUL91	0.1999286
19	MAPE	JUL91	1.2280089
20	MPE	JUL91	-0.050139
21	MAE	JUL91	0.1312115
22	ME	JUL91	-0.001811
23	MAXE	JUL91	0.3732328
24	MINE	JUL91	-0.551605
25	MAXPE	JUL91	3.2692294
26	MINPE	JUL91	-5.954022
27	RSQUARE	JUL91	0.9586828
28	ADJRSQ	JUL91	0.9549267
29	RW_RSQ	JUL91	0.2657801
30	ARSQ	JUL91	0.9474145
31	APC	JUL91	0.044768
32	AIC	JUL91	-77.68559
33	SBC	JUL91	-74.02897
34	CORR	JUL91	0.9791313

In the OUTEST= data set, the DATE variable contains the ID value of the last observation in the data set used to fit the forecasting model. The variable SALES contains the statistic indicated by the value of the _TYPE_ variable. The _TYPE_=N, NRESID, and DF observations contain, respectively, the number of observations read from the data set, the number of nonmissing residuals used to compute the goodness-of-fit statistics, and the number of nonmissing observations minus the number of parameters used in the forecasting model.

The observation that has _TYPE_=SIGMA contains the estimate of the standard deviation of the one-step prediction error computed from the residuals. The _TYPE_=CONSTANT and _TYPE_=LINEAR observations contain the coefficients of the time trend regression. The _TYPE_=AR1, AR2, ..., AR8 observations contain the estimated autoregressive parameters. A missing autoregressive parameter indicates that the autoregressive term at that lag was not retained in the model by the stepwise model selection method. (See the section "STEPAR Method" on page 959 for more information.)

The other observations in the OUTEST= data set contain various goodness-of-fit statistics that measure how well the forecasting model used fits the given data. See the section "OUTEST= Data Set" on page 969 for details.

Controlling the Forecasting Method

The METHOD= option controls which forecasting method is used. The TREND= option controls the degree of the time trend model used. For example, the following statements produce forecasts of SALES as in the preceding example but use the double exponential smoothing method instead of the default STEPAR method:

```
proc forecast data=past interval=month lead=10
              method=expo trend=2
              out=pred outfull outresid
              outest=est outfitstats;
   var sales;
   id date;
run;

proc print data=est;
run;
```

The PRINT procedure prints the OUTEST= data set for the EXPO method, as shown in Figure 17.7.

Figure 17.7 The OUTEST= Data Set for METHOD=EXPO

Obs	_TYPE_	date	sales
1	N	JUL91	25
2	NRESID	JUL91	25
3	DF	JUL91	23
4	WEIGHT	JUL91	0.1055728
5	S1	JUL91	11.427657
6	S2	JUL91	10.316473
7	SIGMA	JUL91	0.2545069
8	CONSTANT	JUL91	12.538841
9	LINEAR	JUL91	0.1311574
10	SST	JUL91	21.28342
11	SSE	JUL91	1.4897965
12	MSE	JUL91	0.0647738
13	RMSE	JUL91	0.2545069
14	MAPE	JUL91	1.9121204
15	MPE	JUL91	-0.816886
16	MAE	JUL91	0.2101358
17	ME	JUL91	-0.094941
18	MAXE	JUL91	0.3127332
19	MINE	JUL91	-0.460207
20	MAXPE	JUL91	2.9243781
21	MINPE	JUL91	-4.967478
22	RSQUARE	JUL91	0.930002
23	ADJRSQ	JUL91	0.9269586
24	RW_RSQ	JUL91	-0.243886
25	ARSQ	JUL91	0.9178285
26	APC	JUL91	0.0699557
27	AIC	JUL91	-66.50591
28	SBC	JUL91	-64.06816
29	CORR	JUL91	0.9772418

See the section "Syntax: FORECAST Procedure" on page 951 for other options that control the forecasting method. See the section "Introduction to Forecasting Methods" on page 946 and the section "Forecasting Methods" on page 959 for an explanation of the different forecasting methods.

Introduction to Forecasting Methods

This section briefly introduces the forecasting methods used by the FORECAST procedure. See textbooks on forecasting and see the section "Forecasting Methods" on page 959 for more detailed discussions of forecasting methods.

The FORECAST procedure combines three basic models to fit time series:

- time trend models for long-term, deterministic change
- autoregressive models for short-term fluctuations

- seasonal models for regular seasonal fluctuations

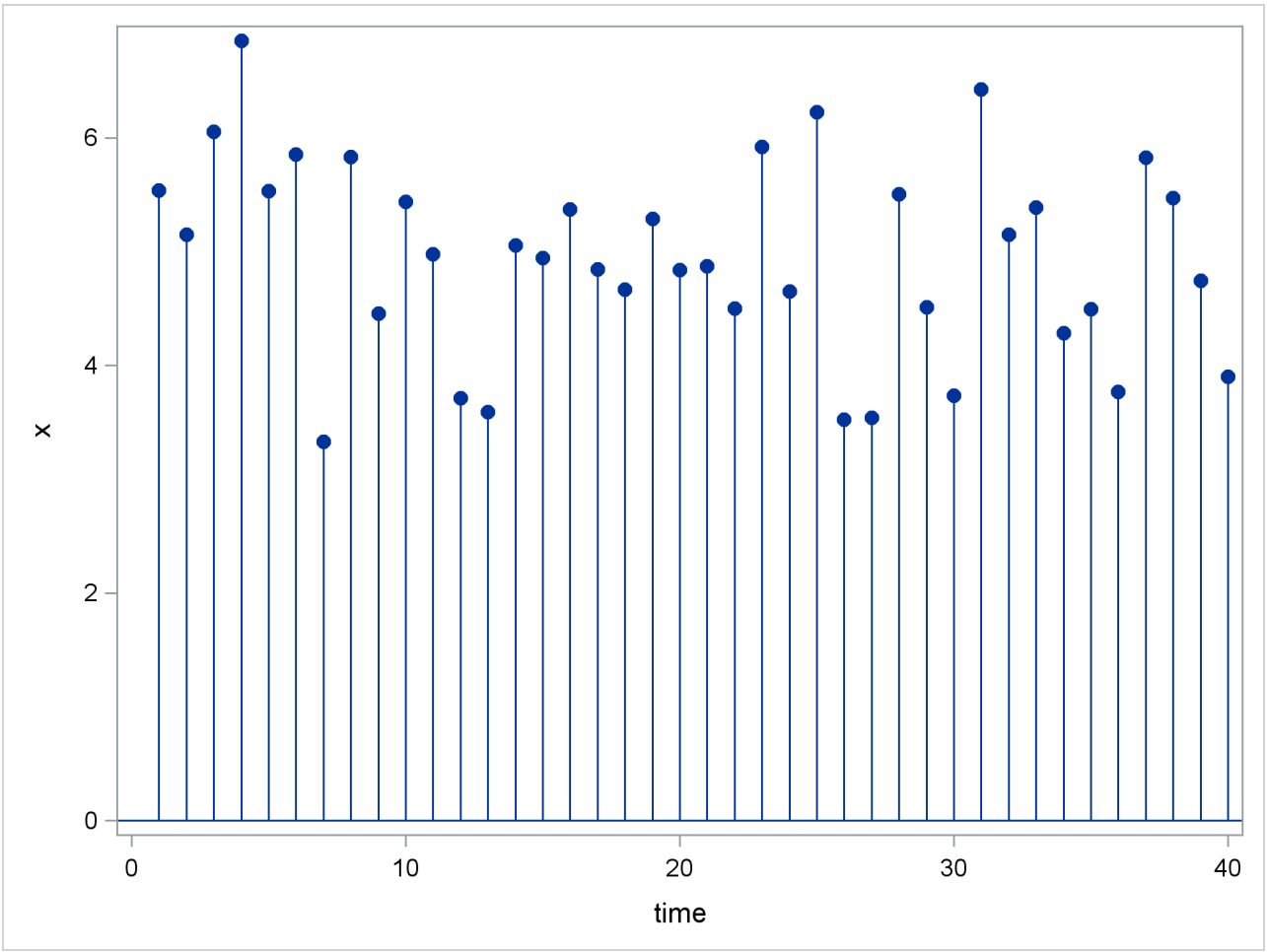
Two approaches to time series modeling and forecasting are *time trend models* and *time series methods*.

Time Trend Models

Time trend models assume that there is some permanent deterministic pattern across time. These models are best suited to data that are not dominated by random fluctuations.

Examining a graphical plot of the time series you want to forecast is often very useful in choosing an appropriate model. The simplest case of a time trend model is one in which you assume the series is a constant plus purely random fluctuations that are independent from one time period to the next. Figure 17.8 shows how such a time series might look.

Figure 17.8 Time Series without Trend



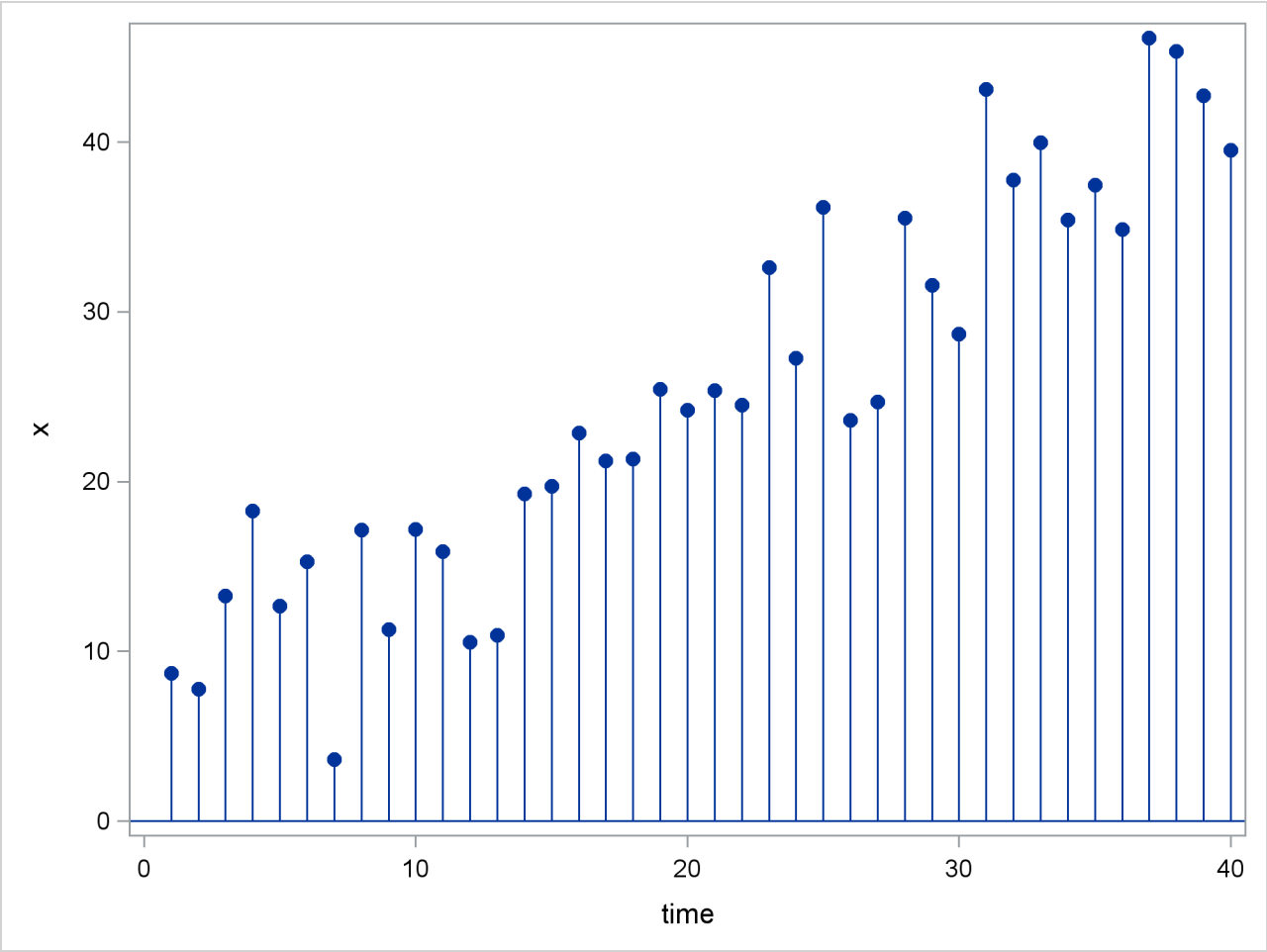
The x_t values are generated according to the equation

$$x_t = b_0 + \epsilon_t$$

where ϵ_t is an independent, zero-mean, random error and b_0 is the true series mean.

Suppose that the series exhibits growth over time, as shown in Figure 17.9.

Figure 17.9 Time Series with Linear Trend



A linear model is appropriate for this data. For the linear model, assume the x_t values are generated according to the equation

$$x_t = b_0 + b_1 t + \epsilon_t$$

The linear model has two parameters. The predicted values for the future are the points on the estimated line. The extension of the polynomial model to three parameters is the quadratic (which forms a parabola). This allows for a constantly changing slope, where the x_t values are generated according to the equation

$$x_t = b_0 + b_1 t + b_2 t^2 + \epsilon_t$$

PROC FORECAST can fit three types of time trend models: constant, linear, and quadratic. For other kinds of trend models, other SAS procedures can be used.

Exponential smoothing fits a time trend model by using a smoothing scheme in which the weights decline geometrically as you go backward in time. The forecasts from exponential smoothing are a time trend, but the trend is based mostly on the recent observations instead of on all the observations equally. How well exponential smoothing works as a forecasting method depends on choosing a good smoothing weight for the series.

To specify the exponential smoothing method, use the METHOD=EXPO option. Single exponential smoothing produces forecasts with a constant trend (that is, no trend). Double exponential smoothing produces forecasts with a linear trend, and triple exponential smoothing produces a quadratic trend. Use the TREND= option with the METHOD=EXPO option to select single, double, or triple exponential smoothing.

The time trend model can be modified to account for regular seasonal fluctuations of the series about the trend. To capture seasonality, the trend model includes a seasonal parameter for each season. Seasonal models can be additive or multiplicative.

$$x_t = b_0 + b_1 t + s(t) + \epsilon_t \qquad \text{(additive)}$$

$$x_t = (b_0 + b_1 t)s(t) + \epsilon_t \qquad \text{(multiplicative)}$$

where *s(t)* is the seasonal parameter for the season that corresponds to time *t*.

The Winters method is similar to exponential smoothing, but it includes seasonal factors. The Winters method can use either additive or multiplicative seasonal factors. Like exponential smoothing, good results with the Winters method depend on choosing good smoothing weights for the series to be forecast.

To specify the multiplicative or additive versions of the Winters method, use the METHOD=WINTERS or METHOD=ADDWINTERS options, respectively. To specify seasonal factors to include in the model, use the SEASONS= option.

Many observed time series do not behave like constant, linear, or quadratic time trends. However, you can partially compensate for the inadequacies of the trend models by fitting time series models to the departures from the time trend, as described in the following sections.

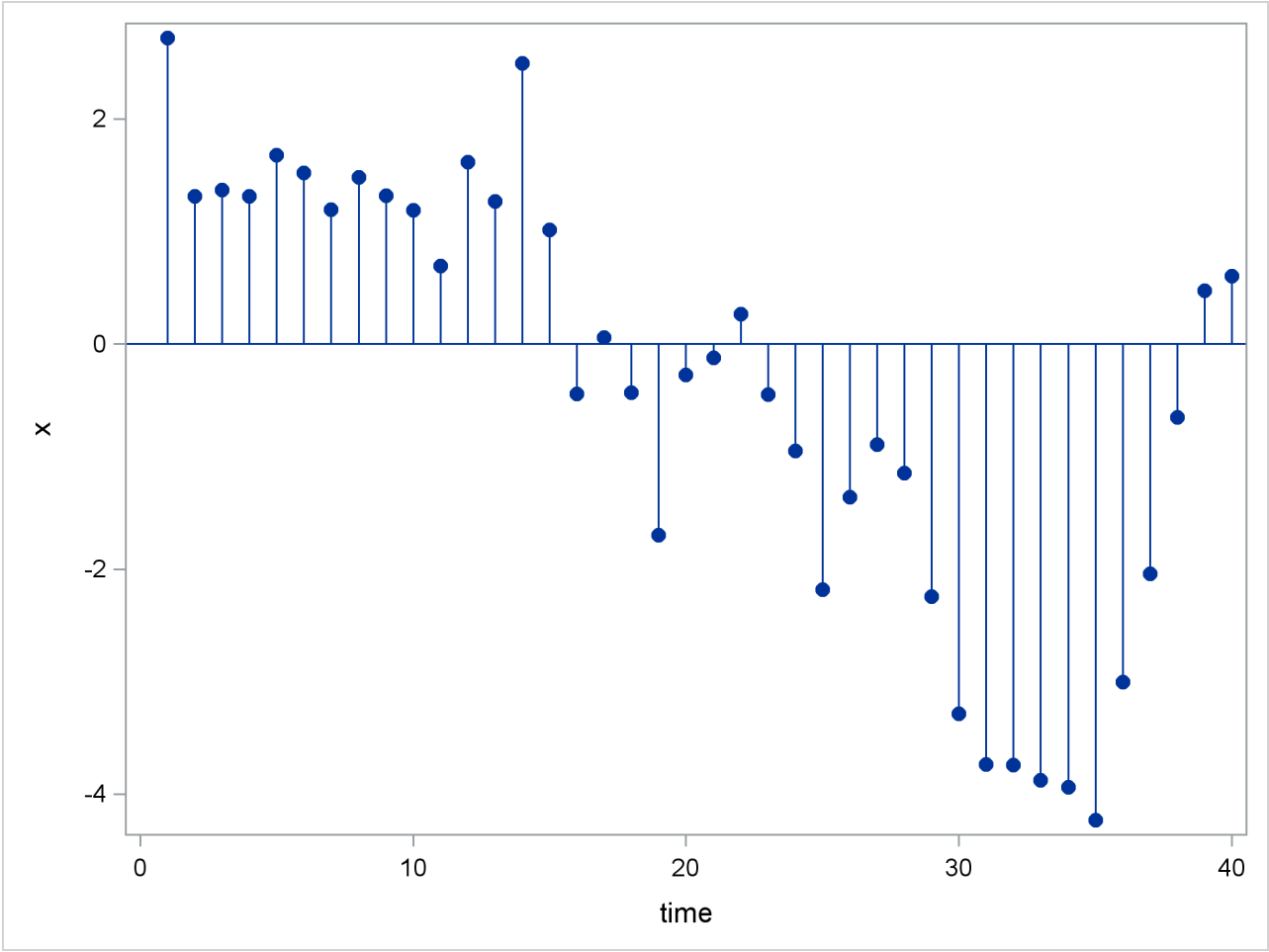
Time Series Methods

Time series models assume the future value of a variable to be a linear function of past values. If the model is a function of past values for a finite number of periods, it is an *autoregressive model* and is written as follows:

$$x_t = a_0 + a_1 x_{t-1} + a_2 x_{t-2} + \ldots + a_p x_{t-p} + \epsilon_t$$

The coefficients a_i are *autoregressive parameters*. One of the simplest cases of this model is the random walk, where the series dances around in purely random jumps. This is illustrated in Figure 17.10.

Figure 17.10 Random Walk Series



The x_t values are generated by the equation

$$x_t = x_{t-1} + \epsilon_t$$

In this type of model, the best forecast of a future value is the present value. However, with other autoregressive models, the best forecast is a weighted sum of recent values. Pure autoregressive forecasts always damp down to a constant (assuming the process is stationary).

Autoregressive time series models can also be used to predict seasonal fluctuations.

Combining Time Trend with Autoregressive Models

Trend models are suitable for capturing long-term behavior, whereas autoregressive models are more appropriate for capturing short-term fluctuations. One approach to forecasting is to combine a deterministic time trend model with an autoregressive model.

The *stepwise autoregressive method* (STEPAR method) combines a time trend regression with an autoregressive model for departures from trend. The combined time trend and autoregressive model is written as follows:

$$x_t = b_0 + b_1 t + b_2 t^2 + u_t$$

$$u_t = a_1 u_{t-1} + a_2 u_{t-2} + \ldots + a_p u_{t-p} + \epsilon_t$$

The autoregressive parameters included in the model for each series are selected by a stepwise regression procedure, so that autoregressive parameters are included only at those lags at which they are statistically significant.

The stepwise autoregressive method is fully automatic. Unlike the exponential smoothing and Winters methods, it does not depend on choosing smoothing weights. However, the STEPAR method assumes that the long-term trend is stable; that is, the time trend regression is fit to the whole series with equal weights for the observations.

The stepwise autoregressive model is used when you specify the METHOD=STEPAR option or do not specify any METHOD= option. To select a constant, linear, or quadratic trend for the time-trend part of the model, use the TREND= option.

Syntax: FORECAST Procedure

The following statements are used with PROC FORECAST:

PROC FORECAST *options* **;**
  BY *variables* **;**
  ID *variables* **;**
  VAR *variables* **;**

Functional Summary

Table 17.1 summarizes the statements and options that control the FORECAST procedure.

Table 17.1 FORECAST Functional Summary

Description	Statement	Option
Statements		
specify model and data set options	PROC FORECAST	
specify BY-group processing	BY	
identify observations	ID	
specify the variables to forecast	VAR	
Input Data Set Options		
specify the input SAS data set	PROC FORECAST	DATA=
specify frequency of the input time series	PROC FORECAST	INTERVAL=
specify increment between observations	PROC FORECAST	INTPER=
specify seasonality	PROC FORECAST	SEASONS=
specify number of periods in a season	PROC FORECAST	SINTPER=
treat zeros at beginning of series as missing	PROC FORECAST	ZEROMISS

Description	Statement	Option
Output Data Set Options		
specify the number of periods ahead to forecast	PROC FORECAST	LEAD=
name output data set to contain the forecasts	PROC FORECAST	OUT=
write actual values to the OUT= data set	PROC FORECAST	OUTACTUAL
write confidence limits to the OUT= data set	PROC FORECAST	OUTLIMIT
write residuals to the OUT= data set	PROC FORECAST	OUTRESID
write standard errors of the forecasts to the OUT= data set	PROC FORECAST	OUTSTD
write one-step-ahead predicted values to the OUT= data set	PROC FORECAST	OUT1STEP
write predicted, actual, and confidence limit values to the OUT= data set	PROC FORECAST	OUTFULL
write all available results to the OUT= data set	PROC FORECAST	OUTALL
specify significance level for confidence limits	PROC FORECAST	ALPHA=
control the alignment of SAS date values	PROC FORECAST	ALIGN=
Parameters and Statistics Output Data Set Options		
write parameter estimates and goodness-of-fit statistics to an output data set	PROC FORECAST	OUTEST=
write additional statistics to OUTEST= data set	PROC FORECAST	OUTESTALL
write Theil statistics to OUTEST= data set	PROC FORECAST	OUTESTTHEIL
write forecast accuracy statistics to OUTEST= data set	PROC FORECAST	OUTFITSTATS
Forecasting Method Options		
specify the forecasting method	PROC FORECAST	METHOD=
specify degree of the time trend model	PROC FORECAST	TREND=
specify smoothing weights	PROC FORECAST	WEIGHT=
specify order of the autoregressive model	PROC FORECAST	AR=
specify significance level for adding AR lags	PROC FORECAST	SLENTRY=
specify significance level for keeping AR lags	PROC FORECAST	SLSTAY=
start forecasting before the end of data	PROC FORECAST	START=
specify criterion for judging singularity	PROC FORECAST	SINGULAR=
limit number of error or warning messages	PROC FORECAST	MAXERRORS=

Description	Statement	Option
Initializing Smoothed Values		
specify number of beginning values to use in calculating starting values	PROC FORECAST	NSTART=
specify number of beginning values to use in calculating initial seasonal parameters	PROC FORECAST	NSSTART=
specify starting values for constant term	PROC FORECAST	ASTART=
specify starting values for linear trend	PROC FORECAST	BSTART=
specify starting values for the quadratic trend	PROC FORECAST	CSTART=

PROC FORECAST Statement

PROC FORECAST *options* **;**

The following options can be specified in the PROC FORECAST statement:

ALIGN=*option*

controls the alignment of SAS dates used to identify output observations. The ALIGN= option allows the following values: BEGINNING | BEG | B, MIDDLE | MID | M, and ENDING | END | E. BEGINNING is the default.

ALPHA=*value*

specifies the significance level to use in computing the confidence limits of the forecast. The value of the ALPHA= option must be between 0.01 and 0.99. You should use only two digits for the ALPHA= option because PROC FORECAST rounds the value to the nearest percent (ALPHA=0.101 is the same as ALPHA=0.10). The default is ALPHA=0.05, which produces 95% confidence limits.

AR=*n*

NLAGS=*n*

specifies the maximum order of the autoregressive model. The AR= option is valid only for METHOD=STEPAR. The default value of *n* depends on the INTERVAL= option and on the number of observations in the DATA= data set. See the section "STEPAR Method" on page 959 for details.

ASTART=*value*

ASTART=*(value . . .)*

specifies starting values for the constant term for the exponential smoothing, Winters, and additive Winters methods. This option is ignored if METHOD=STEPAR. The values specified are associated with the variables in the VAR statement in the order in which the variables are listed. See the section "Starting Values for EXPO, WINTERS, and ADDWINTERS Methods" on page 966 for details.

BSTART=*value*

BSTART=*(value . . .)*

specifies starting values for the linear trend for the exponential smoothing, Winters, and additive Winters methods. The values specified are associated with the variables in the VAR statement in the order in which the variables are listed. This option is ignored if METHOD=STEPAR or TREND=1. See the section "Starting Values for EXPO, WINTERS, and ADDWINTERS Methods" on page 966 for details.

CSTART=*value*

CSTART=*(value . . .)*

specifies starting values for the quadratic trend for the exponential smoothing, Winters, and additive Winters methods. The values specified are associated with the variables in the VAR statement in the order in which the variables are listed. This option is ignored if METHOD=STEPAR or TREND=1 or 2. See the section "Starting Values for EXPO, WINTERS, and ADDWINTERS Methods" on page 966 for details.

DATA=*SAS-data-set*

names the SAS data set that contains the input time series for the procedure to forecast. If the DATA= option is not specified, the most recently created SAS data set is used.

INTERVAL=*interval*

specifies the frequency of the input time series. For example, if the input data set consists of quarterly observations, then INTERVAL=QTR should be used. See Chapter 5, "Date Intervals, Formats, and Functions," for more details about the intervals available.

INTPER=*n*

when the INTERVAL= option is not used, specifies an increment (other than 1) to use in generating the values of the ID variable for the forecast observations in the output data set.

LEAD=*n*

specifies the number of periods ahead to forecast. The default is LEAD=12.

The LEAD= value is relative to the last observation in the input data set and not to the end of a particular series. Thus, if a series has missing values at the end, the actual number of forecasts computed for that series will be greater than the LEAD= value.

MAXERRORS=*n*

limits the number of warning and error messages produced during the execution of the procedure to the specified value. The default is MAXERRORS=50.

This option is particularly useful in BY-group processing where it can be used to suppress the recurring messages.

METHOD=*method-name*

specifies the method to use to model the series and generate the forecasts.

METHOD=STEPAR specifies the stepwise autoregressive method.

METHOD=EXPO specifies the exponential smoothing method.

METHOD=WINTERS specifies the Holt-Winters exponentially smoothed trend-seasonal method.

METHOD=ADDWINTERS specifies the additive seasonal factors variant of the Winters method.

For more information, see the section "Forecasting Methods" on page 959. The default is METHOD=STEPAR.

NSTART=*n*

NSTART=MAX

specifies the number of beginning values of the series to use in calculating starting values for the trend parameters in the exponential smoothing, Winters, and additive Winters methods. This option is ignored if METHOD=STEPAR.

For METHOD=EXPO, *n* beginning values of the series are used in forming the exponentially smoothed values S1, S2, and S3, where *n* is the value of the NSTART= option. The parameters are initialized by fitting a time trend regression to the first *n* nonmissing values of the series.

For METHOD=WINTERS or METHOD=ADDWINTERS, *n* beginning complete seasonal cycles are used to compute starting values for the trend parameters. For example, for monthly data the seasonal cycle is one year, and NSTART=2 specifies that the first 24 observations at the beginning of each series are used for the time trend regression used to calculate starting values.

When NSTART=MAX is specified, all the observations are used. The default for METHOD=EXPO is NSTART=8; the default for METHOD=WINTERS or METHOD=ADDWINTERS is NSTART=2. See the section "Starting Values for EXPO, WINTERS, and ADDWINTERS Methods" on page 966 for details.

NSSTART=*n*

NSSTART=MAX

specifies the number of beginning values of the series to use in calculating starting values for seasonal parameters for METHOD=WINTERS or METHOD=ADDWINTERS. The seasonal parameters are initialized by averaging over the first *n* values of the series for each season, where *n* is the value of the NSSTART= option. When NSSTART=MAX is specified, all the observations are used.

If NSTART= is specified, but NSSTART= is not, NSSTART= defaults to the value specified for NSTART=. If neither NSTART= nor NSSTART= is specified, then the default is NSSTART=2. This option is ignored if METHOD=STEPAR or METHOD=EXPO. See the section "Starting Values for EXPO, WINTERS, and ADDWINTERS Methods" on page 966 for details.

OUT=*SAS-data-set*

names the output data set to contain the forecasts. If the OUT= option is not specified, the data set is named by using the DATA*n* convention. See the section "OUTEST= Data Set" on page 969 for details.

OUTACTUAL

writes the actual values to the OUT= data set.

OUTALL

provides all the output control options (OUTLIMIT, OUT1STEP, OUTACTUAL, OUTRESID, and OUTSTD).

OUTEST=*SAS-data-set*

names an output data set to contain the parameter estimates and goodness-of-fit statistics. When the OUTEST= option is not specified, the parameters and goodness-of-fit statistics are not stored. See the section "OUTEST= Data Set" on page 969 for details.

OUTESTALL
writes additional statistics to the OUTEST= data set. This option is the same as specifying both OUTESTTHEIL and OUTFITSTATS.

OUTESTTHEIL
writes Theil forecast accuracy statistics to the OUTEST= data set.

OUTFITSTATS
writes various R-square-type forecast accuracy statistics to the OUTEST= data set.

OUTFULL
provides OUTACTUAL, OUT1STEP, and OUTLIMIT output control options in addition to the forecast values.

OUTLIMIT
writes the forecast confidence limits to the OUT= data set.

OUTRESID
writes the residuals (when available) to the OUT= data set.

OUTSTD
writes the standard errors of the forecasts to the OUT= data set.

OUT1STEP
writes the one-step-ahead predicted values to the OUT= data set.

SEASONS=*interval*

SEASONS= (*interval1 [interval2 [interval3]]* **)**

SEASONS=*n*

SEASONS= (*n1 [n2 [n3]]* **)**
specifies the seasonality for seasonal models. The *interval* can be QTR, MONTH, DAY, or HOUR, or multiples of these (for example, QTR2, MONTH2, MONTH3, MONTH4, MONTH6, HOUR2, HOUR3, HOUR4, HOUR6, HOUR8, and HOUR12).

Alternatively, seasonality can be specified by giving the length of the seasonal cycles. For example, SEASONS=3 means that every group of three observations forms a seasonal cycle. The SEASONS= option is valid only for METHOD=WINTERS or METHOD=ADDWINTERS. See the section "Specifying Seasonality" on page 966 for details.

SINGULAR=*value*
gives the criterion for judging singularity. The default depends on the precision of the computer that you run SAS programs on.

SINTPER=*m*

SINTPER= (*m1 [m2 [m3]]* **)**
specifies the number of periods to combine in forming a season. For example, SEASONS=3 SINTPER=2 specifies that each group of two observations forms a season and that the seasonal cycle repeats every six observations. The SINTPER= option is valid only when the SEASONS= option is used. See the section "Specifying Seasonality" on page 966 for details.

SLENTRY=*value*
controls the significance levels for entry of autoregressive parameters in the STEPAR method. The value of the SLENTRY= option must be between 0 and 1. The default is SLENTRY=0.2. See the section "STEPAR Method" on page 959 for details.

SLSTAY=*value*
controls the significance levels for removal of autoregressive parameters in the STEPAR method. The value of the SLSTAY= option must be between 0 and 1. The default is SLSTAY=0.05. See the section "STEPAR Method" on page 959 for details.

START=*n*
uses the first *n* observations to fit the model and begins forecasting with the *n* +1 observation.

TREND=*n*
specifies the degree of the time trend model. The value of the TREND= option must be 1, 2, or 3. TREND=1 selects the constant trend model; TREND=2 selects the linear trend model; and TREND=3 selects the quadratic trend model. The default is TREND=2, except for METHOD=EXPO, for which the default is TREND=3.

WEIGHT=*w*
WEIGHT= (*w1 [w2 [w3]]* **)**
specifies the smoothing weights for the EXPO, WINTERS, and ADDWINTERS methods. For the EXPO method, only one weight can be specified. For the WINTERS or ADDWINTERS method, *w1* gives the weight for updating the constant component, *w2* gives the weight for updating the linear and quadratic trend components, and *w3* gives the weight for updating the seasonal component. The *w2* and *w3* values are optional. Each value in the WEIGHT= option must be between 0 and 1. For default values, see the section "EXPO Method" on page 960 and the section "WINTERS Method" on page 962.

ZEROMISS
treats zeros at the beginning of a series as missing values. For example, a product can be introduced at a date after the date of the first observation in the data set, and the sales variable for the product can be recorded as zero for the observations prior to the introduction date. The ZEROMISS option says to treat these initial zeros as missing values.

BY Statement

BY *variables* **;**

A BY statement can be used with PROC FORECAST to obtain separate analyses on observations in groups defined by the BY variables.

ID Statement

ID *variables* **;**

The first variable listed in the ID statement identifies observations in the input and output data sets. Usually, the first ID variable is a SAS date or datetime variable. Its values are interpreted and extrapolated according

to the values of the INTERVAL= option. See the section "Data Periodicity and Time Intervals" on page 958 for details.

If more than one ID variable is specified in the ID statement, only the first is used to identify the observations; the rest are just copied to the OUT= data set and will have missing values for forecast observations.

VAR Statement

VAR *variables* **;**

The VAR statement specifies the variables in the input data set that you want to forecast. If no VAR statement is specified, the procedure forecasts all numeric variables except the ID and BY variables.

Details: FORECAST Procedure

Missing Values

The treatment of missing values varies by method. For METHOD=STEPAR, missing values are tolerated in the series; the autocorrelations are estimated from the available data and tapered, if necessary. For the EXPO, WINTERS, and ADDWINTERS methods, missing values after the start of the series are replaced with one-step-ahead predicted values, and the predicted values are applied to the smoothing equations. For the WINTERS method, negative or zero values are treated as missing.

Data Periodicity and Time Intervals

The INTERVAL= option is used to establish the frequency of the time series. For example, INTERVAL=MONTH specifies that each observation in the input data set represents one month. If INTERVAL=MONTH2, each observation represents two months. Thus, there is a two-month time interval between each pair of successive observations, and the data frequency is bimonthly.

See Chapter 5, "Date Intervals, Formats, and Functions," for details about the interval values supported.

The INTERVAL= option is used together with the ID statement to fully describe the observations that make up the time series. The first variable specified in the ID statement is used to identify the observations. Usually, SAS date or datetime values are used for this variable. PROC FORECAST uses the ID variable in the following ways:

- to validate the data periodicity. When the INTERVAL= option is specified, the ID variable is used to check the data and verify that successive observations have valid ID values that correspond to successive time intervals. When the INTERVAL= option is not used, PROC FORECAST verifies that the ID values are nonmissing and in ascending order. A warning message is printed when an invalid ID value is found in the input data set.

- to check for gaps in the input observations. For example, if INTERVAL=MONTH and an input observation for January 1970 is followed by an observation for April 1970, there is a gap in the input data, with two observations omitted. When a gap in the input data is found, a warning message is printed, and PROC FORECAST processes missing values for each omitted input observation.
- to label the forecast observations in the output data set. The values of the ID variable for the forecast observations after the end of the input data set are extrapolated according to the frequency specifications of the INTERVAL= option. If the INTERVAL= option is not specified, the ID variable is extrapolated by incrementing the ID variable value for the last observation in the input data set by the INTPER= value, if specified, or by one.

The ALIGN= option controls the alignment of SAS dates. See Chapter 5, "Date Intervals, Formats, and Functions," for more information.

Forecasting Methods

This section explains the forecasting methods used by PROC FORECAST.

STEPAR Method

In the STEPAR method, PROC FORECAST first fits a time trend model to the series and takes the difference between each value and the estimated trend. (This process is called *detrending*.) Then, the remaining variation is fit by using an autoregressive model.

The STEPAR method fits the autoregressive process to the residuals of the trend model by using a backwards-stepping method to select parameters. Because the trend and autoregressive parameters are fit in sequence rather than simultaneously, the parameter estimates are not optimal in a statistical sense. However, the estimates are usually close to optimal, and the method is computationally inexpensive.

The STEPAR Algorithm

The STEPAR method consists of the following computational steps:

1. Fit the trend model as specified by the TREND= option by using ordinary least-squares regression. This step detrends the data. The default trend model for the STEPAR method is TREND=2, a linear trend model.
2. Take the residuals from step 1 and compute the autocovariances to the number of lags specified by the NLAGS= option.
3. Regress the current values against the lags, using the autocovariances from step 2 in a Yule-Walker framework. Do not bring in any autoregressive parameter that is not significant at the level specified by the SLENTRY= option. (The default is SLENTRY=0.20.) Do not bring in any autoregressive parameter that results in a nonpositive-definite Toeplitz matrix.
4. Find the autoregressive parameter that is least significant. If the significance level is greater than the SLSTAY= value, remove the parameter from the model. (The default is SLSTAY=0.05.) Continue this process until only significant autoregressive parameters remain. If the OUTEST= option is specified, write the estimates to the OUTEST= data set.

5. Generate the forecasts by using the estimated model and output to the OUT= data set. Form the confidence limits by combining the trend variances with the autoregressive variances.

Missing values are tolerated in the series; the autocorrelations are estimated from the available data and tapered if necessary.

This method requires at least three passes through the data: two passes to fit the model and a third pass to initialize the autoregressive process and write to the output data set.

Default Value of the NLAGS= Option

If the NLAGS= option is not specified, the default value of the NLAGS= option is chosen based on the data frequency specified by the INTERVAL= option and on the number of observations in the input data set, if this can be determined in advance. (PROC FORECAST cannot determine the number of input observations before reading the data when a BY statement or a WHERE statement is used or if the data are from a tape format SAS data set or external database. The NLAGS= value must be fixed before the data are processed.)

If the INTERVAL= option is specified, the default NLAGS= value includes lags for up to three years plus one, subject to the maximum of 13 lags or one-third of the number of observations in your data set, whichever is less. If the number of observations in the input data set cannot be determined, the maximum NLAGS= default value is 13. If the INTERVAL= option is not specified, the default is NLAGS=13 or one-third the number of input observations, whichever is less.

If the Toeplitz matrix formed by the autocovariance matrix at a given step is not positive definite, the maximal number of autoregressive lags is reduced.

For example, for INTERVAL=QTR, the default is NLAGS=13 (that is, $4\times3 + 1$) provided that there are at least 39 observations. The NLAGS= option default is always at least 3.

EXPO Method

Exponential smoothing is used when the METHOD=EXPO option is specified. The term *exponential smoothing* is derived from the computational scheme developed by Brown and others (Brown and Meyer 1961; Brown 1962). Estimates are computed with updating formulas that are developed across time series in a manner similar to smoothing.

The EXPO method fits a trend model such that the most recent data are weighted more heavily than data in the early part of the series. The weight of an observation is a geometric (exponential) function of the number of periods that the observation extends into the past relative to the current period. The weight function is

$$w_\tau = \omega(1-\omega)^{t-\tau}$$

where τ is the observation number of the past observation, t is the current observation number, and ω is the weighting constant specified with the WEIGHT= option.

You specify the model with the TREND= option as follows:

- TREND=1 specifies single exponential smoothing (a constant model)
- TREND=2 specifies double exponential smoothing (a linear trend model)
- TREND=3 specifies triple exponential smoothing (a quadratic trend model)

Updating Equations

The single exponential smoothing operation is expressed by the formula

$$S_t = \omega x_t + (1-\omega) S_{t-1}$$

where S_t is the smoothed value at the current period, t is the time index of the current period, and x_t is the current actual value of the series. The smoothed value S_t is the forecast of x_{t+1} and is calculated as the smoothing constant ω times the value of the series, x_t, in the current period plus $(1-\omega)$ times the previous smoothed value S_{t-1}, which is the forecast of x_t computed at time $t-1$.

Double and triple exponential smoothing are derived by applying exponential smoothing to the smoothed series, obtaining smoothed values as follows:

$$S_t^{[2]} = \omega S_t + (1-\omega) S_{t-1}^{[2]}$$

$$S_t^{[3]} = \omega S_t^{[2]} + (1-\omega) S_{t-1}^{[3]}$$

Missing values after the start of the series are replaced with one-step-ahead predicted values, and the predicted value is then applied to the smoothing equations.

The polynomial time trend parameters CONSTANT, LINEAR, and QUAD in the OUTEST= data set are computed from S_T, $S_T^{[2]}$, and $S_T^{[3]}$, the final smoothed values at observation T, the last observation used to fit the model. In the OUTEST= data set, the values of S_T, $S_T^{[2]}$, and $S_T^{[3]}$ are identified by _TYPE_=S1, _TYPE_=S2, and _TYPE_=S3, respectively.

Smoothing Weights

Exponential smoothing forecasts are forecasts for an integrated moving-average process; however, the weighting parameter is specified by the user rather than estimated from the data. Experience has shown that good values for the WEIGHT= option are between 0.05 and 0.3. As a general rule, smaller smoothing weights are appropriate for series with a slowly changing trend, while larger weights are appropriate for volatile series with a rapidly changing trend. If unspecified, the weight defaults to $(1-0.8^{1/trend})$, where *trend* is the value of the TREND= option. This produces defaults of WEIGHT=0.2 for TREND=1, WEIGHT=0.10557 for TREND=2, and WEIGHT=0.07168 for TREND=3.

The ESM procedure can be used to forecast time series by using exponential smoothing with smoothing weights that are optimized automatically. See Chapter 15, "The ESM Procedure."

The Time Series Forecasting System provides for exponential smoothing models and enables you to either specify or optimize the smoothing weights. See Chapter 54, "Getting Started with Time Series Forecasting," for details.

Confidence Limits

The confidence limits for exponential smoothing forecasts are calculated as they would be for an exponentially weighted time trend regression, using the simplifying assumption of an infinite number of observations. The variance estimate is computed by using the mean square of the unweighted one-step-ahead forecast residuals.

More detailed descriptions of the forecast computations can be found in Montgomery and Johnson (1976); Brown (1962).

WINTERS Method

The WINTERS method uses updating equations similar to exponential smoothing to fit parameters for the model

$$x_t = (a + bt)s(t) + \epsilon_t$$

where a and b are the trend parameters and the function $s(t)$ selects the seasonal parameter for the season that corresponds to time t.

The WINTERS method assumes that the series values are positive. If negative or zero values are found in the series, a warning is printed and the values are treated as missing.

The preceding standard WINTERS model uses a linear trend. However, PROC FORECAST can also fit a version of the WINTERS method that uses a quadratic trend. When TREND=3 is specified for METHOD=WINTERS, PROC FORECAST fits the following model:

$$x_t = (a + bt + ct^2)s(t) + \epsilon_t$$

The quadratic trend version of the Winters method is often unstable, and its use is not recommended.

When TREND=1 is specified, the following constant trend version is fit:

$$x_t = as(t) + \epsilon_t$$

The default for the WINTERS method is TREND=2, which produces the standard linear trend model.

Seasonal Factors

The notation $s(t)$ represents the selection of the seasonal factor used for different time periods. For example, if INTERVAL=DAY and SEASONS=MONTH, there are 12 seasonal factors, one for each month in the year, and the time index t is measured in days. For any observation, t is determined by the ID variable and $s(t)$ selects the seasonal factor for the month that t falls in. For example, if t is 9 February 1993 then $s(t)$ is the seasonal parameter for February.

When there are multiple seasons specified, $s(t)$ is the product of the parameters for the seasons. For example, if SEASONS=(MONTH DAY), then $s(t)$ is the product of the seasonal parameter for the month that corresponds to the period t and the seasonal parameter for the day of the week that corresponds to period t. When the SEASONS= option is not specified, the seasonal factors $s(t)$ are not included in the model. See the section "Specifying Seasonality" on page 966 for more information about specifying multiple seasonal factors.

Updating Equations

This section shows the updating equations for the Winters method. In the following formula, x_t is the actual value of the series at time t; a_t is the smoothed value of the series at time t; b_t is the smoothed trend at time t; c_t is the smoothed quadratic trend at time t; $s_{t-1}(t)$ selects the old value of the seasonal factor that corresponds to time t before the seasonal factors are updated.

The estimates of the constant, linear, and quadratic trend parameters are updated by using the following equations:

For TREND=3,

$$a_t = \omega_1 \frac{x_t}{s_{t-1}(t)} + (1 - \omega_1)(a_{t-1} + b_{t-1} + c_{t-1})$$

$$b_t = \omega_2(a_t - a_{t-1} + c_{t-1}) + (1 - \omega_2)(b_{t-1} + 2c_{t-1})$$

$$c_t = \omega_2 \frac{1}{2}(b_t - b_{t-1}) + (1 - \omega_2)c_{t-1}$$

For TREND=2,

$$a_t = \omega_1 \frac{x_t}{s_{t-1}(t)} + (1 - \omega_1)(a_{t-1} + b_{t-1})$$

$$b_t = \omega_2(a_t - a_{t-1}) + (1 - \omega_2)b_{t-1}$$

For TREND=1,

$$a_t = \omega_1 \frac{x_t}{s_{t-1}(t)} + (1 - \omega_1)a_{t-1}$$

In this updating system, the trend polynomial is always centered at the current period so that the intercept parameter of the trend polynomial for predicted values at times after t is always the updated intercept parameter a_t. The predicted value for τ periods ahead is

$$x_{t+\tau} = (a_t + b_t\tau)s_t(t + \tau)$$

The seasonal parameters are updated when the season changes in the data, using the mean of the ratios of the actual to the predicted values for the season. For example, if SEASONS=MONTH and INTERVAL=DAY, then when the observation for the first of February is encountered, the seasonal parameter for January is updated by using the formula

$$s_t(t - 1) = \omega_3 \frac{1}{31} \sum_{i=t-31}^{t-1} \frac{x_i}{a_i} + (1 - \omega_3)s_{t-1}(t - 1)$$

where t is February 1 of the current year, $s_t(t - 1)$ is the seasonal parameter for January updated with the data available at time t, and $s_{t-1}(t - 1)$ is the seasonal parameter for January of the previous year.

When multiple seasons are used, $s_t(t)$ is a product of seasonal factors. For example, if SEASONS=(MONTH DAY) then $s_t(t)$ is the product of the seasonal factors for the month and for the day of the week: $s_t(t) = s_t^m(t)s_t^d(t)$.

The factor $s_t^m(t)$ is updated at the start of each month by using a modification of the preceding formula that adjusts for the presence of the other seasonal by dividing the summands $\frac{x_i}{a_i}$ by the that corresponds to day of the week effect $s_i^d(i)$.

Similarly, the factor $s_t^d(t)$ is updated by using the following formula:

$$s_t^d(t) = \omega_3 \frac{x_t}{a_t s_t^m(t)} + (1 - \omega_3)s_{t-1}^d(t)$$

where $s_{t-1}^d(t)$ is the seasonal factor for the same day of the previous week.

Missing values after the start of the series are replaced with one-step-ahead predicted values, and the predicted value is substituted for x_i and applied to the updating equations.

Normalization

The parameters are normalized so that the seasonal factors for each cycle have a mean of 1.0. This normalization is performed after each complete cycle and at the end of the data. Thus, if INTERVAL=MONTH and SEASONS=MONTH are specified and a series begins with a July value, then the seasonal factors for the series are normalized at each observation for July and at the last observation in the data set. The normalization is performed by dividing each of the seasonal parameters, and multiplying each of the trend parameters, by the mean of the unnormalized seasonal parameters.

Smoothing Weights

The weight for updating the seasonal factors, ω_3, is given by the third value specified in the WEIGHT= option. If the WEIGHT= option is not used, then ω_3 defaults to 0.25; if the WEIGHT= option is used but does not specify a third value, then ω_3 defaults to ω_2. The weight for updating the linear and quadratic trend parameters, ω_2, is given by the second value specified in the WEIGHT= option; if the WEIGHT= option does not specify a second value, then ω_2 defaults to ω_1. The updating weight for the constant parameter, ω_1, is given by the first value specified in the WEIGHT= option. As a general rule, smaller smoothing weights are appropriate for series with a slowly changing trend, while larger weights are appropriate for volatile series with a rapidly changing trend.

If the WEIGHT= option is not used, then ω_1 defaults to $(1 - 0.8^{1/trend})$, where *trend* is the value of the TREND= option. This produces defaults of WEIGHT=0.2 for TREND=1, WEIGHT=0.10557 for TREND=2, and WEIGHT=0.07168 for TREND=3.

The ESM procedure and the Time Series Forecasting System provide for generating forecast models that use Winters Method and enable you to specify or optimize the weights. (See Chapter 15, "The ESM Procedure," and Chapter 54, "Getting Started with Time Series Forecasting," for details.)

Confidence Limits

A method for calculating exact forecast confidence limits for the WINTERS method is not available. Therefore, the approach taken in PROC FORECAST is to assume that the true seasonal factors have small variability about a set of fixed seasonal factors and that the remaining variation of the series is small relative to the mean level of the series. The equations are written

$$s_t(t) = \mathrm{I}(t)(1 + \delta_t)$$

$$x_t = \mu \mathrm{I}(t)(1 + \gamma_t)$$

$$a_t = \xi(1 + \alpha_t)$$

where μ is the mean level and $\mathrm{I}(t)$ are the fixed seasonal factors. Assuming that α_t and δ_t are small, the forecast equations can be linearized and only first-order terms in δ_t and α_t kept. In terms of forecasts for γ_t, this linearized system is equivalent to a seasonal ARIMA model. Confidence limits for γ_t are based on this ARIMA model and converted into confidence limits for x_t using $s_t(t)$ as estimates of $\mathrm{I}(t)$.

The exponential smoothing confidence limits are based on an approximation to a weighted regression model, whereas the preceding Winters confidence limits are based on an approximation to an ARIMA model. You can use METHOD=WINTERS without the SEASONS= option to do exponential smoothing and get confidence limits for the EXPO forecasts based on the ARIMA model approximation. These are generally more pessimistic than the weighted regression confidence limits produced by METHOD=EXPO.

ADDWINTERS Method

The ADDWINTERS method is like the WINTERS method except that the seasonal parameters are added to the trend instead of multiplied with the trend. The default TREND=2 model is as follows:

$$x_t = a + bt + s(t) + \epsilon_t$$

The WINTERS method for updating equation and confidence limits calculations described in the preceding section are modified accordingly for the additive version.

Holt Two-Parameter Exponential Smoothing

If the seasonal factors are omitted (that is, if the SEASONS= option is not specified), the WINTERS (and ADDWINTERS) method reduces to the Holt two-parameter version of exponential smoothing. Thus, the WINTERS method is often referred to as the Holt-Winters method.

Double exponential smoothing is a special case of the Holt two-parameter smoother. The double exponential smoothing results can be duplicated with METHOD=WINTERS by omitting the SEASONS= option and appropriately setting the WEIGHT= option. Letting $\alpha = \omega(2-\omega)$ and $\beta = \omega/(2-\omega)$, the following statements produce the same forecasts:

```
proc forecast method=expo trend=2 weight=ω ...;

proc forecast method=winters trend=2 weight=(α, β) ...;
```

Although the forecasts are the same, the confidence limits are computed differently.

Choice of Weights for EXPO, WINTERS, and ADDWINTERS Methods

For the EXPO, WINTERS, and ADDWINTERS methods, properly chosen smoothing weights are of critical importance in generating reasonable results. There are several factors to consider in choosing the weights.

The noisier the data, the lower should be the weight given to the most recent observation. Another factor to consider is how quickly the mean of the time series is changing. If the mean of the series is changing rapidly, relatively more weight should be given to the most recent observation. The more stable the series over time, the lower should be the weight given to the most recent observation.

Note that the smoothing weights should be set separately for each series; weights that produce good results for one series might be poor for another series. Since PROC FORECAST does not have a feature to use different weights for different series, when forecasting multiple series with the EXPO, WINTERS, or ADDWINTERS method it might be desirable to use different PROC FORECAST steps with different WEIGHT= options.

For the Winters method, many combinations of weight values might produce unstable *noninvertible* models, even though all three weights are between 0 and 1. When the model is noninvertible, the forecasts depend strongly on values in the distant past, and predictions are determined largely by the starting values. Unstable models usually produce poor forecasts. The Winters model can be unstable even if the weights are optimally chosen to minimize the in-sample MSE. See Archibald (1990) for a detailed discussion of the unstable region of the parameter space of the Winters model.

Optimal weights and forecasts for exponential smoothing models can be computed by using the ESM and ARIMA procedures and by the Time Series Forecasting System.

Starting Values for EXPO, WINTERS, and ADDWINTERS Methods

The exponential smoothing method requires starting values for the smoothed values S_0, $S_0^{[2]}$, and $S_0^{[3]}$. The Winters and additive Winters methods require starting values for the trend coefficients and seasonal factors.

By default, starting values for the trend parameters are computed by a time trend regression over the first few observations for the series. Alternatively, you can specify the starting value for the trend parameters with the ASTART=, BSTART=, and CSTART= options.

The number of observations used in the time trend regression for starting values depends on the NSTART= option. For METHOD=EXPO, NSTART= beginning values of the series are used, and the coefficients of the time trend regression are then used to form the initial smoothed values S_0, $S_0^{[2]}$, and $S_0^{[3]}$.

For METHOD=WINTERS or METHOD=ADDWINTERS, n complete seasonal cycles are used to compute starting values for the trend parameter, where n is the value of the NSTART= option. For example, for monthly data the seasonal cycle is one year, so NSTART=2 specifies that the first 24 observations at the beginning of each series are used for the time trend regression used to calculate starting values.

The starting values for the seasonal factors for the WINTERS and ADDWINTERS methods are computed from seasonal averages over the first few complete seasonal cycles at the beginning of the series. The number of seasonal cycles averaged to compute starting seasonal factors is controlled by the NSSTART= option. For example, for monthly data with SEASONS=12 or SEASONS=MONTH, the first n January values are averaged to get the starting value for the January seasonal parameter, where n is the value of the NSSTART= option.

The $s_0(i)$ seasonal parameters are set to the ratio (for WINTERS) or difference (for ADDWINTERS) of the mean for the season to the overall mean for the observations used to compute seasonal starting values.

For example, if METHOD=WINTERS, INTERVAL=DAY, SEASON=(MONTH DAY), and NSTART=2 (the default), the initial seasonal parameter for January is the ratio of the mean value over days in the first two Januaries after the start of the series (that is, after the first nonmissing value) to the mean value for all days read for initialization of the seasonal factors. Likewise, the initial factor for Sundays is the ratio of the mean value for Sundays to the mean of all days read.

For the ASTART=, BSTART=, and CSTART= options, the values specified are associated with the variables in the VAR statement in the order in which the variables are listed (the first value with the first variable, the second value with the second variable, and so on). If there are fewer values than variables, default starting values are used for the later variables. If there are more values than variables, the extra values are ignored.

Specifying Seasonality

Seasonality of a time series is a regular fluctuation about a trend. This is called seasonality because the time of year is the most common source of periodic variation. For example, sales of home heating oil are regularly greater in winter than during other times of the year.

Seasonality can be caused by many things other than weather. In the United States, sales of nondurable goods are greater in December than in other months because of the Christmas shopping season. The term seasonality is also used for cyclical fluctuation at periods other than a year. Often, certain days of the week cause regular fluctuation in daily time series, such as increased spending on leisure activities during weekends.

Three kinds of seasonality are supported in PROC FORECAST: time-of-year, day-of-week, and time-of-day. The seasonal part of the model is specified by using the SEASONS= option. The values for the SEASONS= option are listed in Table 17.2.

Table 17.2 The SEASONS= Option

SEASONS= Value	Cycle Length	Type of Seasonality
QTR	yearly	time of year
MONTH	yearly	time of year
DAY	weekly	day of week
HOUR	daily	time of day

The three kinds of seasonality can be combined. For example, SEASONS=(MONTH DAY HOUR) specifies that 24 hour-of-day seasons are nested within 7 day-of-week seasons, which in turn are nested within 12 month-of-year seasons. The different kinds of intervals can be listed in the SEASONS= option in any order. Thus, SEASONS=(HOUR DAY MONTH) is the same as SEASONS=(MONTH DAY HOUR). Note that the Winters method smoothing equations might be less stable when multiple seasonal factors are used.

Multiple period seasons can also be used. For example, SEASONS=QTR2 specifies two semiannual time-of-year seasons. The grouping of observations into multiple period seasons starts with the first interval in the seasonal cycle. Thus, MONTH2 seasons are January–February, March–April, and so on. (There is no provision for shifting seasonal intervals; thus, there is no way to specify seasons December–January, February–March, April–May, and so on.)

For multiple period seasons, the number of intervals combined to form the seasons must evenly divide and be less than the basic cycle length. For example, with SEASONS=MONTH*n*, the basic cycle length is 12, so MONTH2, MONTH3, MONTH4, and MONTH6 are valid SEASONS= values (because 2, 3, 4, and 6 evenly divide 12 and are less than 12), but MONTH5 and MONTH12 are not valid SEASONS= values.

The frequency of the seasons must not be greater than the frequency of the input data. For example, you cannot specify SEASONS=MONTH if INTERVAL=QTR or SEASONS=MONTH if INTERVAL=MONTH2. You also cannot specify two seasons of the same basic cycle. For example, SEASONS=(MONTH QTR) or SEASONS=(MONTH2 MONTH4) is not allowed.

Alternatively, the seasonality can be specified by giving the number of seasons in the SEASONS= option. SEASONS=*n* specifies that there are n seasons, with observations 1, $n + 1$, $2n + 1$, and so on in the first season, observations 2, $n + 2$, $2n + 2$, and so on in the second season, and so forth.

The options SEASONS=*n* and SINTPER=*m* cause PROC FORECAST to group the input observations into n seasons, with m observations to a season, which repeat every nm observations. The options SEASONS=(n_1 n_2) and SINTPER=(m_1 m_2) produce n_1 seasons with m_1 observations to a season nested within n_2 seasons with $n_1 m_1 m_2$ observations to a season.

If the SINTPER=*m* option is used with the SEASONS= option, the SEASONS= interval is multiplied by the SINTPER= value. For example, specifying both SEASONS=(QTR HOUR) and SINTPER=(2 3) is the same as specifying SEASONS=(QTR2 HOUR3) and also the same as specifying SEASONS=(HOUR3 QTR2).

Data Requirements

You should have ample data for the series that you forecast by using PROC FORECAST. However, the results might be poor unless you have a good deal more than the minimum amount of data the procedure allows. The minimum number of observations required for the different methods is as follows:

- If METHOD=STEPAR is used, the minimum number of nonmissing observations required for each series forecast is the TREND= option value plus the value of the NLAGS= option. For example, using NLAGS=13 and TREND=2, at least 15 nonmissing observations are needed.
- If METHOD=EXPO is used, the minimum is the TREND= option value.
- If METHOD=WINTERS or ADDWINTERS is used, the minimum number of observations is either the number of observations in a complete seasonal cycle or the TREND= option value, whichever is greater. (However, there should be data for several complete seasonal cycles, or the seasonal factor estimates might be poor.) For example, for the seasonal specifications SEASONS=MONTH, SEASONS=(QTR DAY), or SEASONS=(MONTH DAY HOUR), the longest cycle length is one year, so at least one year of data is required. At least two years of data is recommended.

OUT= Data Set

The FORECAST procedure writes the forecast to the output data set named by the OUT= option. The OUT= data set contains the following variables:

- the BY variables
- _TYPE_, a character variable that identifies the type of observation
- _LEAD_, a numeric variable that indicates the number of steps ahead in the forecast. The value of _LEAD_ is 0 for the one-step-ahead forecasts before the start of the forecast period.
- the ID statement variables
- the VAR statement variables, which contain the result values as indicated by the _TYPE_ variable value for the observation

The FORECAST procedure processes each of the input variables listed in the VAR statement and writes several observations for each forecast period to the OUT= data set. The observations are identified by the value of the _TYPE_ variable. The options OUTACTUAL, OUTALL, OUTLIMIT, OUTRESID, OUT1STEP, OUTFULL, and OUTSTD control which types of observations are included in the OUT= data set.

The values of the variable _TYPE_ are as follows:

ACTUAL — The VAR statement variables contain actual values from the input data set. The OUTACTUAL option writes the actual values. By default, only the observations for the forecast period are output.

FORECAST — The VAR statement variables contain forecast values. The OUT1STEP option writes the one-step-ahead predicted values for the observations used to fit the model.

RESIDUAL — The VAR statement variables contain residuals. The residuals are computed by subtracting the forecast value from the actual value ($residual = actual - forecast$). The OUTRESID option writes observations for the residuals.

L*nn* — The VAR statement variables contain lower *nn* % confidence limits for the forecast values for the future observations specified by the LEAD= option. The value of *nn* depends on the ALPHA= option; with the default ALPHA=0.05, the _TYPE_ value is L95 for the lower confidence limit observations. The OUTLIMIT option writes observations for the upper and lower confidence limits.

U*nn* — The VAR statement variables contain upper *nn* % confidence limits for the forecast values for the future observations specified by the LEAD= option. The value of *nn* depends on the ALPHA= option; with the default ALPHA=0.05, the _TYPE_ value is U95 for the upper confidence limit observations. The OUTLIMIT option writes observations for the upper and lower confidence limits.

STD — The VAR statement variables contain standard errors of the forecast values. The OUTSTD option writes observations for the standard errors of the forecast.

If no output control options are specified, PROC FORECAST outputs only the forecast values for the forecast periods.

The _TYPE_ variable can be used to subset the OUT= data set. For example, the following data step splits the OUT= data set into two data sets, one that contains the forecast series and the other that contains the residual series. For example

```
proc forecast out=out outresid ...;
   ...
run;

data fore resid;
   set out;
   if _TYPE_='FORECAST' then output fore;
   if _TYPE_='RESIDUAL' then output resid;
run;
```

See Chapter 4, "Working with Time Series Data," for more information about processing time series data sets in this format.

OUTEST= Data Set

The FORECAST procedure writes the parameter estimates and goodness-of-fit statistics to an output data set when the OUTEST= option is specified. The OUTEST= data set contains the following variables:

- the BY variables
- the first ID variable, which contains the value of the ID variable for the last observation in the input data set used to fit the model

- _TYPE_, a character variable that identifies the type of each observation
- the VAR statement variables, which contain statistics and parameter estimates for the input series. The values contained in the VAR statement variables depend on the _TYPE_ variable value for the observation.

The observations contained in the OUTEST= data set are identified by the _TYPE_ variable. The OUTEST= data set might contain observations with the following _TYPE_ values:

AR1–AR*n* The observation contains estimates of the autoregressive parameters for the series. Two-digit lag numbers are used if the value of the NLAGS= option is 10 or more; in that case these _TYPE_ values are AR01–AR*n*. These observations are output for the STEPAR method only.

CONSTANT The observation contains the estimate of the constant or intercept parameter for the time trend model for the series. For the exponential smoothing and the Winters' methods, the trend model is centered (that is, t =0) at the last observation used for the fit.

LINEAR The observation contains the estimate of the linear or slope parameter for the time trend model for the series. This observation is output only if you specify TREND=2 or TREND=3.

N The observation contains the number of nonmissing observations used to fit the model for the series.

QUAD The observation contains the estimate of the quadratic parameter for the time trend model for the series. This observation is output only if you specify TREND=3.

SIGMA The observation contains the estimate of the standard deviation of the error term for the series.

S1–S3 The observations contain exponentially smoothed values at the last observation. _TYPE_=S1 is the final smoothed value of the single exponential smooth. _TYPE_=S2 is the final smoothed value of the double exponential smooth. _TYPE_=S3 is the final smoothed value of the triple exponential smooth. These observations are output for METHOD=EXPO only.

S_*name* The observation contains estimates of the seasonal parameters. For example, if SEASONS=MONTH, the OUTEST= data set contains observations with _TYPE_=S_JAN, _TYPE_=S_FEB, _TYPE_=S_MAR, and so forth.

For multiple-period seasons, the names of the first and last interval of the season are concatenated to form the season name. Thus, for SEASONS=MONTH4, the OUTEST= data set contains observations with _TYPE_=S_JANAPR, _TYPE_=S_MAYAUG, and _TYPE_=S_SEPDEC.

When the SEASONS= option specifies numbers, the seasonal factors are labeled _TYPE_=S_*i*_*j*. For example, SEASONS=(2 3) produces observations with _TYPE_ values of S_1_1, S_1_2, S_2_1, S_2_2, and S_2_3. The observation with _TYPE_=S_*i*_*j* contains the seasonal parameters for the *j*th season of the *i*th seasonal cycle.

These observations are output only for METHOD=WINTERS or METHOD=ADDWINTERS.

WEIGHT The observation contains the smoothing weight used for exponential smoothing. This is the value of the WEIGHT= option. This observation is output for METHOD=EXPO only.

WEIGHT1 | WEIGHT2 | WEIGHT3 The observations contain the weights used for smoothing the WINTERS or ADDWINTERS method parameters (specified by the WEIGHT= option). _TYPE_=WEIGHT1 is the weight used to smooth the CONSTANT parameter. _TYPE_=WEIGHT2 is the weight used to smooth the LINEAR and QUAD parameters. _TYPE_=WEIGHT3 is the weight used to smooth the seasonal parameters. These observations are output only for the WINTERS and ADDWINTERS methods.

NRESID The observation contains the number of nonmissing residuals, n, used to compute the goodness-of-fit statistics. The residuals are obtained by subtracting the one-step-ahead predicted values from the observed values.

SST The observation contains the total sum of squares for the series, corrected for the mean. $SST = \sum_{t=1}^{n} (y_t - \overline{y})^2$, where $\overline{y}$ is the series mean.

SSE The observation contains the sum of the squared residuals, uncorrected for the mean. $SSE = \sum_{t=1}^{n} (y_t - \hat{y}_t)^2$, where $\hat{y}$ is the one-step predicted value for the series.

MSE The observation contains the mean squared error, calculated from one-step-ahead forecasts. $MSE = \frac{1}{n-k} SSE$, where k is the number of parameters in the model.

RMSE The observation contains the root mean squared error. $RMSE = \sqrt{MSE}$.

MAPE The observation contains the mean absolute percent error.

$MAPE = \frac{100}{n} \sum_{t=1}^{n} |(y_t - \hat{y}_t)/y_t|$.

MPE The observation contains the mean percent error.

$MPE = \frac{100}{n} \sum_{t=1}^{n} (y_t - \hat{y}_t)/y_t$.

MAE The observation contains the mean absolute error. $MAE = \frac{1}{n} \sum_{t=1}^{n} |y_t - \hat{y}_t|$.

ME The observation contains the mean error. $MAE = \frac{1}{n} \sum_{t=1}^{n} (y_t - \hat{y}_t)$.

MAXE The observation contains the maximum error (the largest residual).

MINE The observation contains the minimum error (the smallest residual).

MAXPE The observation contains the maximum percent error.

MINPE The observation contains the minimum percent error.

RSQUARE The observation contains the R square statistic, $R^2 = 1 - SSE/SST$. If the model fits the series badly, the model error sum of squares *SSE* might be larger than *SST* and the R square statistic will be negative.

ADJRSQ The observation contains the adjusted R square statistic.

$ADJRSQ = 1 - (\frac{n-1}{n-k})(1 - R^2)$.

ARSQ The observation contains Amemiya's adjusted R square statistic.

$ARSQ = 1 - (\frac{n+k}{n-k})(1 - R^2)$.

RW_RSQ The observation contains the random walk R square statistic (Harvey's R^2_{D} statistic that uses the random walk model for comparison).

$RW\_RSQ = 1 - (\frac{n-1}{n}) SSE/RWSSE$,

where $RWSSE = \sum_{t=2}^{n} (y_t - y_{t-1} - \mu)^2$ and $\mu = \frac{1}{n-1} \sum_{t=2}^{n} (y_t - y_{t-1})$.

AIC The observation contains Akaike's information criterion.

$AIC = n \ln(SSE/n) + 2k$.

SBC	The observation contains Schwarz's Bayesian criterion. $SBC = n\ln(SSE/n) + k\ln(n)$.
APC	The observation contains Amemiya's prediction criterion. $APC = \frac{1}{n}SST(\frac{n+k}{n-k})(1-R^2) = (\frac{n+k}{n-k})\frac{1}{n}SSE$.
CORR	The observation contains the correlation coefficient between the actual values and the one-step-ahead predicted values.
THEILU	The observation contains Theil's U statistic that uses original units. See Maddala (1977, pp. 344–345), and Pindyck and Rubinfeld (1981, pp. 364–365) for more information about Theil statistics.
RTHEILU	The observation contains Theil's U statistic calculated using relative changes.
THEILUM	The observation contains the bias proportion of Theil's U statistic.
THEILUS	The observation contains the variance proportion of Theil's U statistic.
THEILUC	The observation contains the covariance proportion of Theil's U statistic.
THEILUR	The observation contains the regression proportion of Theil's U statistic.
THEILUD	The observation contains the disturbance proportion of Theil's U statistic.
RTHEILUM	The observation contains the bias proportion of Theil's U statistic, calculated by using relative changes.
RTHEILUS	The observation contains the variance proportion of Theil's U statistic, calculated by using relative changes.
RTHEILUC	The observation contains the covariance proportion of Theil's U statistic, calculated by using relative changes.
RTHEILUR	The observation contains the regression proportion of Theil's U statistic, calculated by using relative changes.
RTHEILUD	The observation contains the disturbance proportion of Theil's U statistic, calculated by using relative changes.

Examples: FORECAST Procedure

Example 17.1: Forecasting Auto Sales

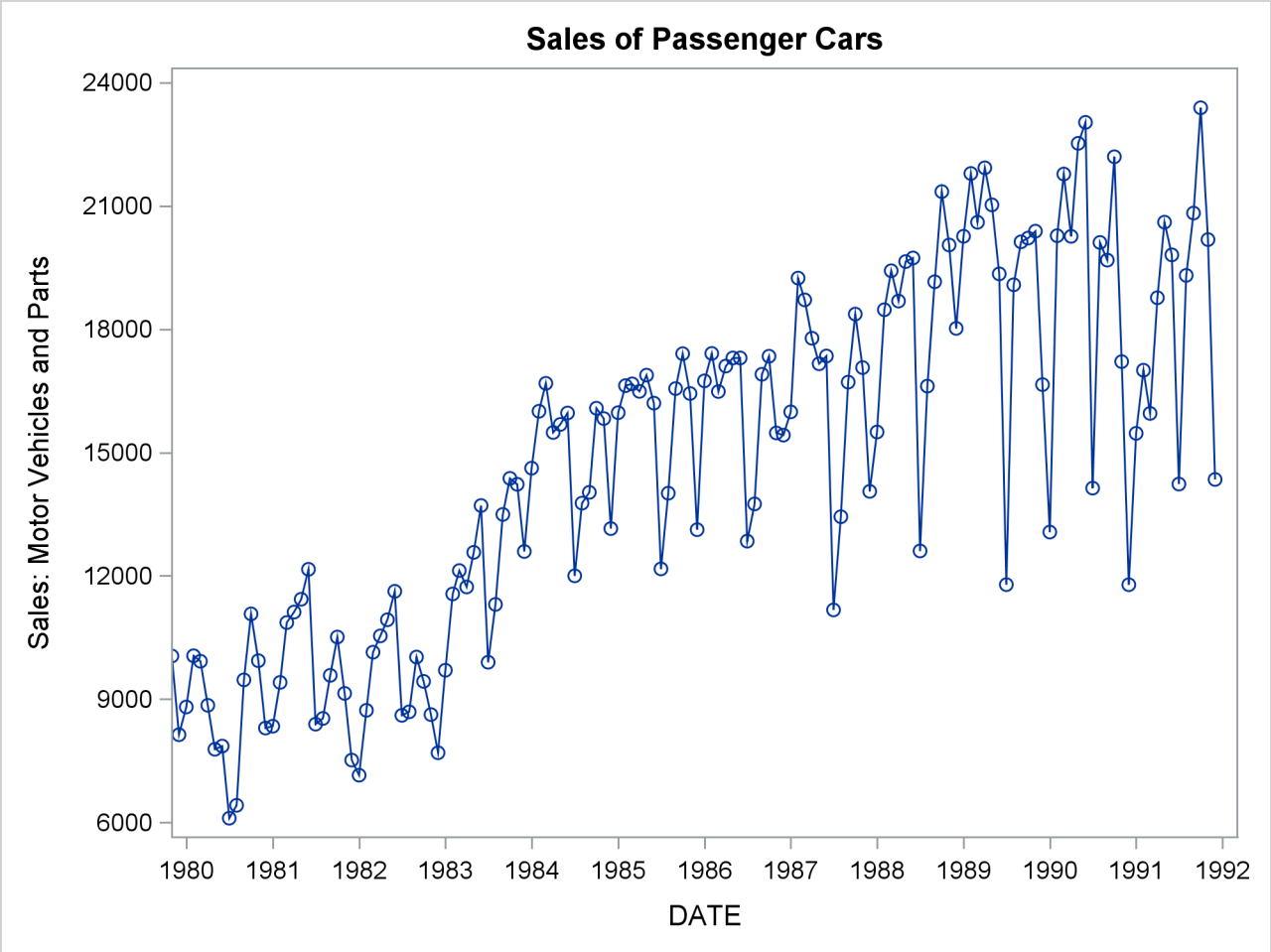
This example uses the Winters method to forecast the monthly U. S. sales of passenger cars series (VEHICLES) from the data set SASHELP.USECON. These data are taken from *Business Statistics*, published by the U. S. Bureau of Economic Analysis.

The following statements plot the series. The plot is shown in Output 17.1.1.

```
title1 "Sales of Passenger Cars";

symbol1 i=spline v=dot;
```

```
axis2 label=(a=-90 r=90 "Vehicles and Parts" )
      order=(6000 to 24000 by 3000);

title1 "Sales of Passenger Cars";
proc sgplot data=sashelp.usecon;
   series x=date y=vehicles / markers;
   xaxis values=('1jan80'd to '1jan92'd by year);
   yaxis values=(6000 to 24000 by 3000);
   format date year4.;
run;
```

Output 17.1.1 Monthly Passenger Car Sales



The following statements produce the forecast:

```
proc forecast data=sashelp.usecon interval=month
              method=winters seasons=month lead=12
              out=out outfull outresid outest=est;
   id date;
   var vehicles;
   where date >= '1jan80'd;
run;
```

The INTERVAL=MONTH option indicates that the data are monthly, and the ID DATE statement gives the dating variable. The METHOD=WINTERS specifies the Winters smoothing method. The LEAD=12 option forecasts 12 months ahead. The OUT=OUT option specifies the output data set, while the OUTFULL and OUTRESID options include in the OUT= data set the predicted and residual values for the historical period and the confidence limits for the forecast period. The OUTEST= option stores various statistics in an output data set. The WHERE statement is used to include only data from 1980 on.

The following statements print the OUT= data set (first 20 observations):

```
title2 'The OUT= Data Set';
proc print data=out (obs=20) noobs;
run;
```

The listing of the output data set produced by PROC PRINT is shown in part in Output 17.1.2.

Output 17.1.2 The OUT= Data Set Produced by PROC FORECAST (First 20 Observations)

Sales of Passenger Cars
The OUT= Data Set

DATE	_TYPE_	_LEAD_	VEHICLES
JAN80	ACTUAL	0	8808.00
JAN80	FORECAST	0	8046.52
JAN80	RESIDUAL	0	761.48
FEB80	ACTUAL	0	10054.00
FEB80	FORECAST	0	9284.31
FEB80	RESIDUAL	0	769.69
MAR80	ACTUAL	0	9921.00
MAR80	FORECAST	0	10077.33
MAR80	RESIDUAL	0	-156.33
APR80	ACTUAL	0	8850.00
APR80	FORECAST	0	9737.21
APR80	RESIDUAL	0	-887.21
MAY80	ACTUAL	0	7780.00
MAY80	FORECAST	0	9335.24
MAY80	RESIDUAL	0	-1555.24
JUN80	ACTUAL	0	7856.00
JUN80	FORECAST	0	9597.50
JUN80	RESIDUAL	0	-1741.50
JUL80	ACTUAL	0	6102.00
JUL80	FORECAST	0	6833.16

The following statements print the OUTEST= data set:

```
title2 'The OUTEST= Data Set: WINTERS Method';
proc print data=est;
run;
```

The PROC PRINT listing of the OUTEST= data set is shown in Output 17.1.3.

Output 17.1.3 The OUTEST= Data Set Produced by PROC FORECAST

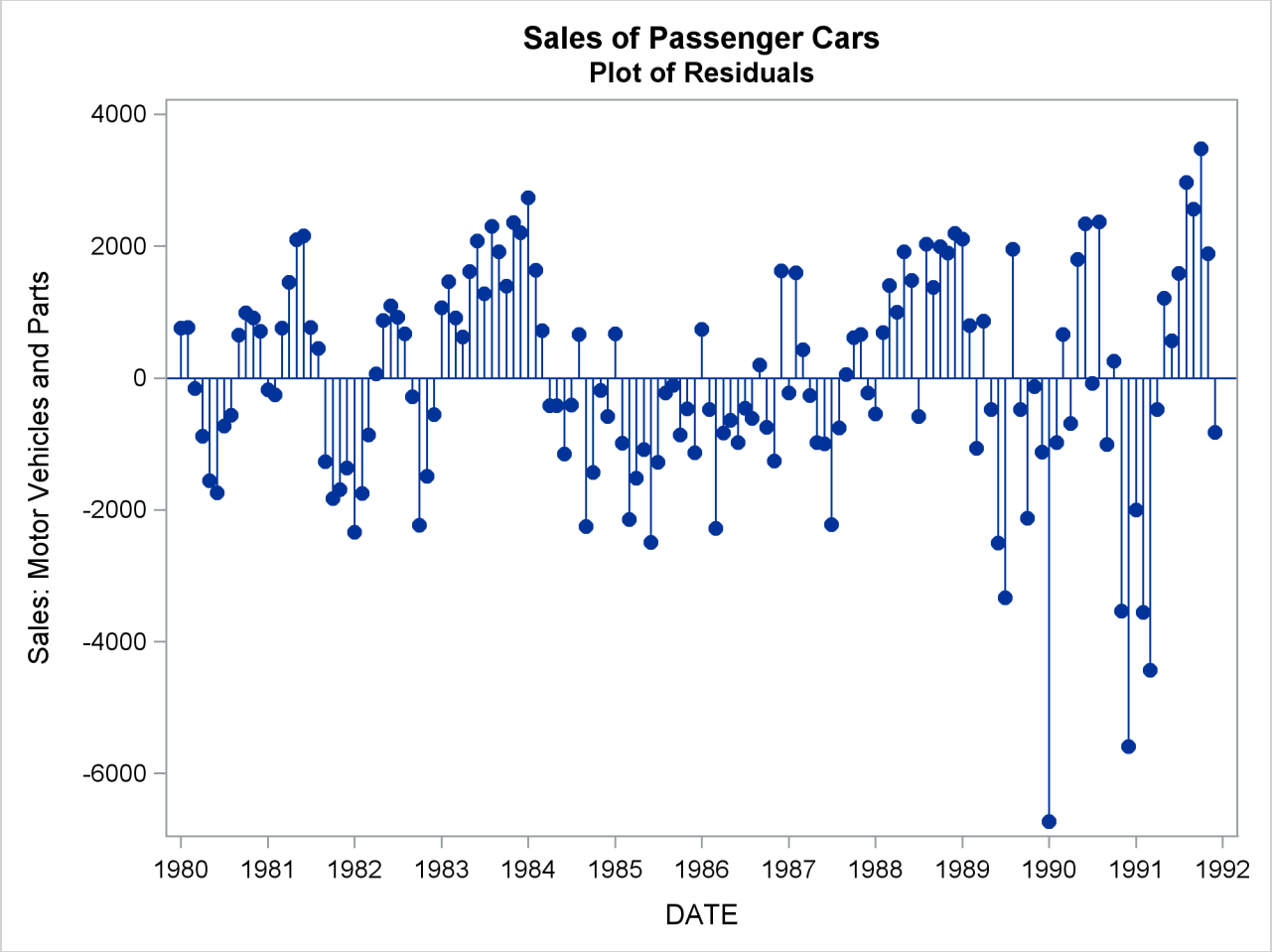
Sales of Passenger Cars
The OUTEST= Data Set: WINTERS Method

Obs	_TYPE_	DATE	VEHICLES
1	N	DEC91	144
2	NRESID	DEC91	144
3	DF	DEC91	130
4	WEIGHT1	DEC91	0.1055728
5	WEIGHT2	DEC91	0.1055728
6	WEIGHT3	DEC91	0.25
7	SIGMA	DEC91	1741.481
8	CONSTANT	DEC91	18577.368
9	LINEAR	DEC91	4.804732
10	S_JAN	DEC91	0.8909173
11	S_FEB	DEC91	1.0500278
12	S_MAR	DEC91	1.0546539
13	S_APR	DEC91	1.074955
14	S_MAY	DEC91	1.1166121
15	S_JUN	DEC91	1.1012972
16	S_JUL	DEC91	0.7418297
17	S_AUG	DEC91	0.9633888
18	S_SEP	DEC91	1.051159
19	S_OCT	DEC91	1.1399126
20	S_NOV	DEC91	1.0132126
21	S_DEC	DEC91	0.802034
22	SST	DEC91	2.63312E9
23	SSE	DEC91	394258270
24	MSE	DEC91	3032755.9
25	RMSE	DEC91	1741.481
26	MAPE	DEC91	9.4800217
27	MPE	DEC91	-1.049956
28	MAE	DEC91	1306.8534
29	ME	DEC91	-42.95376
30	RSQUARE	DEC91	0.8502696

The following statements plot the residuals. The plot is shown in Output 17.1.4.

```
title1 "Sales of Passenger Cars";
title2 'Plot of Residuals';
proc sgplot data=out;
   where _type_ = 'RESIDUAL';
   needle x=date y=vehicles / markers markerattrs=(symbol=circlefilled);
   xaxis values=('1jan80'd to '1jan92'd by year);
   format date year4.;
run;
```

Output 17.1.4 Residuals from Winters Method

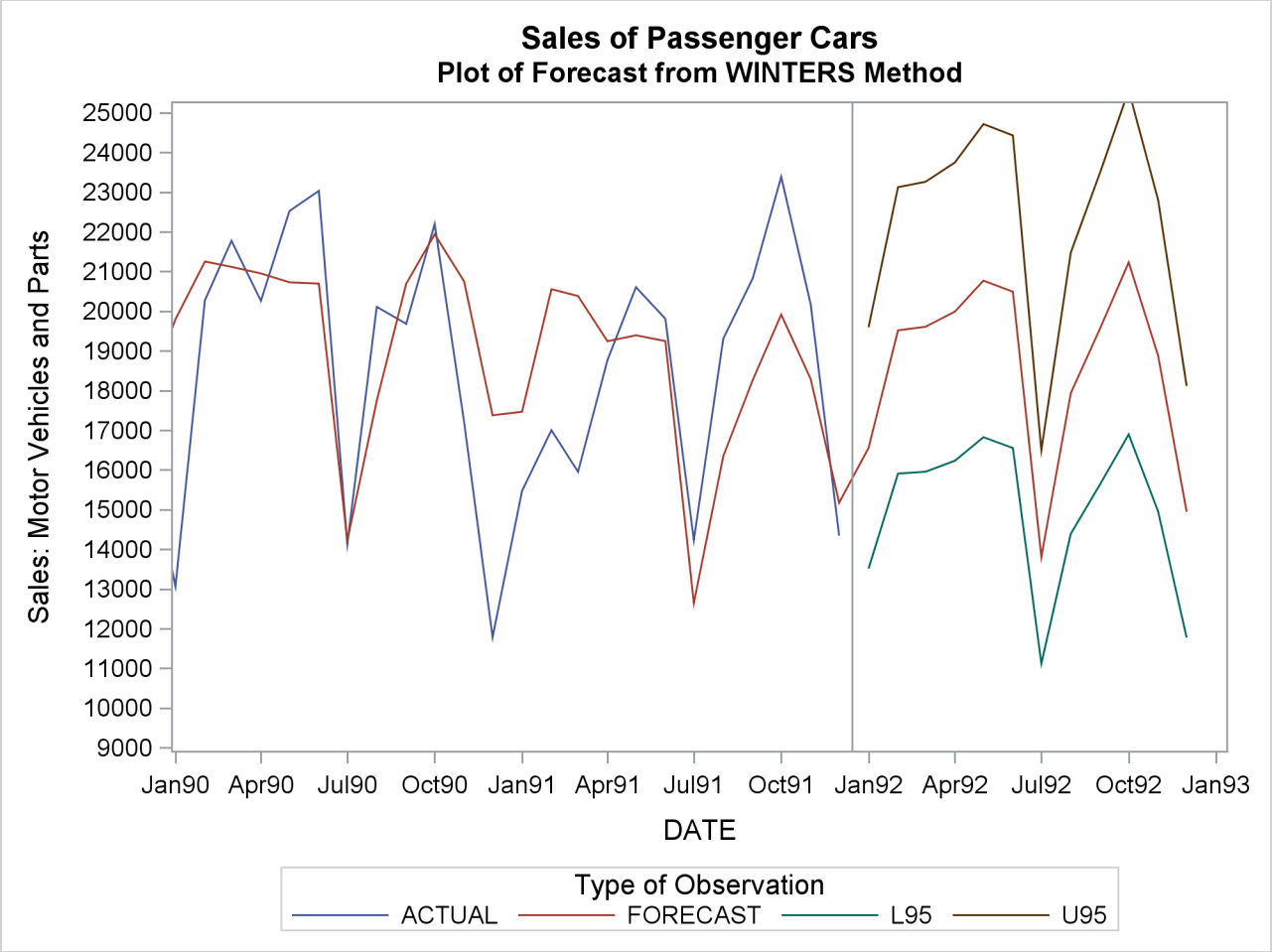


The following statements plot the forecast and confidence limits. The last two years of historical data are included in the plot to provide context for the forecast plot. A reference line is drawn at the start of the forecast period.

```
title1 "Sales of Passenger Cars";
title2 'Plot of Forecast from WINTERS Method';
proc sgplot data=out;
   series x=date y=vehicles / group=_type_ lineattrs=(pattern=1);
   where _type_ ^= 'RESIDUAL';
   refline '15dec91'd / axis=x;
   yaxis values=(9000 to 25000 by 1000);
   xaxis values=('1jan90'd to '1jan93'd by qtr);
run;
```

The plot is shown in Output 17.1.5.

Output 17.1.5 Forecast of Passenger Car Sales



Example 17.2: Forecasting Retail Sales

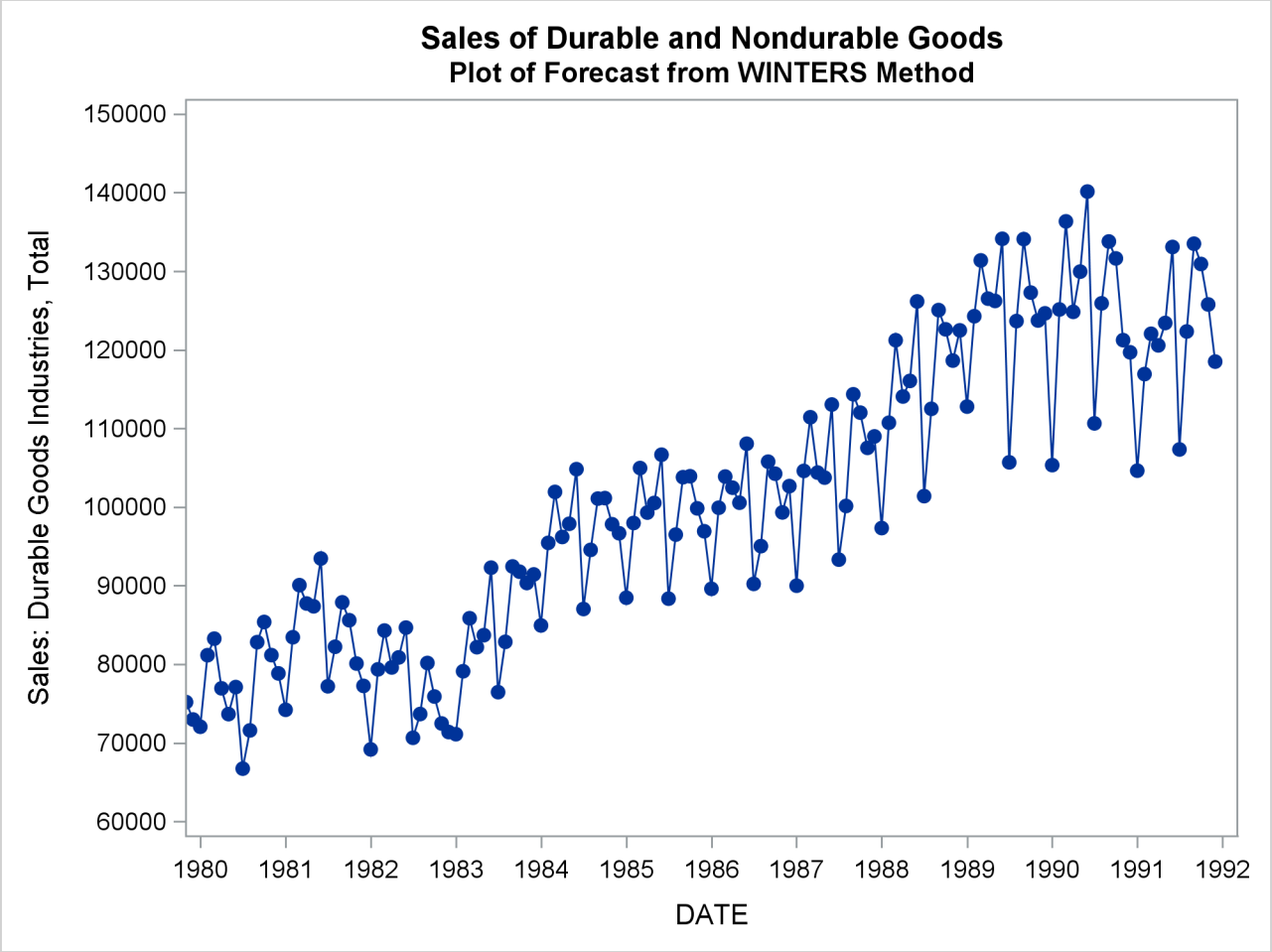
This example uses the stepwise autoregressive method to forecast the monthly U. S. sales of durable goods (DURABLES) and nondurable goods (NONDUR) from the SASHELP.USECON data set. The data are from *Business Statistics*, published by the U.S. Bureau of Economic Analysis. The following statements plot the series:

```
title1 'Sales of Durable and Nondurable Goods';
title2 'Plot of Forecast from WINTERS Method';
proc sgplot data=sashelp.usecon;
   series x=date y=durables / markers markerattrs=(symbol=circlefilled);
   xaxis values=('1jan80'd to '1jan92'd by year);
   yaxis values=(60000 to 150000 by 10000);
   format date year4.;
run;

title1 'Sales of Durable and Nondurable Goods';
title2 'Plot of Forecast from WINTERS Method';
proc sgplot data=sashelp.usecon;
```

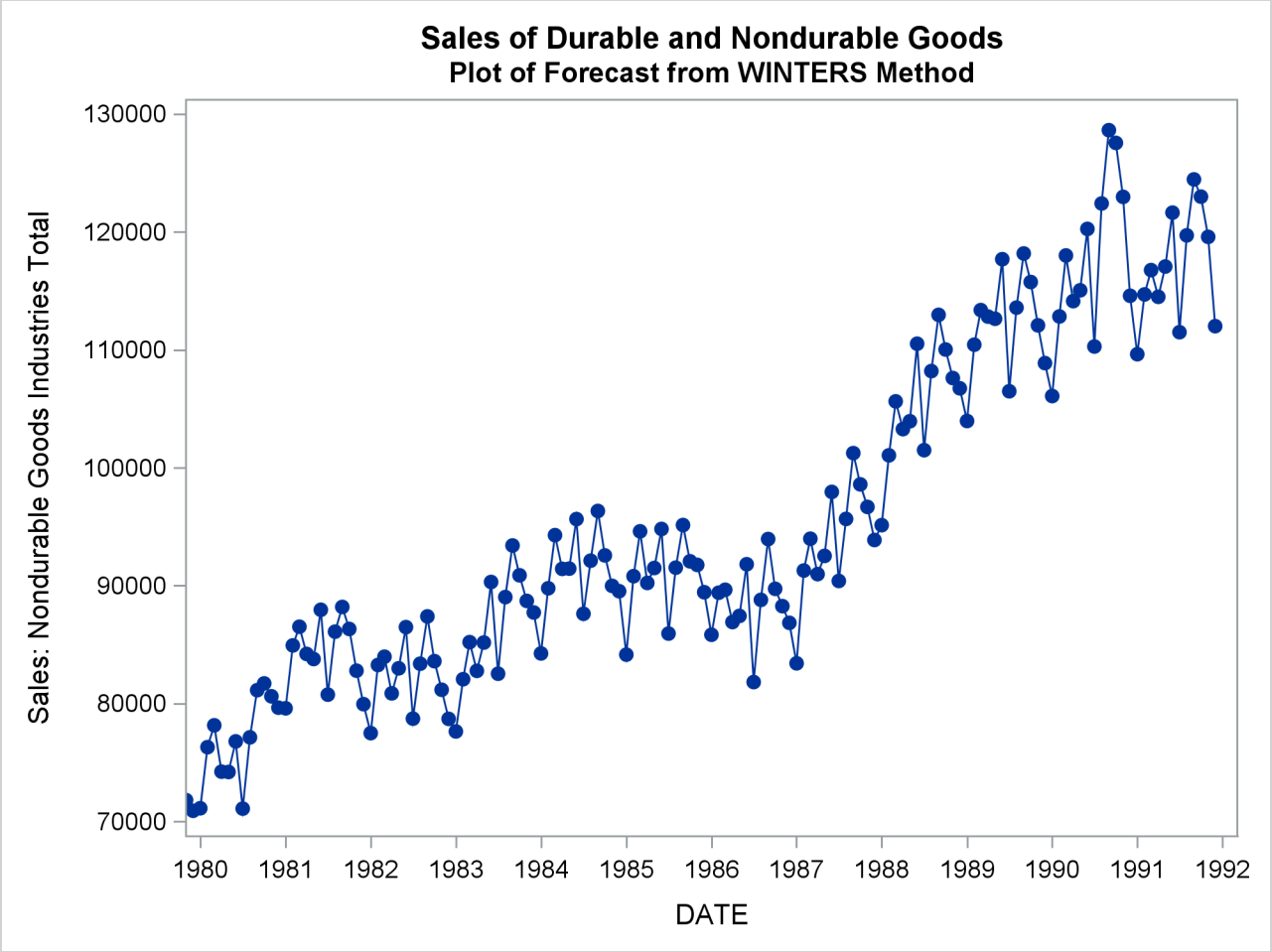
```
    series x=date y=nondur / markers markerattrs=(symbol=circlefilled);
    xaxis values=('1jan80'd to '1jan92'd by year);
    yaxis values=(70000 to 130000 by 10000);
    format date year4.;
run;
```

The plots are shown in Output 17.2.1 and Output 17.2.2.

Output 17.2.1 Durable Goods Sales



Output 17.2.2 Nondurable Goods Sales



The following statements produce the forecast:

```
title1 "Forecasting Sales of Durable and Nondurable Goods";

proc forecast data=sashelp.usecon interval=month
              method=stepar trend=2 lead=12
              out=out outfull outest=est;
   id date;
   var durables nondur;
   where date >= '1jan80'd;
run;
```

The following statements print the OUTEST= data set.

```
title2 'OUTEST= Data Set: STEPAR Method';
proc print data=est;
run;
```

The PROC PRINT listing of the OUTEST= data set is shown in Output 17.2.3.

Output 17.2.3 The OUTEST= Data Set Produced by PROC FORECAST

Forecasting Sales of Durable and Nondurable Goods
OUTEST= Data Set: STEPAR Method

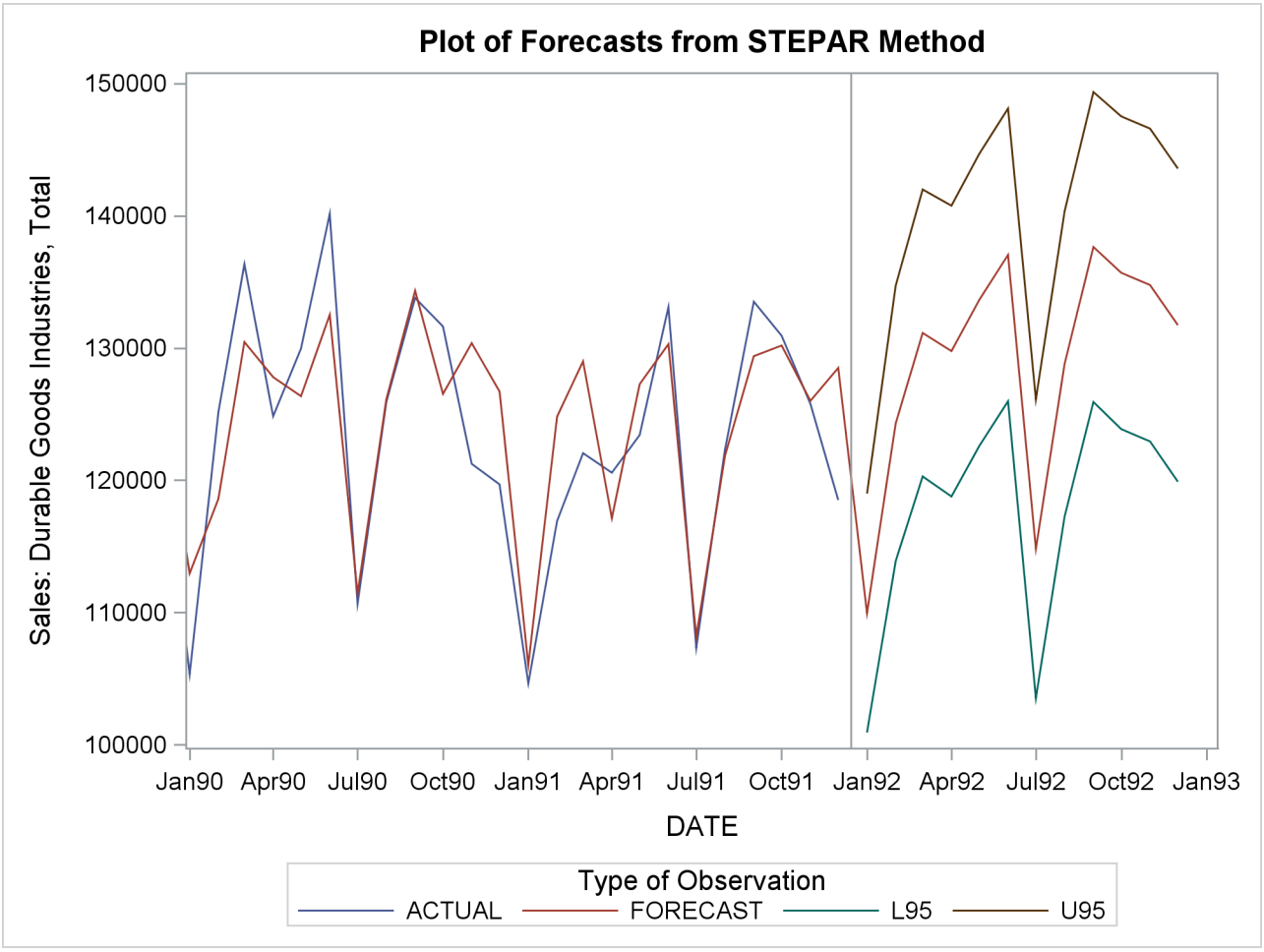
Obs	_TYPE_	DATE	DURABLES	NONDUR
1	N	DEC91	144	144
2	NRESID	DEC91	144	144
3	DF	DEC91	137	139
4	SIGMA	DEC91	4519.451	2452.2642
5	CONSTANT	DEC91	71884.597	73190.812
6	LINEAR	DEC91	400.90106	308.5115
7	AR01	DEC91	0.5844515	0.8243265
8	AR02	DEC91	.	.
9	AR03	DEC91	.	.
10	AR04	DEC91	.	.
11	AR05	DEC91	.	.
12	AR06	DEC91	0.2097977	.
13	AR07	DEC91	.	.
14	AR08	DEC91	.	.
15	AR09	DEC91	.	.
16	AR10	DEC91	-0.119425	.
17	AR11	DEC91	.	.
18	AR12	DEC91	0.6138699	0.8050854
19	AR13	DEC91	-0.556707	-0.741854
20	SST	DEC91	4.923E10	2.8331E10
21	SSE	DEC91	1.88157E9	544657337
22	MSE	DEC91	13734093	3918398.1
23	RMSE	DEC91	3705.9538	1979.4944
24	MAPE	DEC91	2.9252601	1.6555935
25	MPE	DEC91	-0.253607	-0.085357
26	MAE	DEC91	2866.675	1532.8453
27	ME	DEC91	-67.87407	-29.63026
28	RSQUARE	DEC91	0.9617803	0.9807752

The following statements plot the forecasts and confidence limits. The last two years of historical data are included in the plots to provide context for the forecast. A reference line is drawn at the start of the forecast period.

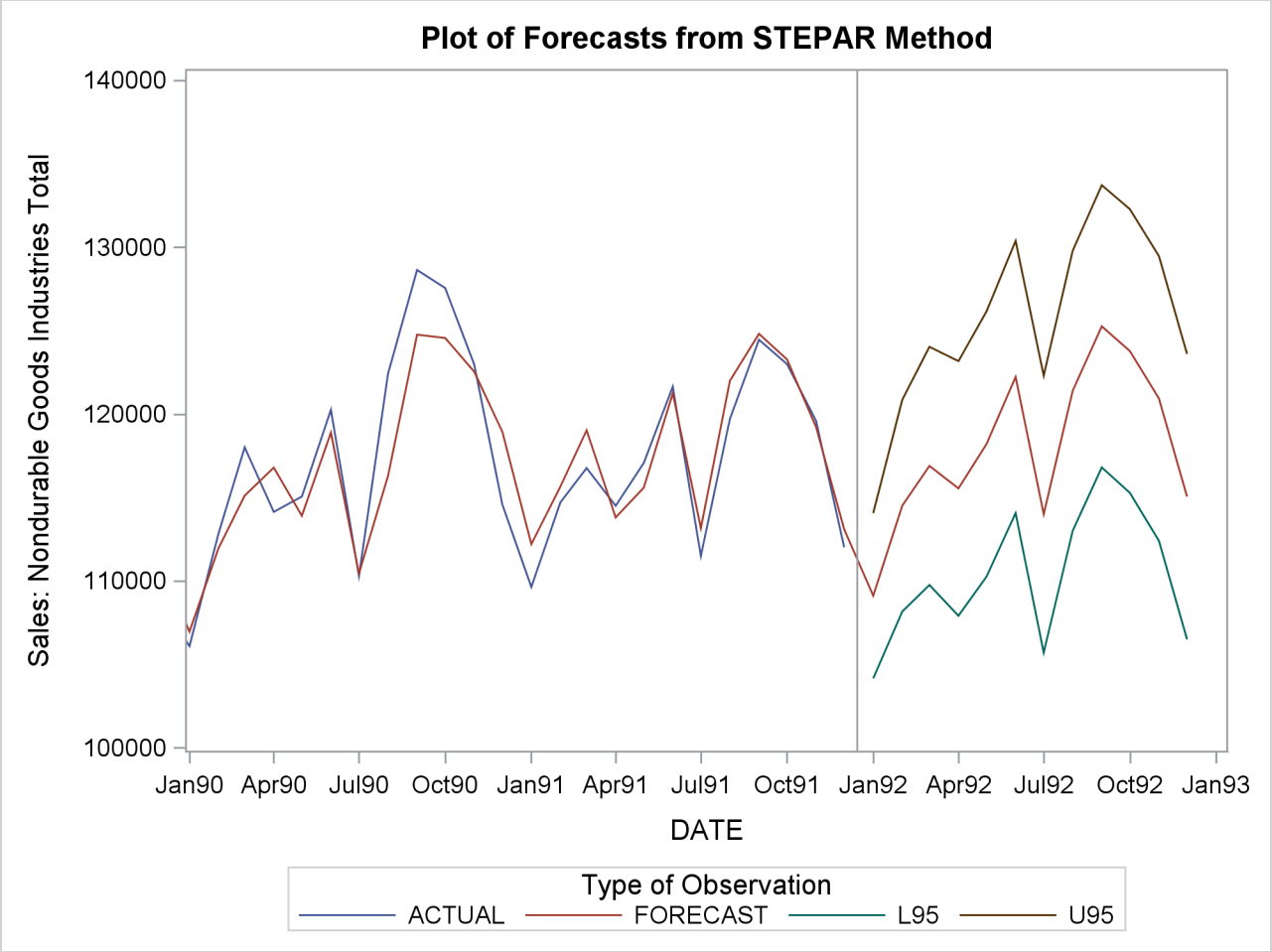
```
title1 'Plot of Forecasts from STEPAR Method';
proc sgplot data=out;
   series x=date y=durables / group=_type_;
   xaxis values=('1jan90'd to '1jan93'd by qtr);
   yaxis values=(100000 to 150000 by 10000);
   refline '15dec91'd / axis=x;
run;

proc sgplot data=out;
   series x=date y=nondur / group=_type_;
   xaxis values=('1jan90'd to '1jan93'd by qtr);
   yaxis values=(100000 to 140000 by 10000);
   refline '15dec91'd / axis=x;
run;
```

The plots are shown in Output 17.2.4 and Output 17.2.5.

Output 17.2.4 Forecast of Durable Goods Sales



Output 17.2.5 Forecast of Nondurable Goods Sales



Example 17.3: Forecasting Petroleum Sales

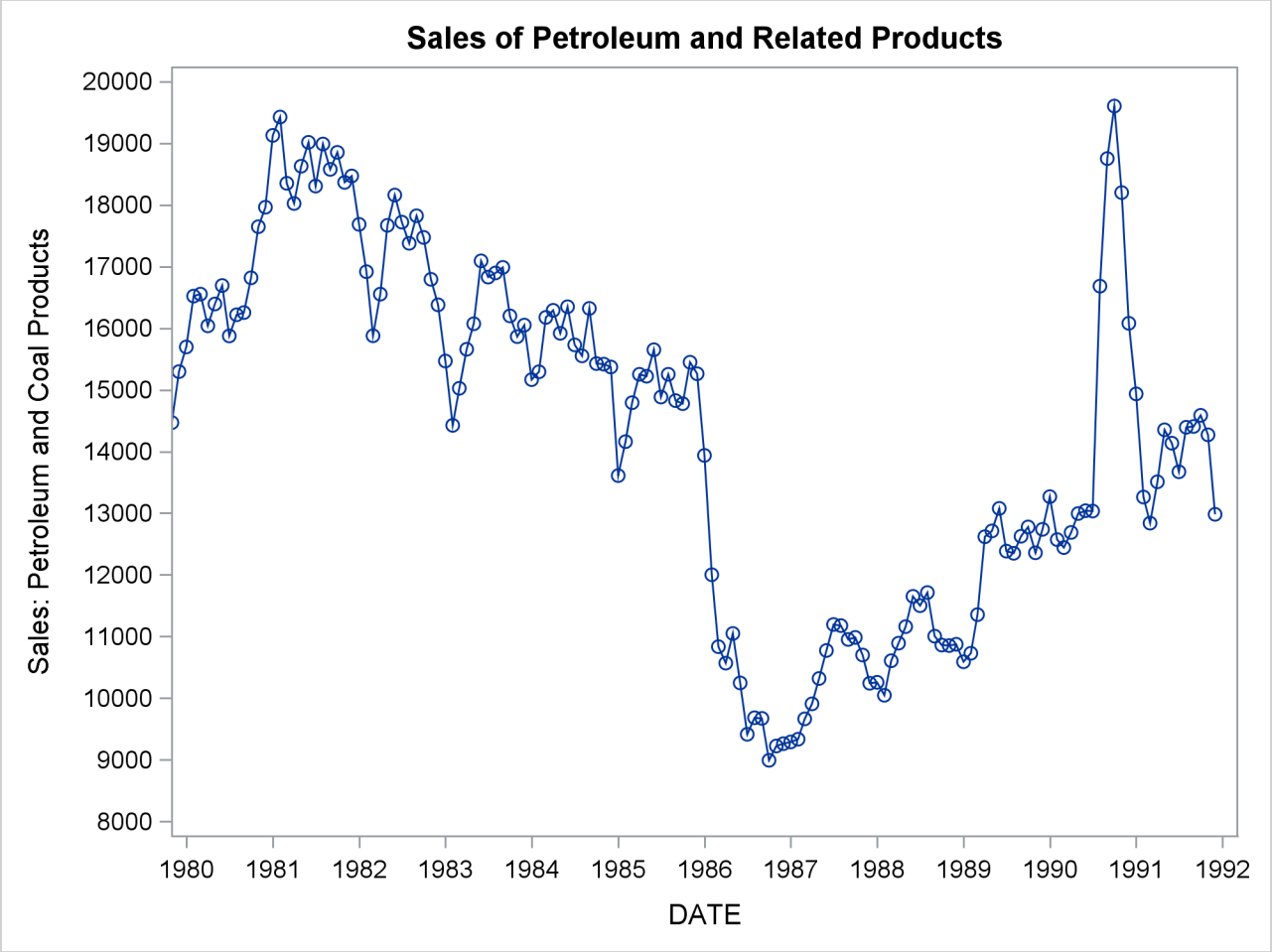
This example uses the double exponential smoothing method to forecast the monthly U. S. sales of petroleum and related products series (PETROL) from the data set SASHELP.USECON. These data are taken from *Business Statistics*, published by the U.S. Bureau of Economic Analysis.

The following statements plot the PETROL series:

```
title1 "Sales of Petroleum and Related Products";
proc sgplot data=sashelp.usecon;
   series x=date y=petrol / markers;
   xaxis values=('1jan80'd to '1jan92'd by year);
   yaxis values=(8000 to 20000 by 1000);
   format date year4.;
run;
```

The plot is shown in Output 17.3.1.

Output 17.3.1 Sales of Petroleum and Related Products



The following statements produce the forecast:

```
proc forecast data=sashelp.usecon interval=month
              method=expo trend=2 lead=12
              out=out outfull outest=est;
   id date;
   var petrol;
   where date >= '1jan80'd;
run;
```

The following statements print the OUTEST= data set:

```
title2 'OUTEST= Data Set: EXPO Method';
proc print data=est;
run;
```

The PROC PRINT listing of the output data set is shown in Output 17.3.2.

Output 17.3.2 The OUTEST= Data Set Produced by PROC FORECAST

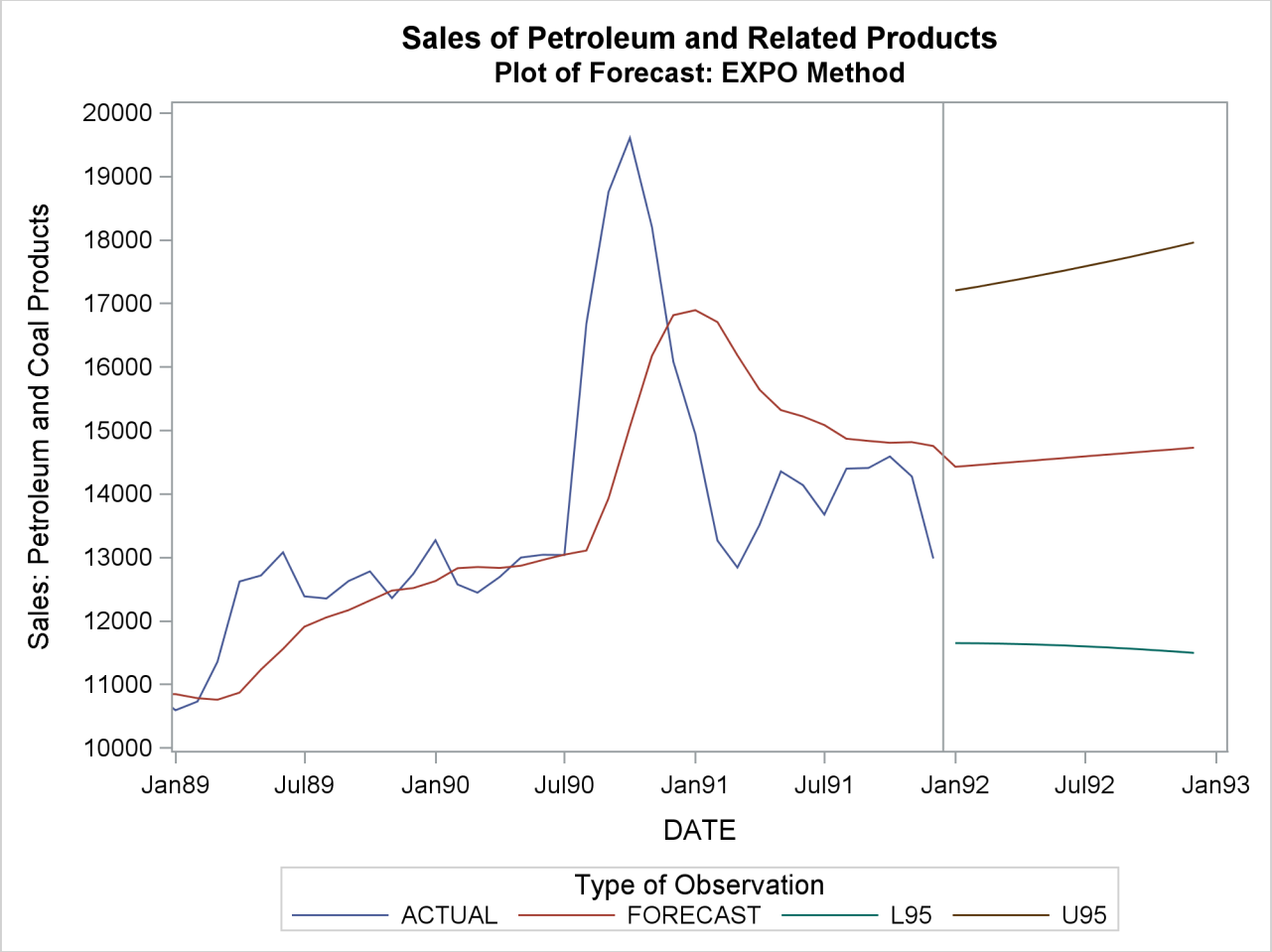
Sales of Petroleum and Related Products
OUTEST= Data Set: EXPO Method

Obs	_TYPE_	DATE	PETROL
1	N	DEC91	144
2	NRESID	DEC91	144
3	DF	DEC91	142
4	WEIGHT	DEC91	0.1055728
5	S1	DEC91	14165.259
6	S2	DEC91	13933.435
7	SIGMA	DEC91	1281.0945
8	CONSTANT	DEC91	14397.084
9	LINEAR	DEC91	27.363164
10	SST	DEC91	1.17001E9
11	SSE	DEC91	233050838
12	MSE	DEC91	1641203.1
13	RMSE	DEC91	1281.0945
14	MAPE	DEC91	6.5514467
15	MPE	DEC91	-0.147168
16	MAE	DEC91	891.04243
17	ME	DEC91	8.2148584
18	RSQUARE	DEC91	0.8008122

The plot of the forecast is shown in Output 17.3.3.

```
title1 "Sales of Petroleum and Related Products";
title2 'Plot of Forecast: EXPO Method';
proc sgplot data=out;
   series x=date y=petrol / group=_type_;
   xaxis values=('1jan89'd to '1jan93'd by qtr);
   yaxis values=(10000 to 20000 by 1000);
   refline '15dec91'd / axis=x;
run;
```

Output 17.3.3 Forecast of Petroleum and Related Products



References

Ahlburg, D. A. (1984). "Forecast Evaluation and Improvement Using Theil's Decomposition." *Journal of Forecasting* 3:345–351.

Aldrin, M., and Damsleth, E. (1989). "Forecasting Non-seasonal Time Series with Missing Observations." *Journal of Forecasting* 8:97–116.

Archibald, B. C. (1990). "Parameter Space of the Holt-Winters' Model." *International Journal of Forecasting* 6:199–209.

Bails, D. G., and Peppers, L. C. (1982). *Business Fluctuations: Forecasting Techniques and Applications*. Englewood Cliffs, NJ: Prentice-Hall.

Bartolomei, S. M., and Sweet, A. L. (1989). "A Note on the Comparison of Exponential Smoothing Methods for Forecasting Seasonal Series." *International Journal of Forecasting* 5:111–116.

Bliemel, F. (1973). "Theil's Forecast Accuracy Coefficient: A Clarification." *Journal of Marketing Research* 10:444–446.

Bowerman, B. L., and O'Connell, R. T. (1979). *Time Series and Forecasting: An Applied Approach*. North Scituate, MA: Duxbury Press.

Box, G. E. P., and Jenkins, G. M. (1976). *Time Series Analysis: Forecasting and Control*. Rev. ed. San Francisco: Holden-Day.

Bretschneider, S. I., Carbone, R., and Longini, R. L. (1979). "An Adaptive Approach to Time Series Forecasting." *Decision Sciences* 10:232–244.

Brown, R. G. (1962). *Smoothing, Forecasting, and Prediction of Discrete Time Series*. New York: Prentice-Hall.

Brown, R. G., and Meyer, R. F. (1961). "The Fundamental Theorem of Exponential Smoothing." *Operations Research* 9:673–685.

Bureau of Economic Analysis (1992). *Business Statistics*. 27th ed. Washington, DC: US Department of Commerce.

Chatfield, C. (1978). "The Holt-Winters Forecasting Procedure." *Journal of the Royal Statistical Society, Series C* 27:264–279.

Chatfield, C., and Prothero, D. L. (1973). "Box-Jenkins Seasonal Forecasting: Problems in a Case Study." *Journal of the Royal Statistical Society, Series A* 136:295–315.

Chow, W. M. (1965). "Adaptive Control of the Exponential Smoothing Constant." *Journal of Industrial Engineering* 16:314–317.

Cogger, K. O. (1974). "The Optimality of General-Order Exponential Smoothing." *Operations Research* 22:858–867.

Cox, D. R. (1961). "Prediction by Exponentially Weighted Moving Averages and Related Methods." *Journal of the Royal Statistical Society, Series B* 23:414–422.

Fair, R. C. (1986). "Evaluating the Predictive Accuracy of Models." In *Handbook of Econometrics*, vol. 3, edited by Z. Griliches, and M. D. Intriligator, 1979–1995. New York: North-Holland.

Fildes, R. (1979). "Quantitative Forecasting—the State of the Art: Extrapolative Models." *Journal of the Operational Research Society* 30:691–710.

Gardner, E. S., Jr. (1984). "The Strange Case of the Lagging Forecasts." *Interfaces* 14:47–50.

Gardner, E. S., Jr. (1985). "Exponential Smoothing: The State of the Art." *Journal of Forecasting* 4:1–38.

Granger, C. W. J., and Newbold, P. (1977). *Forecasting Economic Time Series*. New York: Academic Press.

Harvey, A. C. (1984). "A Unified View of Statistical Forecasting Procedures." *Journal of Forecasting* 3:245–275.

Ledolter, J., and Abraham, B. (1984). "Some Comments on the Initialization of Exponential Smoothing." *Journal of Forecasting* 3:79–84.

Maddala, G. S. (1977). *Econometrics*. New York: McGraw-Hill.

Makridakis, S. G., Wheelwright, S. C., and McGee, V. E. (1983). *Forecasting: Methods and Applications*. 2nd ed. New York: John Wiley & Sons.

McKenzie, E. (1984). "General Exponential Smoothing and the Equivalent ARMA Process." *Journal of Forecasting* 3:333–344.

Montgomery, D. C., and Johnson, L. A. (1976). *Forecasting and Time Series Analysis*. New York: McGraw-Hill.

Muth, J. F. (1960). "Optimal Properties of Exponentially Weighted Forecasts." *Journal of the American Statistical Association* 55:299–306.

Pierce, D. A. (1979). "R^2 Measures for Time Series." *Journal of the American Statistical Association* 74:901–910.

Pindyck, R. S., and Rubinfeld, D. L. (1981). *Econometric Models and Econometric Forecasts*. 2nd ed. New York: McGraw-Hill.

Raine, J. E. (1971). "Self-Adaptive Forecasting Reconsidered." *Decision Sciences* 2:181–191.

Roberts, S. A. (1982). "A General Class of Holt-Winters Type Forecasting Models." *Management Science* 28:808–820.

Theil, H. (1966). *Applied Economic Forecasting*. Amsterdam: North-Holland.

Trigg, D. W., and Leach, A. G. (1967). "Exponential Smoothing with an Adaptive Response Rate." *Operational Research Quarterly* 18:53–59.

Winters, P. R. (1960). "Forecasting Sales by Exponentially Weighted Moving Averages." *Management Science* 6:324–342.

Chapter 18
The HPCDM Procedure (Experimental)

Contents

Overview: HPCDM Procedure **990**
Getting Started: HPCDM Procedure **992**
Estimating a Simple Compound Distribution Model 992
Analyzing the Effect of Parameter Uncertainty on the Compound Distribution 996
Scenario Analysis 998
Syntax: HPCDM Procedure **1006**
Functional Summary 1006
PROC HPCDM Statement 1008
BY Statement 1013
DISTBY Statement 1013
EXTERNALCOUNTS Statement 1014
OUTPUT Statement 1014
OUTSUM Statement 1015
PERFORMANCE Statement 1018
SEVERITYMODEL Statement 1018
Programming Statements 1019
Details: HPCDM Procedure **1019**
Specifying Scenario Data in the DATA= Data Set 1019
Simulation Procedure 1020
Simulation of Adjusted Compound Distribution Sample 1027
Parameter Perturbation Analysis 1035
Descriptive Statistics 1035
Input Specification 1037
Output Data Sets 1038
Displayed Output 1040
ODS Graphics 1041
Examples: HPCDM Procedure **1043**
Example 18.1: Estimating the Probability Distribution of Insurance Payments 1043
Example 18.2: Using Externally Simulated Count Data 1047
References **1055**

Overview: HPCDM Procedure

In many loss modeling applications, the loss events are analyzed by modeling the severity (magnitude) of loss and the frequency (count) of loss separately. The primary goal of preparing these models is to estimate the aggregate loss—that is, the total loss that occurs over a period of time for which the frequency model is applicable. For example, an insurance company might want to assess the expected and worst-case losses for a particular business line, such as automobile insurance, over an entire year given the models for the number of losses in a year and the severity of each loss. A bank might want to assess the value-at-risk (VaR), a measure of the worst-case loss, for a portfolio of assets given the frequency and severity models for each asset type.

Loss severity and loss frequency are random variables, so the aggregate loss is also a random variable. Instead of preparing a point estimate of the expected aggregate loss, it is more desirable to estimate its probability distribution, because this enables you to infer various aspects of the aggregate loss such as measures of location, scale (variability), and shape in addition to percentiles. For example, the value-at-risk that banks or insurance companies use to compute regulatory capital requirements is usually the estimate of the 97.5th or 99th percentile from the aggregate loss distribution.

Let N represent the frequency random variable for the number of loss events that occur in the time period of interest. Let X represent the severity random variable for the magnitude of one loss event. Then, the aggregate loss S is defined as

$$S = \sum_{j=1}^{N} X_j$$

The goal is to estimate the probability distribution of S. Let $F_X(x)$ denote the cumulative distribution function (CDF) of X, $F_X^{*n}(x)$ denote the n-fold convolution of the CDF of X, and $\Pr(N = n)$ denote the probability of seeing n losses as per the frequency distribution. The CDF of S is theoretically computable as

$$F_S(s) = \sum_{n=0}^{\infty} \Pr(N = n) \cdot F_X^{*n}(x)$$

This probability distribution model of S, characterized by the CDF $F_S(s)$, is referred to as a *compound distribution model* (CDM). The HPCDM procedure computes an estimate of the CDM, given the distribution models of X and N.

PROC HPCDM accepts the severity model of X as estimated by the SEVERITY procedure. It accepts the frequency model of N as estimated by the COUNTREG procedure. Both the SEVERITY and COUNTREG procedures are part of SAS/ETS software. Both procedures allow models of X and N to be conditional on external factors (regressors). In particular, you can model the severity distribution such that its scale parameter depends on severity regressors, and you can model the frequency distribution such that its mean depends on frequency regressors. The frequency model can also be a zero-inflated model. PROC HPCDM uses the estimates of model parameters and the values of severity and frequency regressors to estimate the compound distribution model.

Direct computation of F_S is usually a difficult task because of the need to compute the n-fold convolution. Klugman, Panjer, and Willmot (1998, Ch. 4) suggest some relatively efficient recursion and inversion methods for certain combinations of severity and frequency distributions. However, those methods assume that distributions of N and X are fixed and all Xs are identically distributed. When the distributions of X

and N are conditional on regressors, each set of regressor values results in a different distribution. So you must repeat the recursion and inversion methods for each combination of regressor values, and this repetition makes these methods prohibitively expensive. PROC HPCDM instead estimates the compound distribution by using a Monte Carlo simulation method, which can use all available computational resources to generate a sufficiently large, representative sample of the compound distribution while accommodating the dependence of distributions of X and N on external factors. Conceptually, the simulation method works as follows:

1. Use the specified frequency model to draw a value N, which represents the number of loss events.
2. Use the specified severity model to draw N values, each of which represents the magnitude of loss for each of the N loss events.
3. Add the N severity values from step 2 to compute aggregate loss S as

$$S = \sum_{j=1}^{N} X_j$$

   This forms one sample point of the CDM.

Steps 1 through 3 are repeated M number of times, where M is specified by you, to obtain the representative sample of the CDM. PROC HPCDM analyzes this sample to compute empirical estimates of various summary statistics of the compound distribution such as the mean, variance, skewness, and kurtosis in addition to percentiles such as the median, the 95th percentile, the 99th percentile, and so on. You can also use PROC HPCDM to write the entire simulated sample to an output data set and to produce the plot of the empirical distribution function (EDF), which serves as a nonparametric estimate of F_S.

The simulation process gets more complicated when the frequency and severity models contain regression effects. The CDM is then conditional on the given values of regressors. The simulation process essentially becomes a scenario analysis, because you need to specify the expected values of the regressors that together represent the scenario for which you want to estimate the CDM. PROC HPCDM enables you to specify an input data set that contains the scenario. If you are modeling a group of entities together (such as a portfolio of multiple assets or a group of insurance policies), each with a different set of characteristics, then the scenario consists of more than one observation, and each observation corresponds to a different entity. PROC HPCDM enables you to specify such a group scenario in the input data set and performs a realistic simulation of loss events that each entity can generate.

PROC HPCDM also enables you to specify externally simulated counts. This is useful if you have an empirical frequency model or if you estimate the frequency model by using a method other than PROC COUNTREG and simulate counts by using such a model. You can specify M replications of externally simulated counts. For each of the replications, in step 1 of the simulation, instead of using the frequency model, PROC HPCDM uses the count N that you specify. If the severity model contains regression effects, then you can specify the scenario to simulate for each of the M replications.

If the parameters of your severity and frequency models have uncertainty associated with them, and they usually do, then you can use PROC HPCDM to conduct parameter perturbation analysis to assess the effect of parameter uncertainty on the estimates of CDM. If you specify that P perturbed samples be generated, then the parameter set is perturbed P times, and each time PROC HPCDM makes a random draw from either the univariate normal distribution of each parameter or the multivariate normal distribution over all parameters. For each of the P perturbed parameter sets, a full compound distribution sample is simulated and summarized.

This process yields P number of estimates for each summary statistic and percentile, which are then used to provide you with estimates of the location and variability of each summary statistic and percentile.

You can also use PROC HPCDM to compute the distribution of an aggregate *adjusted* loss. For example, in insurance applications, you might want to compute the distribution of the *amount paid* in a given time period after applying adjustments such as deductible and policy limit to each individual loss. PROC HPCDM enables you to specify SAS programming statements to adjust each severity value. If X_j^a represents the adjusted severity value, then PROC HPCDM computes S^a, an aggregate adjusted loss, as

$$S^a = \sum_{j=1}^{N} X_j^a$$

All the analyses that PROC HPCDM conducts for the aggregate unadjusted loss, including scenario analysis and parameter perturbation analysis, are also conducted for the aggregate adjusted loss, thereby giving you a comprehensive picture of the adjusted compound distribution model.

Getting Started: HPCDM Procedure

This section outlines the use of the HPCDM procedure to fit compound distribution models. The examples are intended as a gentle introduction to some of the features of the procedure.

Estimating a Simple Compound Distribution Model

This example illustrates the simplest use of PROC HPCDM. Assume that you are an insurance company that has used the historical data about the number of losses per year and the severity of each loss to determine that the Poisson distribution is the best distribution for the loss frequency and that the gamma distribution is the best distribution for the severity of each loss. Now, you want to estimate the distribution of an aggregate loss to determine the worst-case loss that can be incurred by your policyholders in a year. In other words, you want to estimate the compound distribution of $S = \sum_{i=1}^{N} X_i$, where the loss frequency, N, follows the fitted Poisson distribution and the severity of each loss event, X_i, follows the fitted gamma distribution.

If your historical count and severity data are stored in the data sets Work.ClaimCount and Work.ClaimSev, respectively, then you need to ensure that you use the following PROC COUNTREG and PROC SEVERITY steps to fit and store the parameter estimates of the frequency and severity models:

```
/* Fit an intercept-only Poisson count model and
   write estimates to an item store */
proc countreg data=claimcount;
   model numLosses= / dist=poisson;
   store countStorePoisson;
run;

/* Fit severity models and write estimates to a data set */
proc severity data=claimsev criterion=aicc outest=sevest covout plots=none;
   loss lossValue;
   dist _predefined_;
run;
```

The STORE statement in the PROC COUNTREG step saves the count model information, including the parameter estimates, in the Work.CountStorePoisson item store. An item store contains the model information in a binary format that cannot be modified after it is created. You can examine the contents of an item store that is created by a PROC COUNTREG step by specifying a combination of the RESTORE= option and the SHOW statement in another PROC COUNTREG step. For more information, see Chapter 12, "The COUNTREG Procedure."

The OUTEST= option in the PROC SEVERITY statement stores the estimates of all the fitted severity models in the Work.SevEst data set. Let the best severity model that the PROC SEVERITY step chooses be the gamma distribution model.

You can now submit the following PROC HPCDM step to simulate an aggregate loss sample of size 10,000 by specifying the count model's item store in the COUNTSTORE= option and the severity model's data set of estimates in the SEVERITYEST= option:

```
/* Simulate and estimate Poisson-gamma compound distribution model */
proc hpcdm countstore=countStorePoisson severityest=sevest
           seed=13579 nreplicates=10000 plots=(edf(alpha=0.05) density)
           print=(summarystatistics percentiles);
   severitymodel gamma;
   output out=aggregateLossSample samplevar=aggloss;
   outsum out=aggregateLossSummary mean stddev skewness kurtosis
          p01 p05 p95 p995=var pctlpts=90 97.5;
run;
```

The SEVERITYMODEL statement requests that an aggregate sample be generated by compounding only the gamma distribution and the frequency distribution. Specifying the SEED= value helps you get an identical sample each time you execute this step, provided that you use the same execution environment. In the single-machine mode of execution, the execution environment is the combination of the operating environment and the number of threads that are used for execution. In the distributed computing mode, the execution environment is the combination of the operating environment, the number of nodes, and the number of threads that are used for execution on each node.

Upon completion, PROC HPCDM creates the two output data sets that you specify in the OUT= options of the OUTPUT and OUTSUM statements. The Work.AggregateLossSample data set contains 10,000 observations such that the value of the AggLoss variable in each observation represents one possible aggregate loss value that you can expect to see in one year. Together, the set of the 10,000 values of the AggLoss variable represents one sample of compound distribution. PROC HPCDM uses this sample to compute the empirical estimates of various summary statistics and percentiles of the compound distribution. The Work.AggregateLossSummary data set contains the estimates of mean, standard deviation, skewness, and kurtosis that you specify in the OUTSUM statement. It also contains the estimates of the 1st, 5th, 90th, 95th, 97.5th, and 99.5th percentiles that you specify in the OUTSUM statement. The value-at-risk (VaR) is an aggregate loss value such that there is a very low probability that an observed aggregate loss value exceeds the VaR. One of the commonly used probability levels to define VaR is 0.005, which makes the 99.5th percentile an empirical estimate of the VaR. Hence, the OUTSUM statement of this example stores the 99.5th percentile in a variable named VaR. VaR is one of the widely used measures of worst-case risk.

Some of the default output and some of the output that you have requested by specifying the PRINT= option are shown in Figure 18.1.

Figure 18.1 Information, Summary Statistics, and Percentiles of the Poisson-Gamma Compound Distribution

The HPCDM Procedure
Severity Model: Gamma
Count Model: Poisson

Compound Distribution Information	
Severity Model	Gamma Distribution
Count Model	Poisson Model in Item Store WORK.COUNTSTOREPOISSON

Sample Summary Statistics			
Mean	4062.8	**Median**	3349.7
Standard Deviation	3429.6	**Interquartile Range**	4456.4
Variance	11761948.0	**Minimum**	0
Skewness	1.14604	**Maximum**	26077.4
Kurtosis	1.76466	**Sample Size**	10000

Sample Percentiles	
Percentile	**Value**
1	0
5	0
25	1449.1
50	3349.7
75	5905.5
90	8792.6
95	10672.5
97.5	12391.7
99	14512.5
99.5	15877.9
Percentile Method = 5	

The "Sample Summary Statistics" table indicates that for the given parameter estimates of the Poisson frequency and gamma severity models, you can expect to see a mean aggregate loss of 4,062.8 and a median aggregate loss of 3,349.7 in a year. The "Sample Percentiles" table indicates that there is a 0.5% chance that the aggregate loss exceeds 15,877.9, which is the VaR estimate, and a 2.5% chance that the aggregate loss exceeds 12,391.7. These summary statistic and percentile estimates provide a quantitative picture of the compound distribution. You can also visually analyze the compound distribution by examining the plots that PROC HPCDM prepares. The first plot in Figure 18.2 shows the empirical distribution function (EDF), which is a nonparametric estimate of the cumulative distribution function (CDF). The second plot shows the histogram and the kernel density estimate, which are nonparametric estimates of the probability density function (PDF).

Figure 18.2 Nonparametric CDF and PDF Plots of the Poisson-Gamma Compound Distribution

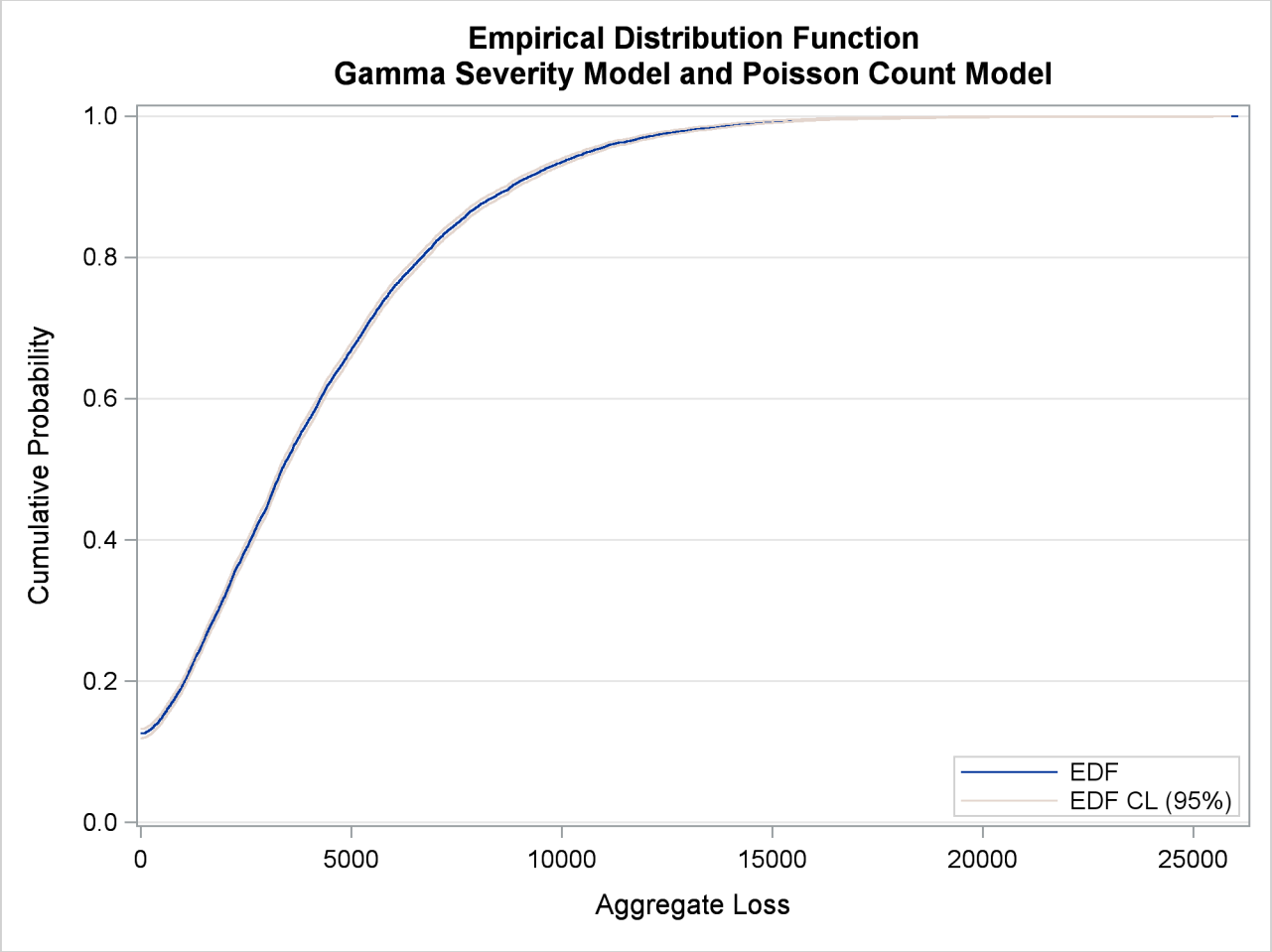
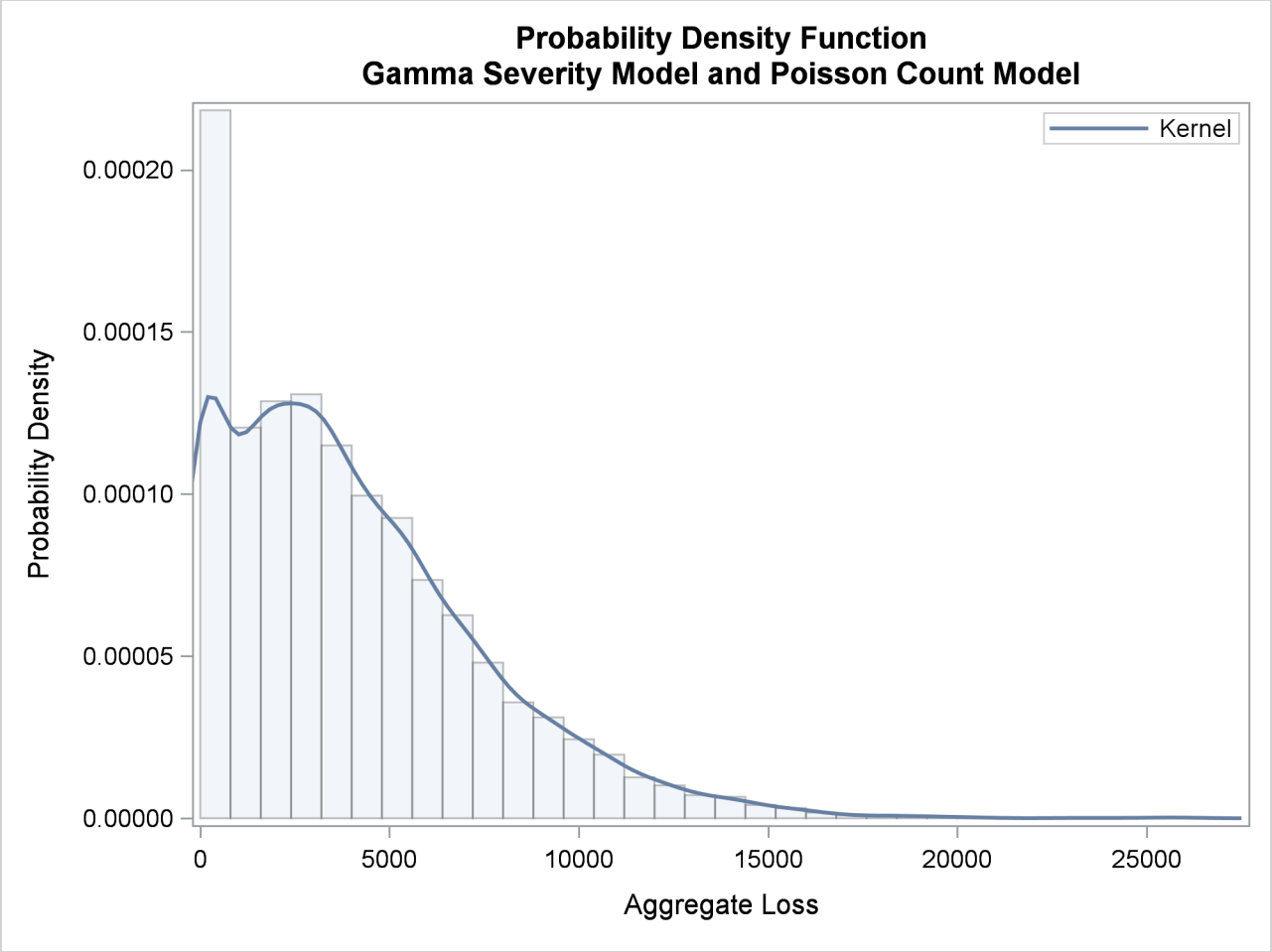


Figure 18.2 *continued*



The plots confirm the right skew that is indicated by the estimate of skewness in Figure 18.1 and a relatively fat tail, which is indicated by comparing the maximum and the 99.5th percentiles in Figure 18.1.

Analyzing the Effect of Parameter Uncertainty on the Compound Distribution

Continuing with the previous example, note that you have fitted the frequency and severity models by using the historical data. Even if you choose the best-fitting models, the true underlying models are not known exactly. This fact is reflected in the uncertainty that is associated with the parameters of your models. Any compound distribution estimate that is computed by using these uncertain parameter estimates is inherently uncertain. You can request that PROC HPCDM conduct parameter perturbation analysis, which assesses the effect of the parameter uncertainty on the estimates of the compound distribution by simulating multiple samples, each with parameters that are randomly perturbed from their mean estimates.

The following PROC HPCDM step adds the NPERTURBEDSAMPLES= option to the PROC HPCDM statement to request that perturbation analysis be conducted and the PRINT=PERTURBSUMMARY option to request that a summary of the perturbation analysis be displayed:

```
/* Conduct parameter perturbation analysis of
   the Poisson-gamma compound distribution model */
proc hpcdm countstore=countStorePoisson severityest=sevest
           seed=13579 nreplicates=10000 nperturbedsamples=30
           print(only)=(perturbsummary) plots=none;
   severitymodel gamma;
   output out=aggregateLossSample samplevar=aggloss;
   outsum out=aggregateLossSummary mean stddev skewness kurtosis
          p01 p05 p95 p995=var pctlpts=90 97.5;
run;
```

The Work.AggregateLossSummary data set contains the specified summary statistics and percentiles for all 30 perturbed samples. You can identify a perturbed sample by the value of the _DRAWID_ variable. The first few observations of the Work.AggregateLossSummary data set are shown in Figure 18.3. For the first observation, the value of the _DRAWID_ variable is 0, which represents an unperturbed sample—that is, the aggregate sample that is simulated without perturbing the parameters from their means.

Figure 18.3 Summary Statistics and Percentiles of the Perturbed Samples

SEVERITYMODEL	_COUNTMODEL_	_DRAWID_	_SAMPLEVAR_	N	MEAN	STDDEV
Gamma	Poisson	0	aggloss	10000	4062.76	3429.57
Gamma	Poisson	1	aggloss	10000	4008.04	3406.22
Gamma	Poisson	2	aggloss	10000	4426.67	3719.94
Gamma	Poisson	3	aggloss	10000	3991.87	3480.10
Gamma	Poisson	4	aggloss	10000	3807.58	3303.61
Gamma	Poisson	5	aggloss	10000	4083.70	3429.83
Gamma	Poisson	6	aggloss	10000	4185.82	3525.20
Gamma	Poisson	7	aggloss	10000	3882.99	3372.81
Gamma	Poisson	8	aggloss	10000	4092.94	3483.60
Gamma	Poisson	9	aggloss	10000	4039.82	3454.69
Gamma	Poisson	10	aggloss	10000	3851.17	3287.52

SKEWNESS	KURTOSIS	P01	P05	P90	P95	P97_5	var
1.14604	1.76466	0	0	8792.64	10672.49	12391.70	15877.89
1.10747	1.43304	0	0	8658.62	10521.82	12279.33	16152.05
1.14337	1.66525	0	0	9484.05	11522.70	13523.54	17575.20
1.23233	2.07634	0	0	8672.80	10568.25	12472.90	16969.77
1.08965	1.15633	0	0	8375.09	10319.59	11884.11	15255.16
1.08043	1.31018	0	0	8836.78	10707.19	12399.09	16236.24
1.12642	1.49282	0	0	9095.46	11056.46	12752.18	16519.99
1.22931	1.95615	0	0	8515.35	10371.84	12245.23	16153.91
1.10040	1.47077	0	0	8923.13	10757.13	12522.34	16275.95
1.17185	1.84608	0	0	8696.09	10679.34	12611.43	16350.84
1.12302	1.60240	0	0	8383.29	10129.41	11725.89	15303.35

The PRINT=PERTURBSUMMARY option in the preceding PROC HPCDM step produces the "Sample Perturbation Analysis" and "Sample Percentile Perturbation Analysis" tables that are shown in Figure 18.4. The tables show that you can expect a mean aggregate loss of about 4,049.1 and the standard error of the mean is 193.6. If you want to use the VaR estimate to determine the amount of reserves that you need to maintain to cover the worst-case loss, then you should consider not only the mean estimate of the 99.5th

percentile, which is about 16,339.1, but also the standard error of 692.8 to account for the effect of uncertainty in your frequency and severity parameter estimates.

Figure 18.4 Summary of Perturbation Analysis of the Poisson-Gamma Compound Distribution

The HPCDM Procedure
Severity Model: Gamma
Count Model: Poisson

Sample Perturbation Analysis

Statistic	Estimate	Standard Error
Mean	4049.1	193.55480
Standard Deviation	3448.5	132.43375
Variance	11909479	919586.4
Skewness	1.14075	0.04610
Kurtosis	1.64953	0.27146

Number of Perturbed Samples = 30
Size of Each Sample = 10000

Sample Percentile Perturbation Analysis

Percentile	Estimate	Standard Error
0	0	0
1	0	0
5	0	0
25	1386.8	114.41389
50	3368.2	185.13314
75	5944.8	265.53061
90	8756.0	365.86765
95	10663.6	441.16381
97.5	12454.8	519.67311
99	14685.6	620.49261
99.5	16339.1	692.79352

Number of Perturbed Samples = 30
Size of Each Sample = 10000

Scenario Analysis

The distributions of loss frequency and loss severity often depend on exogenous variables (regressors). For example, the number of losses and the severity of each loss that an automobile insurance policyholder incurs might depend on the characteristics of the policyholder and the characteristics of the vehicle. When you fit frequency and severity models, you need to account for the effects of such regressors on the probability distributions of the counts and severity. The COUNTREG procedure enables you to model regression effects on the mean of the count distribution, and the SEVERITY procedure enables you to model regression effects on the scale parameter of the severity distribution. When you use these models to estimate the compound distribution model of the aggregate loss, you need to specify a set of values for all the regressors, which represents the state of the world for which the simulation is conducted. This is referred to as the what-if or scenario analysis.

Consider that you, as an automobile insurance company, have postulated that the distribution of the loss event frequency depends on five regressors (external factors): age of the policyholder, gender, type of car, annual miles driven, and policyholder's education level. Further, the distribution of the severity of each loss depends on three regressors: type of car, safety rating of the car, and annual household income of the policyholder (which can be thought of as a proxy for the luxury level of the car). Note that the frequency model regressors and severity model regressors can be different, as illustrated in this example.

Let these regressors be recorded in the variables Age (scaled by a factor of 1/50), Gender (1: female, 2: male), CarType (1: sedan, 2: sport utility vehicle), AnnualMiles (scaled by a factor of 1/5,000), Education (1: high school graduate, 2: college graduate, 3: advanced degree holder), CarSafety (scaled to be between 0 and 1, the safest being 1), and Income (scaled by a factor of 1/100,000), respectively. Let the historical data about the number of losses that various policyholders incur in a year be recorded in the NumLoss variable of the Work.LossCounts data set, and let the severity of each loss be recorded in the LossAmount variable of the Work.Losses data set.

The following PROC COUNTREG step fits the count regression model and stores the fitted model information in the Work.CountregModel item store:

```
/* Fit negative binomial frequency model for the number of losses */
proc countreg data=losscounts;
   model numloss = age gender carType annualMiles education / dist=negbin;
   store work.countregmodel;
run;
```

You can examine the parameter estimates of the count model that are stored in the Work.CountregModel item store by submitting the following statements:

```
/* Examine the parameter estimates for the model in the item store */
proc countreg restore=work.countregmodel;
   show parameters;
run;
```

The "Parameter Estimates" table that is displayed by the SHOW statement is shown in Figure 18.5.

Figure 18.5 Parameter Estimates of the Count Regression Model

ITEM STORE CONTENTS: WORK.COUNTREGMODEL

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	0.910479	0.090515	10.06	<.0001
age	1	-0.626803	0.058547	-10.71	<.0001
gender	1	1.025034	0.032099	31.93	<.0001
carType	1	0.615165	0.031153	19.75	<.0001
annualMiles	1	-1.010276	0.017512	-57.69	<.0001
education	1	-0.280246	0.021677	-12.93	<.0001
_Alpha	1	0.318403	0.020090	15.85	<.0001

The following PROC SEVERITY step fits the severity scale regression models for all the common distributions that are predefined in PROC SEVERITY:

```
/* Fit severity models for the magnitude of losses */
proc severity data=losses plots=none outest=work.sevregest print=all;
   loss lossamount;
   scalemodel carType carSafety income;
   dist _predef_;
   nloptions maxiter=100;
run;
```

The comparison of fit statistics of various scale regression models is shown in Figure 18.6. The scale regression model that is based on the lognormal distribution is deemed the best-fitting model according to the likelihood-based statistics, whereas the scale regression model that is based on the generalized Pareto distribution (GPD) is deemed the best-fitting model according to the EDF-based statistics.

Figure 18.6 Severity Model Comparison

The SEVERITY Procedure

All Fit Statistics

Distribution	-2 Log Likelihood		AIC		AICC		BIC		KS		AD		CvM	
Burr	127231		127243		127243		127286		7.75407		224.47578		27.41346	
Exp	128431		128439		128439		128467		6.13537		181.83094		12.33919	
Gamma	128324		128334		128334		128370		7.54562		276.13156		24.59515	
Igauss	127434		127444		127444		127480		6.15855		211.51908		17.70942	
Logn	127062	*	127072	*	127072	*	127107	*	6.77687		212.70400		21.47945	
Pareto	128166		128176		128176		128211		5.37453		110.53673		7.07119	
Gpd	128166		128176		128176		128211		5.37453	*	110.53660	*	7.07116	*
Weibull	128429		128439		128439		128475		6.21268		190.81178		13.45425	

Note: The asterisk (*) marks the best model according to each column's criterion.

Now, you are ready to analyze the distribution of the aggregate loss that can be expected from a specific policyholder—for example, a 59-year-old male policyholder with an advanced degree who earns 159,870 and drives a sedan that has a very high safety rating about 11,474 miles annually. First, you need to encode and scale this information into the appropriate regressor variables of a data set. Let that data set be named Work.SinglePolicy, with an observation as shown in Figure 18.7.

Figure 18.7 Scenario Analysis Data for One Policyholder

age	gender	carType	annualMiles	education	carSafety	income
1.18	2	1	2.2948	3	0.99532	1.5987

Now, you can submit the following PROC HPCDM step to analyze the compound distribution of the aggregate loss that is incurred by the policyholder in the Work.SinglePolicy data set in a given year by using the frequency model from the Work.CountregModel item store and the two best severity models, lognormal and GPD, from the Work.SevRegEst data set:

```
/* Simulate the aggregate loss distribution for the scenario
   with single policyholder */
proc hpcdm data=singlePolicy nreplicates=10000 seed=13579 print=all
           countstore=work.countregmodel severityest=work.sevregest;
   severitymodel logn gpd;
   outsum out=onepolicysum mean stddev skew kurtosis median
          pctlpts=97.5 to 99.5 by 1;
run;
```

The displayed results from the preceding PROC HPCDM step are shown in Figure 18.8.

When you use a severity scale regression model, it is recommended that you verify the severity scale regressors that are used by PROC HPCDM by examining the Scale Model Regressors row of the "Compound Distribution Information" table. PROC HPCDM detects the severity regressors automatically by examining the variables in the SEVERITYEST= and DATA= data sets. If those data sets contain variables that you did not include in the SCALEMODEL statement in PROC SEVERITY, then such variables can be treated as severity regressors. One common mistake that can lead to this situation is to fit a severity model by using the BY statement and forget to specify the identical BY statement in the PROC HPCDM step; this can cause PROC HPCDM to treat BY variables as scale model regressors. In this example, Figure 18.8 confirms that the correct set of scale model regressors is detected.

Figure 18.8 Scenario Analysis Results for One Policyholder with Lognormal Severity Model

The HPCDM Procedure
Severity Model: Logn
Count Model: NegBin(p=2)

Compound Distribution Information	
Severity Model	Lognormal Distribution
Scale Model Regressors	carType carSafety income
Count Model	NegBin(p=2) Model in Item Store WORK.COUNTREGMODEL

Sample Summary Statistics			
Mean	217.61476	**Median**	0
Standard Deviation	429.35923	**Interquartile Range**	269.28863
Variance	184349.3	**Minimum**	0
Skewness	3.94508	**Maximum**	4986.6
Kurtosis	22.88738	**Sample Size**	10000

Sample Percentiles	
Percentile	**Value**
0	0
1	0
5	0
25	0
50	0
75	269.28863
95	999.83713
97.5	1417.7
98.5	1774.0
99	2036.5
99.5	2590.1
Percentile Method = 5	

The "Sample Summary Statistics" and "Sample Percentiles" tables in Figure 18.8 show estimates of the aggregate loss distribution for the lognormal severity model. The average expected loss is about 218, and the worst-case loss, if approximated by the 97.5th percentile, is about 1,418. The percentiles table shows that the distribution is highly skewed to the right; this is also confirmed by the skewness estimate. The median

estimate of 0 can be interpreted in two ways. One way is to conclude that the policyholder will not incur any loss in 50% of the years during which he or she is insured. The other way is to conclude that 50% of policyholders who have the characteristics of this policyholder will not incur any loss in a given year. However, there is a 2.5% chance that the policyholder will incur a loss that exceeds 1,418 in any given year and a 0.5% chance that the policyholder will incur a loss that exceeds 2,590 in any given year.

If the aggregate loss sample is simulated by using the GPD severity model, then the results are as shown in Figure 18.9. The average and worst-case losses are 212 and 1,388, respectively. These estimates are very close to the values that are predicted by the lognormal severity model.

Figure 18.9 Scenario Analysis Results for One Policyholder with GPD Severity Model

The HPCDM Procedure
Severity Model: Gpd
Count Model: NegBin(p=2)

Compound Distribution Information	
Severity Model	Generalized Pareto Distribution
Scale Model Regressors	carType carSafety income
Count Model	NegBin(p=2) Model in Item Store WORK.COUNTREGMODEL

Sample Summary Statistics			
Mean	211.70312	**Median**	0
Standard Deviation	403.70696	**Interquartile Range**	269.23607
Variance	162979.3	**Minimum**	0
Skewness	3.23359	**Maximum**	4233.1
Kurtosis	14.36690	**Sample Size**	10000

Sample Percentiles	
Percentile	**Value**
0	0
1	0
5	0
25	0
50	0
75	269.23607
95	1003.0
97.5	1387.7
98.5	1700.7
99	1912.0
99.5	2294.3
Percentile Method = 5	

The scenario that you just analyzed contains only one policyholder. You can extend the scenario to include multiple policyholders. Let the Work.GroupOfPolicies data set record information about five different policyholders, as shown in Figure 18.10.

Figure 18.10 Scenario Analysis Data for Multiple Policyholders

policyholderId	age	gender	carType	annualMiles	education	carSafety	income
1	1.18	2	1	2.2948	3	0.99532	1.59870
2	0.66	2	1	2.6718	2	0.86412	0.84459
3	0.64	2	2	1.9528	1	0.86478	0.50177
4	0.46	1	2	2.6402	2	0.27062	1.18870
5	0.62	1	1	1.7294	1	0.32830	0.37694

The following PROC HPCDM step conducts a scenario analysis for the aggregate loss that is incurred by all five policyholders in the Work.GroupOfPolicies data set together in one year:

```
/* Simulate the aggregate loss distribution for the scenario
   with multiple policyholders */
proc hpcdm data=groupOfPolicies nreplicates=10000 seed=13579 print=all
           countstore=work.countregmodel severityest=work.sevregest
           plots=(conditionaldensity(rightq=0.95)) nperturbedSamples=50;
   severitymodel logn gpd;
   outsum out=multipolicysum mean stddev skew kurtosis median
         pctlpts=97.5 to 99.5 by 1;
run;
```

The preceding PROC HPCDM step conducts perturbation analysis by simulating 50 perturbed samples. The perturbation summary results for the lognormal severity model are shown in Figure 18.11, and the results for the GPD severity model are shown in Figure 18.12. If the severity of each loss follows the fitted lognormal distribution, then you can expect that the group of policyholders together incurs an average loss of 5,331 $\pm$ 560 and a worst-case loss of 15,859 $\pm$ 1,442 when you define the worst-case loss as the 97.5th percentile.

Figure 18.11 Perturbation Analysis of Losses from Multiple Policyholders with Lognormal Severity Model

The HPCDM Procedure
Severity Model: Logn
Count Model: NegBin(p=2)

Compound Distribution Information	
Severity Model	Lognormal Distribution
Scale Model Regressors	carType carSafety income
Count Model	NegBin(p=2) Model in Item Store WORK.COUNTREGMODEL

Sample Perturbation Analysis		
Statistic	**Estimate**	**Standard Error**
Mean	5331.3	559.52182
Standard Deviation	4170.6	346.61321
Variance	17514137	2979306.9
Skewness	2.02770	0.24997
Kurtosis	9.14611	3.75927
Number of Perturbed Samples = 50		
Size of Each Sample = 10000		

Figure 18.11 *continued*

Sample Percentile Perturbation Analysis		
Percentile	Estimate	Standard Error
1	216.43966	65.57200
5	765.60278	143.70919
25	2401.0	324.11066
50	4342.7	498.47507
75	7139.4	739.01751
95	13185.9	1217.8
97.5	15858.5	1441.8
98.5	17886.4	1585.0
99	19553.1	1693.9
99.5	22646.0	2001.5
Number of Perturbed Samples = 50		
Size of Each Sample = 10000		

If the severity of each loss follows the fitted GPD distribution, then you can expect an average loss of 5,294 $\pm$ 539 and a worst-case loss of 15,128 $\pm$ 1,340.

If you decide to use the 99.5th percentile to define the worst-case loss, then the worst-case loss is 22,646 $\pm$ 2,002 for the lognormal severity model and 20,539 $\pm$ 1,798 for the GPD severity model. The numbers for lognormal and GPD are well within one standard error of each other, which indicates that the aggregate loss distribution is less sensitive to the choice of these two severity distributions in this particular example; you can use the results from either of them.

Figure 18.12 Perturbation Analysis of Losses from Multiple Policyholders with GPD Severity Model

The HPCDM Procedure
Severity Model: Gpd
Count Model: NegBin(p=2)

Compound Distribution Information	
Severity Model	Generalized Pareto Distribution
Scale Model Regressors	carType carSafety income
Count Model	NegBin(p=2) Model in Item Store WORK.COUNTREGMODEL

Sample Perturbation Analysis		
Statistic	Estimate	Standard Error
Mean	5294.3	538.96406
Standard Deviation	3922.0	337.97495
Variance	15496384	2679730.2
Skewness	1.47924	0.10402
Kurtosis	3.66621	0.85390
Number of Perturbed Samples = 50		
Size of Each Sample = 10000		

Figure 18.12 *continued*

Sample Percentile Perturbation Analysis

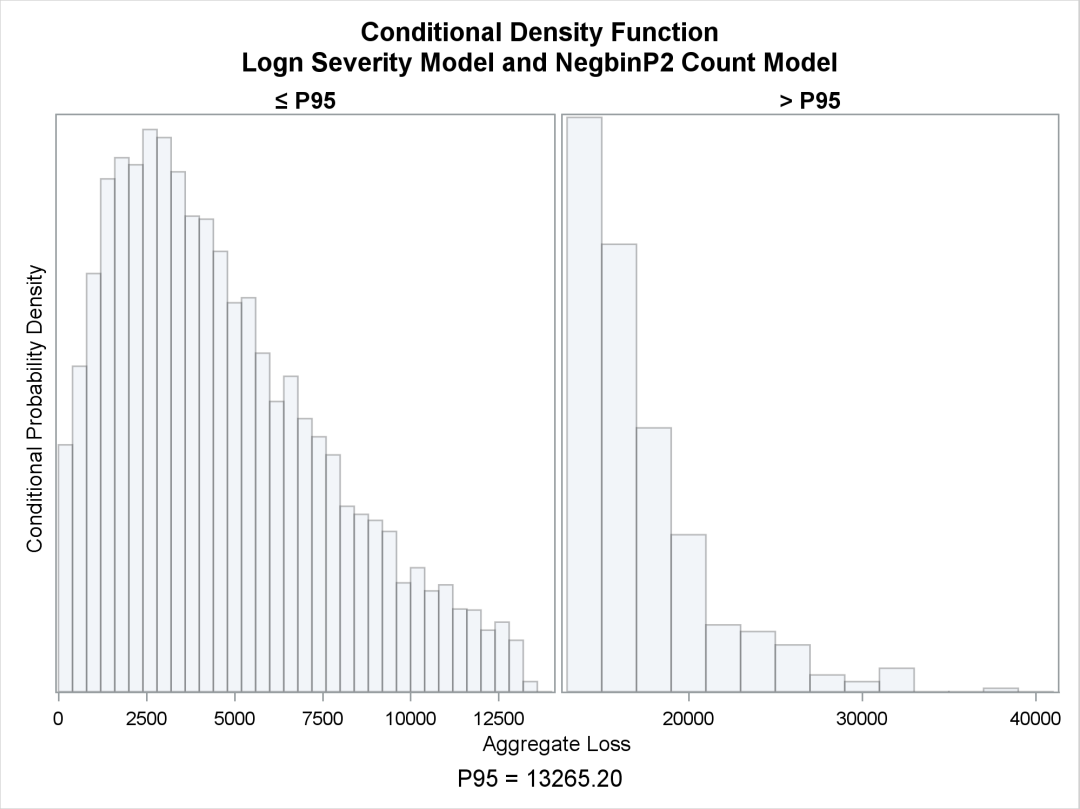
Percentile	Estimate	Standard Error
1	173.75335	62.52659
5	728.89376	137.91935
25	2422.1	314.53125
50	4424.5	488.71163
75	7204.9	710.03240
95	12851.4	1180.1
97.5	15127.8	1340.4
98.5	16823.9	1482.5
99	18206.7	1621.8
99.5	20538.9	1797.6

Number of Perturbed Samples = 50

Size of Each Sample = 10000

The PLOTS=CONDITIONALDENSITY option that is used in the preceding PROC HPCDM step prepares the conditional density plots for the body and right-tail regions of the density function of the aggregate loss. The plots for the aggregate loss sample that is generated by using the lognormal severity model are shown in Figure 18.13. The plot on the left side is the plot of $\Pr(Y|Y \leq 13{,}265)$, where the limit 13,265 is the 95th percentile as specified by the RIGHTQ=0.95 option. The plot on the right side is the plot of $\Pr(Y|Y > 13{,}265)$, which helps you visualize the right-tail region of the density function. You can also request the plot of the left tail by specifying the LEFTQ= suboption of the CONDITIONALDENSITY option if you want to explore the details of the left tail region. Note that the conditional density plots are always produced by using the unperturbed sample.

Figure 18.13 Conditional Density Plots for the Aggregate Loss of Multiple Policyholders



Syntax: HPCDM Procedure

The following statements are used with the HPCDM procedure:

PROC HPCDM *options* ;
BY *variable-list* ;
DISTBY *replication-id-variable* ;
SEVERITYMODEL *severity-model-list* ;
EXTERNALCOUNTS *COUNT=frequency-variable < ID=replication-id-variable >* ;
OUTPUT *OUT=SAS-data-set < variable-name-options > < / out-option >* ;
OUTSUM *OUT=SAS-data-set statistic-keyword< =variable-name > < ... statistic-keyword< =variable-name > > < outsum-options >* ;
PERFORMANCE *options* ;
Programming statements ;

Functional Summary

Table 18.1 summarizes the statements and options available in the HPCDM procedure.

Table 18.1 HPCDM Functional Summary

Description	Statement	Option
Statements		
Specifies the names of severity distribution models	SEVERITYMODEL	
Specifies externally simulated count data	EXTERNALCOUNTS	
Specifies where and how the full simulated samples are written	OUTPUT	
Specifies where and how the summary statistics of simulated samples are written	OUTSUM	
Specifies performance options	PERFORMANCE	
Specifies programming statements that define an objective function	Programming statements	
Data Set Options		
Specifies the input data set	PROC HPCDM	DATA=
Specifies the output data set for the full simulated samples	OUTPUT	OUT=
Specifies the output data set for the summary statistics	OUTSUM	OUT=
Model Input Options		
Specifies the variable that contains externally simulated counts	EXTERNALCOUNTS	COUNT=
Specifies the item store that contains the frequency (count) model	PROC HPCDM	COUNTSTORE=
Specifies the replicate identifier variable for external counts	EXTERNALCOUNTS	ID=
Specifies the input data set for parameter estimates of the severity models	PROC HPCDM	SEVERITYEST=
Simulation Options		
Specifies the adjusted severity symbol in the programming statements	PROC HPCDM	ADJUSTEDSEVERITY=
Specifies the number of parameter-perturbed samples to be simulated	PROC HPCDM	NPERTURBEDSAMPLES=
Specifies a number that controls the size of the simulated sample	PROC HPCDM	NREPLICATES=
Specifies a seed for the internal pseudo-random number generator	PROC HPCDM	SEED=
Output Preparation Options		
Specifies the variable for the aggregate adjusted loss sample	OUTPUT	ADJSAMPLEVAR=
Specifies the names of the variables for percentiles	OUTSUM	PCTLNAME=

Table 18.1 *continued*

Description	Statement	Option
Specifies the decimal precision to form default percentile variable names	OUTSUM	PCTLNDEC=
Specifies percentiles to compute and report	OUTSUM	PCTLPTS=
Specifies the method to compute the percentiles	PROC HPCDM	PCTLDEF=
Specifies that all perturbed samples be written to the OUT= data set	OUTPUT	PERTURBOUT
Specifies the variable for the aggregate loss sample	OUTPUT	SAMPLEVAR=
Specifies the denominator for computing second- and higher-order moments	PROC HPCDM	VARDEF=
Displayed Output and Plotting Options		
Suppresses all displayed and graphical output	PROC HPCDM	NOPRINT
Specifies which displayed output to prepare	PROC HPCDM	PRINT=
Specifies which graphical output to prepare	PROC HPCDM	PLOTS=

PROC HPCDM Statement

PROC HPCDM *options* **;**

The PROC HPCDM statement invokes the procedure. You can specify the following *options*, which are listed in alphabetical order.

ADJUSTEDSEVERITY=*symbol-name*

ADJSEV=*symbol-name*

names the symbol that represents the adjusted severity value in the SAS programming statements that you specify. The *symbol-name* is a SAS name that conforms to the naming conventions of a SAS variable. For more information, see the section "Programming Statements" on page 1019.

COUNTSTORE=*SAS-item-store*

names the item store that contains all the information about the frequency (count) model. The COUNTREG procedure generates this item store when you use the STORE statement.

The exogenous variables in the frequency model, if any, are deduced from this item store. The DATA= data set must contain all those variables.

You must specify this option if you do not specify the EXTERNALCOUNTS statement. This option is ignored if you specify the EXTERNALCOUNTS statement, because PROC HPCDM does not need to simulate frequency counts internally when you specify externally simulated counts.

If you specify the COUNTSTORE= option, then you cannot specify the BY statement in PROC HPCDM, and vice versa.

If you specify the COUNTSTORE= option and execute the HPCDM procedure in distributed mode, then the distributed data access mode for the DATA= data set must be either client-data (local-data)

mode or through-the-client mode—that is, the DATA= data set should not be stored on a distributed database appliance. For more information about data access modes, see the section "Data Access Modes" on page 67 of Chapter 3, "Shared Concepts and Topics."

DATA=*SAS-data-set*

names the input data set that contains the values of regression variables in frequency or severity models and severity adjustment variables that you use in the programming statements.

The DATA= data set specifies information about the scenario for which you want to estimate the aggregate loss distribution. The interpretation of the contents of the data set and the supported distributed data access modes depend on whether you specify the EXTERNALCOUNTS statement. For more information, see the section "Specifying Scenario Data in the DATA= Data Set" on page 1019.

NOPRINT

turns off all displayed and graphical output. If you specify this option, then PROC HPCDM ignores any value that you specify for the PRINT= or PLOTS= option.

NPERTURBEDSAMPLES=*number*

NPERTURB=*number*

requests that parameter perturbation analysis be conducted. The model parameters are perturbed the specified *number* of times and a separate full sample is simulated for each set of perturbed parameter values. The summary statistics and percentiles are computed for each such perturbed sample, and their values are aggregated across the samples to compute the mean and standard deviation of each summary statistic and percentile.

The parameter perturbation procedure makes random draws of parameter values from a multivariate normal distribution if the covariance estimates of the parameters are available in the SEVERITYEST= data set for the severity model and in the COUNTSTORE= store for the count model. If covariance estimates are not available, then for each parameter, a random draw is made from the univariate normal distribution that has mean and standard deviation equal to the point estimate and the standard error, respectively, of that parameter. If neither covariance nor standard error estimates are available, then perturbation analysis is not conducted.

If you specify the PRINT=ALL or PRINT=PERTURBSUMMARY option, then the summary of perturbation analysis is printed for the core summary statistics and the percentiles of the aggregate loss distribution. If you specify the OUTSUM statement, then the requested summary statistics are written to the OUTSUM= data set for each perturbed sample. You can also optionally request that each perturbed sample be written in its entirety to the OUT= data set by specifying the PERTURBOUT option in the OUTPUT statement.

For more information on the parameter perturbation analysis, see the section "Parameter Perturbation Analysis" on page 1035.

NREPLICATES=*number*

NREP=*number*

specifies a *number* that controls the size of the compound distribution sample that PROC HPCDM simulates. The *number* is interpreted differently based on whether you specify the EXTERNALCOUNTS statement.

If you do not specify the EXTERNALCOUNTS statement, then the sample size is equal to the *number* that you specify for this option. If you do not specify this option, then a default value of 100,000 is used.

If you specify the EXTERNALCOUNTS statement, then the number of replicates that you specify in the DATA= data set is multiplied by the *number* that you specify for this option to get the total size of the compound distribution sample. If you do not specify this option, then a default value of 1 is used.

PCTLDEF=*percentile-method*

specifies the method to compute the percentiles of the compound distribution. The *percentile-method* can be 1, 2, 3, 4, or 5. The default method is 5. For more information, see the description of the PCTLDEF= option in the UNIVARIATE procedure in the *Base SAS Procedures Guide: Statistical Procedures*.

PLOTS <(*global-plot-options***)> =***plot-request-option*

PLOTS <(*global-plot-options***)> =(***plot-request-option . . . plot-request-option***)**

specifies the desired graphical output.

By default, the HPCDM procedure produces no graphical output.

You can specify the following *global-plot-option*:

ONLY

turns off the default graphical output and prepares only the requested plots.

If you specify more than one *plot-request-option*, then separate them with spaces and enclose them in parentheses. The following *plot-request-options* are available:

ALL

displays all the graphical output.

CONDITIONALDENSITY (*conditional-density-plot-options***)**

CONDPDF (*conditional-density-plot-options***)**

prepares a group of plots of the conditional density functions estimates. The group contains at most three plots, each conditional on the value of the aggregate loss being in one of the three regions that are defined by the quantiles that you specify in the following *conditional-density-plot-options*:

LEFTQ=*number*

specifies the quantile in the range (0,1) that marks the end of the left-tail region. If you specify a value of l for *number*, then the left-tail region is defined as the set of values that are less than or equal to q_l, where q_l is the lth quantile. For the left-tail region, nonparametric estimates of the conditional probability density function $f_S^l(s) = \Pr[S = s | S \leq q_l]$ are plotted. The value of q_l is estimated by the $100l$th percentile of the simulated compound distribution sample.

If you do not specify this option or you specify a missing value for this option, then the left-tail region is not plotted.

RIGHTQ=*number*

specifies the quantile in the range (0,1) that marks the beginning of the right-tail region. If you specify a value of r for *number*, then the right-tail region is defined as the set of values that are greater than q_r, where q_r is the rth quantile. For the right-tail region, nonparametric estimates of the conditional probability density function $f_S^r(s) = \Pr[S = s | S > q_r]$ are plotted. The value of q_r is estimated by the $100r$th percentile of the simulated compound distribution sample.

If you do not specify this option or you specify a missing value for this option, then the right-tail region is not plotted.

You must specify nonmissing value for at least one of the preceding two options. For the region between the LEFTQ= and RIGHTQ= quantiles, which is referred to as the central or body region, nonparametric estimates of the conditional probability density function $f_S^c(s) = \Pr[S = s | q_l < S \leq q_r]$ are plotted. If you do not specify a LEFTQ= value, then q_l is assumed to be 0. If you do not specify a RIGHTQ= value, then q_r is assumed to be ∞.

DENSITY
prepares a plot of the nonparametric estimates of the probability density function (in particular, histogram and kernel density estimates) of the compound distribution.

EDF < (*edf-plot-option*) >
prepares a plot of the nonparametric estimates of the cumulative distribution function of the compound distribution.

You can request that the confidence interval be plotted by specifying the following *edf-plot-option*:

ALPHA=*number*
specifies the confidence level in the (0,1) range that is used for computing the confidence intervals for the EDF estimates. If you specify a value of α for *number*, then the upper and lower confidence limits for the confidence level of $100(1 - \alpha)$ are plotted.

NONE
displays none of the graphical output. If you specify this option, then it overrides all other plot request options. The default graphical output is also suppressed.

Note that if the simulated sample size is large, then it can take a significant amount of time and memory to prepare the plots.

PRINT < (*global-display-option*) > =*display-option*

PRINT < (*global-display-option*) > =(*display-option . . . display-option***)**
specifies the desired displayed output. If you specify more than one *display-option*, then separate them with spaces and enclose them in parentheses.

You can specify the following *global-display-option*:

ONLY
turns off the default displayed output and displays only the requested output.

You can specify the following *display-options*:

ALL
displays all the output.

NONE
displays none of the output. If you specify this option, then it overrides all other display options. The default displayed output is also suppressed.

PERCENTILES
displays the percentiles of the compound distribution sample. This includes all the predefined percentiles, percentiles that you request in the OUTSUM statement, and percentiles that you specify for preparing conditional density plots.

PERTURBSUMMARY
displays the mean and standard deviation of the summary statistics and percentiles that are taken across all the samples produced by perturbing the model parameters. This option is valid only if you specify the NPERTURBEDSAMPLES= option in the PROC HPCDM statement.

SUMMARYSTATISTICS | SUMSTAT
displays the summary statistics of the compound distribution sample.

If you do not specify the PRINT= option or the ONLY *global-display-option*, then the default displayed output is equivalent to specifying PRINT=(SUMMARYSTATISTICS).

SEED=*number*
specifies the integer to use as the seed in generating the pseudo-random numbers that are used for simulating severity and frequency values. If you do not specify the seed or if *number* is negative or 0, then the time of day from the computer's clock is used as the seed.

SEVERITYEST=*SAS-data-set*
names the input data set that contains the parameter estimates for the severity model. The format of this data set must be the same as the OUTEST= data set that is produced by the SEVERITY procedure.

The names of the regression variables in the scale regression model, if any, are deduced from this data set. In particular, PROC HPCDM assumes that all the variables in the SEVERITYEST= data set that do not appear in the following list are scale regression variables:

- BY variables
- _MODEL_, _TYPE_, _NAME_, and _STATUS_ variables
- variables that represent distribution parameters

The DATA= data set must contain all the regressors in the scale regression model.

To ensure that PROC HPCDM correctly matches the values of regressors and the values of regression parameter estimates, you might need to rename the regressors in the DATA= data set so that their names match the names of the regressors that you specify in the SCALEMODEL statement of the PROC SEVERITY step that fits the severity model.

If you specify a BY statement in the PROC SEVERITY step that creates the SEVERITYEST= data set, then you must specify an identical BY statement in the PROC HPCDM step. Otherwise, PROC HPCDM detects the BY variables as regression variables in the scale regression model, which might produce unexpected results.

VARDEF=*divisor*
specifies the divisor to use in the calculation of variance, standard deviation, kurtosis, and skewness of the compound distribution sample. If the sample size is *N*, then you can specify one of the following values for the *divisor*:

DF
sets the divisor for variance to $N - 1$. This is the default. This also changes the definitions of skewness and kurtosis.

N
sets the divisor to N.

For more information, see the section "Descriptive Statistics" on page 1035.

BY Statement

BY *variable-list* **;**

You can use the BY statement in the HPCDM procedure to process the input data set in groups of observations defined by the BY variables.

If you specify the BY statement, then PROC HPCDM expects the input data set to be sorted in the order of the BY variables unless you specify the NOTSORTED option.

The BY statement is always supported in the single-machine mode of execution. For the distributed mode, it is supported only when the DATA= data set resides on the client machine. In other words, the BY statement is supported only in the client-data (or local-data) mode of the distributed computing model and not for any of the alongside modes, such as the alongside-the-database or alongside HDFS mode.

If you specify the COUNTSTORE= option, then the BY group processing is not supported.

DISTBY Statement

DISTBY *replication-id-variable* **;**

A DISTBY statement is necessary if and only if you specify an ID= variable in the EXTERNALCOUNTS statement. In fact, the *replication-id-variable* must be the same as the ID= variable. This is especially important in the distributed mode of execution, because when the observations in the DATA= data set are distributed to the grid nodes, by specifying the *replication-id-variable* as a DISTBY variable, you are instructing PROC HPCDM to make sure that the observations that have the same value for the *replication-id-variable* are always kept together on one grid node. This is required for correct simulation of the CDM in the presence of the ID= variable.

Contrast this to the BY variables that you specify in the BY statement. The observations of a BY group might be split across all the nodes of the grid, but the observations of a DISTBY group, which is nested within a BY group, are never split across the nodes of the grid.

The *replication-id-variable* must not appear in the BY statement. However, the DATA= data set must be sorted as if the *replication-id-variable* were listed after the BY variables in the BY statement.

Even though the DISTBY statement is important primarily in distributed mode, you must also specify it in single-machine mode.

EXTERNALCOUNTS Statement

EXTERNALCOUNTS *COUNT=frequency-variable* < *ID=replication-id-variable* > **;**

The EXTERNALCOUNTS statement enables you to specify externally simulated frequency counts. By default, PROC HPCDM internally simulates the number of loss events by using the frequency model input (COUNTSTORE= item store). However, if you specify the EXTERNALCOUNTS statement, then PROC HPCDM uses the counts that you specify in the DATA= data set and simulates only the severity values internally.

If you specify more than one EXTERNALCOUNTS statement, only the first one is used.

You must specify the following option in the EXTERNALCOUNTS statement:

COUNT=*count-variable*
specifies the variable that contains the simulated counts. This variable must be present in the DATA= data set.

You can also specify the following option in the EXTERNALCOUNTS statement:

ID=*replication-id-variable*
specifies the variable that contains the replicate identifier. This variable must be present in the DATA= data set. Furthermore, you must specify the DISTBY statement with *replication-id-variable* as the only DISTBY variable to ensure correct simulation.

The observations of DATA= data set must be arranged such that the values of the ID= variable are in increasing order in each BY group or in the entire data set if you do not specify the BY statement.

If you do not specify the ID= option, then PROC HPCDM assumes that each observation represents one replication. In other words, the observation number serves as the default replication identifier.

The simulation process of using the external counts to generate the compound distribution sample is described in the section "Simulation with External Counts" on page 1023.

OUTPUT Statement

OUTPUT *OUT=SAS-data-set* < *variable-name-options* > < / *out-option* > **;**

The OUTPUT statement enables you to specify the data set to output the generated compound distribution sample.

If you specify more than one OUTPUT statement, only the first one is used.

You must specify the output data set by using the following option:

OUT=*SAS-data-set*
OUTSAMPLE=*SAS-data-set*
specifies the output data set to contain the simulated compound distribution sample. If you specify programming statements to adjust individual severity values, then this data set contains both unadjusted and adjusted samples.

You can specify the following *variable-name-options* to control the names of the variables created in the OUT= data set:

ADJSAMPLEVAR=*variable-name*
specifies the name of the variable to contain the adjusted compound distribution sample in the OUT= data set. If you do not specify ADJSAMPLEVAR= option, then "_AGGADJSEV_" is used by default.

This option is ignored if you do not specify the ADJUSTEDSEVERITY= option and the programming statements to adjust the simulated severity values.

SAMPLEVAR=*variable-name*
specifies the name of the variable to contain the simulated sample in the OUT= data set. If you do not specify SAMPLEVAR= option, then "_AGGSEV_" is used by default.

Further, you can request that the perturbed samples be written to the OUT= data set by specifying the following *out-option*:

PERTURBOUT
specifies that all the perturbed samples be written to the OUT= data set. Each perturbed sample is identified by the _DRAWID_ variable in the OUT= data set. A value of 0 for the _DRAWID_ variable indicates an unperturbed sample.

Separate compound distribution samples are generated for each combination of specified severity and frequency models. The _SEVERITYMODEL_ and _COUNTMODEL_ columns in the OUT= data set identify the severity and frequency models, respectively, that are used to generate the sample in the SAMPLEVAR= and ADJSAMPLEVAR= variables.

OUTSUM Statement

OUTSUM *OUT=SAS-data-set statistic-keyword<=variable-name> <... statistic-keyword<=variable-name>> <outsum-options>* **;**

The OUTSUM statement enables you to specify the data set in which PROC HPCDM writes the summary statistics of the compound distribution samples.

If you specify more than one OUTSUM statement, only the first one is used.

You must specify the output data set by using the following option:

OUT=*SAS-data-set*
OUTSUM=*SAS-data-set*
specifies the output data set that contains the summary statistics of each of the simulated compound distribution samples. You can control the summary statistics that appear in this data set by specifying different *statistic-keywords* and *outsum-options*.

If you execute the HPCDM procedure in distributed mode, only the client-data (local-data) and through-the-client data access modes are supported for this data set. In other words, the libref that you specify for this data set should not point to a distributed database appliance. For more information about data access modes, see the section "Data Access Modes" on page 67 of Chapter 3, "Shared Concepts and Topics."

You can request that one or more predefined statistics of the compound distribution sample be written to the OUTSUM= data set. For each specification of the form *statistic-keyword<=variable-name>*, the statistic that is specified by the *statistic-keyword* is written to a variable named *variable-name*. If you do not specify

the *variable-name*, then the statistic is written to a variable named *statistic-keyword*. You can specify the following *statistic-keywords*:

KURTOSIS
KURT
specifies the kurtosis of the compound distribution sample.

MEAN
specifies the mean of the compound distribution sample.

MEDIAN
Q2
P50
specifies the median (the 50th percentile) of the compound distribution sample.

P01
specifies the 1st percentile of the compound distribution sample.

P05
specifies the 5th percentile of the compound distribution sample.

P95
specifies the 95th percentile of the compound distribution sample.

P99
specifies the 99th percentile of the compound distribution sample.

P99_5
P995
specifies the 99.5th percentile of the compound distribution sample.

Q1
P25
specifies the lower or 1st quartile (the 25th percentile) of the compound distribution sample.

Q3
P75
specifies the upper or 3rd quartile (the 75th percentile) of the compound distribution sample.

QRANGE
specifies the interquartile range (Q3–Q1) of the compound distribution sample.

SKEWNESS
SKEW
specifies the skewness of the compound distribution sample.

STDDEV
STD
specifies the standard deviation of the compound distribution sample.

All percentiles are computed by using the method that you specify for the PCTLDEF= option in the PROC HPCDM statement. You can also request additional percentiles to be reported in the OUTSUM= data set by specifying the following *outsum-options*:

PCTLPTS=*percentile-list*

specifies one or more percentiles that you want to be computed and written to the OUTSUM= data set. This option is useful if you need to request percentiles that are not available in the preceding list of *statistic-keyword* values. Each percentile value must belong to the (0,100) open interval. The *percentile-list* is a comma-separated list of numbers. You can also use a list notation of the form "*< number1 > to < number2 > by < increment >*". For example, the following two options are equivalent:

```
pctlpts=10, 20, 99.6, 99.7, 99.8, 99.9
pctlpts=10, 20, 99.6 to 99.9 by 0.1
```

The name of the variable for a given percentile value is decided by the PCTLNAME= option.

PCTLNAME=*percentile-variable-name-list*

specifies the names of the variables that contain the estimates of the percentiles that you request by using the PCTLPTS= option.

If you do not specify the PCTLNAME= option, then each percentile value *t* in the list of values in the PCTLPTS= option is written to the variable named "P*t*," where the decimal point in *t*, if any, is replaced by an underscore.

The *percentile-variable-name-list* is a space-separated list of names. You can also use a shortcut notation of <prefix>*m*–<prefix>*n* for two integers *m* and *n* ($m < n$) to generate the following list of names: <prefix>*m*, <prefix>$m + 1$, ..., and <prefix>*n*. For example, the following two options are equivalent:

```
pctlname=p1 p2 pc5 pc6 pc7 pc8 pc9 pc10
pctlname=p1 p2 pc5-pc10
```

The name in *j*th position of the expanded name list of the PCTLNAME= option is used to create a variable for a percentile value in the *j*th position of the expanded value list of the PCTLPTS= option. If you specify k_n names in the PCTLNAME= option and k_v percentile values in the PCTLPTS= option, and if $k_n < k_v$, then the first k_n percentiles are written to the variables that you specify and the remaining $k_v - k_n$ percentiles are written to the variables that have the name of the form P*t*, where *t* is the text representation of the percentile value that is formed by retaining at most PCTLNDEC= digits after the decimal point and replacing the decimal point with an underscore ('_'). For example, assume you specify the options

```
pctlpts=10, 20, 99.3 to 99.5 by 0.1, 99.995
pctlname=pten ptwenty ninenine3-ninenine5
```

Then PROC HPCDM writes the 10th and 20th percentiles to pten and ptwenty variables, respectively; the 99.3rd through 99.5th percentiles to ninenine3, ninenine4, and ninenine5 variables, respectively; and the remaining 99.995th percentile to the P99_995 variable.

If a percentile value in the PCTLPTS= option matches a percentile value implied by one of the predefined percentile statistics and you specify the corresponding *statistic-keyword*, then the variable name that is implied by the *statistic-keyword< =variable-name >* specification takes precedence over the name that you specify in the PCTLNAME= option. For example, assume you specify the predefined percentile statistic of P95 as in the OUTSUM statement

```
outsum out=mypctls p95=ninetyfifth
        pctlpts=95 to 99 by 1 pctlname=pct95-pct99;
```

Then the 95th percentile is written to the ninetyfifth variable instead of the pct95 variable that the PCTLNAME= option implies.

PCTLNDEC=*integer-value*
specifies the maximum number of decimal places to use while creating the names of the variables for the percentile values in the PCTLPTS= option. The default value is 3. For example, for a percentile value of 99.9995, PROC HPCDM creates a variable named P99_999 by default, but if you specify PCTLNDEC=4, then the variable is named P99_9995.

The PCTLNDEC= option is used only for percentile values for which you do not specify a name in the PCTLNAME= option.

Note that all variable names in the OUTSUM= data set have a limit of 32 characters. If a name exceeds that limit, then it is truncated to contain only the first 32 characters. For more information about the variables in the OUTSUM= data set, see the section "Output Data Sets" on page 1038.

PERFORMANCE Statement

PERFORMANCE *options* **;**

The PERFORMANCE statement defines performance parameters for distributed and multithreaded computing, passes variables that describe the distributed computing environment, and requests detailed results about the performance characteristics of PROC HPCDM.

You can also use the PERFORMANCE statement to control whether a high-performance analytical procedure executes in single-machine or distributed mode.

For more information about the PERFORMANCE statement, see the section "PERFORMANCE Statement" on page 87 of Chapter 3, "Shared Concepts and Topics."

SEVERITYMODEL Statement

SEVERITYMODEL *severity-model-list* **;**

The SEVERITYMODEL statement specifies one or more severity distribution models that you want to use in simulating a compound distribution sample. The *severity-model-list* is a space-separated list of names of severity models that you would use with PROC SEVERITY's DIST statement. The SEVERITYEST= data set must contain all the severity models in the list. If you specify a name that does not appear in the _MODEL_ column of the SEVERITYEST= data set, then that name is ignored.

If you specify more than one SEVERITYMODEL statement, only the first one is used.

If you do not specify a SEVERITYMODEL statement, then this is equivalent to specifying all the severity models that appear in the SEVERITYEST= data set.

A compound distribution sample is generated for each of the severity models by compounding that severity model with the frequency model that you specify in the COUNTSTORE= item store or the external frequency model that is encoded by the COUNT= variable that you specify in the EXTERNALCOUNTS statement.

Programming Statements

In PROC HPCDM, you can use a series of programming statements that use variables in the DATA= data set to adjust an individual severity value. The adjusted severity values are aggregated to form a separate adjusted compound distribution sample.

The programming statements are executed for each simulated individual severity value. The observation of the input data set that is used to evaluate the programming statements is determined by the simulation procedure that is described in the section "Simulation Procedure" on page 1020.

For more information, see the section "Simulation of Adjusted Compound Distribution Sample" on page 1027.

Details: HPCDM Procedure

Specifying Scenario Data in the DATA= Data Set

A scenario represents a state of the world for which you want to estimate the distribution of aggregate losses. The state consists of one or more entities that generate the loss events. For example, an entity might be an individual who has an insurance policy or an organization that has a workers' compensation policy. Each entity has some characteristics of its own and some external factors that affect the frequency with which it generates the losses and the severity of each loss. For example, characteristics of an individual with an automobile insurance policy can include various demographics of the individual and various features of the automobile. Characteristics of an organization with a workers' compensation policy can be the number of employees, revenue, ratio of temporary to permanent employees, and so on. The organization can also be affected by external macroeconomic factors such as GDP and unemployment of the country where the organization operates and factors that affect its industry. You need to quantify and specify all these characteristics as external factors (regressors) when you fit severity and frequency models.

You should specify all the information about a scenario in the DATA= data set that you specify in the PROC HPCDM statement. Each observation in the DATA= data set encodes the characteristics of an entity. For proper simulation of severities, you must specify in the DATA= data set all the characteristics that you use as regressors in the severity scale regression models. When you use the COUNTSTORE= option to specify the frequency model, you must specify in the DATA= data set all the characteristics that you use as regressors in the frequency model in order to properly simulate the counts. All the regressors are expected to have nonmissing values. If any of the regressors have a missing value in an observation, then that observation is ignored.

The information in the DATA= data set is interpreted as follows, based on whether you specify the EXTERNALCOUNTS statement:

- If you do not specify the EXTERNALCOUNTS statement, then all the observations in the data set form a scenario. The observations are used together to compute one random draw from the compound distribution. The total number of draws is equal to the value that you specify in the NREPLICATES= option. The simulation process is described in the section "Simulation with Regressors and No External Counts" on page 1021 and illustrated using an example in the section "Illustration of Aggregate Loss Simulation Process" on page 1022.

In this case, the distributed data access mode for the DATA= data set must be either client-data (local-data) mode or through-the-client mode—that is, the DATA= data set should not be stored on a distributed appliance. For more information about data access modes, see the section "Data Access Modes" on page 67 of Chapter 3, "Shared Concepts and Topics."

- If you specify the EXTERNALCOUNTS statement, then the DATA= data set is expected to contain multiple replications (draws) of the frequency counts that you simulate externally for a scenario. The DATA= data set must contain the COUNT= variable that you specify in the EXTERNALCOUNTS statement. The replications are identified by the observation number or the ID= variable that you specify in the EXTERNALCOUNTS statement. For each observation in a given replication, the COUNT= variable is expected to contain the count of losses that are generated by the entity associated with that observation. All the observations of a given replication are used together to compute one random draw from the compound distribution. The size of the compound distribution sample is equal to the number of distinct replications that you specify in the DATA= data set, multiplied by the value that you specify in the NREPLICATES= option. The simulation process is described in the section "Simulation with External Counts" on page 1023 and illustrated using an example in the section "Illustration of the Simulation Process with External Counts" on page 1024.

  In this case, the distributed data access mode for the DATA= data set can be any of the supported data access modes. For more information about data access modes, see the section "Data Access Modes" on page 67 of Chapter 3, "Shared Concepts and Topics."

In both cases, an observation can also contain severity adjustment variables that you can use to adjust the severity of the losses generated by that entity, based on some policy rules. For more information about simulating the adjusted compound distribution sample, see the section "Simulation of Adjusted Compound Distribution Sample" on page 1027.

If you specify severity and frequency models that have no regression effects in them and if you do not specify externally simulated counts in the EXTERNALCOUNTS statement, then you do not need to specify the DATA= data set. This case corresponds to a fixed scenario that is represented entirely by the distribution parameters of the models.

Simulation Procedure

PROC HPCDM selects a simulation procedure based on whether you specify external counts or you request that PROC HPCDM simulate the counts, and whether the severity or frequency models contain regression effects. The following sections describe the process for the different scenarios.

Simulation with No Regressors and No External Counts

If you specify severity and frequency models that have no regression effects in them, and if you do not specify externally simulated counts in the EXTERNALCOUNTS statement, then PROC HPCDM uses the following simulation procedure.

The process is described for one severity distribution, *dist*. If you specify multiple severity distributions in the SEVERITYMODEL statement, then the process is repeated for each specified distribution.

The following steps are repeated M times to generate a compound distribution sample of size M, where M is the value that you specify in the NREPLICATES= option or the default value of that option:

1. Use the frequency model that you specify in the COUNTSTORE= option to draw a value N from the count distribution. N is the number of loss events that are expected to occur in the time period that is being simulated.
2. Draw N values, X_j $(j = 1, \ldots, N)$, from the severity distribution *dist* with parameters that you specify in the SEVERITYEST= data set.
3. Add the N severity values that are drawn in step 2 to compute one point S from the compound distribution as

$$S = \sum_{j=1}^{N} X_j$$

Note that although it is more common to fit the frequency model with regressors, PROC COUNTREG enables you to fit a frequency model without regressors. If you do not specify any regressors in the MODEL statement of the COUNTREG procedure, then it fits a model that contains only an intercept.

Simulation with Regressors and No External Counts

If the severity or frequency models have regression effects and if you do not specify externally simulated counts in the EXTERNALCOUNTS statement, then you must specify a DATA= data set to provide values of the regression variables, which together represent a scenario for which you want to simulate the CDM. In this case, PROC HPCDM uses the following simulation procedure.

The process is described for one severity distribution. If you specify multiple severity distributions in the SEVERITYMODEL statement, then the process is repeated for each specified distribution.

Note that you are doing scenario analysis when regression effects are present. Let K denote the number of observations that form the scenario. This is the number of observations either in the current BY group or in the entire DATA= data set if you do not specify the BY statement. If $K > 1$, then you are modeling the scenario for a group of entities. If $K = 1$, then you are modeling the scenario for one entity.

The following steps are repeated M times to generate a compound distribution sample of size M, where M is the value that you specify in the NREPLICATES= option or the default value of that option:

1. For each observation k $(k = 1, \ldots, K)$, a count N_k is drawn from the frequency model that you specify in the COUNTSTORE= option. The parameters of this model are determined by the frequency regressors in observation k. N_k represents the number of loss events that are expected to be generated by entity k in the time period that is being simulated.
2. Counts from all observations are added to compute $N = \sum_{k=1}^{K} N_k$. N is the total number of loss events that are expected to occur in the time period that is being simulated.
3. N number of random draws are made from the severity distribution, and they are added to generate one point of the compound distribution sample. Each of the N draws uses one of the K observations. If you specify a scale regression model for the severity distribution, then the scale parameter of the severity distribution is determined by the values of the severity regressors in the observation that is chosen for that draw.

If you specify the BY statement, then a separate sample of size M is created for each BY group in the DATA= data set.

Illustration of Aggregate Loss Simulation Process

As an illustration of the simulation process, consider a very simple example of analyzing the distribution of an aggregate loss that is incurred by a set of policyholders of an automobile insurance company in a period of one year. It is postulated that the frequency and severity distributions depend on three variables: Age, Gender (1: female, 2: male), and CarType (1: sedan, 2: sport utility vehicle). So these variables are used as regressors while you fit the count model and severity scale regression model by using the COUNTREG and SEVERITY procedures, respectively. Now, consider that you want to use the fitted frequency and severity models to estimate the distribution of the aggregate loss that is incurred by a set of five policyholders together. Let the characteristics of the five policyholders be encoded in a SAS data set named Work.Scenario that has the following contents:

```
Obs      age      gender      carType
1        30       2           1
2        25       1           2
3        45       2           2
4        33       1           1
5        50       1           1
```

The column Obs contains the observation number. It is shown only for the purpose of illustration. It need not be present in the data set. The following PROC HPCDM step simulates the scenario in the Work.Scenario data set:

```
proc hpcdm data=scenario
        severityest=<severity parameter estimates data set>
        countstore=<count model store> nreplicates=<sample size>;
    severitymodel <severity distribution name(s)>;
run;
```

The following process generates a sample from the aggregate loss distribution for the scenario in the Work.Scenario data set:

1. Use the values Age=30, Gender=2, and CarType=1 in the first observation to draw a count from the count distribution. Let that count be 2. Repeat the process for the remaining four observations. Let the counts be as shown in the Count column in the following table:

   ```
   Obs      age      gender      carType   count
   1        30       2           1         2
   2        25       1           2         1
   3        45       2           2         2
   4        33       1           1         3
   5        50       1           1         0
   ```

   Note that the Count column is shown for illustration only; it is not added as a variable to the DATA= data set.

2. The simulated counts from all the observations are added to get a value of $N = 8$. This means that for this particular sample point, you expect a total of eight loss events in a year from these five policyholders.

3. For the first observation, the scale parameter of the severity distribution is computed by using the values Age=30, Gender=2, and CarType=1. That value of the scale parameter is used together with estimates of the other parameters from the SEVERITYEST= data set to make two draws from the severity distribution. Each of the draws simulates the magnitude of the loss that is expected from the first policyholder. The process is repeated for the remaining four policyholders. The fifth policyholder does not generate any loss event for this particular sample point, so no severity draws are made by using the fifth observation. Let the severity draws, rounded to integers for convenience, be as shown in the _SEV_ column in the following table:

Obs	age	gender	carType	count	_sev_		
1	30	2	1	2	350	2100	
2	25	1	2	1	4500		
3	45	2	2	2	700	4300	
4	33	1	1	3	600	1500	950
5	50	1	1	0			

Note that the _SEV_ column is shown for illustration only; it is not added as a variable to the DATA= data set.

PROC HPCDM adds the severity values of the eight draws to compute an aggregate loss value of 15,000. After recording this amount in the sample, the process returns to step 1 to compute the next point in the aggregate loss sample. For example, in the second iteration, the count distribution of each policyholder might generate one loss event for a total of five loss events, and the five severity draws from the severity distributions that govern each of the policyholders might add up to 5,000. Then, the value of 5,000 is recorded as the second point in the aggregate loss sample. The process continues until *M* aggregate loss sample points are simulated, where the *M* is the value that you specify in the NREPLICATES= option.

Simulation with External Counts

If you specify externally simulated counts by using the EXTERNALCOUNTS statement, then each replication in the input data set represents the loss events generated by an entity. An entity can be an individual or organization for which you want to estimate the compound distribution. If an entity has any characteristics that are used as external factors (regressors) in developing the severity scale regression model, then you must specify the values of those factors in the DATA= data set. If you specify the ID= variable, then multiple observations for the same replication ID represent different entities in a group for which you are simulating the CDM.

PROC HPCDM uses the following simulation procedure in the presence of externally simulated counts.

The process is described for one severity distribution. If you specify multiple severity distributions in the SEVERITYMODEL statement, then the process is repeated for each specified distribution.

Let there be *M* distinct replications in the current BY group of the DATA= data set or in the entire DATA= data set if you do not specify the BY statement. A replication is identified by either the observation number or the value of the ID= variable that you specify in the EXTERNALCOUNTS statement.

For each of the M values of the replication identifier, the following steps are executed R times, where R is the value of the NREPLICATES= option or the default value of that option:

1. Compute the total number of losses, N. If there are K ($K \geq 1$) observations for the current value of the replication identifier, then $N = \sum_{k=1}^{K} N_k$, where N_k is the value of the COUNT= variable for observation k.
2. N number of random draws are made from the severity distribution, and they are added to generate one point of the compound distribution sample.

This process generates a compound distribution sample of size $M \times R$. If you specify the BY statement, then a separate sample of size $M \times R$ is created for each BY group in the DATA= data set.

Illustration of the Simulation Process with External Counts

In order to illustrate the simulation process, consider the following simple example. In this example, your severity model does not contain any regressors. An example that uses a severity scale regression model is illustrated later. Assume that you have made 10 random draws from an external count model and recorded them in the ExtCount variable of a SAS data set named Work.Counts1 as follows:

```
Obs      extCount
1        3
2        2
3        0
4        1
5        3
6        4
7        1
8        2
9        0
10       5
```

Because the data set does not contain an ID= variable, the observation number that is shown in the Obs column acts as the replicate identifier. The following PROC HPCDM step simulates an aggregate loss sample by using the Work.Counts1 data set:

```
proc hpcdm data=work.counts1 nreplicates=5
        severityest=<severity parameter estimates data set>;
    severitymodel <severity distribution name(s)>;
    externalcounts count=extCount;
run;
```

The simulation process works as follows:

1. For the first replication, which is associated with the first observation, three severity values are drawn from the severity distribution by using the parameter estimates that you specify in the SEVERITYEST= data set. If the severity values are 150, 500, and 320, then their sum of 970 is recorded as the first point of the aggregate loss sample. Because the value of the NREPLICATES= option is 5, this process of drawing three severity values and adding them to form a point of the aggregate loss sample is repeated four more times to generate a total of five sample points that correspond to the first observation.

2. For the second replication, two severity values are drawn from the severity distribution. If the severity values are 450 and 100, then their sum of 550 is recorded as a point of the aggregate loss sample. This process of drawing two severity values and adding them to form a point of the aggregate loss sample is repeated four more times to generate a total of five sample points that correspond to the second observation.

3. The process continues until all the replications, which are observations in this case, are exhausted.

The process results in an aggregate loss sample of size 50, which is equal to the number of replications in the data set (10) multiplied by the value of the NREPLICATES= option (5).

Now, consider an example in which the severity models in the SEVERITYEST= data set are scale regression models. In this case, the severity distribution that is used for drawing the severity value is decided by the values of regressors in the observation that is being processed. Consider that you want to simulate the aggregate loss that is incurred by one policyholder and you have recorded, in the ExtCount variable, the results of 10 random draws from an external count model. The DATA= data set has the following contents:

Obs	age	gender	carType	extCount
1	30	2	1	5
2	30	2	1	2
3	30	2	1	0
4	30	2	1	1
5	30	2	1	3
6	30	2	1	4
7	30	2	1	1
8	30	2	1	2
9	30	2	1	0
10	30	2	1	5

The simulation process in this case is the same as the process in the previous case of no regressors, except that the severity distribution that is used for drawing the severity values has a scale parameter that is determined by the values of the regressors Age, Gender, and CarType in the observation that is being processed. In this particular example, all observations have the same value for all regressors, indicating that you are modeling a scenario in which the characteristics of the policyholder do not change during the time for which you have simulated the number of events. You can also model a scenario in which the characteristics of the policyholder change by recording those changes in the values of the appropriate regressors.

Extending this example further, consider that you want to analyze the distribution of the aggregate loss that is incurred by a group of policyholders, as in the example in the section "Illustration of Aggregate Loss Simulation Process" on page 1022. Let the Work.Counts2 data set record multiple replications of the number of losses that might be generated by each policyholder. The contents of the Work.Counts2 data set are as follows:

Obs	replicateId	age	gender	carType	extCount
1	1	30	2	1	2
2	1	25	1	2	1
3	1	45	2	2	3
4	1	33	1	1	5
5	1	50	1	1	1
6	2	30	2	1	3
7	2	25	1	2	2
8	1	45	2	2	0
9	2	33	1	1	4
10	2	50	1	1	1

The ReplicateId variable records the identifier for the replication. Each replication contains multiple observations, such that each observation represents one of the policyholders that you are analyzing. For simplicity, only the first two replications are shown here.

The following PROC HPCDM step simulates an aggregate loss sample by using the Work.Counts2 data set:

```
proc hpcdm data=work.counts2 nreplicates=3
        severityest=<severity parameter estimates data set>;
    severitymodel <severity distribution name(s)>;
    distby replicateId;
    externalcounts count=extCount id=replicateId;
    output out=aggloss samplevar=totalLoss;
run;
```

When you specify an ID= variable in the EXTERNALCOUNTS statement, you must specify the same ID= variable in the DISTBY statement in order for the procedure to work correctly in a distributed computing environment. Further, the DATA= set must be sorted in ascending order of the ID= variable values.

The simulation process works as follows:

1. First, the five observations of the first replication (ReplicateId=1 are analyzed. For the first observation (Obs=1), the scale parameter of the severity distribution is computed by using the values Age=30, Gender=2, and CarType=1. That value of the scale parameter is used together with estimates of the other parameters from the SEVERITYEST= data set to make two draws from the severity distribution. Next, the regressor values of the second observation are used to compute the scale parameter of the severity distribution, which is used to make one severity draw. The process continues such that the regressor values in the third, fourth, and fifth observations are used to decide the severity distribution to make three, five, and one draws from, respectively. Let the severity values that are drawn from the observations of this replication be as shown in the _SEV_ column in the following table, where the _SEV_ column is shown for illustration only; it is not added as a variable to the DATA= data set:

Obs	replicateId	age	gender	carType	extCount	_sev_
1	1	30	2	1	2	700 500
2	1	25	1	2	1	5000
3	1	45	2	2	3	900 1400 300
4	1	33	1	1	5	350 2000 150 800 600
5	1	50	1	1	1	250

The values of all 12 severity draws are added to compute and record the value of 12,950 as the first point of the aggregate loss sample. Because you specify NREPLICATES=3 in the PROC HPCDM step, this process of making 12 severity draws from the respective observations is repeated two more times to generate a total of three sample points for the first replication.

2. The five observations of the second replication (ReplicateId=2) are analyzed next to draw three, two, four, and one severity values from the severity distributions, with scale parameters that are decided by the regressor values in the sixth, seventh, ninth, and tenth observations, respectively. The 10 severity values are added to form a point of the aggregate loss sample. This process of making 10 severity draws from the respective observations is repeated two more times to generate a total of three sample points for the second replication.

If your Work.Counts2 data set contains 10,000 distinct values of ReplicateId, then 30,000 observations are written to the Work.AggLoss data set that you specify in the OUTPUT statement of the preceding PROC HPCDM step. Because you specify SAMPLEVAR=TotalLoss in the OUTPUT statement, the aggregate loss sample is available in the TotalLoss column of the Work.AggLoss data set.

Simulation of Adjusted Compound Distribution Sample

If you specify programming statements that adjust the severity value, then a separate adjusted compound distribution sample is also generated.

Your programming statements are expected to implement an adjustment function f that uses the unadjusted severity value, X_j, to compute and return an adjusted severity value, X_j^a. To compute X_j^a, you might also use the sum of unadjusted severity values and the sum of adjusted severity values.

Formally, if N denotes the number of loss events that are to be simulated for the current replication of the simulation process, then for the severity draw, X_j, of the jth loss event ($j = 1, \ldots, N$), the adjusted severity value is

$$X_j^a = f(X_j, S_{j-1}, S_{j-1}^a)$$

where $S_{j-1} = \sum_{l=1}^{j-1} X_l$ is the aggregate unadjusted loss before X_j is generated and $S_{j-1}^a = \sum_{l=1}^{j-1} X_l^a$ is the aggregate adjusted loss before X_j is generated. The initial values of both types of aggregate losses are set to 0. In other words, $S_0 = 0$ and $S_0^a = 0$.

The aggregate adjusted loss for the replication is S_N^a, which is denoted by S^a for simplicity, and is defined as

$$S^a = \sum_{j=1}^{N} X_j^a$$

In your programming statements that implement f, you can use the following keywords as placeholders for the input arguments of the function f:

SEV
indicates the placeholder for X_j, the unadjusted severity value. PROC HPCDM generates this value as described in the section "Simulation with No Regressors and No External Counts" on page 1020 (step 2) or the section "Simulation with Regressors and No External Counts" on page 1021 (step 3). PROC HPCDM supplies this value to your program.

CUMSEV
indicates the placeholder for S_{j-1}, the sum of unadjusted severity values that PROC HPCDM generates before X_j is generated. PROC HPCDM supplies this value to your program.

CUMADJSEV
indicates the placeholder for S^a_{j-1}, the sum of adjusted severity values that are computed by your programming statements before X_j is generated and adjusted. PROC HPCDM supplies this value to your program.

In your programming statements, you must assign the value of X^a_j, the output of function f, to a symbol that you specify in the ADJUSTEDSEVERITY= option in the PROC HPCDM statement. PROC HPCDM uses the final assigned value of this symbol as the value of X^a_j.

You can use most DATA step statements and functions in your program. The DATA step file and the data set I/O statements (for example, INPUT, FILE, SET, and MERGE) are not available. However, some functionality of the PUT statement is supported. For more information, see the section "PROC FCMP and DATA Step Differences" in *Base SAS Procedures Guide*.

The simulation process that generates the aggregate adjusted loss sample is identical to the process that is described in the section "Simulation with Regressors and No External Counts" on page 1021 or the section "Simulation with External Counts" on page 1023, except that after making each of the N severity draws, PROC HPCDM executes your severity adjustment programming statements to compute the adjusted severity (X^a_j). All the N adjusted severity values are added to compute S^a, which forms a point of the aggregate adjusted loss sample. The process is illustrated using an example in the section "Illustration of Aggregate Adjusted Loss Simulation Process" on page 1030.

Using Severity Adjustment Variables

If you do not specify the DATA= data set, then your ability to adjust the severity value is limited, because you can use only the current severity draw, sums of unadjusted and adjusted severity draws that are made before the current draw, and some constant numbers to encode your adjustment policy. That is sufficient if you want to estimate the distribution of aggregate adjusted loss for only one entity. However, if you are simulating a scenario that contains more than one entity, then it might be more useful if the adjustment policy depends on factors that are specific to each entity that you are simulating. To do that, you must specify the DATA= data set and encode such factors as *adjustment variables* in the DATA= data set. Let A denote the set of values of the adjustment variables. Then, the form of the adjustment function f that computes the adjusted severity value becomes

$$X^a_j = f(X_j, S_{j-1}, S^a_{j-1}, A)$$

PROC HPCDM reads the values of adjustment variables from the DATA= data set and supplies the set of those values (A) to your severity adjustment program. For an invocation of f with an unadjusted severity value of X_j, the values in set A are read from the same observation that is used to simulate X_j.

All adjustment variables that you use in your program must be present in the DATA= data set. You must not use any keyword for a placeholder symbol as a name of any variable in the DATA= data set, whether the variable is a severity adjustment variable or a regressor in the frequency or severity model. Further, the following restrictions apply to the adjustment variables:

- You can use only numeric-valued variables in PROC HPCDM programming statements. This restriction also implies that you cannot use SAS functions or call routines that require character-valued arguments, unless you pass those arguments as constant (literal) strings or characters.

- You cannot use functions that create lagged versions of a variable in PROC HPCDM programming statements. If you need lagged versions, then you can use a DATA step before the PROC HPCDM step to add those versions to the input data set.

The use of adjustment variables is illustrated using an example in the section "Illustration of Aggregate Adjusted Loss Simulation Process" on page 1030.

Aggregate Adjusted Loss Simulation for a Multi-entity Scenario

If you are simulating a scenario that consists of multiple entities, then you can use some additional pieces of information in your severity adjustment program. Let the scenario consist of K entities and let N_k denote the number of loss events that are incurred by kth entity ($k = 1, \ldots, K$) in the current iteration of the simulation process. The total number of severity draws that need to be made is $N = \sum_{k=1}^{K} N_k$. The aggregate adjusted loss is now defined as

$$S^a = \sum_{k=1}^{K} \sum_{j=1}^{N_k} X^a_{k,j}$$

where $X^a_{k,j}$ is an adjusted severity value of the jth draw ($j = 1, \ldots, N_k$) for the kth entity, and the form of the adjustment function f that computes $X^a_{k,j}$ is

$$X^a_{k,j} = f(X_{k,j}, S_{k,j-1}, S^a_{k,j-1}, S_{n-1}, S^a_{n-1}, A)$$

where $X_{k,j}$ is the value of the jth draw of unadjusted severity for the kth entity. $S_{k,j-1} = \sum_{l=1}^{j-1} X_{k,l}$ and $S^a_{k,j-1} = \sum_{l=1}^{j-1} X^a_{k,l}$ are the aggregate unadjusted loss and the aggregate adjusted loss, respectively, for the kth entity before $X_{k,j}$ is generated. The index n ($n = 1, \ldots, N$) keeps track of the total number of severity draws, across all entities, that are made before $X_{k,j}$ is generated. So $S_{n-1} = \sum_{l=1}^{n-1} X_l$ and $S^a_{n-1} = \sum_{l=1}^{n-1} X^a_l$ are the aggregate unadjusted loss and aggregate adjusted loss, respectively, for all the entities that are processed before $X_{k,j}$ is generated. Note that S_{n-1} and S^a_{n-1} include the $j-1$ draws that are made for the kth entity before $X_{k,j}$ is generated.

The initial values of all types of aggregate losses are set to 0. In other words, $S_0 = 0$, $S^a_0 = 0$, and for all values of k, $S_{k,0} = 0$ and $S^a_{k,0} = 0$.

PROC HPCDM uses the final value that you assign to the ADJUSTEDSEVERITY= symbol in your programming statements as the value of $X^a_{k,j}$.

In your severity adjustment program, you can use the following two additional placeholder keywords:

CUMSEVFOROBS
indicates the placeholder for $S_{k,j-1}$, which is the total loss that is incurred by the kth entity before the current loss event. PROC HPCDM supplies this value to your program.

CUMADJSEVFOROBS
indicates the placeholder for $S^a_{k,j-1}$, which is the total adjusted loss that is incurred by the kth entity before the current loss event. PROC HPCDM supplies this value to your program.

The previously described placeholder symbols _CUMSEV_ and _CUMADJSEV_ represent S_{n-1} and S^a_{n-1}, respectively. If you have only one entity in the scenario ($K = 1$), then the values of _CUMSEVFOROBS_ and _CUMADJSEVFOROBS_ are identical to the values of _CUMSEV_ and _CUMADJSEV_, respectively.

There is one caveat when a scenario consists of more than one entity ($K > 1$) and when you use any of the symbols for cumulative severity values (_CUMSEV_, _CUMADJSEV_, _CUMSEVFOROBS_, or _CUMADJSEVFOROBS_) in your programming statements. In this case, to make the simulation realistic, it is important to randomize the order of N severity draws across K entities. For more information, see the section "Randomizing the Order of Severity Draws across Observations of a Scenario" on page 1032.

Illustration of Aggregate Adjusted Loss Simulation Process

This section continues the example in the section "Simulation with Regressors and No External Counts" on page 1021 to illustrate the simulation of aggregate adjusted loss.

Recall that the earlier example simulates a scenario that consists of five policyholders. Assume that you want to compute the distribution of the aggregate amount paid to all the policyholders in a year, where the payment for each loss is decided by a deductible and a per-payment limit. To begin with, you must record the deductible and limit information in the input DATA= data set. The following table shows the DATA= data set from the earlier example, extended to include two variables, Deductible and Limit:

Obs	age	gender	carType	deductible	limit
1	30	2	1	250	5000
2	25	1	2	500	3000
3	45	2	2	100	2000
4	33	1	1	500	5000
5	50	1	1	200	2000

The variables Deductible and Limit are referred to as severity adjustment variables, because you need to use them to compute the adjusted severity. Let AmountPaid represent the value of adjusted severity that you are interested in. Further, let the following SAS programming statements encode your logic of computing the value of AmountPaid:

```
amountPaid = MAX(_sev_ - deductible, 0);
amountPaid = MIN(amountPaid, MAX(limit - _cumadjsevforobs_, 0));
```

PROC HPCDM supplies your program with values of the placeholder symbols _SEV_ and _CUMADJSEVFOROBS_, which represent the value of the current unadjusted severity draw and the sum of adjusted severity values from the previous draws, respectively, for the observation that is being processed. The use of _CUMADJSEVFOROBS_ helps you ensure that the payment that is made to a given policyholder in a year does not exceed the limit that is recorded in the Limit variable.

In order to simulate a sample for the aggregate of AmountPaid, you need to submit a PROC HPCDM step whose structure is like the following:

```
proc hpcdm data=<data set name> adjustedseverity=amountPaid
       severityest=<severity parameter estimates data set>
       countstore=<count model store>;
    severitymodel <severity distribution name(s)>;

    amountPaid = MAX(_sev_ - deductible, 0);
    amountPaid = MIN(amountPaid, MAX(limit - _cumadjsevforobs_, 0));
run;
```

The simulation process of one replication that generates one point of the aggregate loss sample and the corresponding point of the aggregate adjusted loss sample is as follows:

1. Use the values Age=30, Gender=2, and CarType=1 in the first observation to draw a count from the count distribution. Let that count be 3. Repeat the process for the remaining four observations. Let the counts be as shown in the Count column in the following table:

Obs	age	gender	carType	deductible	limit	count
1	30	2	1	250	5000	2
2	25	1	2	500	3000	1
3	45	2	2	100	2000	2
4	33	1	1	500	5000	3
5	50	1	1	200	2000	0

   Note that the Count column is shown for illustration only; it is not added as a variable to the DATA= data set.

2. The simulated counts from all the observations are added to get a value of $N = 8$. This means that for this particular replication, you expect a total of eight loss events in a year from these five policyholders.

3. For the first observation, the scale parameter of the severity distribution is computed by using the values Age=30, Gender=2, and CarType=1. That value of the scale parameter is used together with estimates of the other parameters from the SEVERITYEST= data set to make two draws from the severity distribution. The process is repeated for the remaining four policyholders. The fifth policyholder does not generate any loss event for this particular replication, so no severity draws are made by using the fifth observation. Let the severity draws, rounded to integers for convenience, be as shown in the _SEV_ column in the following table, where the _SEV_ column is shown for illustration only; it is not added as a variable to the DATA= data set:

Obs	age	gender	carType	deductible	limit	count	_sev_		
1	30	2	1	250	5000	2	350	2100	
2	25	1	2	500	3000	1	4500		
3	45	2	2	100	2000	2	700	4300	
4	33	1	1	200	5000	3	600	1500	950
5	50	1	1	200	2000	0			

   The sample point for the aggregate unadjusted loss is computed by adding the severity values of eight draws, which gives an aggregate loss value of 15,000. The unadjusted aggregate loss is also referred to as the ground-up loss.

   For each of the severity draws, your severity adjustment programming statements are executed to compute the adjusted severity, which is the value of AmountPaid in this case. For the draws in the preceding table, the values of AmountPaid are as follows:

Obs	deductible	limit	_sev_	_cumadjsevforobs_	amountPaid
1	250	5000	350	0	100
1	250	5000	2100	100	1850
2	500	3000	4500	0	3000
3	100	2000	700	0	600
3	100	2000	4300	600	1400
4	200	5000	600	0	400
4	200	5000	1500	400	1300
4	200	5000	950	1700	750

The adjusted severity values are added to compute the cumulative payment value of 9,400, which forms the first sample point for the aggregate adjusted loss.

After recording the aggregate unadjusted and aggregate adjusted loss values in their respective samples, the process returns to step 1 to compute the next sample point unless the specified number of sample points have been simulated.

In this particular example, you can verify that the order in which the 8 loss events are simulated does not affect the aggregate adjusted loss. As a simple example, consider the following order of draws that is different from the consecutive order that was used in the preceding table:

Obs	deductible	limit	_sev_	_cumadjsevforobs_	amountPaid
4	200	5000	600	0	400
3	100	2000	4300	0	2000
1	250	5000	350	0	100
3	100	2000	700	2000	0
4	200	5000	950	400	750
1	250	5000	2100	100	1850
2	500	3000	4500	0	3000
4	200	5000	1500	1150	1300

Although the payments that are made for individual loss events differ, the aggregate adjusted loss is still 9,400.

However, in general, when you use a cumulative severity value such as _CUMADJSEVFOROBS_ in your program, the order in which the draws are processed affects the final value of aggregate adjusted loss. For more information, see the sections "Randomizing the Order of Severity Draws across Observations of a Scenario" on page 1032 and "Illustration of the Need to Randomize the Order of Severity Draws" on page 1033.

Randomizing the Order of Severity Draws across Observations of a Scenario

If you specify a scenario that consists of a group of more than one entity, then it is assumed that each entity generates its loss events independently from other entities. In other words, the time at which the loss event of one entity is generated or recorded is independent of the time at which the loss event of another entity is generated or recorded. If entity k generates N_k loss events, then the total number of loss events for a group of K entities is $N = \sum_{k=1}^{K} N_k$. To simulate the aggregate loss for this group, N severity draws are made and aggregated to compute one point of the compound distribution sample. However, to honor the assumption of independence among entities, the order of those N severity draws must be randomized across K entities such that no entity is preferred over another.

The K entities are represented by K observations of the scenario in the DATA= data set. If you specify external counts, the K observations correspond to the observations that have the same replication identifier value. If you do not specify the external counts, then the K observations correspond to all the observations in the BY group or in the entire DATA= set if you do not specify the BY statement.

The randomization process over K observations is implemented as follows. First, one of the K observations is chosen at random and one severity value is drawn from the severity distribution implied by that observation, then another observation is chosen at random and one severity value is drawn from its implied severity distribution, and so on. In each step, the total number of events that are simulated for the selected observation k is incremented by 1. When all N_k events for an observation k are simulated, observation k is retired and the process continues with the remaining observations until a total of N severity draws are made. Let $k(j)$

denote a function that implements this randomization by returning an observation k ($k = 1, \ldots, K$) for the jth draw ($j = 1, \ldots, N$). The aggregate loss computation can then be formally written as

$$S = \sum_{j=1}^{N} X_{k(j)}$$

where $X_{k(j)}$ denotes the severity value that is drawn by using observation $k(j)$.

If you do not specify a scale regression model for severity, then all severity values are drawn from the same severity distribution. However, if you specify a scale regression model for severity, then the severity draw is made from the severity distribution that is determined by the values of regressors in observation k. In particular, the scale parameter of the distribution depends on the values of regressors in observation k. If $R(l)$ denotes the scale regression model for observation l and $X_{R(l)}$ denotes the severity value drawn from scale regression model $R(l)$, then the aggregate loss computation can be formally written as

$$S = \sum_{j=1}^{N} X_{R(k(j))}$$

This randomization process is especially important in the context of simulating an adjusted compound distribution sample when your severity adjustment program uses the aggregate adjusted severity observed so far to adjust the next severity value. For an illustration of the need to randomize in such cases, see the next section.

Illustration of the Need to Randomize the Order of Severity Draws

This section uses the example of the section "Illustration of Aggregate Adjusted Loss Simulation Process" on page 1030, but with the following PROC HPCDM step:

```
proc hpcdm data=<data set name> adjustedseverity=amountPaid
       severityest=<severity parameter estimates data set>
       countstore=<count model store>;
    severitymodel <severity distribution name(s)>;

    if (_cumadjsev_ > 15000) then
        amountPaid = 0;
    else do;
        penaltyFactor = MIN(3, 15000/(15000 - _cumadjsev_));
        amountPaid = MAX(0, _sev_ - deductible * penaltyFactor);
    end;
run;
```

The severity adjustment statements in the preceding steps compute the value of AmountPaid by using the following provisions in the insurance policy:

- There is a limit of 15,000 on the total amount that can be paid in a year to the group of policyholders that is being simulated. The amount of payment for each loss event depends on the total amount of payments before that loss event.
- The penalty for incurring more losses is imposed in the form of an increased deductible. In particular, the deductible is increased by the ratio of the maximum cumulative payment (15,000) to the amount that remains available to pay for future losses in the year. The factor by which the deductible can be raised has a limit of three.

This example illustrates only step 3 of the simulation process, where randomization is done. It assumes that step 2 of the simulation process is identical to the step 2 in the example in the section "Illustration of Aggregate Adjusted Loss Simulation Process" on page 1030. At the beginning of step 3, let the severity draws from all the observations be as shown in the _SEV_ column in the following table:

Obs	age	gender	carType	deductible	count	_sev_		
1	30	2	1	250	2	350	2100	
2	25	1	2	500	1	4500		
3	45	2	2	100	2	700	4300	
4	33	1	1	200	3	600	1500	950
5	50	1	1	200	0			

If the order of these eight draws is not randomized, then all the severity draws for the first observation are adjusted before all the severity draws of the second observation, and so on. The execution of the severity adjustment program leads to the following sequence of values for AmountPaid:

Obs	deductible	_sev_	_cumadjsev_	penaltyFactor	amountPaid
1	250	350	0	1	100
1	250	2100	100	1.0067	1848.32
2	500	4500	1948.32	1.1493	3925.36
3	100	700	5873.68	1.6436	535.64
3	100	4300	6409.32	1.7461	4125.39
4	200	600	10534.72	3	0
4	200	1500	10534.72	3	900
4	200	950	11434.72	3	350

The preceding sequence of simulating loss events results in a cumulative payment of 11,784.72.

If the sequence of draws is randomized over observations, then the computation of the cumulative payment might proceed as follows for one instance of randomization:

Obs	deductible	_sev_	_cumadjsev_	penaltyFactor	amountPaid
2	500	4500	0	1	4000
1	250	350	4000	1.3636	9.09
3	100	700	4009.09	1.3648	563.52
4	200	950	4572.61	1.4385	662.30
4	200	1500	5234.91	1.5361	1192.78
1	250	2100	6427.69	1.7498	1662.54
4	200	600	8090.24	2.1708	165.83
3	100	4300	8256.07	2.2242	4077.58

In this example, a policyholder is identified by the value in the Obs column. As the table indicates, PROC HPCDM randomizes the order of loss events not only across policyholders but also across the loss events that a given policyholder incurs. The particular sequence of loss events that is shown in the table results in a cumulative payment of 12,333.65. This differs from the cumulative payment that results from the previously considered nonrandomized sequence of loss events, which tends to penalize the fourth policyholder by always processing her payments after all other payments, with a possibility of underestimating the total paid amount. This comparison not only illustrates that the order of randomization affects the aggregate adjusted loss sample but also corroborates the arguments about the importance of order randomization that are made at the beginning of the section "Randomizing the Order of Severity Draws across Observations of a Scenario" on page 1032.

Parameter Perturbation Analysis

It is important to realize that most of the parameters of the frequency and severity models are estimated and there is uncertainty associated with the parameter estimates. Any compound distribution estimate that is computed by using these uncertain parameter estimates is inherently uncertain. The aggregate loss sample that is simulated by using the mean estimates of the parameters is just one possible sample from the compound distribution. If information about parameter uncertainty is available, then it is recommended that you conduct parameter perturbation analysis that generates multiple samples of the compound distribution, in which each sample is simulated by using a set of perturbed parameter estimates. You can use the NPERTURBEDSAMPLES= option in the PROC HPCDM statement to specify the number of perturbed samples to be generated. The set of perturbed parameter estimates is created by making a random draw of the parameter values from their joint probability distribution. If you specify NPERTURBEDSAMPLES=*P*, then PROC HPCDM creates *P* sets of perturbed parameters and each set is used to simulate a full aggregate sample. The summary analysis of *P* such aggregate loss samples results in a set of *P* estimates for each summary statistic and percentile of the compound distribution. The mean and standard deviation of this set of *P* estimates quantify the uncertainty that is associated with the compound distribution.

The parameter uncertainty information is available in the form of either the variance-covariance matrix of the parameter estimates or standard errors of the parameters estimates. If the variance-covariance matrix is available and is positive definite, then PROC HPCDM assumes that the joint probability distribution of the parameter estimates is a multivariate normal distribution, $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$, where the mean vector $\boldsymbol{\mu}$ is the set of point parameter estimates and Σ is the variance-covariance matrix. If the variance-covariance matrix is not available or is not positive definite, then PROC HPCDM assumes that each parameter has a univariate normal distribution, $\mathcal{N}(\mu, \sigma^2)$, where μ is the point estimate of the parameter and σ is the standard error of the parameter estimate.

For severity models, the point parameter estimates are expected to be available in the SEVERITYEST= data set in observations for which _TYPE_='EST', the standard errors are expected to be available in the SEVERITYEST= data set in observations for which _TYPE_='STDERR', and the variance-covariance matrix is expected to be available in the SEVERITYEST= data set in observations for which _TYPE_='COV'. If you use the SEVERITY procedure to create the SEVERITYEST= data set, then you need to specify the COVOUT option in the PROC SEVERITY statement to make the variance-covariance estimates available in the SEVERITYEST= data set.

For the frequency model, you must use the COUNTREG procedure to create the COUNTSTORE= item store, which always contains the point estimates, standard errors, and variance-covariance matrix of the parameters.

If you specify the ADJUSTEDSEVERITY= option in the PROC HPCDM statement, then a separate perturbation analysis is conducted for the distribution of the aggregate adjusted loss.

Descriptive Statistics

This section provides computational details for the descriptive statistics that are computed for each aggregate loss sample. You can also save these statistics in an OUTSUM= data set by specifying appropriate keywords in the OUTSUM statement.

This section gives specific details about the moment statistics. For more information about the methods of computing percentile statistics, see the description of the PCTLDEF= option in the UNIVARIATE procedure in the *Base SAS Procedures Guide: Statistical Procedures*.

Standard algorithms (Fisher 1973) are used to compute the moment statistics. The computational methods that the HPCDM procedure uses are consistent with those that other SAS procedures use for calculating descriptive statistics.

Mean

The sample mean is calculated as

$$\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n}$$

where n is the size of the generated aggregate loss sample and y_i is the ith value of the aggregate loss.

Standard Deviation

The standard deviation is calculated as

$$s = \sqrt{\frac{1}{d}\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

where n is the size of the generated aggregate loss sample, y_i is the ith value of the aggregate loss, $\bar{y}$ is the sample mean, and d is the divisor controlled by the VARDEF= option in the PROC HPCDM statement:

$$d = \begin{cases} n-1 & \text{if VARDEF=DF (default)} \\ n & \text{if VARDEF=N} \end{cases}$$

Skewness

The sample skewness, which measures the tendency of the deviations to be larger in one direction than in the other, is calculated as

$$\frac{1}{d_s}\sum_{i=1}^{n}\left(\frac{y_i - \bar{y}}{s}\right)^3$$

where n is the size of the generated aggregate loss sample, y_i is the ith value of the aggregate loss, $\bar{y}$ is the sample mean, s is the sample standard deviation, and d_s is the divisor controlled by the VARDEF= option in the PROC HPCDM statement:

$$d_s = \begin{cases} \frac{(n-1)(n-2)}{n} & \text{if VARDEF=DF (default)} \\ n & \text{if VARDEF=N} \end{cases}$$

If VARDEF=DF, then n must be greater than 2.

The sample skewness can be positive or negative; it measures the asymmetry of the data distribution and estimates the theoretical skewness $\sqrt{\beta_1} = \mu_3 \mu_2^{-\frac{3}{2}}$, where μ_2 and μ_3 are the second and third central moments. Observations that are normally distributed should have a skewness near zero.

Kurtosis

The sample kurtosis, which measures the heaviness of tails, is calculated as in Table 18.2 depending on the value that you specify in the VARDEF= option.

Table 18.2 Formulas for Kurtosis

VARDEF Value	Formula
DF (default)	$\frac{n(n+1)}{(n-1)(n-2)(n-3)} \sum_{i=1}^{n} \left(\frac{y_i - \bar{y}}{s} \right)^4 - \frac{3(n-1)^2}{(n-2)(n-3)}$
N	$\frac{1}{n} \sum_{i=1}^{n} \left(\frac{y_i - \bar{y}}{s} \right)^4 - 3$

In these formulas, n is the size of the generated aggregate loss sample, y_i is the ith value of the aggregate loss, $\bar{y}$ is the sample mean, and s is the sample standard deviation. If VARDEF=DF, then n must be greater than 3.

The sample kurtosis measures the heaviness of the tails of the data distribution. It estimates the adjusted theoretical kurtosis denoted as $\beta_2 - 3$, where $\beta_2 = \frac{\mu_4}{\mu_2{}^2}$ and μ_4 is the fourth central moment. Observations that are normally distributed should have a kurtosis near zero.

Input Specification

PROC HPCDM accepts the DATA= and SEVERITYEST= data sets and the COUNTSTORE= item store as input. This section details the information that they are expected to contain.

DATA= Data Set

If you specify the BY statement, then the DATA= data set must contain all the BY variables that you specify in the BY statement and the data set must be sorted by the BY variables unless the BY statement includes the NOTSORTED option.

If the severity models in the SEVERITYEST= data set contain any scale regressors, then all those regressors must be present in the DATA= data set.

If you specify the programming statements to compute an aggregate adjusted loss, and if your specified ADJUSTEDSEVERITY= symbol depends on severity adjustment variables, then the DATA= data set must contain all such variables.

The rest of the contents of the DATA= data set depends on whether you specify the EXTERNALCOUNTS statement. If you specify the EXTERNALCOUNTS statement, then the DATA= data set is expected to contain the COUNT= and ID= variables that you specify in the EXTERNALCOUNTS statement. If you do not specify the EXTERNALCOUNTS statement, then the DATA= data set must contain all the regressors, including zero model regressors, that are present in the count model that the COUNTSTORE= item store contains.

You do not need to specify the DATA= data set if all the following conditions are true:

- You do not specify the BY statement.
- You specify a SEVERITYEST= data set such that none of the severity models are scale regression models.
- You do not specify the EXTERNALCOUNTS statement.
- You specify a COUNTSTORE= item store such that the count model contains no count regressors.
- Your severity adjustment programming statements, if you specify any, do not use any external input.

SEVERITYEST= Data Set

The SEVERITYEST= data set is expected to contain the parameter estimates of the severity models. This is a required data set; you must specify it whenever you use PROC HPCDM.

The SEVERITYEST= data set must have the same format as the OUTEST= data set that is created by the SEVERITY procedure. For more information, see the description of the OUTEST= data set in the SEVERITY procedure in the *SAS/ETS User's Guide*.

If you specify the BY statement, then the SEVERITYEST= data set must contain all the BY variables that you specify in the BY statement. If you do not specify the NOTSORTED option in the BY statement, then the SEVERITYEST= data set must be sorted by the BY variables.

COUNTSTORE= Item Store

The COUNTSTORE= item store is expected to be created by using the STORE statement in the COUNTREG procedure. You must specify the COUNTSTORE= item store when you do not specify the EXTERNALCOUNTS statement. For more information, see the description of the STORE statement in the COUNTREG procedure in the *SAS/ETS User's Guide*.

Output Data Sets

PROC HPCDM writes the output data sets that you specify in the OUT= option of the OUTPUT and OUTSUM statements. The contents of these output data sets are described in the sections "OUTSAMPLE= Data Set" on page 1038 and "OUTSUM= Data Set" on page 1039, respectively.

OUTSAMPLE= Data Set

The OUTSAMPLE= data set records the full sample of the aggregate loss and aggregate adjusted loss.

If you specify the BY statement, then the data are organized in BY groups and the data set contains variables that you specify in the BY statement. In addition, the OUTSAMPLE= data set contains the following variables:

SEVERITYMODEL
: indicates the name of the severity distribution model.

COUNTMODEL
: indicates the name of the count model. If you specify the EXTERNALCOUNTS statement, then the value of this variable is "_EXTERNAL_". If you specify the COUNTSTORE= option, then the value of this variable is "_COUNTSTORE_".

<unadjusted sample variable>
: indicates the value of the unadjusted aggregate loss. The name of this variable is the value of the SAMPLEVAR= option in the OUTPUT statement. If you do not specify the SAMPLEVAR= option, then the variable is named _AGGSEV_.

<adjusted sample variable>
: indicates the value of the adjusted aggregate loss. This variable is created only when you specify the programming statements and the ADJUSTEDSEVERITY= option in the PROC HPCDM statement. The name of this variable is the value of the ADJSAMPLEVAR= option in the OUTPUT statement. If you do not specify the ADJSAMPLEVAR= option, then the variable is named _AGGADJSEV_.

DRAWID
: indicates the identifier for the perturbed sample. This variable is created only when you specify the NPERTURBEDSAMPLES= option in the PROC HPCDM statement. The value of this variable identifies the perturbed sample. A value of 0 for the _DRAWID_ variable indicates an unperturbed sample.

OUTSUM= Data Set

The OUTSUM= data set records the summary statistics and percentiles of the compound distributions of aggregate loss and aggregate adjusted loss. Only the estimates that you request in the OUTSUM statement are written to the OUTSUM= data set. For more information about the method of naming the variables that correspond to the summary statistics or percentiles, see the description of the OUTSUM statement.

If you specify the BY statement, then the data are organized in BY groups and the data set contains variables that you specify in the BY statement. In addition, the OUTSUM= data set contains the following variables:

SEVERITYMODEL
: indicates the name of the severity distribution model.

COUNTMODEL
: indicates the name of the count model. If you specify the EXTERNALCOUNTS statement, then the value of this variable is "_EXTERNAL_". If you specify the COUNTSTORE= option, then the value of this variable is "_COUNTSTORE_".

SAMPLEVAR
: indicates the name of the aggregate loss sample. For an unadjusted sample, the value of the variable is the value of the SAMPLEVAR= option that you specify in the OUTPUT statement or the default value of "_AGGSEV_". For an adjusted sample, the value of the variable is the value of the ADJSAMPLEVAR= option that you specify in the OUTPUT statement or the default value of "_AGGADJSEV_".

DRAWID
: indicates the identifier for the perturbed sample. This variable is created only when you specify the NPERTURBEDSAMPLES= option in the PROC HPCDM statement. The value of this variable identifies the perturbed sample. A value of 0 for _DRAWID_ indicates an unperturbed sample.

Displayed Output

The HPCDM procedure optionally produces displayed output by using the Output Delivery System (ODS). All output is controlled by the PRINT= option in the PROC HPCDM statement. Table 18.3 relates the PRINT= options to ODS tables.

Table 18.3 ODS Tables Produced in PROC HPCDM

ODS Table Name	Description	Option
CompoundInfo	Compound distribution information	Default
DataSummary	Input data summary	Default
Percentiles	Percentiles of the aggregate loss sample	PRINT=PERCENTILES
PerformanceInfo	Execution environment information that pertains to the computational performance	Default
PerturbedPctlSummary	Perturbation analysis of percentiles	PRINT=PERTURBSUMMARY and NPERTURBEDSAMPLES > 0
PerturbedSummary	Perturbation analysis of summary statistics	PRINT=PERTURBSUMMARY and NPERTURBEDSAMPLES > 0
SummaryStatistics	Summary statistics of the aggregate loss sample	PRINT=SUMMARYSTATISTICS
Timing	Timing information for various computational stages of the procedure	DETAILS (PERFORMANCE statement)

PRINT= Option

This section provides detailed descriptions of the tables that are displayed by using different PRINT= options.

- If you do not specify the PRINT= option and if you do not specify the NOPRINT or PRINT=NONE options, then by default PROC HPCDM produces the CompoundInfo, DataSummary, and SummaryStatistics ODS tables.

  The "Compound Distribution Information" table (ODS name: CompoundInfo) displays the information about the severity and count models.

  The "Input Data Summary" table (ODS name: DataSummary) is displayed when you specify the DATA= data set. The table displays the total number of observations and the valid number of observations in the data set. If you specify the EXTERNALCOUNTS statement, then the table also displays the number of replications and total number of loss events across all replications.

- If you specify PRINT=PERCENTILES, the "Percentiles" table (ODS name: Percentiles) is displayed for the distribution of the aggregate loss. The table contains estimates of all the predefined percentiles in addition to the percentiles that you request in the OUTSUM statement.

If you specify the programming statements and the ADJUSTEDSEVERITY= symbol, then an additional table is displayed for the distribution of the aggregate adjusted loss. This table also contains estimates of all the predefined percentiles in addition to the percentiles that you request in the OUTSUM statement.

- If you specify PRINT=PERTURBSUMMARY, two tables are displayed for the distribution of the aggregate loss. The "Perturbed Summary Statistics" table (ODS name: PerturbedSummary) displays the summary of the effect of perturbing model parameters on the following five summary statistics of the distribution: mean, standard deviation, variance, skewness, and kurtosis. The "Perturbed Percentiles" table (ODS name: PerturbedPctlSummary) displays the perturbation summary for all the predefined percentiles in addition to the percentiles that you request in the OUTSUM statement.

  The tables are displayed only if you specify a value greater than 0 for the NPERTURBEDSAMPLES= option.

  If you specify a value of *P* for the NPERTURBEDSAMPLES= option, then for each summary statistic and percentile, an average and standard error of the set of *P* values of that summary statistic or percentile are displayed in the respective perturbation summary tables.

  If you specify the programming statements and the ADJUSTEDSEVERITY= symbol, then additional perturbation summary tables are displayed for the distribution of the aggregate adjusted loss.

- If you specify PRINT=SUMMARYSTATISTICS, the "Summary Statistics" table (ODS name: SummaryStatistics) is displayed for the distribution of the aggregate loss. The table contains estimates of the following summary statistics: the number of observations in the sample, maximum value in the sample, minimum value in the sample, mean, median, standard deviation, interquartile range, variance, skewness, and kurtosis.

  If you specify the programming statements and the ADJUSTEDSEVERITY= symbol, then an additional table of summary statistics is displayed for the distribution of the aggregate adjusted loss.

Performance Information

The "Performance Information" table (ODS name: PerformanceInfo) is produced by default. It displays information about the execution mode. For single-machine mode, the table displays the number of threads that are used. For distributed mode, the table displays the grid mode (symmetric or asymmetric), the number of compute nodes, and the number of threads per node.

If you specify the DETAILS option in the PERFORMANCE statement, PROC HPCDM also produces a "Timing" table (ODS name: Timing) that displays elapsed times (absolute and relative) for the main tasks of the procedure.

ODS Graphics

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*).

Before you create graphs, ODS Graphics must be enabled (for example, with the ODS GRAPHICS ON statement). For more information about enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" in that chapter.

The overall appearance of graphs is controlled by ODS styles. Styles and other aspects of using ODS Graphics are discussed in the section "A Primer on ODS Statistical Graphics" in that chapter.

This section describes the use of ODS for creating graphics with the HPCDM procedure.

NOTE: If you request simulation of an aggregate loss sample of large size, either by specifying a large value for the NREPLICATES= option or by including a large number of replicates in the DATA= data set that you specify in conjunction with the EXTERNALCOUNTS statement, then it is recommended that you not request any plots, because creating plots that have large numbers of points can require a very large amount of hardware resources and can take a very long time. You can disable the generation of plots either by submitting the ODS GRAPHICS OFF statement before submitting the PROC HPCDM step or by specifying the PLOTS=NONE option in the PROC HPCDM statement. It is recommended that you request plots only when the sample size is less than 100,000.

ODS Graph Names

PROC HPCDM assigns a name to each graph that it creates by using ODS. You can use these names to selectively refer to the graphs. The names are listed in Table 18.4.

Table 18.4 ODS Graphics Produced by PROC HPCDM

ODS Graph Name	Plot Description	PLOTS= Option
ConditionalDensityPlot	Conditional density plot	CONDITIONALDENSITY
DensityPlot	Probability density function plot	DENSITY
EDFPlot	Empirical distribution function plot	EDF

Conditional Density Plot

The conditional density plot helps you visually analyze two or three regions of the compound distribution by displaying a density function estimate that is conditional on the values of the aggregate loss that fall in those regions. You can specify the region boundaries in terms of quantiles by using the LEFTQ= and RIGHTQ= suboptions of the PLOTS=CONDITIONALDENSITY option. This is especially useful if you want to see the distribution of aggregate loss values in the right- and left-tail regions.

If you specify the programming statements and the ADJUSTEDSEVERITY= symbol, then a separate set of conditional density plots are displayed for the aggregate adjusted loss.

Probability Density Function Plot

The probability density function (PDF) plot shows the nonparametric estimates of the PDF of the aggregate loss distribution. This plot includes histogram and kernel density estimates.

If you specify the programming statements and the ADJUSTEDSEVERITY= symbol, then a separate density plot is displayed for the aggregate adjusted loss.

Empirical Distribution Function Plot

The empirical density function (EDF) plot shows the nonparametric estimate of the cumulative distribution function of the aggregate loss distribution. You can specify the ALPHA= suboption of the PLOTS=EDF

option to request that the upper and lower confidence limits be plotted for each EDF estimate. By default, the confidence interval is not plotted.

If you specify the programming statements and the ADJUSTEDSEVERITY= symbol, then a separate EDF plot is displayed for the aggregate adjusted loss.

Examples: HPCDM Procedure

Example 18.1: Estimating the Probability Distribution of Insurance Payments

The primary outcome of running PROC HPCDM is the estimate of the compound distribution of aggregate loss, given the distributions of frequency and severity of the individual losses. This aggregate loss is often referred to as the ground-up loss. If you are an insurance company or a bank, you are also interested in acting on the ground-up loss by computing an entity that is derived from the ground-up loss. For example, you might want to estimate the distribution of the amount that you are expected to pay for the losses or the distribution of the amount that you can offload onto another organization, such as a reinsurance company. PROC HPCDM enables you to specify a severity adjustment program, which is a sequence of SAS programming statements that adjust the severity of the individual loss event to compute the entity of interest. Your severity adjustment program can use external information that is recorded as variables in the observations of the DATA= data set in addition to placeholder symbols for information that PROC HPCDM generates internally, such as the severity of the current loss event (_SEV_) and the sum of the adjusted severity values of the events that have been simulated thus far for the current sample point (_CUMADJSEV_). If you are doing a scenario analysis such that a scenario contains more than one observation, then you can also access the cumulative severity and cumulative adjusted severity for the current observation by using the _CUMSEVFOROBS_ and _CUMADJSEVFOROBS_ symbols.

This example continues the example of the section "Scenario Analysis" on page 998 to illustrate how you can estimate the distribution of the aggregate amount that is paid to a group of policyholders. Let the amount that is paid to an individual policyholder be computed by using what is usually referred to as a *disappearing deductible* (Klugman, Panjer, and Willmot 1998, Ch. 2). If X denotes the ground-up loss that a policyholder incurs, d denotes the lower limit on the deductible, d' denotes the upper limit on the deductible, and u denotes the limit on the total payments that are made to a policyholder in a year, then Y, the amount that is paid to the policyholder for each loss event, is defined as follows:

$$Y = \begin{cases} 0 & X \le d \\ d' \frac{X-d}{d'-d} & d < X \le d' \\ X & d' < X \le u \\ u & X > u \end{cases}$$

You can encode this logic by using a set of SAS programming statements.

Extend the Work.GroupOfPolicies data set in the example in the section "Scenario Analysis" on page 998 to include the following three additional variables for each policyholder: LowDeductible to record d, HighDeductible to record d', and Limit to record u. The data set contains the observations as shown in Output 18.1.1.

Output 18.1.1 Scenario Analysis Data for Multiple Policyholders with Policy Provisions

policyholderId	age	gender	carType	annualMiles	education	carSafety	income
1	1.18	2	1	2.2948	3	0.99532	1.59870
2	0.66	2	2	2.8148	1	0.05625	0.67539
3	0.82	1	2	1.6130	2	0.84146	1.05940
4	0.44	1	1	1.2280	3	0.14324	0.24110
5	0.44	1	1	0.9670	2	0.08656	0.65979

lowDeductible	highDeductible	limit	annualLimit
400	1400	7500	10000
300	1300	2500	20000
100	1100	5000	10000
300	800	5000	20000
100	1100	5000	20000

The following PROC HPCDM step estimates the compound distributions of the aggregate loss and the aggregate amount that is paid to the group of policyholders in the Work.GroupOfPolicies data set by using the count model that is stored in the Work.CountregModel item store and the lognormal severity model that is stored in the Work.SevRegEst data set:

```
/* Simulate the aggregate loss distribution and aggregate adjusted
   loss distribution for the scenario with multiple policyholders */
proc hpcdm data=groupOfPolicies nreplicates=10000 seed=13579 print=all
           countstore=work.countregmodel severityest=work.sevregest
           plots=(edf pdf) nperturbedSamples=50
           adjustedseverity=amountPaid;
   severitymodel logn;

   if (_sev_ <= lowDeductible) then
      amountPaid = 0;
   else do;
      if (_sev_ <= highDeductible) then
         amountPaid = highDeductible *
            (_sev_-lowDeductible)/(highDeductible-lowDeductible);
      else
         amountPaid = MIN(_sev_, limit); /* imposes per-loss payment limit */
   end;
run;
```

The preceding step uses a severity adjustment program to compute the value of the symbol AmountPaid and specifies that symbol in the ADJUSTEDSEVERITY= option in the PROC HPCDM step. The program is executed for each simulated loss event. The PROC HPCDM supplies your program with the value of the severity in the _SEV_ placeholder symbol.

The "Sample Summary Statistics" table in Output 18.1.2 shows the summary statistics of the compound distribution of the aggregate ground-up loss. The "Adjusted Sample Summary Statistics" table shows the summary statistics of the compound distribution of the aggregate AmountPaid. The average aggregate payment is about 4,391, as compared to the average aggregate ground-up loss of 5,963.

Output 18.1.2 Summary Statistics of Compound Distributions of the Total Loss and Total Amount Paid

The HPCDM Procedure
Severity Model: Logn
Count Model: NegBin(p=2)

Compound Distribution Information	
Severity Model	Lognormal Distribution
Scale Model Regressors	carType carSafety income
Count Model	NegBin(p=2) Model in Item Store WORK.COUNTREGMODEL

Sample Summary Statistics			
Mean	5962.5	**Median**	4748.4
Standard Deviation	4825.7	**Interquartile Range**	5330.3
Variance	23286981.0	**Minimum**	0
Skewness	2.31699	**Maximum**	68187.9
Kurtosis	11.67151	**Sample Size**	10000

Adjusted Sample Summary Statistics			
Mean	4391.4	**Median**	3751.4
Standard Deviation	3185.1	**Interquartile Range**	4140.4
Variance	10145084.4	**Minimum**	0
Skewness	1.10371	**Maximum**	24291.2
Kurtosis	1.62694	**Sample Size**	10000

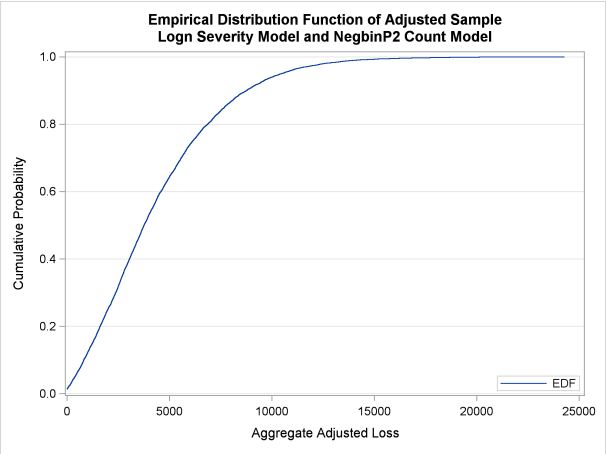
The perturbation summary of the distribution of AmountPaid is shown in Output 18.1.3. It shows that you can expect to pay a median of 3,786 $\pm$ 420 to this group of five policyholders in a year. Also, if the 99.5th percentile defines the worst case, then you can expect to pay 15,588 $\pm$ 1,197 in the worst-case.

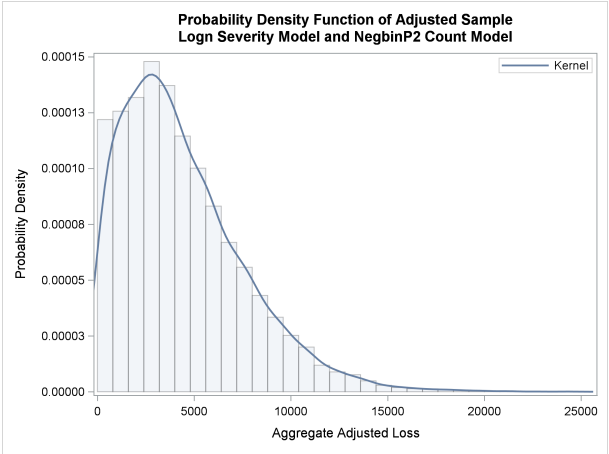
Output 18.1.3 Perturbation Summary of the Total Amount Paid

Adjusted Sample Percentile Perturbation Analysis		
Percentile	**Estimate**	**Standard Error**
1	7.96974	20.33033
5	398.70254	113.56342
25	1995.0	290.18465
50	3785.9	420.19002
75	6119.1	569.62667
95	10403.7	828.00404
99	14088.1	1105.4
99.5	15588.0	1196.8
Number of Perturbed Samples = 50		
Size of Each Sample = 10000		

The empirical distribution function (EDF) and probability density function plots of the aggregate adjusted loss are shown in Output 18.1.4. Both plots indicate a heavy-tailed distribution of the total amount paid.

Output 18.1.4 PDF and EDF Plots of the Compound Distribution of the Total Amount Paid





Now consider that, in the future, you want to modify the policy provisions to add a limit on the total amount of payment that is made to an individual policyholder in one year and to impose a group limit of 15,000 on the total amount of payments that are made to the group as a whole in one year. You can analyze the effects of these modified policy provisions on the distribution of the aggregate paid amount by recording the individual policyholder's annual limit in the AnnualLimit variable of the input data set and then modifying your severity adjustment program by using the placeholder symbols _CUMADJSEVFOROBS_ and _CUMADJSEV_ as shown in the following PROC HPCDM step:

```
/* Simulate the aggregate loss distribution and aggregate adjusted
   loss distribution for the modified set of policy provisions */
proc hpcdm data=groupOfPolicies nreplicates=10000 seed=13579 print=all
           countstore=work.countregmodel severityest=work.sevregest
           plots=none nperturbedSamples=50
           adjustedseverity=amountPaid;
   severitymodel logn;

   if (_sev_ <= lowDeductible) then
      amountPaid = 0;
   else do;
      if (_sev_ <= highDeductible) then
         amountPaid = highDeductible *
            (_sev_-lowDeductible)/(highDeductible-lowDeductible);
      else
         amountPaid = MIN(_sev_, limit); /* imposes per-loss payment limit */

      /* impose policyholder's annual limit */
      amountPaid = MIN(amountPaid, MAX(0,annualLimit - _cumadjsevforobs_));

      /* impose group's annual limit */
      amountPaid = MIN(amountPaid, MAX(0,15000 - _cumadjsev_));
   end;
run;
```

The results of the perturbation analysis for these modified policy provisions are shown in Output 18.1.5. When compared to the results of Output 18.1.3, the additional policy provisions of restricting the total payment to the policyholder and the group have kept the median payment unchanged, but the provisions have reduced the worst-case payment (99.5th percentile) to 14,683 $\pm$ 440 from 15,588 $\pm$ 1,197.

Output 18.1.5 Perturbation Summary of the Total Amount Paid for Modified Policy Provisions

The HPCDM Procedure
Severity Model: Logn
Count Model: NegBin(p=2)

Adjusted Sample Percentile Perturbation Analysis

Percentile	Estimate	Standard Error
0	0	0
1	7.96974	20.33033
5	398.70254	113.56342
25	1995.0	290.18465
50	3785.9	420.19002
75	6119.1	569.62667
95	10377.5	795.71616
99	13767.5	855.56936
99.5	14683.0	440.01020

Number of Perturbed Samples = 50

Size of Each Sample = 10000

Example 18.2: Using Externally Simulated Count Data

The COUNTREG procedure enables you to estimate count regression models that are based on the most commonly used discrete distributions, such as the Poisson, negative binomial (both $p = 1$ and $p = 2$), and Conway-Maxwell-Poisson distributions. PROC COUNTREG also enables you to fit zero-inflated models that are based on Poisson, negative binomial ($p = 2$), and Conway-Maxwell-Poisson distributions. However, there might be situations in which you want to use some other method of fitting count regression models. For example, if you are modeling the number of loss events that are incurred by two financial instruments such that there is some dependency between the two, then you might use some multivariate frequency modeling methods and simulate the counts for each instrument by using the dependency structure between the count model parameters of the two instruments. As another example, you might want to use different types of count models for different BY groups in your data; this is not possible in PROC COUNTREG in SAS/ETS 13.1 and earlier. So you need to simulate the counts for such BY groups externally. PROC HPCDM enables you to supply externally simulated counts by using the EXTERNALCOUNTS statement. PROC HPCDM then does not need to simulate the counts internally; it simulates only the severity of each loss event by using the severity model estimates in the SEVERITYEST= data set. The process is described and illustrated in the section "Simulation with External Counts" on page 1023.

Consider that you are a bank, and as part of quantifying your operational risk, you want to estimate the aggregate loss distributions for two lines of business, retail banking and commercial banking, by using some key risk indicators (KRIs). Assume that your model fitting and model selection process has determined that the Poisson regression model and negative binomial regression model are the best-fitting count models for number of loss events that are incurred in the retail banking and commercial banking businesses, respectively. Let CorpKRI1, CorpKRI2, CbKRI1, CbKRI2, and CbKRI3 be the KRIs that are used in the count regression model of the commercial banking business, and let CorpKRI1, RbKRI1, and RbKRI2 be the KRIs that are used in the count regression model of the retail banking business. Some examples of corporate-level KRIs (CorpKRI1 and CorpKRI2 in this example) are the ratio of temporary to permanent employees and the

number of security breaches that are reported during a year. Some examples of KRIs that are specific to the commercial banking business (CbKRI1, CbKRI2, and CbKRI3 in this example) are number of credit defaults, proportion of financed assets that are movable, and penalty claims against your bank because of processing delays. Some examples of KRIs that are specific to the retail banking business (RbKRI1 and RbKRI2 in this example) are number of credit cards that are reported stolen, fraction of employees who have not undergone fraud detection training, and number of forged drafts and checks that are presented in a year.

Let the severity of each loss event in the commercial banking business be dependent on two KRIs, CorpKRI1 and CbKRI2. Let the severity of each loss event in the retail banking business be dependent on three KRIs, CorpKRI2, RbKRI1, and RbKRI3. Note that for each line of business, the set of KRIs that are used for the severity model is different from the set of KRIs that are used for the count model, although there is some overlap between the two sets. Further, the severity model for retail banking includes a new regressor (RbKRI3) that is not used for any of the count models. Such use of different sets of KRIs for count and severity models is typical of real-world applications.

Let the parameter estimates of the negative binomial and Poisson regression models, as determined by PROC COUNTREG, be available in the Work.CountEstEx2NB2 and Work.CountEstEx2Poisson data sets, respectively. These data sets are produced by using the OUTEST= option in the respective PROC COUNTREG statements. Let the parameter estimates of the best-fitting severity models, as determined by PROC SEVERITY, be available in the Work.SevEstEx2Best data set. You can find the code to prepare these data sets in the PROC HPCDM sample program *hcdmex02.sas*.

Now, consider that you want to estimate the distribution of the aggregate loss for a scenario, which is represented by a specific set of KRI values. The following DATA step illustrates one such scenario:

```
/* Generate a scenario data set for a single operating condition */
data singleScenario (keep=corpKRI1 corpKRI2 cbKRI1 cbKRI2 cbKRI3
                          rbKRI1 rbKRI2 rbKRI3);
   array x{8} corpKRI1 corpKRI2 cbKRI1 cbKRI2 cbKRI3 rbKRI1 rbKRI2 rbKRI3;
   call streaminit(5151);
   do i=1 to dim(x);
      x(i) = rand('NORMAL');
   end;
   output;
run;
```

The Work.SingleScenario data set contains all the KRIs that are included in the count and severity models of both business lines. Note that if you standardize or scale the KRIs while fitting the count and severity models, then you must apply the same standardization or scaling method to the values of the KRIs that you specify in the scenario. In this particular example, all KRIs are assumed to be standardized.

The following DATA step uses the scenario in the Work.SingleScenario data set to simulate 10,000 replications of the number of loss events that you might observe for each business line and writes the simulated counts to the NumLoss variable of the Work.LossCounts1 data set:

```
/* Simulate multiple replications of the number of loss events that
   you can expect in the scenario being analyzed */
data lossCounts1 (keep=line corpKRI1 corpKRI2 cbKRI2 rbKRI1 rbKRI3 numloss);
   array cxR{3} corpKRI1 rbKRI1 rbKRI2;
   array cbetaR{4} _TEMPORARY_;
   array cxC{5} corpKRI1 corpKRI2 cbKRI1 cbKRI2 cbKRI3;
   array cbetaC{6} _TEMPORARY_;

   retain theta;
   if _n_ = 1 then do;
      call streaminit(5151);
      * read count model estimates *;
      set countEstEx2NB2(where=(line='CommercialBanking' and _type_='PARM'));
      cbetaC(1) = Intercept;
      do i=1 to dim(cxC);
         cbetaC(i+1) = cxC(i);
      end;
      alpha = _Alpha;
      theta = 1/alpha;

      set countEstEx2Poisson(where=(line='RetailBanking' and _type_='PARM'));
      cbetaR(1) = Intercept;
      do i=1 to dim(cxR);
         cbetaR(i+1) = cxR(i);
      end;
   end;

   set singleScenario;
   do iline=1 to 2;
      if (iline=1) then line = 'CommercialBanking';
      else line = 'RetailBanking';
      do repid=1 to 10000;
         * draw from count distribution *;
         if (iline=1) then do;
            xbeta = cbetaC(1);
            do i=1 to dim(cxC);
               xbeta = xbeta + cxC(i) * cbetaC(i+1);
            end;
            Mu = exp(xbeta);
            p = theta/(Mu+theta);
            numloss = rand('NEGB',p,theta);
         end;
         else do;
            xbeta = cbetaR(1);
            do i=1 to dim(cxR);
               xbeta = xbeta + cxR(i) * cbetaR(i+1);
            end;
            numloss = rand('POISSON', exp(xbeta));
         end;
         output;
      end;
   end;
run;
```

The Work.LossCounts1 data set contains the NumLoss variable in addition to the KRIs that are used by the severity regression model, which are needed by PROC HPCDM to simulate the aggregate loss.

By default, PROC HPCDM computes an aggregate loss distribution by using each of the severity models that you specify in the SEVERITYMODEL statement. However, you can restrict PROC HPCDM to use only a subset of the severity models for a given BY group by modifying the SEVERITYEST= data set to include only the estimates of the desired severity models in each BY group, as illustrated in the following DATA step:

```
/* Keep only the best severity model for each business line
   and set coefficients of unused regressors in each model to 0 */
data sevestEx2Best;
    set sevestEx2;
    if ((line = 'CommercialBanking' and _model_ = 'Logn')) then do;
        corpKRI2 = 0; rbKRI1 = 0; rbKRI3 = 0;
        output;
    end;
    else if ((line = 'RetailBanking' and _model_ = 'Gamma')) then do;
        corpKRI1 = 0; cbKRI2 = 0;
        output;
    end;
run;
```

Note that the preceding DATA step also sets the coefficients of the unused regressors in each model to 0. This is important because PROC HPCDM uses all the regressors that it detects from the SEVERITYEST= data set for each severity model.

Now, you are ready to estimate the aggregate loss distribution for each line of business by submitting the following PROC HPCDM step, in which you specify the EXTERNALCOUNTS statement to request that external counts in the NumLoss variable of the DATA= data set be used for simulation of the aggregate loss:

```
/* Estimate the distribution of the aggregate loss for both
   lines of business by using the externally simulated counts */
proc hpcdm data=lossCounts1 seed=13579 print=all
           severityest=sevestEx2Best;
   by line;
   externalcounts count=numloss;
   severitymodel logn gamma;
run;
```

Each observation in the Work.LossCounts1 data set represents one replication of the external counts simulation process. For each such replication, the preceding PROC HPCDM step makes as many severity draws from the severity distribution as the value of the NumLoss variable and adds the severity values from those draws to compute one sample point of the aggregate loss. The severity distribution that is used for making the severity draws has a scale parameter value that is decided by the KRI values in the given observation and the regression parameter values that are read from the Work.SevEstEx2Best data set.

The summary statistics and percentiles of the aggregate loss distribution for the commercial banking business, which uses the lognormal severity model, are shown in Output 18.2.1. The "Input Data Summary" table indicates that each of the 10,000 observations in the BY group is treated as one replication and that there are a total of 19,028 loss events produced by all the replications together. For the scenario in the Work.SingleScenario data set, you can expect the commercial banking business to incur an average aggregate loss of 653 units, as shown in the "Sample Summary Statistics" table, and the chance that the loss will exceed 4,728 units is 0.5%, as shown in the "Sample Percentiles" table.

Output 18.2.1 Aggregate Loss Summary for Commercial Banking Business

The HPCDM Procedure

line=CommercialBanking

Input Data Summary	
Name	WORK.LOSSCOUNTS1
Observations	10000
Valid Observations	10000
Replications	10000
Total Count	19028

line=CommercialBanking

Sample Summary Statistics			
Mean	653.06881	**Median**	362.81582
Standard Deviation	865.23039	**Interquartile Range**	863.64757
Variance	748623.6	**Minimum**	0
Skewness	2.92720	**Maximum**	13391.9
Kurtosis	15.94551	**Sample Size**	10000

line=CommercialBanking

Sample Percentiles	
Percentile	**Value**
0	0
1	0
5	0
25	56.32294
50	362.81582
75	919.97051
95	2309.3
99	3910.0
99.5	4727.7
Percentile Method = 5	

For the retail banking business, which uses the gamma severity model, the "Sample Percentiles" table in Output 18.2.2 indicates that the median operational loss of that business is about 71 units and the chance that the loss will exceed 380 units is about 1%.

Output 18.2.2 Aggregate Loss Percentiles for Retail Banking Business

line=RetailBanking

Sample Percentiles	
Percentile	**Value**
0	0
1	0
5	0
25	0
50	71.23738
75	141.95906
95	272.95890
99	379.77242
99.5	433.09783
Percentile Method = 5	

When you conduct the simulation and estimation for a scenario that contains only one observation, you assume that the operating environment does not change over the period of time that is being analyzed. That assumption might be valid for shorter durations and stable business environments, but often the operating environments change, especially if you are estimating the aggregate loss over a longer period of time. So you might want to include in your scenario all the possible operating environments that you expect to see during the analysis time period. Each environment is characterized by its own set of KRI values. For example, the operating conditions might change from quarter to quarter, and you might want to estimate the aggregate loss distribution for the entire year. You start the estimation process for such scenarios by creating a scenario data set. The following DATA step creates the Work.MultiConditionScenario data set, which consists of four operating environments, one for each quarter:

```
/* Generate a scenario data set for multiple operating conditions */
data multiConditionScenario (keep=opEnvId corpKRI1 corpKRI2
      cbKRI1 cbKRI2 cbKRI3 rbKRI1 rbKRI2 rbKRI3);
   array x{8} corpKRI1 corpKRI2 cbKRI1 cbKRI2 cbKRI3 rbKRI1 rbKRI2 rbKRI3;
   call streaminit(5151);
   do opEnvId=1 to 4;
      do i=1 to dim(x);
         x(i) = rand('NORMAL');
      end;
      output;
   end;
run;
```

All four observations of the Work.MultiConditionScenario data set together form one scenario. When simulating the external counts for such multi-entity scenarios, one replication consists of the possible number of loss events that can occur as a result of each of the four operating environments. In any given replication, some operating environments might not produce any loss event or all four operating environments might produce some loss events. Assume that you use a DATA step to create the Work.LossCounts2 data set that contains, for each business line, 10,000 replications of the loss counts and that you identify each replication by using the RepId variable. You can find the DATA step code to prepare the Work.LossCounts2 data set in the PROC HPCDM sample program *hcdmex02.sas*.

Output 18.2.3 shows some observations of the Work.LossCounts2 data set for each business line. For the first replication (RepId=1) of the commercial banking business, only operating environments 3 and 4 incur loss events, whereas the other environments incur no loss events. For the second replication (RepId=2), all operating environments incur at least one loss event. For the first replication (RepId=1) of the retail banking business, operating environments 2, 3, and 4 incur two, one, and three loss events, respectively.

Output 18.2.3 Snapshot of the External Counts Data with Replication Identifier

line	opEnvId	corpKRI1	corpKRI2	cbKRI2	rbKRI1	rbKRI3	repid	numloss
CommercialBanking	1	0.45224	0.40661	-0.33680	-1.08692	-2.20557	1	0
CommercialBanking	2	-0.03799	0.98670	-0.03752	1.94589	1.22456	1	0
CommercialBanking	3	-0.29120	-0.45239	0.98855	-0.37208	-1.51534	1	3
CommercialBanking	4	0.87499	-0.67812	-0.04839	-1.44881	0.78221	1	1
CommercialBanking	1	0.45224	0.40661	-0.33680	-1.08692	-2.20557	2	2
CommercialBanking	2	-0.03799	0.98670	-0.03752	1.94589	1.22456	2	5
CommercialBanking	3	-0.29120	-0.45239	0.98855	-0.37208	-1.51534	2	12
CommercialBanking	4	0.87499	-0.67812	-0.04839	-1.44881	0.78221	2	12
RetailBanking	1	0.45224	0.40661	-0.33680	-1.08692	-2.20557	1	0
RetailBanking	2	-0.03799	0.98670	-0.03752	1.94589	1.22456	1	2
RetailBanking	3	-0.29120	-0.45239	0.98855	-0.37208	-1.51534	1	1
RetailBanking	4	0.87499	-0.67812	-0.04839	-1.44881	0.78221	1	3
RetailBanking	1	0.45224	0.40661	-0.33680	-1.08692	-2.20557	2	2
RetailBanking	2	-0.03799	0.98670	-0.03752	1.94589	1.22456	2	2
RetailBanking	3	-0.29120	-0.45239	0.98855	-0.37208	-1.51534	2	0
RetailBanking	4	0.87499	-0.67812	-0.04839	-1.44881	0.78221	2	1

You can now use this simulated count data to estimate the distribution of the aggregate loss that is incurred in all four operating environments by submitting the following PROC HPCDM step, in which you specify the replication identifier variable RepId in the ID= option of the EXTERNALCOUNTS statement:

```
/* Estimate the distribution of the aggregate loss for both
   lines of business by using the externally simulated counts
   for the multiple operating environments */
proc hpcdm data=lossCounts2 seed=13579 print=all
           severityest=sevestEx2Best plots=density;
   by line;
   distby repid;
   externalcounts count=numloss id=repid;
   severitymodel logn gamma;
run;
```

Note that when you specify the ID= variable in the EXTERNALCOUNTS statement, you must also specify that variable in the DISTBY statement. Within each BY group, for each value of the RepId variable, one point of the aggregate loss sample is simulated by using the process that is described in the section "Simulation with External Counts" on page 1023.

The summary statistics and percentiles of the distribution of the aggregate loss, which is the aggregate of the losses across all four operating environments, are shown in Output 18.2.4 for the commercial banking business. The "Input Data Summary" table indicates that there are 10,000 replications in the BY group and that a total of 145,721 loss events are generated across all replications. The "Sample Percentiles" table indicates that you can expect a median aggregate loss of 4,460 units and a worst-case loss, as defined by the 99.5th percentile, of 16,304 units from the commercial banking business when you combine losses that result from all four operating environments.

Output 18.2.4 Aggregate Loss Summary for the Commercial Banking Business in Multiple Operating Environments

The HPCDM Procedure

line=CommercialBanking

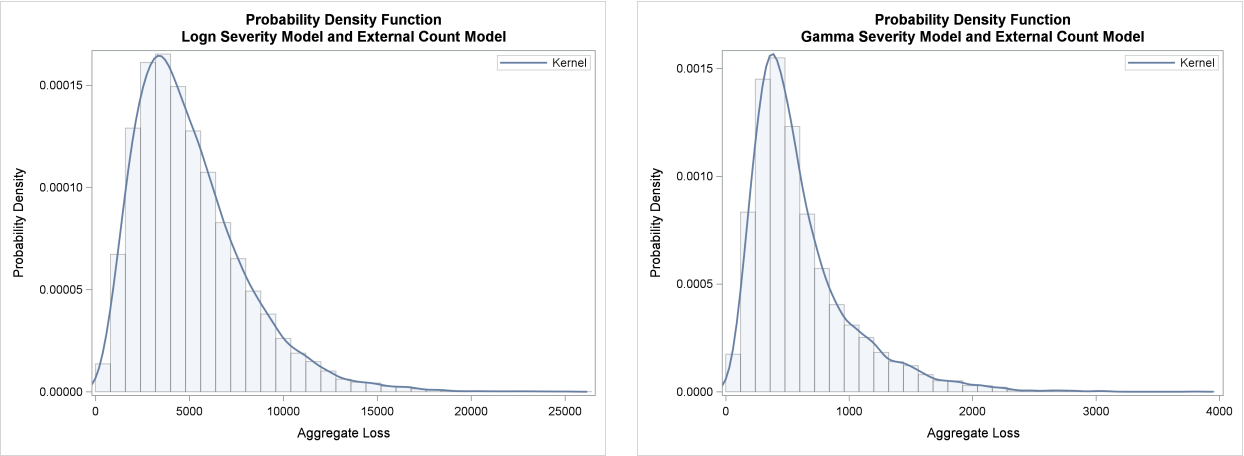
Input Data Summary	
Name	WORK.LOSSCOUNTS2
Observations	40000
Valid Observations	40000
Replications	10000
Total Count	145721

line=CommercialBanking

Sample Percentiles	
Percentile	**Value**
1	763.82070
5	1427.8
25	2933.7
50	4459.5
75	6539.3
95	10649.2
99	14606.4
99.5	16303.7
Percentile Method = 5	

The probability density functions of the aggregate loss for the commercial and retail banking businesses are shown in Output 18.2.5. In addition to the difference in scales of the losses in the two businesses, you can see that the aggregate loss that is incurred in the commercial banking business has a heavier right tail than the aggregate loss that is incurred in the retail banking business.

Output 18.2.5 Density Plots of the Aggregate Losses for Commercial Banking (left) and Retail Banking (right) Businesses



References

Fisher, R. A. (1973). *Statistical Methods for Research Workers*. 14th ed. New York: Hafner Publishing.

Klugman, S. A., Panjer, H. H., and Willmot, G. E. (1998). *Loss Models: From Data to Decisions*. New York: John Wiley & Sons.

Chapter 19
The HPCOPULA Procedure

Contents

Overview: HPCOPULA Procedure **1057**
PROC HPCOPULA Features 1058
Getting Started: HPCOPULA Procedure **1058**
Syntax: HPCOPULA Procedure **1059**
Functional Summary 1059
PROC HPCOPULA Statement 1059
DEFINE Statement 1060
SIMULATE Statement 1060
PERFORMANCE Statement 1061
VAR Statement 1061
Details: HPCOPULA Procedure **1062**
Sklar's Theorem 1062
Dependence Measures 1062
Normal Copula 1063
Student's *t* copula 1064
Archimedean Copulas 1064
OUTUNIFORM= Data Sets 1066
Examples: HPCOPULA Procedure **1066**
Example 19.1: Simulating Default Times 1066
References **1069**

Overview: HPCOPULA Procedure

The HPCOPULA procedure is a high-performance version of the SAS/ETS COPULA procedure, which simulates data from a specified copula. Unlike the COPULA procedure, which can be run only on an individual workstation, the HPCOPULA procedure takes advantage of a computing environment in which the optimization task can be distributed to one or more nodes. In addition, each node can use one or more threads to perform the optimization on its subset of the data. When several nodes are used and each node uses several threads to carry out its part of the work, the result is a highly parallel computation that provides a dramatic gain in performance.

You can use the HPCOPULA procedure to read and write data in distributed form and perform analyses either in single-machine mode or in distributed mode. For more information about the execution mode of SAS High-Performance Analytics procedures, see the section "Processing Modes" on page 62 in Chapter 3, "Shared Concepts and Topics."

The HPCOPULA procedure is specifically designed to operate in the high-performance distributed environment. By default, PROC HPCOPULA performs computations in multiple threads.

PROC HPCOPULA Features

The HPCOPULA procedure enables you to simulate a specified copula, and it supports the following types of copulas:

- normal copula
- *t* copula
- Archimedean copulas:
  - Clayton copula
  - Frank copula
  - Gumbel copula

Getting Started: HPCOPULA Procedure

This example illustrates the use of PROC HPCOPULA. The data are daily returns on several major stocks. The main purpose of this example is to simulate from the joint distribution of stock returns a new sample of a specified size, provided that the parameter estimates of the copula model that is used are available.

In the following statements, the DEFINE statement specifies a normal copula named COP, and the CORR= option specifies that the data set Estimates be used as the source for the model parameters. The NDRAWS=1000000 option in the SIMULATE statement generates one million observations from the normal copula. The OUTUNIFORM= option specifies the name of the SAS data set to contain the simulated sample that has uniform marginal distributions. The PERFORMANCE statement requests that the analytic computations use two nodes in the distributed computing environment and two threads in each node. Note that this syntax does not require the DATA= option.

```
/* Copula simulation of uniforms */
proc hpcopula;
   var ret_ibm ret_msft ret_bp ret_ko ret_duk;
   define cop normal (corr = estimates);
   simulate cop / ndraws     = 1000000
                  outuniform = simulated_uniforms;
   PERFORMANCE nodes=2 nthreads=2 details;
run;
```

The simulated data are contained in the new SAS data set, Simulated_Uniforms.

Syntax: HPCOPULA Procedure

The following statements are available in the HPCOPULA procedure:

PROC HPCOPULA *options* ;
  VAR *variables* ;
  DEFINE *name copula-type* < *(parameter-value-options . . .)* > ;
  SIMULATE < *copula-name-list* > */ options* ;

Functional Summary

Table 19.1 summarizes the statements and options that the HPCOPULA procedure uses.

Table 19.1 PROC HPCOPULA Functional Summary

Description	Statement	Option
Data Set Options		
Specifies the input data set that contains the correlation matrix for elliptical copulas	DEFINE	CORR=
Declaring the Role of Variables		
Specifies the names of the variables to use in copula fitting or in simulation	VAR	
Copula Simulation Options		
Specifies the random sample size	SIMULATE	NDRAWS=
Specifies the random number generator seed	SIMULATE	SEED=
Output Control Options		
Specifies the output data set to contain the random samples from the simulation with uniform marginal distribution	SIMULATE	OUTUNIFORM=

PROC HPCOPULA Statement

PROC HPCOPULA ;

The PROC HPCOPULA statement invokes the HPCOPULA procedure.

DEFINE Statement

DEFINE *name copula-type* < *(parameter-value-options ...)* > **;**

The DEFINE statement specifies the relevant information about the copula that is used for the simulation. You can specify the following arguments:

name specifies the name of the copula definition. You can be use this *name* later in the SIMULATE statement.

copula-type specifies the type of copula. You must specify one of the following copula types, which are described in the section "Details: HPCOPULA Procedure" on page 1062:

NORMAL fits the normal copula.

T fits the *t* copula.

CLAYTON fits the Clayton copula.

FRANK fits the Frank copula.

GUMBEL fits the Gumbel copula.

parameter-value-options

specify the input parameters that are used to simulate the specified copula. These options must be appropriate for the type of copula specified. You can specify the following *parameter-value-options*:

CORR=*SAS-data-set*

specifies the data set that contains the correlation matrix to use for elliptical copulas. If the correlation matrix is valid but its elements are not submitted in order, then you must provide the variable names in the first column of the matrix, and these names must match the variable names in the VAR statement. See Output 19.1.1 for an example of a correlation matrix input in this form. If the correlation matrix elements are submitted in order, the first column of variable names is not required. You can use this option for normal and *t* copulas.

DF=*value*

specifies the degrees of freedom. You can use this option for *t* copulas.

THETA=*value*

specifies the parameter value for the Archimedean copulas.

The DEFINE statement is used with the SIMULATE statement.

SIMULATE Statement

SIMULATE < *copula-name-list* >*/options* **;**

The SIMULATE statement simulates data from a specified copula model. The copula name specification is the name of a defined copula as specified by *name* in the DEFINE statement.

NDRAWS=*integer*
specifies the number of draws to generate for this simulation. By default, NDRAWS=100.

OUTUNIFORM=*SAS-data-set*
specifies the output data set to contain the result of the simulation in uniform margins. You can use this option when MARGINALS=UNIFORM or MARGINALS=EMPIRICAL. If MARGINALS=EMPIRICAL, then this option enables you to obtain the samples that are simulated from the joint distribution specified by the copula, where all marginal distributions are uniform. The data are not created if you do not specify this option.

SEED=*integer*
specifies the seed for generating random numbers for the simulation. If you do not provide the seed, a random number is used as the seed.

PERFORMANCE Statement

PERFORMANCE < *performance-options* > **;**

The PERFORMANCE statement specifies *performance-options* to control the multithreaded and distributed computing environment and requests detailed performance results of the HPCOPULA procedure. You can also use the PERFORMANCE statement to control whether the HPCOPULA procedure executes in SMP or MPP mode. You can specify the following *performance-options*:

DETAILS
requests a table that shows a timing breakdown of the PROC HPCOPULA steps.

NODES=*n*
specifies the number of nodes in the distributed computing environment, provided that the data are not processed alongside the database.

NTHREADS=*n*
specifies the number of threads for analytic computations and overrides the SAS system option THREADS | NOTHREADS. If you do not specify the NTHREADS= option, PROC HPCOPULA creates one thread per CPU for the analytic computations.

For more information about the PERFORMANCE statement, see the section "PERFORMANCE Statement" on page 87 in Chapter 3, "Shared Concepts and Topics."

VAR Statement

VAR *variables* **;**

The VAR statement specifies the variable names in the input data set that is specified by the DATA= option in the PROC HPCOPULA statement. The subset of variables in the data set is used for the copula models in the FIT statement. If there is no input data set, the VAR statement creates the list of variable names for the SIMULATE statement.

Details: HPCOPULA Procedure

Sklar's Theorem

The copula models are tools for studying the dependence structure of multivariate distributions. The usual joint distribution function contains the information both about the marginal behavior of the individual random variables and about the dependence structure between the variables. The copula is introduced to decouple the marginal properties of the random variables and the dependence structures. An m-dimensional *copula* is a joint distribution function on $[0, 1]^m$, where all marginal distributions are standard uniform. The common notation for a copula is $C(u_1, \ldots, u_m)$.

The Sklar (1959) theorem shows the importance of copulas in modeling multivariate distributions. The first part of the theorem states that a copula can be derived from any joint distribution functions, and the second part asserts the opposite: that any copula can be combined with any set of marginal distributions to result in a multivariate distribution function. The theorem follows:

- Let F be a joint distribution function, and let $F_j, j = 1, \ldots, m$, be the marginal distributions. Then there exists a copula $C : [0, 1]^m \rightarrow [0, 1]$ such that

  $$F(x_1, \ldots, x_m) = C(F_1(x_1), \ldots, F_m(x_m))$$

  for all $x_1, \ldots, x_m$ in $[-\infty, \infty]$. Moreover, if the margins are continuous, then C is unique; otherwise C is uniquely determined on $\mathrm{Ran}F_1 \times \ldots \times \mathrm{Ran}F_m$, where $\mathrm{Ran}F_j = F_j([-\infty, \infty])$ is the range of F_j.

- The converse is also true. That is, if C is a copula and $F_1, \ldots, F_m$ are univariate distribution functions, then the multivariate function that is defined in the preceding equation is a joint distribution function with marginal distributions $F_j, j = 1, \ldots, m$.

Dependence Measures

There are three basic types of dependence measures: linear correlation, rank correlation, and tail dependence. Linear correlation is given by

$$\rho \equiv \mathrm{corr}(X, Y) = \frac{\mathrm{cov}(X, Y)}{\sqrt{\mathrm{var}(X)}\sqrt{\mathrm{var}(Y)}}$$

The linear correlation coefficient contains very limited information about the joint properties of the variables. A well-known property is that zero correlation does not imply independence, whereas independence implies zero correlation. In addition, there are distinct bivariate distributions that have the same marginal distribution and the same correlation coefficient. These results suggest that caution must be used in interpreting the linear correlation.

Another statistical measure of dependence is rank correlation, which is nonparametric. For example, Kendall's tau is the covariance between the sign statistics $X_1 - \tilde{X}_1$ and $X_2 - \tilde{X}_2$, where $(\tilde{X}_1, \tilde{X}_2)$ is an independent copy of (X_1, X_2):

$$\rho_\tau \equiv E[\mathrm{sign}(X_1 - \tilde{X}_1)(X_2 - \tilde{X}_2)]$$

The sign function (sometimes written as sgn) is defined as

$$\text{sign}(x) = \begin{cases} -1 & \text{if} x \leq 0 \\ 0 & \text{if} x = 0 \\ 1 & \text{if} x \geq 0 \end{cases}$$

Spearman's rho is the correlation between the transformed random variables:

$$\rho_S(X_1, X_2) \equiv \rho(F_1(X_1), F_2(X_2))$$

The variables are transformed by their distribution functions so that the transformed variables are uniformly distributed on $[0, 1]$. The rank correlations depend only on the copula of the random variables and are indifferent to the marginal distributions. Like linear correlation, rank correlation has its limitations. In particular, different copulas result in the same rank correlation.

A third measure, tail dependence, focuses on only part of the joint properties between the variables. Tail dependence measures the dependence when both variables have extreme values. Formally, they can be defined as the conditional probabilities of quantile exceedances. There are two types of tail dependence:

- Upper tail dependence is defined as

  $$\lambda_u(X_1, X_2) \equiv \lim_{q->1^-} P(X_2 > F_2^{-1}(q) | X_1 > F_1^{-1}(q))$$

  when the limit exists and $\lambda_u \in [0, 1]$. Here F_j^{-1} is the quantile function (that is, the inverse of the CDF).
- Lower tail dependence is defined symmetrically.

Normal Copula

Let $u_j \sim U(0, 1)$ for $j = 1, \ldots, m$, where $U(0, 1)$ represents the uniform distribution on the $[0, 1]$ interval. Let Σ be the correlation matrix, where $m(m-1)/2$ parameters satisfy the positive semidefiniteness constraint. The normal copula can be written as

$$C_\Sigma(u_1, u_2, \ldots u_m) = \mathbf{\Phi}_\Sigma\Big(\Phi^{-1}(u_1), \ldots \Phi^{-1}(u_m)\Big)$$

where Φ is the distribution function of a standard normal random variable and $\mathbf{\Phi}_\Sigma$ is the m-variate standard normal distribution with mean vector 0 and covariance matrix Σ. That is, the distribution $\mathbf{\Phi}_\Sigma$ is $N_m(0, \Sigma)$.

Simulation

For the normal copula, the input of the simulation is the correlation matrix Σ. The normal copula can be simulated by the following steps, in which $\boldsymbol{U} = (U_1, \ldots, U_m)$ denotes one random draw from the copula:

1. Generate a multivariate normal vector $\boldsymbol{Z} \sim N(0, \Sigma)$, where Σ is an m-dimensional correlation matrix.
2. Transform the vector $\boldsymbol{Z}$ into $\boldsymbol{U} = (\Phi(Z_1), \ldots, \Phi(Z_m))^T$, where Φ is the distribution function of univariate standard normal.

The first step can be achieved by Cholesky decomposition of the correlation matrix $\Sigma = LL^T$, where L is a lower triangular matrix with positive elements on the diagonal. If $\tilde{\boldsymbol{Z}} \sim N(0, I)$, then $L\tilde{\boldsymbol{Z}} \sim N(0, \Sigma)$.

Student's *t* copula

Let $\Theta = \{(\nu, \Sigma) : \nu \in (1, \infty), \Sigma \in \mathbb{R}^{m \times m}\}$, and let t_ν be a univariate t distribution with ν degrees of freedom.

The Student's t copula can be written as

$$C_\Theta(u_1, u_2, \ldots u_m) = \boldsymbol{t}_{\nu,\Sigma}\left(t_\nu^{-1}(u_1), t_\nu^{-1}(u_2), \ldots, t_\nu^{-1}(u_m)\right)$$

where $\boldsymbol{t}_{\nu,\Sigma}$ is the multivariate Student's t distribution that has a correlation matrix Σ with ν degrees of freedom.

Simulation

The input parameters for the simulation are (ν, Σ). The t copula can be simulated by the following steps:

1. Generate a multivariate vector $\boldsymbol{X} \sim t_m(\nu, 0, \Sigma)$ that follows the centered t distribution with ν degrees of freedom and correlation matrix Σ.
2. Transform the vector $\boldsymbol{X}$ into $\boldsymbol{U} = (t_\nu(X_1), \ldots, t_\nu(X_m))^T$, where t_ν is the distribution function of univariate t distribution with ν degrees of freedom.

To simulate centered multivariate t random variables, you can use the property that $\boldsymbol{X} \sim t_m(\nu, 0, \Sigma)$ if $\boldsymbol{X} = \sqrt{\nu/s}\boldsymbol{Z}$, where $\boldsymbol{Z} \sim N(0, \Sigma)$ and the univariate random variable $s \sim \chi^2_\nu$.

Archimedean Copulas

Overview of Archimedean Copulas

Let function $\phi : [0, 1] \to [0, \infty)$ be a strict Archimedean copula generator function, and suppose that its inverse ϕ^{-1} is completely monotonic on $[0, \infty)$. A strict generator is a decreasing function $\phi : [0, 1] \to [0, \infty)$ that satisfies $\phi(0) = \infty$ and $\phi(1) = 0$. A decreasing function $f(t) : [a, b] \to (-\infty, \infty)$ is completely monotonic if it satisfies

$$(-1)^k \frac{d^k}{dt^k} f(t) \geq 0, k \in \mathbb{N}, t \in (a, b)$$

An Archimedean copula is defined as follows:

$$C(u_1, u_2, \ldots, u_m) = \phi^{-1}\left(\phi(u_1) + \cdots + \phi(u_m)\right)$$

The Archimedean copulas available in the HPCOPULA procedure are the Clayton copula, the Frank copula, and the Gumbel copula.

Clayton Copula

Let the generator function $\phi(u) = \theta^{-1}\left(u^{-\theta} - 1\right)$. A Clayton copula is defined as

$$C_\theta(u_1, u_2, \ldots, u_m) = \left[\sum_{i=1}^{m} u_i^{-\theta} - m + 1\right]^{-1/\theta}$$

where $\theta > 0$.

Frank Copula

Let the generator function be

$$\phi(u) = -\log\left[\frac{\exp(-\theta u) - 1}{\exp(-\theta) - 1}\right]$$

A Frank copula is defined as

$$C_\theta(u_1, u_2, \ldots, u_m) = \frac{1}{\theta}\log\left\{1 + \frac{\prod_{i=1}^{m}[\exp(-\theta u_i) - 1]}{[\exp(-\theta) - 1]^{m-1}}\right\}$$

where $\theta \in (-\infty, \infty)\backslash\{0\}$ for $m = 2$ and $\theta > 0$ for $m \geq 3$.

Gumbel Copula

Let the generator function $\phi(u) = (-\log u)^\theta$. A Gumbel copula is defined as

$$C_\theta(u_1, u_2, \ldots, u_m) = \exp\left\{-\left[\sum_{i=1}^{m}(-\log u_i)^\theta\right]^{1/\theta}\right\}$$

where $\theta > 1$.

Simulation

Suppose that the generator of the Archimedean copula is ϕ. Then the simulation method that uses a Laplace-Stieltjes transformation of the distribution function is given by Marshall and Olkin (1988), where $\tilde{F}(t) = \int_0^\infty e^{-tx} dF(x)$:

1. Generate a random variable V that has the distribution function F such that $\tilde{F}(t) = \phi^{-1}(t)$.
2. Draw samples from the independent uniform random variables $X_1, \ldots, X_m$.
3. Return $\boldsymbol{U} = (\tilde{F}(-\log(X_1)/V), \ldots \tilde{F}(-\log(X_m)/V))^T$.

The Laplace-Stieltjes transformations are as follows:

- For the Clayton copula, $\tilde{F} = (1 + t)^{-1/\theta}$, and the distribution function F is associated with a gamma random variable that has a shape parameter of θ^{-1} and a scale parameter of 1.
- For the Gumbel copula, $\tilde{F} = \exp(-t^{1/\theta})$, and F is the distribution function of the stable variable $\mathrm{St}(\theta^{-1}, 1, \gamma, 0)$, where $\gamma = [\cos(\pi/(2\theta))]^\theta$.

- For the Frank copula where $\theta > 0$, $\tilde{F} = -\log\{1 - \exp(-t)[1 - \exp(-\theta)]\}/\theta$, and F is a discrete probability function $P(V = k) = (1 - \exp(-\theta))^k/(k\theta)$. This probability function is related to a logarithmic random variable that has a parameter value of $1 - e^{-\theta}$.

For more information about simulating a random variable from a stable distribution, see Theorem 1.19 in Nolan (2010). For more information about simulating a random variable from a logarithmic series, see Chapter 10.5 in Devroye (1986).

For a Frank copula where $m = 2$ and $\theta < 0$, the simulation can be done through conditional distributions as follows:

1 Draw independent v_1, v_2 from a uniform distribution.

2 Let $u_1 = v_1$.

3 Let $u_2 = -\frac{1}{\theta}\log\left(1 + \frac{v_2(1-e^{-\theta})}{v_2(e^{-\theta v_1}-1)-e^{-\theta v_1}}\right)$.

OUTUNIFORM= Data Sets

The number of columns and the names of columns in OUTUNIFORM= data sets match the number and names of the *variables* in the VAR statement.

Examples: HPCOPULA Procedure

Example 19.1: Simulating Default Times

Suppose the correlation structure that is required for a normal copula function is already known. For example, the correlation structure can be estimated from the historical data on default times in some industries, but this estimation is not within the scope of this example. The correlation structure is saved in a SAS data set called Inparm. The following statements and their output in Output 19.1.1 show that the correlation parameter is set at 0.8:

```
proc print data = inparm;
run;
```

Output 19.1.1 Copula Correlation Matrix

Obs	Y1	Y2
1	1.0	0.8
2	0.8	1.0

The following statements use PROC HPCOPULA to simulate the data:

```
option set=GRIDHOST="&GRIDHOST";
option set=GRIDINSTALLLOC="&GRIDINSTALLLOC";


/* simulate the data from bivariate normal copula */
proc hpcopula;
   var Y1-Y2;
   define cop normal (corr=inparm);
   simulate cop /
            ndraws     = 1000000
            seed       = 1234
            outuniform = normal_unifdata;
   PERFORMANCE nodes=4 nthreads=4 details
               host="&GRIDHOST" install="&GRIDINSTALLLOC";
run;
```

The VAR statement specifies the list of variables that contains the simulated data. The DEFINE statement assigns the name COP and specifies a normal copula that reads the correlation matrix from the Inparm data set. The SIMULATE statement refers to the COP label that is defined in the VAR statement and specifies several options: the NDRAWS= option specifies a sample size, the SEED= option specifies 1234 as the random number generator seed, and the OUTUNIFORM=NORMAL_UNIFDATA option names the output data set to contain the result of simulation in uniforms. The PERFORMANCE statement requests that the analytic computations be performed on four nodes in the distributed computing environment and four threads on each node. Output 19.1.2 shows the run time of this particular simulation experiment.

Output 19.1.2 Run-Time Performance

Performance Information	
Host Node	<< your grid host >>
Install Location	<< your grid install location >>
Execution Mode	Distributed
Number of Compute Nodes	4
Number of Threads per Node	4

Procedure Task Timing		
Task	**Seconds**	**Percent**
Communication to Client	0.00	0.00%
Simulation of Model	0.03	0.07%
Writing of Output Data	45.20	99.93%

The following DATA step transforms the variables from zero-one uniformly distributed to nonnegative exponentially distributed with parameter 0.5 and adds three indicator variables to the data set: SURVIVE1 and SURVIVE2 are equal to 1 if company 1 or company 2, respectively, has remained in business for more than three years, and SURVIVE is equal to 1 if both companies survived the same period together.

```
/* default time has exponential marginal distribution with parameter 0.5 */
data default;
   set normal_unifdata;
   array arr{2} Y1-Y2;
   array time{2} time1-time2;
   array surv{2} survive1-survive2;
   lambda = 0.5;
   do i=1 to 2;
      time[i] = -log(1-arr[i])/lambda;
      surv[i] = 0;
      if (time[i] >3) then surv[i]=1;
   end;
   survive = 0;
   if (time1 >3) && (time2 >3) then survive = 1;
run;
```

The first analysis step is to look at correlations between survival times of the two companies. You can perform this step by using the CORR procedure as follows:

```
proc corr data = default pearson kendall;
   var time1 time2;
run;
```

Output 19.1.3 shows the output of this code. The output contains some descriptive statistics and two measures of correlation: Pearson and Kendall. Both measures indicate high and statistically significant dependence between the life spans of the two companies.

Output 19.1.3 Default Time Descriptive Statistics and Correlations

The CORR Procedure

2 Variables:	time1	time2

Simple Statistics

Variable	N	Mean	Std Dev	Median	Minimum	Maximum
time1	1000000	2.00042	1.99724	1.38664	1.78961E-6	28.39277
time2	1000000	2.00190	2.00064	1.38787	2.24931E-6	30.50949

Pearson Correlation Coefficients, N = 1000000
Prob > |r| under H0: Rho=0

	time1	time2
time1	1.00000	0.76950 <.0001
time2	0.76950 <.0001	1.00000

Kendall Tau b Correlation Coefficients, N = 1000000
Prob > |tau| under H0: Tau=0

	time1	time2
time1	1.00000	0.58998 <.0001
time2	0.58998 <.0001	1.00000

The second and final step is to empirically estimate the default probabilities of the two companies. This is done by using the FREQ procedure as follows:

```
proc freq data=default;
   table survive survive1-survive2;
run;
```

The results are shown in Output 19.1.4.

Output 19.1.4 Probabilities of Default

The FREQ Procedure

survive	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	852314	85.23	852314	85.23
1	147686	14.77	1000000	100.00

survive1	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	776565	77.66	776565	77.66
1	223435	22.34	1000000	100.00

survive2	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	776382	77.64	776382	77.64
1	223618	22.36	1000000	100.00

Output 19.1.4 shows that the empirical default probabilities are 78% and 78%. Assuming that these companies are independent yields the probability estimate that both companies default during the period of three years as 0.75*0.78=0.59 (61%). Comparing this naive estimate with the much higher actual 85% joint default probability illustrates that neglecting the correlation between the two companies significantly underestimates the probability of default.

%clearpage

References

Devroye, L. (1986). *Non-uniform Random Variate Generation*. New York: Springer-Verlag. http://luc.devroye.org/rnbookindex.html.

Marshall, A. W., and Olkin, I. (1988). "Families of Multivariate Distributions." *Journal of the American Statistical Association* 83:834–841.

Nolan, J. P. (2010). *Stable Distributions: Models for Heavy Tailed Data*. Boston: Birkhäuser.

Sklar, A. (1959). "Fonctions de répartition à *n* dimensions et leurs marges." *Publications de l'Institut de Statistique de L'Université de Paris* 8:229–231.

Chapter 20
The HPCOUNTREG Procedure

Contents

Overview: HPCOUNTREG Procedure **1072**
PROC HPCOUNTREG Features 1072
Getting Started: HPCOUNTREG Procedure **1073**
Syntax: HPCOUNTREG Procedure **1076**
Functional Summary 1077
PROC HPCOUNTREG Statement 1078
BOUNDS Statement 1082
BY Statement 1082
CLASS Statement 1082
DISPMODEL Statement 1083
FREQ Statement 1083
INIT Statement 1084
MODEL Statement 1084
OUTPUT Statement 1085
PERFORMANCE Statement 1087
RESTRICT Statement 1088
TEST Statement 1088
WEIGHT Statement 1089
ZEROMODEL Statement 1090
Details: HPCOUNTREG Procedure **1091**
Missing Values 1091
Poisson Regression 1091
Conway-Maxwell-Poisson Regression 1092
Negative Binomial Regression 1096
Zero-Inflated Count Regression Overview 1098
Zero-Inflated Poisson Regression 1099
Zero-Inflated Conway-Maxwell-Poisson Regression 1100
Zero-Inflated Negative Binomial Regression 1102
Parameter Naming Conventions for the RESTRICT, TEST, BOUNDS, and INIT Statements 1103
Computational Resources 1107
Covariance Matrix Types 1107
Displayed Output 1108
OUTPUT OUT= Data Set 1109
OUTEST= Data Set 1109
ODS Table Names 1109

Examples: The HPCOUNTREG Procedure **1110**
Example 20.1: High-Performance Zero-Inflated Poisson Model 1110
References **1114**

Overview: HPCOUNTREG Procedure

The HPCOUNTREG procedure is a high-performance version of the COUNTREG procedure in SAS/ETS software. Like the COUNTREG procedure, the HPCOUNTREG procedure fits regression models in which the dependent variable takes on nonnegative integer or count values. Unlike the COUNTREG procedure, which can be run only on an individual workstation, the HPCOUNTREG procedure takes advantage of a computing environment that enables it to distribute the optimization task among one or more nodes. In addition, each node can use one or more threads to carry out the optimization on its subset of the data. When several nodes are employed, with each node using several threads to carry out its part of the work, the result is a highly parallel computation that provides a dramatic gain in performance.

The HPCOUNTREG procedure enables you to read and write data in distributed form and perform analyses in distributed mode and single-machine mode. For information about how to affect the execution mode of SAS high-performance analytical procedures, see the section "Processing Modes" on page 62 in Chapter 3, "Shared Concepts and Topics."

The HPCOUNTREG procedure is specifically designed to operate in the high-performance distributed environment. By default, PROC HPCOUNTREG performs computations in multiple threads.

PROC HPCOUNTREG Features

The HPCOUNTREG procedure estimates the parameters of a count regression model by maximum likelihood techniques.

The HPCOUNTREG procedure supports the following models for count data:

- Poisson regression
- Conway-Maxwell-Poisson regression
- negative binomial regression with quadratic and linear variance functions (Cameron and Trivedi 1986)
- zero-inflated Poisson (ZIP) model (Lambert 1992)
- zero-inflated Conway-Maxwell-Poisson (ZICMP) model
- zero-inflated negative binomial (ZINB) model
- fixed-effects and random-effects Poisson models for panel data
- fixed-effects and random-effects negative binomial models for panel data

The following list summarizes some basic features of the HPCOUNTREG procedure:

- can perform analysis on a massively parallel high-performance appliance
- reads input data in parallel and writes output data in parallel when the data source is the appliance database
- is highly multithreaded during all phases of analytic execution
- has model-building syntax that uses CLASS and effect-based MODEL statements familiar from SAS/ETS analytic procedures
- performs maximum likelihood estimation
- supports multiple link functions
- uses the WEIGHT statement for weighted analysis
- uses the FREQ statement for grouped analysis
- uses the OUTPUT statement to produce a data set that contains predicted probabilities and other observationwise statistics

Getting Started: HPCOUNTREG Procedure

Except for its ability to operate in the high-performance distributed environment, the HPCOUNTREG procedure is similar in use to other regression model procedures in the SAS System. For example, the following statements are used to estimate a Poisson regression model:

```
proc hpcountreg data=one ;
   model y = x / dist=poisson ;
run;
```

The response variable *y* is numeric and has nonnegative integer values.

This section illustrates two simple examples that use PROC HPCOUNTREG. The data are taken from Long (1997). This study examines how factors such as gender (fem), marital status (mar), number of young children (kid5), prestige of the graduate program (phd), and number of articles published by a scientist's mentor (ment) affect the number of articles (art) published by the scientist.

The first 10 observations are shown in Figure 20.1.

Figure 20.1 Article Count Data

Obs	art	fem	mar	kid5	phd	ment
1	3	0	1	2	1.38000	8.0000
2	0	0	0	0	4.29000	7.0000
3	4	0	0	0	3.85000	47.0000
4	1	0	1	1	3.59000	19.0000
5	1	0	1	0	1.81000	0.0000
6	1	0	1	1	3.59000	6.0000
7	0	0	1	1	2.12000	10.0000
8	0	0	1	0	4.29000	2.0000
9	3	0	1	2	2.58000	2.0000
10	3	0	1	1	1.80000	4.0000

The following SAS statements estimate the Poisson regression model. The model is executed in the distributed computing environment with two threads and four nodes.

```
/*-- Poisson Regression --*/
proc hpcountreg data=long97data;
   model art = fem mar kid5 phd ment / dist=poisson method=quanew;
   performance nthreads=2 nodes=4 details;
run;
```

The "Model Fit Summary" table that is shown in Figure 20.2 lists several details about the model. By default, the HPCOUNTREG procedure uses the Newton-Raphson optimization technique. The maximum log-likelihood value is shown, in addition to two information measures—Akaike's information criterion (AIC) and Schwarz's Bayesian information criterion (SBC)—which can be used to compare competing Poisson models. Smaller values of these criteria indicate better models.

Figure 20.2 Estimation Summary Table for a Poisson Regression

The HPCOUNTREG Procedure

Model Fit Summary	
Dependent Variable	art
Number of Observations	915
Data Set	WORK.LONG97DATA
Model	Poisson
Log Likelihood	-1651
Maximum Absolute Gradient	0.0002080
Number of Iterations	13
Optimization Method	Quasi-Newton
AIC	3314
SBC	3343

Figure 20.3 shows the parameter estimates of the model and their standard errors. All covariates are significant predictors of the number of articles, except for the prestige of the program (phd), which has a *p*-value of 0.6271.

Figure 20.3 Parameter Estimates of Poisson Regression

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	0.3046	0.1030	2.96	0.0031
fem	1	-0.2246	0.05461	-4.11	<.0001
mar	1	0.1552	0.06137	2.53	0.0114
kid5	1	-0.1849	0.04013	-4.61	<.0001
phd	1	0.01282	0.02640	0.49	0.6271
ment	1	0.02554	0.002006	12.73	<.0001

To allow for variance greater than the mean, you can fit the negative binomial model instead of the Poisson model by specifying the DIST=NEGBIN option, as shown in the following statements. Whereas the Poisson model requires that the conditional mean and conditional variance be equal, the negative binomial model allows for overdispersion, in which the conditional variance can exceed the conditional mean.

```
/*-- Negative Binomial Regression --*/
proc hpcountreg data=long97data;
   model art = fem mar kid5 phd ment / dist=negbin(p=2) method=quanew;
   performance nthreads=2 nodes=4 details;
run;
```

Figure 20.4 shows the fit summary and Figure 20.5 shows the parameter estimates.

Figure 20.4 Estimation Summary Table for a Negative Binomial Regression

The HPCOUNTREG Procedure

Model Fit Summary

Dependent Variable	art
Number of Observations	915
Data Set	WORK.LONG97DATA
Model	NegBin
Log Likelihood	-1561
Maximum Absolute Gradient	0.0000666
Number of Iterations	16
Optimization Method	Quasi-Newton
AIC	3136
SBC	3170

Figure 20.5 Parameter Estimates of Negative Binomial Regression

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	0.2561	0.1386	1.85	0.0645
fem	1	-0.2164	0.07267	-2.98	0.0029
mar	1	0.1505	0.08211	1.83	0.0668
kid5	1	-0.1764	0.05306	-3.32	0.0009
phd	1	0.01527	0.03604	0.42	0.6718
ment	1	0.02908	0.003470	8.38	<.0001
_Alpha	1	0.4416	0.05297	8.34	<.0001

The parameter estimate for _Alpha of *0.4416* is an estimate of the dispersion parameter in the negative binomial distribution. A t test for the hypothesis $H_0 : \alpha = 0$ is provided. It is highly significant, indicating overdispersion ($p < 0.0001$).

The null hypothesis $H_0 : \alpha = 0$ can be also tested against the alternative $\alpha > 0$ by using the likelihood ratio test, as described by Cameron and Trivedi (1998, pp. 45, 77–78). The likelihood ratio test statistic is equal to $-2(\mathcal{L}_P - \mathcal{L}_{NB}) = -2(-1651 + 1561) = 180$, which is highly significant, providing strong evidence of overdispersion.

Syntax: HPCOUNTREG Procedure

The following statements are available in the HPCOUNTREG procedure. Items within angle brackets (< >) or square brackets ([ ]) are optional.

PROC HPCOUNTREG <*options*> ;
BOUNDS *bound1* [*, bound2 . . .*] ;
BY *variables* ;
CLASS *variables* ;
DISPMODEL *dependent variable* ∼ < *dispersion-related regressors* > ;
FREQ *freq-variable* ;
INIT *initialization1* < *, initialization2 . . .* > ;
MODEL *dependent-variable* = *regressors* </ *options*> ;
OUTPUT <*output-options*> ;
PERFORMANCE *performance-options* ;
RESTRICT *restriction1* [*, restriction2 . . .*] ;
TEST *equation1* < *, equation2. . .* > / < *test-options* > ;
WEIGHT *variable* </ *option*> ;
ZEROMODEL *dependent-variable* ∼ *zero-inflated-regressors* </ *options*> ;

There can be only one MODEL statement. The ZEROMODEL statement, if used, must appear after the MODEL statement. If a FREQ or WEIGHT statement is specified more than once, the variable specified in the first instance is used.

Functional Summary

Table 20.1 summarizes the statements and options used with the HPCOUNTREG procedure.

Table 20.1 PROC HPCOUNTREG Functional Summary

Description	Statement	Option
Data Set Options		
Specifies the input data set	HPCOUNTREG	DATA=
Specifies the identification variable for panel data analysis	HPCOUNTREG	GROUPID=
Writes parameter estimates to an output data set	HPCOUNTREG	OUTEST=
Writes estimates to an output data set	OUTPUT	OUT=
Specifies BY-group processing	BY	
Specifies an optional frequency variable	FREQ	
Specifies an optional weight variable	WEIGHT	
Printing Control Options		
Prints the correlation matrix of the estimates	HPCOUNTREG	CORRB
Prints the covariance matrix of the estimates	HPCOUNTREG	COVB
Suppresses the normal printed output	HPCOUNTREG	NOPRINT
Requests all printing options	HPCOUNTREG	PRINTALL
Options to Control the Optimization Process		
Specifies maximum number of iterations allowed	HPCOUNTREG	MAXITER=
Selects the iterative minimization method to use	HPCOUNTREG	METHOD=
Specifies maximum number of iterations allowed	HPCOUNTREG	MAXITER=
Specifies maximum number of function calls	HPCOUNTREG	MAXFUNC=
Specifies the upper limit of CPU time in seconds	HPCOUNTREG	MAXTIME=
Specifies absolute function convergence criterion	HPCOUNTREG	ABSCONV=
Specifies absolute function convergence criterion	HPCOUNTREG	ABSFCONV=
Specifies absolute gradient convergence criterion	HPCOUNTREG	ABSGCONV=
Specifies relative function convergence criterion	HPCOUNTREG	FCONV=
Specifies relative gradient convergence criterion	HPCOUNTREG	GCONV=
Specifies absolute parameter convergence criterion	HPCOUNTREG	ABSXCONV=
Specifies matrix singularity criterion	HPCOUNTREG	SINGULAR=
Sets boundary restrictions on parameters	BOUNDS	
Sets initial values for parameters	INIT	
Sets linear restrictions on parameters	RESTRICT	
Model Estimation Options		
Specifies the dispersion variables	DISPMODEL	
Specifies the type of model	HPCOUNTREG	DIST=
Specifies the type of covariance matrix	HPCOUNTREG	COVEST=
Specifies the type of error components model for panel data	MODEL	ERRORCOMP=

Description	Statement	Option
Suppresses the intercept parameter	MODEL	NOINT
Specifies the offset variable	MODEL	OFFSET=
Specifies the parameterization for the Conway-Maxwell-Poisson (CMP) model	MODEL	PARAMETER=
Specifies the zero-inflated offset variable	ZEROMODEL	OFFSET=
Specifies the zero-inflated link function	ZEROMODEL	LINK=
Output Control Options		
Includes covariances in the OUTEST= data set	HPCOUNTREG	COVOUT
Includes correlations in the OUTEST= data set	HPCOUNTREG	CORROUT
Outputs SAS variables to the output data set	OUTPUT	COPYVAR=
Outputs the estimates of dispersion for the CMP model	OUTPUT	DISPERSION
Outputs the estimates of GDelta $= \mathbf{g}_i'\boldsymbol{\delta}$ for the CMP model	OUTPUT	GDELTA=
Outputs the estimates of λ for the CMP model	OUTPUT	LAMBDA=
Outputs the estimates of ν for the CMP model	OUTPUT	NU=
Outputs the estimates of μ for the CMP model	OUTPUT	MU=
Outputs the estimates of mode for the CMP model	OUTPUT	MODE=
Outputs the probability that the response variable will take the current value	OUTPUT	PROB=
Outputs probabilities for particular response values	OUTPUT	PROBCOUNT( )
Outputs expected value of response variable	OUTPUT	PRED=
Outputs the estimates of variance for the CMP model	OUTPUT	VARIANCE=
Outputs estimates of XBeta $= \mathbf{x}_i'\boldsymbol{\beta}$	OUTPUT	XBETA=
Outputs estimates of ZGamma $= \mathbf{z}_i'\boldsymbol{\gamma}$	OUTPUT	ZGAMMA=
Outputs probability of a zero value as a result of the zero-generating process	OUTPUT	PROBZERO=
Performance Options		
Requests a table that shows a timing breakdown	PERFORMANCE	DETAILS
Specifies the number of threads to use	PERFORMANCE	NTHREADS=
Specifies the number of nodes to use on the SAS appliance	PERFORMANCE	NODES=

PROC HPCOUNTREG Statement

PROC HPCOUNTREG <*options*> **;**

The following *options* can be used in the PROC HPCOUNTREG statement.

Input Data Set Options

DATA=*SAS-data-set*
specifies the input SAS data set. If the DATA= option is not specified, PROC HPCOUNTREG uses the most recently created SAS data set.

GROUPID=*variable*
specifies an identification variable when a panel data model is estimated. The identification variable is used as a cross-sectional ID variable.

Output Data Set Options

OUTEST=*SAS-data-set*
writes the parameter estimates to the specified output data set.

CORROUT
writes the correlation matrix for the parameter estimates to the OUTEST= data set. This option is valid only if the OUTEST= option is specified.

COVOUT
writes the covariance matrix for the parameter estimates to the OUTEST= data set. This option is valid only if the OUTEST= option is specified.

Printing Options

You can specify the following options in either the PROC HPCOUNTREG statement or the MODEL statement:

CORRB
prints the correlation matrix of the parameter estimates.

COVB
prints the covariance matrix of the parameter estimates.

NOPRINT
suppresses all printed output.

PRINTALL
requests all printing options.

Estimation Control Options

You can specify the following options in either the PROC HPCOUNTREG statement or the MODEL statement:

COVEST=HESSIAN | OP | QML
specifies the type of covariance matrix for the parameter estimates.

The default is COVEST=HESSIAN. You can specify the following values:

HESSIAN	specifies the covariance from the Hessian matrix.
OP	specifies the covariance from the outer product matrix.
QML	specifies the covariance from the outer product and Hessian matrices.

Optimization Control Options

PROC HPCOUNTREG uses the nonlinear optimization (NLO) subsystem to perform nonlinear optimization tasks. You can specify the following *options* in either the PROC HPCOUNTREG statement or the MODEL statement.

ABSCONV=*r*

ABSTOL=*r*

specifies an absolute function value convergence criterion by which minimization stops when $f(\theta^{(k)}) \leq r$. The default value of r is the negative square root of the largest double-precision value, which serves only as a protection against overflows.

ABSFCONV=*r*

ABSFTOL=*r*

specifies an absolute function difference convergence criterion by which minimization stops when the function value has a small change in successive iterations:

$$|f(\theta^{(k-1)}) - f(\theta^{(k)})| \leq r$$

The default is 0.

ABSGCONV=*r*

ABSGTOL=*r*

specifies an absolute gradient convergence criterion. Optimization stops when the maximum absolute gradient element is small:

$$\max_j |g_j(\theta^{(k)})| \leq r$$

The default is 1E–5.

ABSXCONV=*r*

ABSXTOL=*r*

specifies an absolute parameter convergence criterion. Optimization stops when the Euclidean distance between successive parameter vectors is small:

$$\| \theta^{(k)} - \theta^{(k-1)} \|_2 \leq r$$

The default is 0.

FCONV=*r*

FTOL=*r*

specifies a relative function convergence criterion. Optimization stops when a relative change of the function value in successive iterations is small:

$$\frac{|f(\theta^{(k)}) - f(\theta^{(k-1)})|}{|f(\theta^{(k-1)})|} \leq r$$

The default value is 2ϵ, where ϵ denotes the machine precision constant, which is the smallest double-precision floating-point number such that $1 + \epsilon > 1$.

GCONV=*r*

GTOL=*r*

specifies a relative gradient convergence criterion. For all techniques except CONGRA, optimization stops when the normalized predicted function reduction is small:

$$\frac{g(\theta^{(k)})^T [H^{(k)}]^{-1} g(\theta^{(k)})}{|f(\theta^{(k)})|} \leq r$$

For the CONGRA technique (where a reliable Hessian estimate H is not available), the following criterion is used:

$$\frac{\| g(\theta^{(k)}) \|_2^2 \quad \| s(\theta^{(k)}) \|_2}{\| g(\theta^{(k)}) - g(\theta^{(k-1)}) \|_2 \, |f(\theta^{(k)})|} \leq r$$

The default is 1E–8.

MAXFUNC=*i*

MAXFU=*i*

specifies the maximum number of function calls in the optimization process. The default is 1,000.

The optimization can terminate only after completing a full iteration. Therefore, the number of function calls that are actually performed can exceed the number of calls that are specified by this option.

MAXITER=*i*

MAXIT=*i*

specifies the maximum number of iterations in the optimization process. The default is 200.

MAXTIME=*r*

specifies an upper limit of *r* seconds of CPU time for the optimization process. The default value is the largest floating-point double representation of your computer. The time that is specified by this option is checked only once at the end of each iteration. Therefore, the actual run time can be much longer than *r*. The actual run time includes the remaining time needed to finish the iteration and the time needed to generate the output of the results.

METHOD=*value*

specifies the iterative minimization method to use. The default is METHOD=NEWRAP. You can specify the following *values*:

CONGRA specifies the conjugate-gradient method.

DBLDOG specifies the double-dogleg method.

NEWRAP specifies the Newton-Raphson method (this is the default).

NONE specifies that no optimization be performed beyond using the ordinary least squares method to compute the parameter estimates.

NRRIDG specifies the Newton-Raphson Ridge method.

QUANEW specifies the quasi-Newton method.

TRUREG specifies the trust region method.

SINGULAR=*r*
specifies the general singularity criterion that is applied by the HPCOUNTREG procedure in sweeps and inversions. The default is 1E–8.

BOUNDS Statement

BOUNDS *bound1* [*, bound2 . . .*] **;**

The BOUNDS statement imposes simple boundary constraints on the parameter estimates. You can specify any number of BOUNDS statements.

Each *bound* is composed of parameter names, constants, and inequality operators as follows:

item operator item [*operator item* [*operator item . . .*]]

Each *item* is a constant, a parameter name, or a list of parameter names. Each *operator* is <, >, <=, or >=. Parameter names are as shown in the Parameter column of the "Parameter Estimates" table. Please refer to the section "Parameter Naming Conventions for the RESTRICT, TEST, BOUNDS, and INIT Statements" on page 1103 for more information on how parameters are named in the BOUNDS statement.

You can use both the BOUNDS statement and the RESTRICT statement to impose boundary constraints. However, the BOUNDS statement provides a simpler syntax for specifying these kinds of constraints. For more information, see the section "RESTRICT Statement" on page 1088.

The following BOUNDS statement illustrates the use of parameter lists to specify boundary constraints. It constrains the estimates of the parameter for z to be negative, the parameters for x1 through x10 to be between 0 and 1, and the parameter for x1 in the zero-inflation model to be less than 1.

```
bounds z < 0,
       0 < x1-x10 < 1,
       Inf_x1 < 1;
```

BY Statement

BY *variables* **;**

A BY statement can be used with PROC HPCOUNTREG to obtain separate analyses on observations in groups defined by the BY variables. When a BY statement appears, the input data set should be sorted in order of the BY variables.

BY statement processing is not supported when the HPCOUNTREG procedure runs alongside the database or alongside the Hadoop Distributed File System (HDFS). These modes are used if the input data are stored in a database or HDFS and the grid host is the appliance that houses the data.

CLASS Statement

CLASS *variables* **;**

The CLASS statement names the classification variables to be used in the analysis. Classification variables can be either character or numeric.

Class levels are determined from the formatted values of the CLASS variables. Thus, you can use formats to group values into levels. For more information, see the discussion of the FORMAT procedure in *SAS Language Reference: Dictionary*.

DISPMODEL Statement

DISPMODEL *dependent-variable* ∼ < *dispersion-related-regressors* > **;**

The DISPMODEL statement specifies the *dispersion-related-regressors* that are used to model dispersion. This statement is ignored unless you specify DIST=CMPOISSON in the MODEL statement. The *dependent-variable* in the DISPMODEL statement must be the same as the *dependent-variable* in the MODEL statement.

The *dependent-variable* that appears in the DISPMODEL statement is directly used to model dispersion. Each of the q variables to the right of the tilde (∼) has a parameter to be estimated in the regression. For example, let $\mathbf{g}_i'$ be the ith observation's $1 \times (q+1)$ vector of values of the q dispersion explanatory variables (q_0 is set to 1 for the intercept term). Then the dispersion is a function of $\mathbf{g}_i'\boldsymbol{\delta}$, where $\boldsymbol{\delta}$ is the $(q+1)\times 1$ vector of parameters to be estimated, the dispersion model intercept is δ_0, and the coefficients for the q dispersion covariates are $\delta_1, \ldots, \delta_q$. If you specify DISP=CMPOISSON in the MODEL statement but do not include a DISPMODEL statement, then only the intercept term δ_0 is estimated. The "Parameter Estimates" table in the displayed output shows the estimates for the dispersion intercept and dispersion explanatory variables; they are labeled with the prefix "Disp_". For example, the dispersion intercept is labeled "Disp_Intercept". If you specify Age (a variable in your data set) as a dispersion explanatory variable, then the "Parameter Estimates" table labels the corresponding parameter estimate "Disp_Age". The following statements fit a Conway-Maxwell-Poisson model by using the regressors SEX, ILLNESS, and INCOME and by using AGE as a dispersion-related regressor:

```
proc hpcountreg data=docvisit;
   model doctorvisits=sex illness income / dist=cmpoisson;
   dispmodel doctorvisits ~ age;
run;
```

FREQ Statement

FREQ *freq-variable* **;**

The FREQ statement identifies a variable (*freq-variable*) that contains the frequency of occurrence of each observation. PROC HPCOUNTREG treats each observation as if it appears n times, where n is the value of *freq-variable* for the observation. If the value for the observation is not an integer, it is truncated to an integer. If the value is less than 1 or missing, the observation is not used in the model fitting. When the FREQ statement is not specified, each observation is assigned a frequency of 1.

INIT Statement

INIT *initialization1* < *, initialization2 . . .* > **;**

The INIT statement sets initial values for parameters in the optimization.

Each *initialization* is written as a parameter or parameter list, followed by an optional equal sign (=), followed by a number:

parameter <=> *number*

Parameter names are as shown in the Parameter column of the "Parameter Estimates" table. Please refer to the section "Parameter Naming Conventions for the RESTRICT, TEST, BOUNDS, and INIT Statements" on page 1103 for more information on how parameters are named in the INIT statement.

MODEL Statement

MODEL *dependent-variable* **=** *regressors* </ *options*> **;**

The MODEL statement specifies the dependent variable and independent regressor variables for the regression model. The dependent count variable should take only nonnegative integer values from the input data set. PROC HPCOUNTREG rounds any positive noninteger count value to the nearest integer. PROC HPCOUNTREG discards any observation that has a negative count.

Only one MODEL statement can be specified. You can specify the following *options* in the MODEL statement after a slash (/).

DIST=*value*

specifies a type of model to be analyzed. You can specify the following *values*:

POISSON | P specifies the Poisson regression model.

CMPOISSON | C | CMP specifies a Conway-Maxwell-Poisson regression model.

NEGBIN(P=1) specifies the negative binomial regression model that uses a linear variance function.

NEGBIN(P=2) | NEGBIN specifies the negative binomial regression model that uses a quadratic variance function.

ZIPOISSON | ZIP specifies zero-inflated Poisson regression.

ZICMPOISSON | ZICMP specifies a zero-inflated Conway-Maxwell-Poisson regression. The ZERO-MODEL statement must be specified when this model type is specified.

ZINEGBIN | ZINB specifies zero-inflated negative binomial regression.

You can also specify the DIST option in the HPCOUNTREG statement.

ERRORCOMP=FIXED | RANDOM

specifies a type of conditional panel model to be analyzed. You can specify the following model types:

FIXED specifies a fixed-effect error component regression model.

RANDOM specifies a random-effect error component regression model.

NOINT
suppresses the intercept parameter.

OFFSET=*offset-variable*
specifies a variable in the input data set to be used as an offset variable. The *offset-variable* is used to allow the observational units to vary across observations. For example, when the number of shipping accidents could be measured across different time periods or the number of students who participate in an activity could be reported across different class sizes, the observational units need to be adjusted to a common denominator by using the offset variable. The offset variable appears as a covariate in the model with its parameter restricted to 1. The offset variable cannot be the response variable, the zero-inflation offset variable (if any), or any of the explanatory variables. The "Model Fit Summary" table gives the name of the data set variable that is used as the offset variable; it is labeled "Offset."

PARAMETER=MU | LAMBDA
specifies the parameterization for the Conway-Maxwell-Poisson model. The following parameterizations are supported:

LAMBDA	estimates the original Conway-Maxwell-Poisson model (Shmueli et al. 2005).
MU	reparameterizes λ as documented by Guikema and Coffelt (2008), where $\mu = \lambda^{1/\nu}$ and the integral part of μ represents the mode, which can be considered a measure of central tendency (mean).

By default, PARAMETER=MU.

Printing Options

You can specify the following options in either the PROC HPCOUNTREG statement or the MODEL statement:

CORRB
prints the correlation matrix of the parameter estimates.

COVB
prints the covariance matrix of the parameter estimates.

NOPRINT
suppresses all printed output.

PRINTALL
requests all printing options.

OUTPUT Statement

OUTPUT < *output-options* > **;**

The OUTPUT statement creates a new SAS data set that includes variables created by the *output-options*. These variables include the estimates of $\mathbf{x}_i'\boldsymbol{\beta}$, the expected value of the response variable, and the probability of the response variable taking on the current value. Furthermore, if a zero-inflated model was fit, you can request that the output data set contain the estimates of $\mathbf{z}_i'\boldsymbol{\gamma}$ and the probability that the response is zero as a

result of the zero-generating process. For the Conway-Maxwell-Poisson model, the estimates of $\mathbf{g}_i'\boldsymbol{\delta}$, λ, ν, μ, mode, variance, and dispersion are also available. Except for the probability of the current value, these statistics can be computed for all observations in which the regressors are not missing, even if the response is missing. By adding observations that have missing response values to the input data set, you can compute these statistics for new observations or for settings of the regressors that are not present in the data without affecting the model fit.

You can specify only one OUTPUT statement. You can specify the following *output-options*:

OUT=*SAS-data-set*
names the output data set

COPYVAR=*SAS-variable-names*

COPYVARS=*SAS-variable-names*
adds SAS variables to the output data set.

XBETA=*name*
names the variable to contain estimates of $\mathbf{x}_i'\boldsymbol{\beta}$.

PRED=*name*

MEAN=*name*
names the variable to contain the predicted value of the response variable.

PROB=*name*
names the variable to contain the probability that the response variable will take the actual value, $\Pr(Y = y_i)$.

PROBCOUNT(*value1* **<** *value2* **... >)**
outputs the probability that the response variable will take particular values. Each *value* should be a nonnegative integer. If you specify a noninteger, it is rounded to the nearest integer. The *value* can also be a list of the form X TO Y BY Z. For example, PROBCOUNT(0 1 2 TO 10 BY 2 15) requests predicted probabilities for counts 0, 1, 2, 4, 5, 6, 8, 10, and 15. This option is not available for the fixed-effects and random-effects panel models.

ZGAMMA=*name*
names the variable to contain estimates of $\mathbf{z}_i'\boldsymbol{\gamma}$.

PROBZERO=*name*
names the variable to contain the value of φ_i, which is the probability that the response variable will take the value of 0 as a result of the zero-generating process. This variable is written to the output file only if the model is zero-inflated.

GDELTA=*name*
assigns a name to the variable that contains estimates of $\mathbf{g}_i'\boldsymbol{\delta}$ for the Conway-Maxwell-Poisson distribution.

LAMBDA=*name*
assigns a name to the variable that contains the estimate of λ for the Conway-Maxwell-Poisson distribution.

NU=*name*
assigns a name to the variable that contains the estimate of ν for the Conway-Maxwell-Poisson distribution.

MU=*name*
assigns a name to the variable that contains the estimate of μ for the Conway-Maxwell-Poisson distribution.

MODE=*name*
assigns a name to the variable that contains the integral part of μ (mode) for the Conway-Maxwell-Poisson distribution.

VARIANCE=*name*
assigns a name to the variable that contains the estimate of variance for the Conway-Maxwell-Poisson distribution.

DISPERSION=*name*
assigns a name to the variable that contains the value of dispersion for the Conway-Maxwell-Poisson distribution.

PERFORMANCE Statement

PERFORMANCE < *performance-options* > **;**

The PERFORMANCE statement specifies options to control the multithreaded and distributed computing environment and requests detailed results about the performance characteristics of the HPCOUNTREG procedure. You can also use the PERFORMANCE statement to control whether the HPCOUNTREG procedure executes in single-machine or distributed mode. The most commonly used *performance-options* in the PERFORMANCE statement are as follows:

DETAILS
requests a table that shows a timing breakdown of the procedure steps.

NODES=*n*
specifies the number of nodes in the distributed computing environment, provided that the data are not processed alongside the database.

NTHREADS=*n*
specifies the number of threads for analytic computations and overrides the SAS system option THREADS | NOTHREADS. If you do not specify the NTHREADS= option, PROC HPCOUNTREG creates one thread per CPU for the analytic computations.

For more information about the PERFORMANCE statement for high-performance analytical procedures, see the section "PERFORMANCE Statement" on page 87 of Chapter 3, "Shared Concepts and Topics."

RESTRICT Statement

RESTRICT *restriction1* [, *restriction2 . . .*] **;**

The RESTRICT statement imposes linear restrictions on the parameter estimates. You can specify any number of RESTRICT statements.

Each *restriction* is written as an expression, followed by an equality operator (=) or an inequality operator (<, >, <=, >=) and then by a second expression, as follows:

expression operator expression

The *operator* can be =, <, >, <=, or >=.

Restriction expressions can be composed of parameter names, constants, and the following operators: times (∗), plus (+), and minus (−). Parameter names are as shown in the Effect column of the "Parameter Estimates" table. The restriction expressions must be a linear function of the variables.

Parameter names are as shown in the Parameter column of the "Parameter Estimates" table. Please refer to the section "Parameter Naming Conventions for the RESTRICT, TEST, BOUNDS, and INIT Statements" on page 1103 for more information on how parameters are named in the RESTRICT statement.

Lagrange multipliers are reported in the "Parameter Estimates" table for all the active linear constraints. They are identified by the names Restrict1, Restrict2, and so on. The probabilities of these Lagrange multipliers are computed using a beta distribution (LaMotte 1994). Nonactive (nonbinding) restrictions have no effect on the estimation results and are not noted in the output.

The following RESTRICT statement constrains the negative binomial dispersion parameter α to 1, which restricts the conditional variance to be $\mu + \mu^2$:

```
restrict _Alpha = 1;
```

TEST Statement

<label:> **TEST** *<'string'> equation1 <, equation2. . . > / < test-options>* **;**

The TEST statement performs Wald, Lagrange multiplier, and likelihood ratio tests of linear hypotheses about the regression parameters that are specified in the preceding MODEL statement.

You can add a label (which is printed in the output) to a TEST statement in two ways: add an unquoted *label* followed by a colon before the TEST keyword, or add a quoted *string* after the TEST keyword. The unquoted *label* cannot contain any spaces. If you include both an unquoted *label* and a quoted *string*, PROC HPCOUNTREG uses the unquoted *label*. If you specify neither an unquoted *label* nor a quoted *string*, PROC HPCOUNTREG automatically labels the tests.

Each *equation* specifies a linear hypothesis to be tested and consists of regression parameter names and relational operators. The regression parameter names are as shown in the Parameter column of the "Parameter Estimates" table. For more information about how parameters are named in the TEST statement, the section "Parameter Naming Conventions for the RESTRICT, TEST, BOUNDS, and INIT Statements" on page 1103. Only linear equality restrictions and tests are permitted in PROC COUNTREG. Test *equations* can consist

only of algebraic operations that involve the addition symbol (+), subtraction symbol (-), and multiplication symbol (*).

All hypotheses in one TEST statement are tested jointly.

You can specify the following *test-options* after a slash (/):

ALL
requests Wald, Lagrange multiplier, and likelihood ratio tests.

LM
requests the Lagrange multiplier test.

LR
requests the likelihood ratio test.

WALD
requests the Wald test.

By default, the Wald test is performed.

The following illustrates the use of the TEST statement:

```
proc hpcountreg;
   model y = x1 x2 x3;
   test x1 = 0, x2 * .5 + 2 * x3 = 0;
   test _int: test intercept = 0, x3 = 0;
run;
```

The first test investigates the joint hypothesis that

$$\beta_1 = 0$$

and

$$0.5\beta_2 + 2\beta_3 = 0$$

Only linear equality restrictions and tests are permitted in PROC HPCOUNTREG. Tests expressions can consist only of algebraic operations that involve the addition symbol (+), subtraction symbol (-), and multiplication symbol (*).

WEIGHT Statement

WEIGHT *variable* < / *option* > **;**

The WEIGHT statement specifies a variable to supply weighting values to use for each observation in estimating parameters. The log likelihood for each observation is multiplied by the corresponding weight variable value.

If the weight of an observation is nonpositive, that observation is not used in the estimation.

The following *option* can be added to the WEIGHT statement after a slash (/).

NONORMALIZE
does not normalize the weights. (By default, the weights are normalized so that they add up to the actual sample size. The weights w_i are normalized by multiplying them by $\frac{n}{\sum_{i=1}^{n} w_i}$, where n is the sample size.) If the weights are required to be used as they are, then specify the NONORMALIZE option.

ZEROMODEL Statement

ZEROMODEL *dependent-variable* ∼ *zero-inflated-regressors* < / *options* > **;**

The ZEROMODEL statement is required if either ZIP or ZINB is specified in the DIST= option in the MODEL statement. If ZIP or ZINB is specified, then the ZEROMODEL statement must follow the MODEL statement. The dependent variable in the ZEROMODEL statement must be the same as the dependent variable in the MODEL statement.

The zero-inflated (ZI) regressors appear in the equation that determines the probability (φ_i) of a zero count. Each of these q variables has a parameter to be estimated in the regression. For example, let $\mathbf{z}_i'$ be the ith observation's $1 \times (q+1)$ vector of values of the q ZI explanatory variables (w_0 is set to 1 for the intercept term). Then φ_i is a function of $\mathbf{z}_i'\boldsymbol{\gamma}$, where $\boldsymbol{\gamma}$ is the $(q+1) \times 1$ vector of parameters to be estimated. (The zero-inflated intercept is γ_0; the coefficients for the q zero-inflated covariates are $\gamma_1, \ldots, \gamma_q$.) If q is equal to 0 (no ZI explanatory variables are provided), then only the intercept term γ_0 is estimated. The "Parameter Estimates" table in the displayed output shows the estimates for the ZI intercept and ZI explanatory variables; they are labeled with the prefix "Inf_". For example, the ZI intercept is labeled "Inf_intercept". If you specify Age (a variable in your data set) as a ZI explanatory variable, then the "Parameter Estimates" table labels the corresponding parameter estimate "Inf_Age".

You can specify the following *options* in the ZEROMODEL statement after a slash (/):

LINK=LOGISTIC | NORMAL
specifies the distribution function used to compute probability of zeros. The supported distribution functions are as follows:

LOGISTIC specifies logistic distribution.

NORMAL specifies standard normal distribution.

If this option is omitted, then the default ZI link function is logistic.

OFFSET=*zero-inflated-offset-variable*
specifies a variable in the input data set to be used as a zero-inflated (ZI) offset variable. The ZI offset variable *zero-inflated-offset-variable* is included as a term, with coefficient restricted to 1, in the equation that determines the probability (φ_i) of a zero count and represents an adjustment to a common observational unit. The ZI offset variable cannot be the response variable, the offset variable (if any), or any of the explanatory variables. The name of the data set variable that is used as the ZI offset variable is displayed in the "Model Fit Summary" table, where it is labeled as "Inf_offset".

Details: HPCOUNTREG Procedure

Missing Values

Any observations in the input data set that have a missing value for one or more of the regressors are ignored by PROC HPCOUNTREG and not used in the model fit. PROC HPCOUNTREG rounds any positive noninteger count values to the nearest integer and ignores any observations that have a negative count.

If the input data set contains any observations that have missing response values but nonmissing regressors, PROC HPCOUNTREG can compute several statistics and store them in an output data set by using the OUTPUT statement. For example, you can request that the output data set contain the estimates of $\mathbf{x}_i'\boldsymbol{\beta}$, the expected value of the response variable, and the probability that the response variable will take the current value. Furthermore, if a zero-inflated model was fit, you can request that the output data set contain the estimates of $\mathbf{z}_i'\boldsymbol{\gamma}$, and the probability that the response is 0 as a result of the zero-generating process. Note that the presence of such observations (that have missing response values) does not affect the model fit.

Poisson Regression

The most widely used model for count data analysis is Poisson regression. Poisson regression assumes that y_i, given the vector of covariates $\mathbf{x}_i$, is independently Poisson distributed with

$$P(Y_i = y_i|\mathbf{x}_i) = \frac{e^{-\mu_i}\mu_i^{y_i}}{y_i!}, \quad y_i = 0, 1, 2, \ldots$$

and the mean parameter—that is, the mean number of events per period—is given by

$$\mu_i = \exp(\mathbf{x}_i'\boldsymbol{\beta})$$

where $\boldsymbol{\beta}$ is a $(k+1)\times 1$ parameter vector. (The intercept is β_0; the coefficients for the k regressors are $\beta_1, \ldots, \beta_k$.) Taking the exponential of $\mathbf{x}_i'\boldsymbol{\beta}$ ensures that the mean parameter μ_i is nonnegative. It can be shown that the conditional mean is given by

$$E(y_i|\mathbf{x}_i) = \mu_i = \exp(\mathbf{x}_i'\boldsymbol{\beta})$$

Note that the conditional variance of the count random variable is equal to the conditional mean in the Poisson regression model:

$$V(y_i|\mathbf{x}_i) = E(y_i|\mathbf{x}_i) = \mu_i$$

The equality of the conditional mean and variance of y_i is known as *equidispersion*.

The standard estimator for the Poisson model is the maximum likelihood estimator (MLE). Because the observations are independent, the log-likelihood function is written as

$$\mathcal{L} = \sum_{i=1}^{N}(-\mu_i + y_i \ln \mu_i - \ln y_i!) = \sum_{i=1}^{N}(-e^{\mathbf{x}_i'\boldsymbol{\beta}} + y_i\mathbf{x}_i'\boldsymbol{\beta} - \ln y_i!)$$

For more information about the Poisson regression model, see the section "Poisson Regression" on page 622.

The Poisson model has been criticized for its restrictive property that the conditional variance equals the conditional mean. Real-life data are often characterized by *overdispersion*—that is, the variance exceeds the mean. Allowing for overdispersion can improve model predictions because the Poisson restriction of equal mean and variance results in the underprediction of zeros when overdispersion exists. The most commonly used model that accounts for overdispersion is the negative binomial model. Conway-Maxwell-Poisson regression enables you to model both overdispersion and underdispersion.

Conway-Maxwell-Poisson Regression

The Conway-Maxwell-Poisson (CMP) distribution is a generalization of the Poisson distribution that enables you to model both underdispersed and overdispersed data. It was originally proposed by Conway and Maxwell (1962), but its implementation to model under- and overdispersed count data is attributed to Shmueli et al. (2005).

Recall that y_i, given the vector of covariates $\mathbf{x}_i$, is independently Poisson-distributed as

$$P(Y_i = y_i|\mathbf{x}_i) = \frac{e^{-\lambda_i}\lambda_i^{y_i}}{y_i!}, \quad y_i = 0, 1, 2, \ldots$$

The Conway-Maxwell-Poisson distribution is defined as

$$P(Y_i = y_i|\mathbf{x}_i, \mathbf{z}_i) = \frac{1}{Z(\lambda_i, \nu_i)} \frac{\lambda_i^{y_i}}{(y_i!)^{\nu_i}}, \quad y_i = 0, 1, 2, \ldots$$

where the normalization factor is

$$Z(\lambda_i, \nu_i) = \sum_{n=0}^{\infty} \frac{\lambda_i^n}{(n!)^{\nu_i}}$$

and

$$\lambda_i = \exp(\mathbf{x}_i'\boldsymbol{\beta})$$

$$\nu_i = -\exp(\mathbf{g}_i'\delta)$$

The $\boldsymbol{\beta}$ vector is a $(k + 1) \times 1$ parameter vector. (The intercept is β_0, and the coefficients for the k regressors are $\beta_1, \ldots, \beta_k$.) The δ vector is an $(m + 1) \times 1$ parameter vector. (The intercept is represented by δ_0, and the coefficients for the m regressors are $\delta_1, \ldots, \delta_k$.) The covariates are represented by x_i and g_i vectors.

One of the restrictive properties of the Poisson model is that the conditional mean and variance must be equal:

$$E(y_i|\mathbf{x}_i) = V(y_i|\mathbf{x}_i) = \lambda_i = \exp(\mathbf{x}_i'\boldsymbol{\beta})$$

The CMP distribution overcomes this restriction by defining an additional parameter, ν, which governs the rate of decay of successive ratios of probabilities such that

$$P(Y_i = y_i - 1)/P(Y_i = y_i) = \frac{(y_i)^{\nu_i}}{\lambda_i}$$

The introduction of the additional parameter, ν, allows for flexibility in modeling the tail behavior of the distribution. If $\nu = 1$, the ratio is equal to the rate of decay of the Poisson distribution. If $\nu < 1$, the rate of decay decreases, enabling you to model processes that have longer tails than the Poisson distribution (overdispersed data). If $\nu > 1$, the rate of decay increases in a nonlinear fashion, thus shortening the tail of the distribution (underdispersed data).

There are several special cases of the Conway-Maxwell-Poisson distribution. If $\lambda < 1$ and $\nu \to \infty$, the Conway-Maxwell-Poisson results in the Bernoulli distribution. In this case, the data can take only the values 0 and 1, which represents an extreme underdispersion. If $\nu = 1$, the Poisson distribution is recovered with its equidispersion property. When $\nu = 0$ and $\lambda < 1$, the normalization factor is convergent and forms a geometric series,

$$Z(\lambda_i, 0) = \frac{1}{1 - \lambda_i}$$

and the probability density function becomes

$$P(Y = y_i; \lambda_i, \nu_i = 0) = (1 - \lambda_i)\lambda_i^{y_i}$$

The geometric distribution represents a case of severe overdispersion.

Mean, Variance, and Dispersion for the Conway-Maxwell-Poisson Model

The mean and the variance of the Conway-Maxwell-Poisson distribution are defined as

$$E[Y] = \frac{\partial \ln Z}{\partial \ln \lambda}$$

$$V[Y] = \frac{\partial^2 \ln Z}{\partial^2 \ln \lambda}$$

The Conway-Maxwell-Poisson distribution does not have closed-form expressions for its moments in terms of its parameters λ and ν. However, the moments can be approximated. Shmueli et al. (2005) use asymptotic expressions for Z to derive $E(Y)$ and $V(Y)$ as

$$E[Y] \approx \lambda^{1/\nu} + \frac{1}{2\nu} - \frac{1}{2}$$

$$V[Y] \approx \frac{1}{\nu}\lambda^{1/\nu}$$

In the Conway-Maxwell-Poisson model, the summation of infinite series is evaluated using a logarithmic expansion. The mean and variance are calculated as follows for the Shmueli et al. (2005) model:

$$E(Y) = \frac{1}{Z(\lambda, \nu)} \sum_{j=0}^{\infty} \frac{j\lambda^j}{(j!)^\nu}$$

$$V(Y) = \frac{1}{Z(\lambda, \nu)} \sum_{j=0}^{\infty} \frac{j^2\lambda^j}{(j!)^\nu} - E(Y)^2$$

The dispersion is defined as

$$D(Y) = \frac{V(Y)}{E(Y)}$$

Likelihood Function for the Conway-Maxwell-Poisson Model

The likelihood for a set of n independently and identically distributed variables $y_1, y_2, \ldots, y_n$ is written as

$$\begin{aligned} L(y_1, y_2, \ldots, y_n|\lambda, \nu) &= \frac{\prod_{i=1}^{n} \lambda^{y_i}}{(\prod_{i=1}^{n} y_i!)^{\nu}} Z(\lambda, \nu)^{-n} \\ &= \lambda^{\sum_{i=1}^{n} y_i} \exp\left(-\nu \sum_{i=1}^{n} \ln(y_i!)\right) Z(\lambda, \nu)^{-n} \\ &= \lambda^{S_1} \exp(-\nu S_2) Z(\lambda, \nu)^{-n} \end{aligned}$$

where S_1 and S_2 are sufficient statistics for $y_1, y_2, \ldots, y_n$. You can see from the preceding equation that the Conway-Maxwell-Poisson distribution is a member of the exponential family. The log-likelihood function can be written as

$$\mathcal{L} = -n \ln(Z(\lambda, \nu)) + \sum_{i=1}^{n} (y_i \ln(\lambda) - \nu \ln(y_i!))$$

The gradients can be written as

$$\mathcal{L}_\beta = \left(\sum_{k=1}^{N} y_k - n \frac{\lambda Z(\lambda, \nu)_\lambda}{Z(\lambda, \nu)} \right) \mathbf{x}$$

$$\mathcal{L}_\delta = \left(\sum_{k=1}^{N} \ln(y_k!) - n \frac{Z(\lambda, \nu)_\nu}{Z(\lambda, \nu)} \right) \nu \mathbf{z}$$

Conway-Maxwell-Poisson Regression: Guikema and Coffelt (2008) Reparameterization

Guikema and Coffelt (2008) propose a reparameterization of the Shmueli et al. (2005) Conway-Maxwell-Poisson model to provide a measure of central tendency that can be interpreted in the context of the generalized linear model. By substituting $\lambda = \mu^\nu$, the Guikema and Coffelt (2008) formulation is written as

$$P(Y = y_i; \mu, \nu) = \frac{1}{S(\mu, \nu)} \left(\frac{\mu^{y_i}}{y_i!} \right)^{\nu}$$

where the new normalization factor is defined as

$$S(\mu, \nu) = \sum_{j=0}^{\infty} \left(\frac{\mu^j}{j!} \right)^{\nu}$$

In terms of their new formulations, the mean and variance of Y are given as

$$E[Y] = \frac{1}{\nu} \frac{\partial \ln S}{\partial \ln \mu}$$

$$V[Y] = \frac{1}{\nu^2} \frac{\partial^2 \ln S}{\partial^2 \ln \mu}$$

They can be approximated as

$$E[Y] \approx \mu + \frac{1}{2}\nu - \frac{1}{2}$$

$$V[Y] \approx \frac{\mu}{\nu}$$

In the HPCOUNTREG procedure, the mean and variance are calculated according to the following formulas, respectively, for the Guikema and Coffelt (2008) model:

$$E(Y) = \frac{1}{Z(\lambda,\mu)} \sum_{j=0}^{\infty} \frac{j\mu^{\nu j}}{(j!)^{\nu}}$$

$$V(Y) = \frac{1}{Z(\lambda,\mu)} \sum_{j=0}^{\infty} \frac{j^2\mu^{\nu j}}{(j!)^{\nu}} - E(Y)^2$$

In terms of the new parameter μ, the log-likelihood function is specified as

$$\mathcal{L} = \ln(S(\mu,\nu)) + \nu \sum_{i=1}^{N} (y_i \ln(\mu) - \ln(y_i!))$$

and the gradients are calculated as

$$\mathcal{L}_{\beta} = \left(\nu \sum_{i=1}^{N} y_i - \frac{\mu S(\mu,\nu)_{\mu}}{S(\mu,\nu)} \right) \mathbf{x}$$

$$\mathcal{L}_{\delta} = \left(\sum_{i=1}^{N} (y_i \ln(\mu) - \ln(y_i!)) - \frac{S(\mu,\nu)_{\nu}}{S(\mu,\nu)} \right) \nu \mathbf{g}$$

By default, the HPCOUNTREG procedure uses the Guikema and Coffelt (2008) specification. The Shmueli et al. (2005) model can be estimated by specifying the PARAMETER=LAMBDA option. If you specify DISP=CMPOISSON in the MODEL statement and you omit the DISPMODEL statement, the model is estimated according to the Lord, Guikema, and Geedipally (2008) specification, where ν represents a single parameter that does not depend on any covariates. The Lord, Guikema, and Geedipally (2008) specification makes the model comparable to the negative binomial model because it has only one parameter.

The dispersion is defined as

$$D(Y) = \frac{V(Y)}{E(Y)}$$

Using the Guikema and Coffelt (2008) specification results in the integral part of μ representing the mode, which is a reasonable approximation for the mean. The dispersion can be written as

$$D(Y) = \frac{V(Y)}{E(Y)} \approx \frac{\frac{\mu}{\nu}}{\mu + \frac{1}{2}\nu - \frac{1}{2}} \approx \frac{1}{\upsilon}$$

When $\nu < 1$, the variance can be shown to be greater than the mean and the dispersion greater than 1. This is a result of overdispersed data. When $\nu = 1$ and the mean and variance are equal, the dispersion is equal to 1 (Poisson model). When $\nu > 1$, the variance is smaller than the mean and the dispersion is less than 1. This is a result of underdispersed data.

All Conway-Maxwell-Poisson models in the HPCOUNTREG procedure are parameterized in terms of dispersion, where

$$-\ln(\nu) = \delta_0 + \sum_{n=1}^{q} \delta_n g_n$$

Negative values of $\ln(\nu)$ indicate that the data are approximately overdispersed, and positive values of $\ln(\nu)$ indicate that the data are approximately underdispersed.

Negative Binomial Regression

The Poisson regression model can be generalized by introducing an unobserved heterogeneity term for observation i. Thus, the individuals are assumed to differ randomly in a manner that is not fully accounted for by the observed covariates. This is formulated as

$$E(y_i|\mathbf{x}_i, \tau_i) = \mu_i \tau_i = e^{\mathbf{x}_i'\boldsymbol{\beta}+\epsilon_i}$$

where the unobserved heterogeneity term $\tau_i = e^{\epsilon_i}$ is independent of the vector of regressors $\mathbf{x}_i$. Then the distribution of y_i conditional on $\mathbf{x}_i$ and τ_i is Poisson with conditional mean and conditional variance $\mu_i \tau_i$:

$$f(y_i|\mathbf{x}_i, \tau_i) = \frac{\exp(-\mu_i \tau_i)(\mu_i \tau_i)^{y_i}}{y_i!}$$

Let $g(\tau_i)$ be the probability density function of τ_i. Then, the distribution $f(y_i|\mathbf{x}_i)$ (no longer conditional on τ_i) is obtained by integrating $f(y_i|\mathbf{x}_i, \tau_i)$ with respect to τ_i:

$$f(y_i|\mathbf{x}_i) = \int_0^\infty f(y_i|\mathbf{x}_i, \tau_i) g(\tau_i) d\tau_i$$

An analytical solution to this integral exists when τ_i is assumed to follow a gamma distribution. This solution is the negative binomial distribution. If the model contains a constant term, then in order to identify the mean of the distribution, it is necessary to assume that $E(e^{\epsilon_i}) = E(\tau_i) = 1$. Thus, it is assumed that τ_i follows a gamma(θ, θ) distribution with $E(\tau_i) = 1$ and $V(\tau_i) = 1/\theta$,

$$g(\tau_i) = \frac{\theta^\theta}{\Gamma(\theta)} \tau_i^{\theta-1} \exp(-\theta \tau_i)$$

where $\Gamma(x) = \int_0^\infty z^{x-1}\exp(-z)dz$ is the gamma function and θ is a positive parameter. Then, the density of y_i given $\mathbf{x}_i$ is derived as

$$\begin{aligned} f(y_i|\mathbf{x}_i) &= \int_0^\infty f(y_i|\mathbf{x}_i,\tau_i)g(\tau_i)d\tau_i \\ &= \frac{\theta^\theta \mu_i^{y_i}}{y_i!\Gamma(\theta)}\int_0^\infty e^{-(\mu_i+\theta)\tau_i}\tau_i^{\theta+y_i-1}d\tau_i \\ &= \frac{\theta^\theta \mu_i^{y_i}\Gamma(y_i+\theta)}{y_i!\Gamma(\theta)(\theta+\mu_i)^{\theta+y_i}} \\ &= \frac{\Gamma(y_i+\theta)}{y_i!\Gamma(\theta)}\left(\frac{\theta}{\theta+\mu_i}\right)^\theta\left(\frac{\mu_i}{\theta+\mu_i}\right)^{y_i} \end{aligned}$$

If you make the substitution $\alpha = \frac{1}{\theta}$ ($\alpha > 0$), the negative binomial distribution can then be rewritten as

$$f(y_i|\mathbf{x}_i) = \frac{\Gamma(y_i+\alpha^{-1})}{y_i!\Gamma(\alpha^{-1})}\left(\frac{\alpha^{-1}}{\alpha^{-1}+\mu_i}\right)^{\alpha^{-1}}\left(\frac{\mu_i}{\alpha^{-1}+\mu_i}\right)^{y_i}, \quad y_i = 0,1,2,\ldots$$

Thus, the negative binomial distribution is derived as a gamma mixture of Poisson random variables. It has the conditional mean

$$E(y_i|\mathbf{x}_i) = \mu_i = e^{\mathbf{x}_i'\boldsymbol{\beta}}$$

and the conditional variance

$$V(y_i|\mathbf{x}_i) = \mu_i[1+\frac{1}{\theta}\mu_i] = \mu_i[1+\alpha\mu_i] > E(y_i|\mathbf{x}_i)$$

The conditional variance of the negative binomial distribution exceeds the conditional mean. Overdispersion results from neglected unobserved heterogeneity. The negative binomial model with variance function $V(y_i|\mathbf{x}_i) = \mu_i + \alpha\mu_i^2$, which is quadratic in the mean, is referred to as the NEGBIN2 model Cameron and Trivedi (1986). To estimate this model, specify DIST=NEGBIN(P=2) in the MODEL statement. The Poisson distribution is a special case of the negative binomial distribution where $\alpha = 0$. A test of the Poisson distribution can be carried out by testing the hypothesis that $\alpha = \frac{1}{\theta_i} = 0$. A Wald test of this hypothesis is provided (it is the reported t statistic for the estimated α in the negative binomial model).

The log-likelihood function of the negative binomial regression model (NEGBIN2) is given by

$$\begin{aligned} \mathcal{L} = \sum_{i=1}^N \Bigg\{ &\sum_{j=0}^{y_i-1}\ln(j+\alpha^{-1}) - \ln(y_i!) \\ &-(y_i+\alpha^{-1})\ln(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})) + y_i\ln(\alpha) + y_i\mathbf{x}_i'\boldsymbol{\beta}\Bigg\} \end{aligned}$$

where use of the following fact is made if y is an integer:

$$\Gamma(y+a)/\Gamma(a) = \prod_{j=0}^{y-1}(j+a)$$

Cameron and Trivedi (1986) consider a general class of negative binomial models that have mean μ_i and variance function $\mu_i + \alpha\mu_i^p$. The NEGBIN2 model, with $p = 2$, is the standard formulation of the negative binomial model. Models that have other values of p, $-\infty < p < \infty$, have the same density $f(y_i|\mathbf{x}_i)$, except that α^{-1} is replaced everywhere by $\alpha^{-1}\mu^{2-p}$. The negative binomial model NEGBIN1, which sets $p = 1$, has the variance function $V(y_i|\mathbf{x}_i) = \mu_i + \alpha\mu_i$, which is linear in the mean. To estimate this model, specify DIST=NEGBIN(P=1) in the MODEL statement.

The log-likelihood function of the NEGBIN1 regression model is given by

$$\begin{aligned}\mathcal{L} &= \sum_{i=1}^{N}\left\{\sum_{j=0}^{y_i-1}\ln\left(j+\alpha^{-1}\exp(\mathbf{x}_i'\boldsymbol{\beta})\right)\right.\\ &\quad\left.-\ln(y_i!)-\left(y_i+\alpha^{-1}\exp(\mathbf{x}_i'\boldsymbol{\beta})\right)\ln(1+\alpha)+y_i\ln(\alpha)\right\}\end{aligned}$$

For more information about the negative binomial regression model, see the section "Negative Binomial Regression" on page 627.

Zero-Inflated Count Regression Overview

The main motivation for using zero-inflated count models is that real-life data frequently display overdispersion and excess zeros. Zero-inflated count models provide a way to both model the excess zeros and allow for overdispersion. In particular, there are two possible data generation processes for each observation. The result of a Bernoulli trial is used to determine which of the two processes to use. For observation i, Process 1 is chosen with probability φ_i and Process 2 with probability $1-\varphi_i$. Process 1 generates only zero counts. Process 2 generates counts from either a Poisson or a negative binomial model. In general,

$$y_i \sim \begin{cases} 0 & \text{with probability} \quad \varphi_i \\ g(y_i) & \text{with probability} \quad 1-\varphi_i \end{cases}$$

Therefore, the probability of $\{Y_i = y_i\}$ can be described as

$$\begin{aligned} P(y_i = 0|\mathbf{x}_i) &= \varphi_i + (1-\varphi_i)g(0) \\ P(y_i|\mathbf{x}_i) &= (1-\varphi_i)g(y_i), \quad y_i > 0 \end{aligned}$$

where $g(y_i)$ follows either the Poisson or the negative binomial distribution.

If the probability φ_i depends on the characteristics of observation i, then φ_i is written as a function of $\mathbf{z}_i'\boldsymbol{\gamma}$, where $\mathbf{z}_i'$ is the $1 \times (q+1)$ vector of zero-inflated covariates and $\boldsymbol{\gamma}$ is the $(q+1) \times 1$ vector of zero-inflated coefficients to be estimated. (The zero-inflated intercept is γ_0; the coefficients for the q zero-inflated covariates are $\gamma_1, \ldots, \gamma_q$.) The function F that relates the product $\mathbf{z}_i'\boldsymbol{\gamma}$ (which is a scalar) to the probability φ_i is called the zero-inflated link function,

$$\varphi_i = F_i = F(\mathbf{z}_i'\boldsymbol{\gamma})$$

In the HPCOUNTREG procedure, the zero-inflated covariates are indicated in the ZEROMODEL statement. Furthermore, the zero-inflated link function F can be specified as either the logistic function,

$$F(\mathbf{z}_i'\boldsymbol{\gamma}) = \Lambda(\mathbf{z}_i'\boldsymbol{\gamma}) = \frac{\exp(\mathbf{z}_i'\boldsymbol{\gamma})}{1+\exp(\mathbf{z}_i'\boldsymbol{\gamma})}$$

or the standard normal cumulative distribution function (also called the probit function),

$$F(\mathbf{z}_i'\boldsymbol{\gamma}) = \Phi(\mathbf{z}_i'\boldsymbol{\gamma}) = \int_0^{\mathbf{z}_i'\boldsymbol{\gamma}} \frac{1}{\sqrt{2\pi}}\exp(-u^2/2)du$$

The zero-inflated link function is indicated by using the LINK= option in the ZEROMODEL statement. The default ZI link function is the logistic function.

Zero-Inflated Poisson Regression

In the zero-inflated Poisson (ZIP) regression model, the data generation process that is referred to earlier as Process 2 is

$$g(y_i) = \frac{\exp(-\mu_i)\mu_i^{y_i}}{y_i!}$$

where $\mu_i = e^{\mathbf{x}_i'\boldsymbol{\beta}}$. Thus the ZIP model is defined as

$$\begin{aligned} P(y_i = 0|\mathbf{x}_i, \mathbf{z}_i) &= F_i + (1-F_i)\exp(-\mu_i) \\ P(y_i|\mathbf{x}_i, \mathbf{z}_i) &= (1-F_i)\frac{\exp(-\mu_i)\mu_i^{y_i}}{y_i!}, \quad y_i > 0 \end{aligned}$$

The conditional expectation and conditional variance of y_i are given by

$$E(y_i|\mathbf{x}_i, \mathbf{z}_i) = \mu_i(1-F_i)$$

$$V(y_i|\mathbf{x}_i, \mathbf{z}_i) = E(y_i|\mathbf{x}_i, \mathbf{z}_i)(1+\mu_i F_i)$$

Note that the ZIP model (in addition to the ZINB model) exhibits overdispersion because $V(y_i|\mathbf{x}_i, \mathbf{z}_i) > E(y_i|\mathbf{x}_i, \mathbf{z}_i)$.

In general, the log-likelihood function of the ZIP model is

$$\mathcal{L} = \sum_{i=1}^{N} \ln\left[P(y_i|\mathbf{x}_i, \mathbf{z}_i)\right]$$

After a specific link function (either logistic or standard normal) for the probability φ_i is chosen, it is possible to write the exact expressions for the log-likelihood function and the gradient.

ZIP Model with Logistic Link Function

First, consider the ZIP model in which the probability φ_i is expressed by a logistic link function, namely

$$\varphi_i = \frac{\exp(\mathbf{z}_i'\boldsymbol{\gamma})}{1+\exp(\mathbf{z}_i'\boldsymbol{\gamma})}$$

The log-likelihood function is

$$\begin{aligned}\mathcal{L} &= \sum_{\{i:y_i=0\}} \ln\left[\exp(\mathbf{z}_i'\boldsymbol{\gamma}) + \exp(-\exp(\mathbf{x}_i'\boldsymbol{\beta}))\right] \\ &\quad + \sum_{\{i:y_i>0\}} \left[y_i\mathbf{x}_i'\boldsymbol{\beta} - \exp(\mathbf{x}_i'\boldsymbol{\beta}) - \sum_{k=2}^{y_i}\ln(k)\right] \\ &\quad - \sum_{i=1}^{N}\ln\left[1+\exp(\mathbf{z}_i'\boldsymbol{\gamma})\right]\end{aligned}$$

ZIP Model with Standard Normal Link Function

Next, consider the ZIP model in which the probability φ_i is expressed by a standard normal link function: $\varphi_i = \Phi(\mathbf{z}_i'\boldsymbol{\gamma})$. The log-likelihood function is

$$\begin{aligned}\mathcal{L} &= \sum_{\{i:y_i=0\}} \ln\left\{\Phi(\mathbf{z}_i'\boldsymbol{\gamma}) + \left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right]\exp(-\exp(\mathbf{x}_i'\boldsymbol{\beta}))\right\} \\ &\quad + \sum_{\{i:y_i>0\}} \left\{ \ln\left[\left(1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right)\right] - \exp(\mathbf{x}_i'\boldsymbol{\beta}) + y_i\mathbf{x}_i'\boldsymbol{\beta} - \sum_{k=2}^{y_i}\ln(k)\right\}\end{aligned}$$

For more information about the zero-inflated Poisson regression model, see the section "Zero-Inflated Poisson Regression" on page 630.

Zero-Inflated Conway-Maxwell-Poisson Regression

In the Conway-Maxwell-Poisson regression model, the data generation process is defined as

$$P(Y_i = y_i|\mathbf{x}_i,\mathbf{z}_i) = \frac{1}{Z(\lambda_i,\nu_i)}\frac{\lambda_i^{y_i}}{(y_i!)^{\nu_i}}, \quad y_i = 0,1,2,\ldots$$

where the normalization factor is

$$Z(\lambda_i,\nu_i) = \sum_{n=0}^{\infty}\frac{\lambda_i^n}{(n!)^{\nu_i}}$$

and

$$\lambda_i = \exp(\mathbf{x}_i'\boldsymbol{\beta})$$

$$\nu_i = -\exp(\mathbf{g}_i'\delta)$$

The zero-inflated Conway-Maxwell-Poisson model can be written as

$$\begin{aligned} P(y_i|\mathbf{x}_i, \mathbf{z}_i) &= F_i + (1 - F_i)\frac{1}{Z(\lambda_i, \nu_i)}, \quad y_i = 0 \\ P(y_i|\mathbf{x}_i, \mathbf{z}_i) &= (1 - F_i)\frac{1}{Z(\lambda_i, \nu_i)}\frac{\lambda_i^{y_i}}{(y_i!)^{\nu_i}}, \quad y_i > 0 \end{aligned}$$

The conditional expectation and conditional variance of y_i are given respectively by

$$E(y_i|\mathbf{x}_i, \mathbf{z}_i) = (1 - F_i)\frac{1}{Z(\lambda, \nu)}\sum_{j=0}^{\infty}\frac{j\lambda^j}{(j!)^{\nu}}$$

$$V(y_i|\mathbf{x}_i, \mathbf{z}_i) = (1 - F_i)\frac{1}{Z(\lambda, \nu)}\sum_{j=0}^{\infty}\frac{j^2\lambda^j}{(j!)^{\nu}} - E(y_i|\mathbf{x}_i, \mathbf{z}_i)^2$$

The general form of the log-likelihood function for the Conway-Maxwell-Poisson zero-inflated model is

$$\mathcal{L} = \sum_{i=1}^{N} w_i \ln\left[P(y_i|\mathbf{x}_i, \mathbf{z}_i)\right]$$

Zero-Inflated Conway-Maxwell-Poisson Model with Logistic Link Function

For this model, the probability φ_i is expressed by using a logistic link function as

$$\varphi_i = \Lambda(\mathbf{z}_i'\boldsymbol{\gamma}) = \frac{\exp(\mathbf{z}_i'\boldsymbol{\gamma})}{1 + \exp(\mathbf{z}_i'\boldsymbol{\gamma})}$$

The log-likelihood function is

$$\begin{aligned} \mathcal{L} &= \sum_{\{i:y_i=0\}} w_i \ln\left\{\Lambda(\mathbf{z}_i'\boldsymbol{\gamma}) + \left[1 - \Lambda(\mathbf{z}_i'\boldsymbol{\gamma})\right]\frac{1}{Z(\lambda_i, \nu_i)}\right\} \\ &+ \sum_{\{i:y_i>0\}} w_i \left\{\ln\left[\left(1 - \Lambda(\mathbf{z}_i'\boldsymbol{\gamma})\right)\right] - ln(Z(\lambda, \nu)) + (y_i \ln(\lambda) - \nu \ln(y_i!)\right\} \end{aligned}$$

Zero-Inflated Conway-Maxwell-Poisson Model with Normal Link Function

For this model, the probability φ_i is specified by using the standard normal distribution function (probit function): $\varphi_i = \Phi(\mathbf{z}_i'\boldsymbol{\gamma})$.

The log-likelihood function is written as

$$\begin{aligned} \mathcal{L} &= \sum_{\{i:y_i=0\}} w_i \ln\left\{\Phi(\mathbf{z}_i'\boldsymbol{\gamma}) + \left[1 - \Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right]\frac{1}{Z(\lambda_i, \nu_i)}\right\} \\ &+ \sum_{\{i:y_i>0\}} w_i \left\{\ln\left[\left(1 - \Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right)\right] - ln(Z(\lambda, \nu)) + (y_i \ln(\lambda) - \nu \ln(y_i!)\right\} \end{aligned}$$

Zero-Inflated Negative Binomial Regression

The zero-inflated negative binomial (ZINB) model in PROC HPCOUNTREG is based on the negative binomial model that has a quadratic variance function (when DIST=NEGBIN in the MODEL or PROC HPCOUNTREG statement). The ZINB model is obtained by specifying a negative binomial distribution for the data generation process referred to earlier as Process 2:

$$g(y_i) = \frac{\Gamma(y_i + \alpha^{-1})}{y_i!\Gamma(\alpha^{-1})}\left(\frac{\alpha^{-1}}{\alpha^{-1}+\mu_i}\right)^{\alpha^{-1}}\left(\frac{\mu_i}{\alpha^{-1}+\mu_i}\right)^{y_i}$$

Thus the ZINB model is defined to be

$$\begin{aligned} P(y_i = 0|\mathbf{x}_i, \mathbf{z}_i) &= F_i + (1-F_i)\,(1+\alpha\mu_i)^{-\alpha^{-1}} \\ P(y_i|\mathbf{x}_i, \mathbf{z}_i) &= (1-F_i)\,\frac{\Gamma(y_i+\alpha^{-1})}{y_i!\Gamma(\alpha^{-1})}\left(\frac{\alpha^{-1}}{\alpha^{-1}+\mu_i}\right)^{\alpha^{-1}} \\ &\times \left(\frac{\mu_i}{\alpha^{-1}+\mu_i}\right)^{y_i}, \quad y_i > 0 \end{aligned}$$

In this case, the conditional expectation (E) and conditional variance (V) of y_i are

$$E(y_i|\mathbf{x}_i, \mathbf{z}_i) = \mu_i(1-F_i)$$

$$V(y_i|\mathbf{x}_i, \mathbf{z}_i) = E(y_i|\mathbf{x}_i, \mathbf{z}_i)\,[1 + \mu_i(F_i + \alpha)]$$

Like the ZIP model, the ZINB model exhibits overdispersion because the conditional variance exceeds the conditional mean.

ZINB Model with Logistic Link Function

In this model, the probability φ_i is given by the logistic function, namely

$$\varphi_i = \frac{\exp(\mathbf{z}_i'\boldsymbol{\gamma})}{1+\exp(\mathbf{z}_i'\boldsymbol{\gamma})}$$

The log-likelihood function is

$$\begin{aligned} \mathcal{L} &= \sum_{\{i:y_i=0\}} \ln\left[\exp(\mathbf{z}_i'\boldsymbol{\gamma}) + (1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{-\alpha^{-1}}\right] \\ &+ \sum_{\{i:y_i>0\}}\sum_{j=0}^{y_i-1} \ln(j+\alpha^{-1}) \\ &+ \sum_{\{i:y_i>0\}} \left\{-\ln(y_i!) - (y_i+\alpha^{-1})\ln(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})) + y_i\ln(\alpha) + y_i\mathbf{x}_i'\boldsymbol{\beta}\right\} \\ &- \sum_{i=1}^{N} \ln\left[1+\exp(\mathbf{z}_i'\boldsymbol{\gamma})\right] \end{aligned}$$

ZINB Model with Standard Normal Link Function

For this model, the probability φ_i is expressed by the standard normal distribution function (probit function): $\varphi_i = \Phi(\mathbf{z}_i'\boldsymbol{\gamma})$. The log-likelihood function is

$$\begin{aligned}
\mathcal{L} &= \sum_{\{i:y_i=0\}} \ln\left\{\Phi(\mathbf{z}_i'\boldsymbol{\gamma}) + \left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right](1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{-\alpha^{-1}}\right\} \\
&+ \sum_{\{i:y_i>0\}} \ln\left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right] \\
&+ \sum_{\{i:y_i>0\}} \sum_{j=0}^{y_i-1} \left\{\ln(j+\alpha^{-1})\right\} \\
&- \sum_{\{i:y_i>0\}} \ln(y_i!) \\
&- \sum_{\{i:y_i>0\}} (y_i+\alpha^{-1})\ln(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})) \\
&+ \sum_{\{i:y_i>0\}} y_i\ln(\alpha) \\
&+ \sum_{\{i:y_i>0\}} y_i\mathbf{x}_i'\boldsymbol{\beta}
\end{aligned}$$

For more information about the zero-inflated negative binomial regression model, see the section "Zero-Inflated Negative Binomial Regression" on page 634.

Parameter Naming Conventions for the RESTRICT, TEST, BOUNDS, and INIT Statements

This section describes how you can refer to the parameters in the MODEL, ZEROMODEL, and DISPMODEL statements when you use the RESTRICT, TEST, BOUNDS, or INIT statement. The following examples use the RESTRICT statement, but the same remarks apply to naming parameters when you use the TEST, BOUNDS, or INIT statement. The names of the parameters can be seen in the OUTEST= data set.

To impose a restriction on a parameter that is related to a regressor in the MODEL statement, you simply use the name of the regressor itself to refer to its associated parameter. Suppose your model is:

```
model y = x1 x2 x5;
```

where x1 through x5 are continuous variables. If you want to restrict the parameter associated with the regressor x5 to be greater than 1.7, then you should the following statement:

```
RESTRICT x5 > 1.7;
```

To impose a restriction on a parameter associated with a regressor in the ZEROMODEL statement, you can form the name of the parameter by prefixing Inf_ to the name of the regressor. Suppose your MODEL and ZEROMODEL statements are as follows:

```
model y = x1 x2 x5;
zeromodel y ~ x3 x5;
```

If you want to restrict the parameter related to the x5 regressor in the ZEROMODEL statement to be less than 1.0, then you refer to the parameter as Inf_x5 and provide the following statement:

```
RESTRICT Inf_x5 < 1.0;
```

Even though the regressor x5 appears in both the MODEL and ZEROMODEL statements, the parameter associated with x5 in the MODEL statement is, of course, different from the parameter associated with x5 in the ZEROMODEL statement. Thus, when the name of a regressor is used in a RESTRICT statement without any prefix, it refers to the parameter associated with that regressor in the MODEL statement. Meanwhile, when the name of a regressor is used in a RESTRICT statement with the prefix Inf_, it refers to the parameter associated with that regressor in the ZEROMODEL statement. The parameter associated with the intercept in the ZEROMODEL is named Inf_Intercept.

In a similar way, you can form the name of a parameter associated with a regressor in the DISPMODEL statement by prefixing Dsp_ to the name of the regressor. The parameter associated with the intercept in the DISPMODEL is named Dsp_Intercept.

Referring to Class-Level Parameters

When your MODEL includes a classification variable, you can impose restrictions on the parameters associated with each of the levels that are related to the classification variable as follows.

Suppose your classification variable is named C and that it has three levels: 0, 1, 2. Suppose your model is the following:

```
class C;
model y = x1 x2 C;
```

Adding a classification variable as a regressor to your model introduces additional parameters into your model, each of which is associated with one of the levels of the classification variable. You can form the name of the parameter associated with a particular level of your class variable by inserting the underscore character between the name of the classification variable and the value of the level. Thus, to restrict the parameter associated with level 0 of the classification variable C to always be greater than 0.7, you refer to the parameter as C_0 and provide the following statement:

```
RESTRICT C_0 > 0.7;
```

Referring to Parameters Associated with Interactions between Regressors

When a regressor in your model involves an interaction between other regressors, you can impose restrictions on the parameters associated with the interaction.

Suppose you have the following model:

```
model y = x1 x2 x3*x4;
```

You can form the name of the parameter associated with the interaction regressor x3*x4 by replacing the multiplication sign with an underscore. Thus, x3_x4 refers to the parameter that is associated with the interaction regressor x3*x4.

Referring to interactions between regressors and classification variables is handled in the same way. Suppose you have a classification variable that is named C and has three levels: 0, 1, 2. Suppose that your model is the following:

```
class C;
model y = x1 x2 C*x3;
```

The interaction between the continuous variable x3 and the classification variable C introduces three additional parameters, which are named: x3_C_0, x3_C_1, and x3_C_2. Note how, although the order of the terms in the interaction is C followed by x3, the name of the parameter associated with the interaction is formed by placing the name of the continuous variable x3 first, followed by an underscore, followed by the name of the classification variable C, followed by an underscore, and then followed by the level value. Once again, depending on the parameterization you specify in your CLASS statement, for each interaction in your model that involves a classification variable, one of the parameters associated with that interaction might be dropped from your model prior to optimization.

The name of a parameter associated with a nested interaction is formed in a slightly different way. Suppose you have a classification variable that is named C and has three levels: 0, 1, 2. Suppose that your model is the following:

```
class C;
model y = x1 x2 x3(C);
```

The nested interaction between the continuous variable x3 and the classification variable C introduces three additional parameters, which are named: x3_C__0, x3_C__1, and x3_C__2. Note how the name in each case is formed from the name of the regressor by replacing the left and right parentheses with underscores and then appending another underscore followed by the level value.

Referring to Class Level Parameters with Negative Values

When the value of a level is a negative number, you must replace the minus sign with an underscore when you form the name of the parameter that is associated with that particular level of the classification variable. For example, suppose your classification variable is named D and has four levels: –1, 0, 1, 2. Suppose your model is the following:

```
class D;
model y = x1 x2 D;
```

To restrict the parameter that is associated with level –1 of the classification variable D to always be less than 0.4, you refer to the parameter as D__1 (note that there are two underscores in this parameter name: one to connect the name of the classification variable to its value and the other to replace the minus sign in the value itself) and provide following statement:

```
RESTRICT D__1 < 0.4;
```

Dropping a Class Level Parameter to Avoid Collinearity

Depending on the parameterization you impose on your classification variable, one of the parameters associated with its levels might be dropped from your model prior to optimization in order to avoid collinearity. For example, when the default parameterization GLM is imposed, the parameter that is associated with the last level of your classification variable is dropped prior to optimization. If you attempt to impose a restriction on a dropped parameter by using the RESTRICT statement, PROC COUNTREG issues an error message in the log.

For example, suppose again that your classification variable is named C and that it has three levels: 0, 1, 2. Suppose your model is the following:

```
class C;
model y = x1 x2 C;
```

Because no additional options are specified in the CLASS statement, GLM parameterization is assumed. This means that the parameter named C_2 (which is the parameter associated with the last level of your classification variable) will be dropped from your model before the optimizer is invoked. Therefore, an error will be issued if you attempt to restrict the C_2 parameter in any way by referring to it in a RESTRICT statement. For example, the following RESTRICT statement will generate an error:

```
RESTRICT C_2 < 0.3;
```

Referring to Implicit Parameters

For certain model types, one or more implicit parameters will be added to your model prior to optimization. You can impose restrictions on these implicit parameters.

For the Poisson model for which ERRORCOMP=RANDOM is specified, PROC COUNTREG automatically adds the _Alpha parameter to your model.

If no ERRORCOMP= option is specified, for zero-inflated binomial and negative binomial models, PROC COUNTREG adds the _Alpha parameter to the model. If ERRORCOMP=RANDOM is specified for the zero-inflated binomial and negative binomial models, then PROC COUNTREG adds two implicit parameters to the model: _Alpha and _Beta.

For Conway-Maxwell Poisson models that do not include a DISPMODEL statement, the _lnNu parameter is added to the model.

Whenever your model type dictates the addition of one or more of these implicit parameters, you can impose restrictions on the implicit parameters by referring to them by name in a RESTRICT statement. For example, if your model type implies the existence of the _Alpha parameter, you can restrict _Alpha to be greater than 0.2 as follows:

```
RESTRICT _Alpha > 0.2;
```

Computational Resources

The time and memory that PROC HPCOUNTREG requires are proportional to the number of parameters in the model and the number of observations in the data set being analyzed. Less time and memory are required for smaller models and fewer observations. When PROC HPCOUNTREG is run in the high-performance distributed environment, the amount of time required is also affected by the number of nodes and the number of threads per node as specified in the PERFORMANCE statement.

The method that is chosen to calculate the variance-covariance matrix and the optimization method also affect the time and memory resources. All optimization methods available through the METHOD= option have similar memory use requirements. The processing time might differ for each method, depending on the number of iterations and functional calls needed. The data set is read into memory to save processing time. If not enough memory is available to hold the data, the HPCOUNTREG procedure stores the data in a utility file on disk and rereads the data as needed from this file, substantially increasing the execution time of the procedure. The gradient and the variance-covariance matrix must be held in memory. If the model has p parameters including the intercept, then at least $8 * (p + p * (p + 1)/2)$ bytes of memory are needed. The processing time is also a function of the number of iterations needed to converge to a solution for the model parameters. The number of iterations that are needed cannot be known in advance. You can use the MAXITER= option to limit the number of iterations that PROC HPCOUNTREG executes. You can alter the convergence criteria by using the nonlinear optimization options available in the PROC HPCOUNTREG statement. For a list of all the nonlinear optimization options, see "Optimization Control Options" on page 1080.

Covariance Matrix Types

The COVEST= option in the PROC HPCOUNTREG statement enables you to specify the estimation method for the covariance matrix. COVEST=HESSIAN estimates the covariance matrix that is based on the inverse of the Hessian matrix; COVEST=OP uses the outer product of gradients; and COVEST=QML produces the covariance matrix that is based on both the Hessian and outer product matrices. Although all three methods produce asymptotically equivalent results, they differ in computational intensity and produce results that might differ in finite samples. The COVEST=OP option provides the covariance matrix that is typically the easiest to compute. In some cases, the OP approximation is considered more efficient than the Hessian or QML approximation because it contains fewer random elements. The QML approximation is computationally the most complex because it requires both the outer product of gradients and the Hessian matrix. In most cases, the OP or Hessian approximation is preferred to QML. The need for QML approximation arises in cases where the model is misspecified and the information matrix equality does not hold. The default is COVEST=HESSIAN.

Displayed Output

PROC HPCOUNTREG produces the following displayed output.

Model Fit Summary

The "Model Fit Summary" table contains the following information:

- dependent (count) variable name
- number of observations used
- number of missing values in data set, if any
- data set name
- type of model that was fit
- parameterization for the Conway-Maxwell-Poisson model
- offset variable name, if any
- zero-inflated link function, if any
- zero-inflated offset variable name, if any
- log-likelihood value at solution
- maximum absolute gradient at solution
- number of iterations
- AIC value at solution (smaller value indicates better fit)
- SBC value at solution (smaller value indicates better fit)

A line in the "Model Fit Summary" table indicates whether the algorithm successfully converged.

Parameter Estimates

The "Parameter Estimates" table gives the estimates of the model parameters. In zero-inflated (ZI) models, estimates are also given for the ZI intercept and ZI regressor parameters, which are labeled with the prefix "Inf_". For example, the ZI intercept is labeled "Inf_intercept". If you specify "Age" as a ZI regressor, then the "Parameter Estimates" table labels the corresponding parameter estimate "Inf_Age". If you do not list any ZI regressors, then only the ZI intercept term is estimated.

If the DISPMODEL statement is specified for the Conway-Maxwell-Poisson model, the estimates are given for the dispersion intercept, and parameters are labeled with the prefix "Dsp_". For example, the dispersion model intercept is labeled "Dsp_Intercept". If you specify "Education" as a dispersion model regressor, then the "Parameter Estimates" table labels the corresponding parameter estimate "Dsp_Education". If you do not list any dispersion regressors, then only the dispersion intercept is estimated.

"_Alpha" is the negative binomial dispersion parameter. The *t* statistic that is given for "_Alpha" is a test of overdispersion.

Covariance of Parameter Estimates

If you specify the COVB option in the PROC HPCOUNTREG or MODEL statement, the HPCOUNTREG procedure displays the estimated covariance matrix, which is defined as the inverse of the information matrix at the final iteration.

Correlation of Parameter Estimates

If you specify the CORRB option in the PROC HPCOUNTREG or MODEL statement, the HPCOUNTREG procedure displays the estimated correlation matrix, which is based on the Hessian matrix used at the final iteration.

OUTPUT OUT= Data Set

The OUTPUT statement creates a new SAS data set that contains various estimates that you specify. You can request that the output data set contain the estimates of $\mathbf{x}_i'\boldsymbol{\beta}$, the expected value of the response variable, and the probability that the response variable will take the current value. In a zero-inflated model, you can also request that the output data set contain the estimates of $\mathbf{z}_i'\boldsymbol{\gamma}$, and the probability that the response is zero as a result of the zero-generating process. In a Conway-Maxwell-Poisson model, you can also request that the output data set contains estimates of $\mathbf{g}_i'\boldsymbol{\delta}$, λ, ν, μ, mode, variance and dispersion.

Except for the probability of the current value, these statistics can be computed for all observations in which the regressors are not missing, even if the response is missing. By adding observations with missing response values to the input data set, you can compute these statistics for new observations or for settings of the regressors that are not present in the data without affecting the model fit. Because of potential space limitations on the client workstation, the data set that is created by the OUTPUT statement does not contain the variables in the input data set.

OUTEST= Data Set

The OUTEST= data set is made up of at least two rows: the first row (with _TYPE_='PARM') contains each of the parameter estimates in the model, and the second row (with _TYPE_='STD') contains the standard errors for the parameter estimates in the model.

If you use the COVOUT option in the PROC HPCOUNTREG statement, the OUTEST= data set also contains the covariance matrix for the parameter estimates. The covariance matrix appears in the observations with _TYPE_='COV', and the _NAME_ variable labels the rows with the parameter names.

ODS Table Names

PROC HPCOUNTREG assigns a name to each table that it creates. You can use these names to denote the table when you use the Output Delivery System (ODS) to select tables and create output data sets. These table names are listed in Table 20.2.

Table 20.2 ODS Tables Produced in PROC HPCOUNTREG

ODS Table Name	Description	Option
ODS Tables Created by the MODEL Statement		
FitSummary	Summary of nonlinear estimation	Default
ConvergenceStatus	Convergence status	Default
ParameterEstimates	Parameter estimates	Default
CovB	Covariance of parameter estimates	COVB
CorrB	Correlation of parameter estimates	CORRB

Examples: The HPCOUNTREG Procedure

Example 20.1: High-Performance Zero-Inflated Poisson Model

This example shows the use of the HPCOUNTREG procedure with an emphasis on large data set processing and the performance improvements that are achieved by executing in the high-performance distributed environment.

The following DATA step generates one million replicates from the zero-inflated Poisson (ZIP) model. The model contains seven variables and three variables that correspond to the zero-inflated process.

```
data simulate;
   call streaminit(12345);
   array vars x1-x7;
   array zero_vars z1-z3;

   array parms{7}  (.3 .4 .2 .4 -.3 -.5 -.3);
   array zero_parms{3} (-.6 .3 .2);

   intercept=2;
   z_intercept=-1;
   theta=0.5;

   do i=1 to 1000000;
      sum_xb=0;
      sum_gz=0;
      do j=1 to 7;
         vars[j]=rand('NORMAL',0,1);
         sum_xb=sum_xb+parms[j]*vars[j];
      end;
      mu=exp(intercept+sum_xb);
      y_p=rand('POISSON', mu);

      do j=1 to 3;
```

```
            zero_vars[j]=rand('NORMAL',0,1);
            sum_gz = sum_gz+zero_parms[j]*zero_vars[j];
         end;
         z_gamma = z_intercept+sum_gz;
         pzero = cdf('LOGISTIC',z_gamma);
         cut=rand('UNIFORM');
         if cut<pzero then y_p=0;
         output;
      end;
   keep y_p x1-x7 z1-z3;
   run;
```

The following statements estimate a zero-inflated Poisson model.

```
option set=GRIDHOST="&GRIDHOST";
option set=GRIDINSTALLLOC="&GRIDINSTALLLOC";

proc hpcountreg data=simulate dist=zip;
   performance nthreads=2 nodes=1 details
           host="&GRIDHOST" install="&GRIDINSTALLLOC";
   model y_p=x1-x7;
   zeromodel y_p ~ z1-z3;
run;
```

The model is executed in the distributed computing environment on two threads and only one node. These settings are used to obtain a hypothetical environment that might resemble running the HPCOUNTREG procedure on a desktop workstation with a dual-core CPU. To run these statements successfully, you need to set the macro variables GRIDHOST and GRIDINSTALLLOC to resolve to appropriate values, or you can replace the references to the macro variables in the example with the appropriate values. Output 20.1.1 shows the "Performance Information" table for this hypothetical scenario.

Output 20.1.1 Performance Information with One Node and One Thread

Performance Information	
Host Node	<< your grid host >>
Install Location	<< your grid install location >>
Execution Mode	Distributed
Number of Compute Nodes	1
Number of Threads per Node	2

Output 20.1.2 shows the results for the zero-inflated Poisson model. The "Model Fit Summary" table shows detailed information about the model and indicates that all one million observations were used to fit the model. All parameter estimates in the "Parameter Estimates" table are highly significant and correspond to their theoretical values set during the data generating process. The optimization of the model that contains one million observations took 40.77 seconds.

Output 20.1.2 Zero-Inflated Poisson Model Execution on One Node and Two Threads

Model Fit Summary	
Dependent Variable	y_p
Number of Observations	1000000
Data Set	WORK.SIMULATE
Model	ZIP
ZI Link Function	Logistic
Log Likelihood	-2215238
Maximum Absolute Gradient	2.044E-8
Number of Iterations	7
Optimization Method	Newton-Raphson
AIC	4430500
SBC	4430642

Convergence criterion (FCONV=2.220446E-16) satisfied.

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	2.0005	0.000492	4069.80	<.0001
x1	1	0.2995	0.000352	850.17	<.0001
x2	1	0.3998	0.000353	1132.23	<.0001
x3	1	0.2008	0.000352	570.27	<.0001
x4	1	0.3994	0.000353	1132.85	<.0001
x5	1	-0.2995	0.000353	-848.95	<.0001
x6	1	-0.5000	0.000353	-1414.9	<.0001
x7	1	-0.3002	0.000352	-852.14	<.0001
Inf_Intercept	1	-0.9993	0.002521	-396.45	<.0001
Inf_z1	1	-0.6024	0.002585	-233.02	<.0001
Inf_z2	1	0.2976	0.002454	121.25	<.0001
Inf_z3	1	0.1974	0.002430	81.20	<.0001

Procedure Task Timing

Task	Seconds	Percent
Reading and Levelizing Data	0.32	0.78%
Communication to Client	0.03	0.06%
Optimization	40.77	98.52%
Post-Optimization	0.26	0.63%

In the following statements, the PERFORMANCE statement is modified to use a grid with 10 nodes, with each node capable of spawning eight threads:

```
proc hpcountreg data=simulate dist=zip;
   performance nthreads=8 nodes=10 details
               host="&GRIDHOST" install="&GRIDINSTALLLOC";
   model y_p=x1-x7;
   zeromodel y_p ~ z1-z3;
run;
```

Because the two models being estimated are identical, it is reasonable to expect that Output 20.1.2 and Output 20.1.3 would show the same results. However, you can see a significant difference in performance between the two models. The second model, which was run on a grid that used 10 nodes with eight threads each, took only 3.54 seconds instead of 40.77 seconds to optimize.

In certain circumstances, you might observe slight numerical differences in the results, depending on the number of nodes and threads involved. This happens because the order in which partial results are accumulated can make a difference in the final result, owing to the limits of numerical precision and the propagation of error in numerical computations.

Output 20.1.3 Zero-Inflated Poisson Model Execution on 10 Nodes with Eight Threads Each

The HPCOUNTREG Procedure

Model Fit Summary	
Dependent Variable	y_p
Number of Observations	1000000
Data Set	WORK.SIMULATE
Model	ZIP
ZI Link Function	Logistic
Log Likelihood	-2215238
Maximum Absolute Gradient	2.0608E-8
Number of Iterations	7
Optimization Method	Newton-Raphson
AIC	4430500
SBC	4430642

Convergence criterion (FCONV=2.220446E-16) satisfied.

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	2.0005	0.000492	4069.80	<.0001
x1	1	0.2995	0.000352	850.17	<.0001
x2	1	0.3998	0.000353	1132.23	<.0001
x3	1	0.2008	0.000352	570.27	<.0001
x4	1	0.3994	0.000353	1132.85	<.0001
x5	1	-0.2995	0.000353	-848.95	<.0001
x6	1	-0.5000	0.000353	-1414.9	<.0001
x7	1	-0.3002	0.000352	-852.14	<.0001
Inf_Intercept	1	-0.9993	0.002521	-396.45	<.0001
Inf_z1	1	-0.6024	0.002585	-233.02	<.0001
Inf_z2	1	0.2976	0.002454	121.25	<.0001
Inf_z3	1	0.1974	0.002430	81.20	<.0001

Procedure Task Timing

Task	Seconds	Percent
Reading and Levelizing Data	0.02	0.61%
Communication to Client	0.06	1.44%
Optimization	3.54	90.99%
Post-Optimization	0.27	6.96%

As this example suggests, increasing the number of nodes and the number of threads per node improves performance significantly. When you use the parallelism afforded by a high-performance distributed environment, you can see an even more dramatic reduction in the time required for the optimization as the number of observations in the data set increases. When the data set is extremely large, the computations might not even be possible in some cases, given the typical memory resources and computational constraints of a desktop computer. Under such circumstances the high-performance distributed environment becomes a necessity.

References

Cameron, A. C., and Trivedi, P. K. (1986). "Econometric Models Based on Count Data: Comparisons and Applications of Some Estimators and Tests." *Journal of Applied Econometrics* 1:29–53.

Cameron, A. C., and Trivedi, P. K. (1998). *Regression Analysis of Count Data*. Cambridge: Cambridge University Press.

Conway, R. W., and Maxwell, W. L. (1962). "A Queuing Model with State Dependent Service Rates." *Journal of Industrial Engineering* 12:132–136.

Guikema, S. D., and Coffelt, J. P. (2008). "A Flexible Count Data Regression Model for Risk Analysis." *Risk Analysis* 28:213–223.

Lambert, D. (1992). "Zero-Inflated Poisson Regression with an Application to Defects in Manufacturing." *Technometrics* 34:1–14.

LaMotte, L. R. (1994). "A Note on the Role of Independence in *t* Statistics Constructed from Linear Statistics in Regression Models." *American Statistician* 48:238–240.

Long, J. S. (1997). *Regression Models for Categorical and Limited Dependent Variables*. Thousand Oaks, CA: Sage Publications.

Lord, D., Guikema, S. D., and Geedipally, S. R. (2008). "Application of the Conway-Maxwell-Poisson Generalized Linear Model for Analyzing Motor Vehicle Crashes." *Accident Analysis and Prevention* 40:1123–1134.

Shmueli, G., Minka, T. P., Kadane, J. B., Borle, S., and Boatwright, P. (2005). "A Useful Distribution for Fitting Discrete Data: Revival of the Conway-Maxwell-Poisson Distribution." *Journal of the Royal Statistical Society, Series C* 54:127–142.

Chapter 21
The HPPANEL Procedure

Contents

Overview: HPPANEL Procedure **1116**
Getting Started: HPPANEL Procedure **1117**
Syntax: HPPANEL Procedure **1119**
Functional Summary 1119
PROC HPPANEL Statement 1120
ID Statement 1121
MODEL Statement 1121
OUTPUT Statement 1122
PERFORMANCE Statement 1123
RESTRICT Statement 1123
TEST Statement 1124
Details: HPPANEL Procedure **1125**
Specifying the Input Data 1125
Specifying the Regression Model 1125
Specifying the Number of Nodes and Number of Threads 1125
Unbalanced Data 1126
One-Way Fixed-Effects Model 1126
Two-Way Fixed-Effects Model 1127
Balanced Panels 1128
Unbalanced Panels 1129
One-Way Random-Effects Model 1130
Two-Way Random-Effects Model 1130
Between Estimators 1132
Pooled Estimator 1132
Linear Hypothesis Testing 1132
Specification Tests 1133
OUTPUT OUT= Data Set 1134
OUTEST= Data Set 1134
Printed Output 1135
ODS Table Names 1135
Example: HPPANEL Procedure **1136**
Example 21.1: One-Way Random-Effects High-Performance Model 1136
References **1141**

Overview: HPPANEL Procedure

The HPPANEL procedure is a high-performance version of the PANEL procedure in SAS/ETS software. Both procedures analyze a class of linear econometric models that commonly arise when time series and cross-sectional data are combined. This type of data on time series cross-sectional bases is often referred to as panel data. Typical examples of panel data include observations over time about households, countries, firms, trade, and so on. For example, in the case of survey data about household income, the panel is created by repeatedly surveying the same households in different time periods (years).

Unlike the PANEL procedure (which can be run only on an individual workstation), the HPPANEL procedure takes advantage of a computing environment that enables it to distribute the optimization task among one or more nodes. Running on one node is called single-machine, and running on more than one node is called distributed mode. In addition, each node (whether in single-machine mode or in distributed mode) can use one or more threads to carry out the optimization on its subset of the data. When several nodes are used and each node uses several threads to carry out its part of the work, the result is a highly parallel computation that provides a dramatic gain in performance.

NOTE: Distributed mode requires SAS High-Performance Econometrics.

You can use the HPPANEL procedure to read and write data in distributed form and perform analyses in distributed mode or in single-machine mode. For more information about how to affect the execution mode of SAS high-performance analytical procedures, see the section "Processing Modes" on page 62 in Chapter 3, "Shared Concepts and Topics."

The HPPANEL procedure is specifically designed to operate in the high-performance distributed mode. By default, PROC HPPANEL performs computations in multiple threads.

The panel data models can be grouped into several categories that depend on the structure of the error term. The HPPANEL procedure uses the following error structures and the corresponding methods to analyze data:

- one-way and two-way models
- fixed-effects and random-effects models

A one-way model depends only on the cross section to which the observation belongs. A two-way model depends on both the cross section and the time period to which the observation belongs.

Apart from the possible one-way or two-way nature of the effect, the other dimension of difference between the possible specifications is the nature of the cross-sectional or time-series effect. The models are referred to as fixed-effects models if the effects are nonrandom and as random-effects models otherwise.

If the effects are fixed, the models are essentially regression models that have dummy variables that correspond to the specified effects. For fixed-effects models, ordinary least squares (OLS) estimation is the best linear unbiased estimator. Random-effects models use a two-stage approach: In the first stage, variance components are calculated by using methods described by Fuller and Battese (1974); Wansbeek and Kapteyn (1989); Wallace and Hussain (1969); Nerlove (1971). In the second stage, variance components are used to standardize the data, and ordinary least squares (OLS) regression is performed.

Getting Started: HPPANEL Procedure

The following statements use the cost function data from Greene (1990) to estimate the variance components model. The variable Production is the log of output in millions of kilowatt-hours, and the variable Cost is the log of cost in millions of dollars. See Greene (1990) for details.

```
data greene;
   input firm year production cost @@;
datalines;
1 1955   5.36598   1.14867  1 1960   6.03787   1.45185
1 1965   6.37673   1.52257  1 1970   6.93245   1.76627
2 1955   6.54535   1.35041  2 1960   6.69827   1.71109
2 1965   7.40245   2.09519  2 1970   7.82644   2.39480
3 1955   8.07153   2.94628  3 1960   8.47679   3.25967

   ... more lines ...
```

You decide to fit the following model to the data,

$$C_{it} = \text{Intercept} + \boldsymbol{\beta} P_{it} + v_i + e_t + \epsilon_{it} \quad \text{for } i = 1, \ldots, N \text{ and } t = 1, \ldots, T$$

where C_{it} and P_{it} represent the cost and production; and v_i, e_t, and ϵ_{it} are the cross-sectional, time series, and error variance components, respectively.

If you assume that the time and cross-sectional effects are random, four possible estimators are left for the variance components. The following statements choose the Fuller-Battese method to fit this model:

```
proc hppanel data=greene;
   model cost = production / rantwo vcomp = fb;
   id firm year;
   performance nodes=0 nthreads=2;
run;
```

The output of the HPPANEL procedure is shown in Output 21.1.

Figure 21.1 Two-Way Random Effects Results

The HPPANEL Procedure

Model Information	
Data Source	GREENE
Response Variable	cost
Model	RANTWO
Variance Component	FULLER
Execution Mode	Single-Machine

Fit Statistics	
Sum of Squared Error	0.348082
Degrees of Freedom	22
Mean Squared Error	0.015822
Root Mean Squared Error	0.125785
R-Square	0.813624

Variance Component Estimates	
Variance Component for Cross Sections	0.0469
Variance Component for Time Series	0.00906
Variance Component for Error	0.00875

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	-2.99992	0.64778	-4.63	<.0001
production	1	0.74660	0.07618	9.80	<.0001

Printed first is the model description, which reports the method used for estimation and the method used for estimating error components. Printed next is the fit statistics table, and then the variance components estimates. Finally, the table of regression parameter estimates shows the estimates, standard errors, and *t* tests.

Syntax: HPPANEL Procedure

The following statements are available in the HPPANEL procedure:

```
PROC HPPANEL options ;
  ID cross-section-id time-series-id ;
  MODEL response = regressors < /options > ;
  RESTRICT equation1< ,equation2. . . > ;
  TEST equation < ,equation2. . . >< / options > ;
  OUTPUT OUT=SAS-data-set < output-options > ;
  PERFORMANCE < performance-options > ;
```

The ID and MODEL statements are required.

The following sections provide a functional summary of statements and options, then describe the PROC HPPANEL statement, and then describe the other statements in alphabetical order.

Functional Summary

Table 21.1 summarizes the statements and options that you can use in the HPPANEL procedure.

Table 21.1 Functional Summary

Description	Statement	Option
Data Set Options		
Includes correlations in the OUTEST= data set	HPPANEL	CORROUT
Includes covariances in the OUTEST= data set	HPPANEL	COVOUT
Specifies the input data set	HPPANEL	DATA=
Specifies the name of an output SAS data set	OUTPUT	OUT=
Writes parameter estimates to an output data set	HPPANEL	OUTEST=
Variable Role Options		
Specifies the cross-sectional and time ID variables	ID	
Printing Control Options		
Prints correlations of the estimates	HPPANEL	CORRB
Prints covariances of the estimates	HPPANEL	COVB
Suppresses printed output	HPPANEL	NOPRINT
Prints fixed effects	MODEL	PRINTFIXED
Performs tests of linear hypotheses	TEST	
Model Estimation Options		
Estimates the between-groups model	MODEL	BTWNG
Estimates the between-time-periods model	MODEL	BTWNT

Description	Statement	Option
Estimates the one-way fixed-effects model	MODEL	FIXONE
Estimates the one-way fixed-effects model with respect to time	MODEL	FIXONETIME
Estimates the two-way fixed-effects model	MODEL	FIXTWO
Suppresses the intercept term	MODEL	NOINT
Estimates the pooled regression model	MODEL	POOLED
Estimates the one-way random-effects model	MODEL	RANONE
Estimates the two-way random-effects model	MODEL	RANTWO
Specifies the method for the variance components estimator	MODEL	VCOMP=
Specifies linear equality restrictions on the parameters	RESTRICT	
Specifies which tests to perform	TEST	WALD, LM, LR

PROC HPPANEL Statement

PROC HPPANEL *options* **;**

The HPPANEL statement invokes the HPPANEL procedure.

You can specify the following *options*:

DATA=*SAS-data-set*
names the input data set. Only one observation is allowed for each cross section and time period. If you omit the DATA= option, PROC HPPANEL uses the most recently created SAS data set.

CORRB
prints the matrix of estimated correlations between the parameter estimates.

COVB
prints the matrix of estimated covariances between the parameter estimates.

NOPRINT
suppresses the normal printed output.

OUTEST=*SAS-data-set*
names an output data set to contain the parameter estimates. When the OUTEST= option is not specified, the OUTEST= data set is not created. For more information about the structure of the OUTEST= data set, see the section "OUTEST= Data Set" on page 1134.

OUTCOV
COVOUT
writes the standard errors and covariance matrix of the parameter estimates to the OUTEST= data set. For more information, see the section "OUTEST= Data Set" on page 1134.

OUTCORR

CORROUT

writes the correlation matrix of the parameter estimates to the OUTEST= data set. For more information, see the section "OUTEST= Data Set" on page 1134.

In addition, you can specify any of the following MODEL statement options in the PROC HPPANEL statement: FIXONE, FIXONETIME, FIXTWO, RANONE, RANTWO, NOINT, PRINTFIXED, and VCOMP=. Specifying these options in the PROC HPPANEL statement is equivalent to specifying them in the MODEL statement. For a complete description of each of these options, see the section "MODEL Statement" on page 1121.

ID Statement

ID *cross-section-id time-series-id* **;**

The ID statement specifies variables in the input data set that identify the cross section and the time period for each observation. The ID statement is required. Unlike the PANEL procedure, the HPPANEL procedure does not require the data set to be sorted.

MODEL Statement

MODEL *response* = *regressors* < / *options* > **;**

The MODEL statement specifies the regression model, the error structure that is assumed for the regression residuals, and the estimation technique to be used. The *response* variable is regressed on the independent variables (*regressors*). You can specify only one MODEL statement and only one *response*.

You specify the error structure and estimation technique by including one of the following *options* after a slash (/):

BTWNG

estimates the between-groups model.

BTWNT

estimates the between-time-periods model.

FIXONE

estimates a one-way fixed-effects model, which corresponds to cross-sectional effects.

FIXONETIME

estimates a one-way fixed-effects model, which corresponds to time effects.

FIXTWO

estimates a two-way fixed-effects model.

POOLED

estimates the pooled regression model.

RANONE
estimates a one-way random-effects model.

RANTWO
estimates a two-way random-effects model.

By default, a FIXONE estimation is performed.

You can also specify the following *options* after the slash:

NOINT
suppresses the intercept parameter from the model.

PRINTFIXED
prints the fixed effects.

VCOMP=FB | NL | WH | WK
specifies the type of variance component estimator to use. You can specify the following values:

FB	requests the Fuller-Battese estimator.
WK	requests the Wansbeek-Kapteyn estimator.
WH	requests the Wallace-Hussain estimator.
NERLOVE	requests the Nerlove estimator.

By default, VCOMP=WK for both balanced and unbalanced data.

OUTPUT Statement

OUTPUT OUT=*SAS-data-set* < *output-options* > **;**

The OUTPUT statement creates a new SAS data set to contain variables that are specified by the COPYVAR option, the cross-sectional ID (_CSID_), and the time period (_TSID_). This data set also contains the predicted value and the residual if they are specified by *output-options*. When the response values are missing for the observation, all output estimates except the residual are still computed as long as none of the explanatory variables are missing. You can specify only one OUTPUT statement.

You must specify the OUT= option:

OUT=*SAS-data-set*
names the output data set.

You can specify one or more of the following *output-options*:

COPYVAR=(*SAS-variable-names***)**
COPYVARS=(*SAS-variable-names***)**
adds SAS variables to the output data set.

PREDICTED
outputs estimates of predicted dependent variables.

RESIDUAL
outputs estimates of residuals.

PERFORMANCE Statement

PERFORMANCE < *performance-options* > **;**

The PERFORMANCE statement specifies *performance-options* to control the multithreaded and distributed computing environment and requests detailed performance results of the HPPANEL procedure. You can also use the PERFORMANCE statement to control whether the HPPANEL procedure executes in single-machine or distributed mode. You can specify the following *performance-options*:

DETAILS
requests a table that shows a timing breakdown of the procedure steps.

NODES=*n*
specifies the number of nodes in the distributed computing environment, provided that the data are not processed alongside the database.

NTHREADS=*n*
specifies the number of threads for analytic computations and overrides the SAS system option THREADS | NOTHREADS. If you do not specify the NTHREADS= option, PROC HPPANEL creates one thread per CPU for the analytic computations.

The PERFORMANCE statement is documented further in the section "PERFORMANCE Statement" on page 87 in Chapter 3, "Shared Concepts and Topics."

RESTRICT Statement

RESTRICT *equation1* <,*equation2*. . . > **;**

The RESTRICT statement specifies linear equality restrictions on the parameters in the MODEL statement. There can be as many unique restrictions as the number of parameters in the MODEL statement. Multiple RESTRICT statements are understood as joint restrictions on the model's parameters.

Currently, PROC HPPANEL only supports linear equality restrictions. Restriction expressions can be composed only of algebraic operations that involve the addition symbol (+), subtraction symbol (–), and multiplication symbol (*).

The following statements illustrate the use of the RESTRICT statement:

```
proc hppanel;
   id csid tsid;
   model y = x1 x2 x3;
   restrict x1 = 0, x2 * .5 + 2 * x3= 0;
   restrict  x2 = 0, intercept = 0;
run;
```

A RESTRICT statement cannot include a division sign in its formulation. As in the preceding example, you can obtain restrictions on the intercept by using the keyword INTERCEPT.

TEST Statement

TEST *equation1* <*,equation2...*> <*/ options*> **;**

The TEST statement performs Wald, LaGrange multiplier, and likelihood ratio tests of linear hypotheses about the regression parameters in the MODEL statement. Each *equation* specifies a linear hypothesis to be tested. Currently, only linear equality restrictions and tests are permitted in PROC HPPANEL. Test expressions can be composed only of algebraic operations that involve the addition symbol (+), subtraction symbol (–), and multiplication symbol (*). All hypotheses in one TEST statement are tested jointly. Variable names in the equations must correspond to regressors in the preceding MODEL statement, and each name represents the coefficient of the corresponding regressor. In the equality restrictions, you can use the keyword INTERCEPT to refer to the coefficient of the intercept.

You can specify the following *options* after the slash (/):

ALL
specifies Wald, LaGrange multiplier, and likelihood ratio tests.

WALD
specifies the Wald test.

LM
specifies the LaGrange multiplier test.

LR
specifies the likelihood ratio test.

By default, the Wald test is performed.

The following statements illustrate the use of the TEST statement:

```
proc hppanel;
   id csid tsid;
   model y = x1 x2 x3;
   test x1 = 0, x2 * .5 + 2 * x3 = 0;
   test intercept = 0, x3 = 0;
run;
```

The first test investigates the joint hypothesis that

$$\beta_1 = 0$$

and

$$0.5\beta_2 + 2\beta_3 = 0$$

Details: HPPANEL Procedure

Specifying the Input Data

The HPPANEL procedure is similar to other regression procedures in SAS. Suppose you want to regress the variable Y on regressors X1 and X2. Cross sections are identified by the variable State, and time periods are identified by the variable Date. Unlike the PANEL procedure, the HPPANEL procedure does not require the data set to be sorted. To invoke the HPPANEL procedure, you must specify the cross section and time series variables in an ID statement. The following statements shows the correct syntax:

```
proc hppanel data=a;
   id state date;
   model y = x1 x2;
   performance nodes=2 nthreads=4;
run;
```

Specifying the Regression Model

The MODEL statement in PROC HPPANEL is specified like the MODEL statement in other SAS regression procedures: the dependent variable is listed first, followed by an equal sign, followed by the list of regressor variables, as shown in the following statements:

```
proc hppanel data=a;
   id state date;
   model y = x1 x2;
   performance nodes=2 nthreads=4;
run;
```

Specifying the Number of Nodes and Number of Threads

The PERFORMANCE statement in PROC HPPANEL is specified like the PERFORMANCE statement in other SAS high-performance procedures. The following statements execute the model in the distributed computing environment with two threads and four nodes:

```
proc hppanel data=a;
   id state date;
   model y = x1 x2;
   performance nodes=2 nthreads=4;
run;
```

The major advantage of using PROC HPPANEL is that you can incorporate a model for the structure of the random errors. It is important to consider what type of error structure model is appropriate for your data and to specify the corresponding option in the MODEL statement.

The error structure options supported by the HPPANEL procedure are FIXONE, FIXONETIME, FIXTWO, RANONE, and RANTWO. For more information about these methods and the error structures they assume, see the following sections. The following statements fit a Fuller-Battese one-way random-effects model:

```
proc hppanel data=a;
   id state date;
   model y = x1 x2 / ranone vcomp=fb;
   performance nodes=0 nthreads=1;
run;
```

To aid in model specification within this class of models, PROC HPPANEL provides one specification test statistic, the Hausman *m* statistic, which provides information about the appropriateness of the random-effects specification. The *m* statistic is based on the idea that, under the null hypothesis of no correlation between the effects variables and the regressors, ordinary least squares (OLS) and generalized least squares (GLS) are consistent. However, OLS is inefficient. Hence, a test can be based on the result that the covariance between an efficient estimator and its difference from an inefficient estimator is 0. Rejection of the null hypothesis might suggest that the fixed-effects model is more appropriate.

The HPPANEL procedure also provides the Buse R-square measure. This number is interpreted as a measure of the proportion of the transformed sum of squares of the dependent variable that is attributable to the influence of the independent variables. For OLS estimation, the Buse R-square measure is equivalent to the usual R-square measure.

Unbalanced Data

The HPPANEL procedure can process data that have different numbers of time series observations across different cross sections. The missing time series observations are recognized by the absence of time series ID variable values in some of the cross sections in the input data set. Moreover, if an observation that has a particular time series ID value and cross-sectional ID value is present in the input data set but one or more of the model variables are missing, that time series point is treated as missing for that cross section.

One-Way Fixed-Effects Model

The specification for the one-way fixed-effects model is

$$u_{it} = \gamma_i + \epsilon_{it}$$

where the γ_i are nonrandom parameters to be estimated.

Let $\mathbf{Q_0} = \mathrm{diag}(\mathbf{E}_{T_i})$, with $\bar{\mathbf{J}}_{T_i} = \mathbf{J}_{T_i} / T_i$ and $\mathbf{E}_{T_i} = \mathbf{I}_{T_i} - \bar{\mathbf{J}}_{T_i}$, where $\mathbf{J}_{T_i}$ is a matrix of T_i ones.

The matrix $\mathbf{Q_0}$ represents the within transformation. In the one-way model, the within transformation is the conversion of the raw data to deviations from a cross section's mean. The vector $\tilde{\mathbf{x}}_{it}$ is a row of the general matrix $\mathbf{X}_s$, where the subscripted *s* implies that the constant (column of ones) is missing.

Let $\tilde{\mathbf{X}}_s = \mathbf{Q}_0\mathbf{X}_s$ and $\tilde{\mathbf{y}} = \mathbf{Q}_0\mathbf{y}$. The estimator of the slope coefficients is given by

$$\tilde{\beta}_s = (\tilde{\mathbf{X}}_s'\tilde{\mathbf{X}}_s)^{-1}\tilde{\mathbf{X}}_s'\tilde{\mathbf{y}}$$

After the slope estimates have been calculated, the estimation of an intercept or the cross-sectional fixed effects is handled as follows. First, you obtain the cross-sectional effects:

$$\gamma_i = \bar{y}_{i\cdot} - \tilde{\beta}_s\bar{x}_{i\cdot} \quad \text{for} \quad i = 1 \ldots N$$

If the NOINT option is specified, then the dummy variables' coefficients are set equal to the fixed effects. If you want an intercept, then the *i*th dummy variable is obtained from the following expression:

$$D_i = \gamma_i - \gamma_N \quad \text{for} \quad i = 1 \ldots N-1$$

The intercept is the *N*th fixed effect γ_N.

The within-model sum of squared errors is

$$\text{SSE} = \sum_{i=1}^{N}\sum_{t=1}^{T_i}(y_{it} - \gamma_i - \mathbf{X}_s\tilde{\beta}_s)^2$$

The estimated error variance can be written as

$$\hat{\sigma}_\epsilon^2 = \text{SSE}/(M - N - (K-1))$$

Alternatively, an equivalent way to express the error variance is

$$\hat{\sigma}_\epsilon^2 = \tilde{\mathbf{u}}'\mathbf{Q}_0\tilde{\mathbf{u}}/(M - N - (K-1))$$

where the residuals $\tilde{\mathbf{u}}$ are given by $\tilde{\mathbf{u}} = (\mathbf{I}_M - \mathbf{j}_M\mathbf{j}'_M/M)(\mathbf{y} - \mathbf{X}_s\tilde{\beta}_s)$ if there is an intercept and by $\tilde{\mathbf{u}} = (\mathbf{y} - \mathbf{X}_s\tilde{\beta}_s)$ if there is not. The drawback is that the formula changes (but the results do not) with the inclusion of a constant.

The variance covariance matrix of $\tilde{\beta}_s$ is given by

$$\text{Var}\left[\tilde{\beta}_s\right] = \hat{\sigma}_\epsilon^2(\tilde{\mathbf{X}}_s'\tilde{\mathbf{X}}_s)^{-1}$$

The covariance of the dummy variables and the dummy variables with the $\tilde{\beta}_s$ depends on whether the intercept is included in the model. For more information, see the section "One-Way Fixed-Effects Model" on page 1833.

Alternatively, the FIXONETIME model option estimates a one-way model in which the heterogeneity comes from time effects. This option is analogous to re-sorting the data by time and then by cross section, and then running a FIXONE model. The advantage of using the FIXONETIME option is that sorting is avoided and the model remains labeled correctly.

Two-Way Fixed-Effects Model

The specification for the two-way fixed-effects model is

$$u_{it} = \gamma_i + \alpha_t + \epsilon_{it}$$

where the γ_i and α_t are nonrandom parameters to be estimated.

If you do not specify the NOINT option (which suppresses the intercept) in the MODEL statement, the estimates for the fixed effects are reported under the restriction that $\gamma_N = 0$ and $\alpha_T = 0$. If you specify the NOINT option to suppress the intercept, only the restriction $\alpha_T = 0$ is imposed.

Balanced Panels

Assume that the data are balanced (for example, all cross sections have T observations). Then you can write

$$\tilde{y}_{it} = y_{it} - \bar{y}_{i\cdot} - \bar{y}_{\cdot t} + \bar{\bar{y}}$$

$$\tilde{\mathbf{x}}_{it} = \mathbf{x}_{it} - \bar{\mathbf{x}}_{i\cdot} - \bar{\mathbf{x}}_{\cdot t} + \bar{\bar{\mathbf{x}}}$$

where the symbols are as follows:

- y_{it} and $\mathbf{x}_{it}$ are the dependent variable (a scalar) and the explanatory variables (a vector whose columns are the explanatory variables, not including a constant), respectively
- $\bar{y}_{i\cdot}$ and $\bar{\mathbf{x}}_{i\cdot}$ are cross section means
- $\bar{y}_{\cdot t}$ and $\bar{\mathbf{x}}_{\cdot t}$ are time means
- $\bar{\bar{y}}$ and $\bar{\bar{\mathbf{x}}}$ are the overall means

The two-way fixed-effects model is simply a regression of $\tilde{y}_{it}$ on $\tilde{\mathbf{x}}_{it}$. Therefore, the two-way β is given by

$$\tilde{\beta}_s = \left(\tilde{\mathbf{X}}'\tilde{\mathbf{X}}\right)^{-1}\tilde{\mathbf{X}}'\tilde{\mathbf{y}}$$

The following calculations of cross-sectional dummy variables, time dummy variables, and intercepts are similar to how they are calculated in the one-way model:

First, you obtain the net cross-sectional and time effects. Denote the cross-sectional effects by γ and the time effects by α. These effects are calculated from the following relations:

$$\hat{\gamma}_i = \left(\bar{y}_{i\cdot} - \bar{\bar{y}}\right) - \tilde{\beta}_s\left(\bar{x}_{i\cdot} - \bar{\bar{x}}\right)$$

$$\hat{\alpha}_t = \left(\bar{y}_{\cdot t} - \bar{\bar{y}}\right) - \tilde{\beta}_s\left(\bar{x}_{\cdot t} - \bar{\bar{x}}\right)$$

Use the superscript C and T to denote the cross-sectional dummy variables and time dummy variables, respectively. Under the NOINT option, the following equations produce the dummy variables:

$$D_i^C = \hat{\gamma}_i + \hat{\alpha}_T$$

$$D_t^T = \hat{\alpha}_t - \hat{\alpha}_T$$

When an intercept is specified, the equations for dummy variables and intercept are

$$D_i^C = \hat{\gamma}_i - \hat{\gamma}_N$$

$$D_t^T = \hat{\alpha}_t - \hat{\alpha}_T$$

$$\text{Intercept} = \hat{\gamma}_N + \hat{\alpha}_T$$

The sum of squared errors is

$$\text{SSE} = \sum_{i=1}^{N}\sum_{t=1}^{T_i}(y_{it} - \gamma_i - \alpha_t - \mathbf{X}_s\tilde{\beta}_s)^2$$

The estimated error variance is

$$\hat{\sigma}_\epsilon^2 = \mathrm{SSE}/(M - N - T - (K - 1))$$

With or without a constant, the covariance matrix of $\tilde{\beta}_s$ is given by

$$\mathrm{Var}\left[\tilde{\beta}_s\right] = \hat{\sigma}_\epsilon^2(\tilde{\mathbf{X}}_s'\tilde{\mathbf{X}}_s)^{-1}$$

For information about the covariance matrix that is related to dummy variables, see the section "Two-Way Fixed-Effects Model" on page 1835.

Unbalanced Panels

Let $\mathbf{X}_*$ and $\mathbf{y}_*$ be the independent and dependent variables, respectively, that are arranged by time and by cross section within each time period. (Note that the input data set that the PANEL procedure uses must be sorted by cross section and then by time within each cross section.) Let M_t be the number of cross sections that are observed in year t, and let $\sum_t M_t = M$. Let $\mathbf{D}_t$ be the $M_t \times N$ matrix that is obtained from the $N \times N$ identity matrix from which rows that correspond to cross sections that are not observed at time t have been omitted. Consider

$$\mathbf{Z} = (\mathbf{Z}_1, \mathbf{Z}_2)$$

where $\mathbf{Z}_1 = (\mathbf{D}_1', \mathbf{D}_2', \ldots, \mathbf{D}_T')'$ and $\mathbf{Z}_2 = \mathrm{diag}(\mathbf{D}_1\mathbf{j}_N, \mathbf{D}_2\mathbf{j}_N, \ldots, \mathbf{D}_T\mathbf{j}_N)$. The matrix $\mathbf{Z}$ contains the dummy variable structure for the two-way model.

Let

$$\begin{aligned}
\mathbf{\Delta}_N &= \mathbf{Z}_1'\mathbf{Z}_1 \\
\mathbf{\Delta}_T &= \mathbf{Z}_2'\mathbf{Z}_2 \\
\mathbf{A} &= \mathbf{Z}_2'\mathbf{Z}_1 \\
\bar{\mathbf{Z}} &= \mathbf{Z}_2 - \mathbf{Z}_1\Delta_N^{-1}\mathbf{A}' \\
\mathbf{Q} &= \Delta_T - \mathbf{A}\Delta_N^{-1}\mathbf{A}' \\
\mathbf{P} &= (\mathbf{I}_M - \mathbf{Z}_1\Delta_N^{-1}\mathbf{Z}_1') - \bar{\mathbf{Z}}\mathbf{Q}^{-1}\bar{\mathbf{Z}}'
\end{aligned}$$

The estimate of the regression slope coefficients is given by

$$\tilde{\beta}_s = (\mathbf{X}_{*s}'\mathbf{P}\mathbf{X}_{*s})^{-1}\mathbf{X}_{*s}'\mathbf{P}\mathbf{y}_*$$

where $\mathbf{X}_{*s}$ is the $\mathbf{X}_*$ matrix without the vector of 1s.

The estimator of the error variance is

$$\hat{\sigma}_\epsilon^2 = \tilde{\mathbf{u}}'\mathbf{P}\tilde{\mathbf{u}}/(M - T - N + 1 - (K - 1))$$

where the residuals are given by $\tilde{\mathbf{u}} = (\mathbf{I}_M - \mathbf{j}_M\mathbf{j}_M'/M)(\mathbf{y}_* - \mathbf{X}_{*s}\tilde{\beta}_s)$ if there is an intercept in the model and by $\tilde{\mathbf{u}} = \mathbf{y}_* - \mathbf{X}_{*s}\tilde{\beta}_s$ if there is no intercept.

The actual implementation is quite different from the theory. For more information, see the section "Two-Way Fixed-Effects Model" on page 1835.

One-Way Random-Effects Model

The specification for the one-way random-effects model is

$$u_{it} = v_i + \epsilon_{it}$$

Let $\mathbf{Z_0} = \mathrm{diag}(\mathbf{J}_{T_i})$, $\mathbf{P_0} = \mathrm{diag}(\bar{\mathbf{J}}_{T_i})$, and $\mathbf{Q_0} = \mathrm{diag}(\mathbf{E}_{T_i})$, with $\bar{\mathbf{J}}_{T_i} = \mathbf{J}_{T_i}/T_i$ and $\mathbf{E}_{T_i} = \mathbf{I}_{T_i} - \bar{\mathbf{J}}_{T_i}$. Define $\tilde{\mathbf{X}}_s = \mathbf{Q_0}\mathbf{X}_s$. Also define $\tilde{\mathbf{y}} = \mathbf{Q_0}\mathbf{y}$ and $\mathbf{J}$ as a vector of 1s whose length is T_i.

In the one-way model, estimation proceeds in a two-step fashion. First, you obtain estimates of the variance of the σ_ϵ^2 and σ_v^2. There are multiple ways to derive these estimates; PROC HPPANEL provides four options. For more information, see the section "One-Way Random-Effects Model" on page 1841.

After the variance components are calculated from any method, the next task is to estimate the regression model of interest. For each individual, you form a weight (θ_i),

$$\theta_i = 1 - \sigma_\epsilon / w_i$$

$$w_i^2 = T_i \sigma_v^2 + \sigma_\epsilon^2$$

where T_i is the *i*th cross section's time observations.

Taking the θ_i, you form the partial deviations,

$$\tilde{y}_{it} = y_{it} - \theta_i \bar{y}_{i\cdot}$$

$$\tilde{x}_{it} = x_{it} - \theta_i \bar{x}_{i\cdot}$$

where $\bar{y}_{i\cdot}$ and $\bar{x}_{i\cdot}$ are cross section means of the dependent variable and independent variables (including the constant if any), respectively.

The random-effects β is then the result of simple OLS on the transformed data.

Two-Way Random-Effects Model

The specification for the two-way random-effects model is

$$u_{it} = v_i + e_t + \epsilon_{it}$$

As it does for the one-way random-effects model, the HPPANEL procedure provides four options for variance component estimators. However, unbalanced panels present some special concerns that do not occur for one-way random-effects models.

Let $\mathbf{X}_*$ and $\mathbf{y}_*$ be the independent and dependent variables that are arranged by time and by cross section within each time period. (Note that the input data set that the PANEL procedure uses must be sorted by cross section and then by time within each cross section.) Let M_t be the number of cross sections that are observed in time t, and let $\sum_t M_t = M$. Let $\mathbf{D}_t$ be the $M_t \times N$ matrix that is obtained from the $N \times N$ identity matrix from which rows that correspond to cross sections that are not observed at time t have been omitted. Consider

$$\mathbf{Z} = (\mathbf{Z}_1, \mathbf{Z}_2)$$

where $\mathbf{Z}_1 = (\mathbf{D}_1^{'}, \mathbf{D}_2^{'}, \ldots..\mathbf{D}_T^{'})^{'}$ and $\mathbf{Z}_2 = \mathrm{diag}(\mathbf{D}_1\mathbf{j}_N, \mathbf{D}_2\mathbf{j}_N, \ldots\ldots \mathbf{D}_T\mathbf{j}_N)$.

The matrix $\mathbf{Z}$ contains the dummy variable structure for the two-way model.

For notational ease, let

$$\Delta_N = \mathbf{Z}_1^{'}\mathbf{Z}_1$$

$$\Delta_T = \mathbf{Z}_2^{'}\mathbf{Z}_2$$

$$\mathbf{A} = \mathbf{Z}_2^{'}\mathbf{Z}_1$$

$$\bar{\mathbf{Z}} = \mathbf{Z}_2 - \mathbf{Z}_1\Delta_N^{-1}\mathbf{A}^{'}$$

$$\bar{\Delta}_1 = \mathbf{I}_M - \mathbf{Z}_1\Delta_N^{-1}\mathbf{Z}_1^{'}$$

$$\bar{\Delta}_2 = \mathbf{I}_M - \mathbf{Z}_2\Delta_T^{-1}\mathbf{Z}_2^{'}$$

$$\mathbf{Q} = \Delta_T - \mathbf{A}\Delta_N^{-1}\mathbf{A}^{'}$$

$$\mathbf{P} = (\mathbf{I}_M - \mathbf{Z}_1\Delta_N^{-1}\mathbf{Z}_1^{'}) - \bar{\mathbf{Z}}\mathbf{Q}^{-1}\bar{\mathbf{Z}}^{'}$$

PROC HPPANEL provides four methods to estimate the variance components. For more information, see the section "Two-Way Random-Effects Model" on page 1844.

After the estimates of the variance components are calculated, you can proceed to the final estimation. If the panel is balanced, partial mean deviations are used as follows

$$\tilde{y}_{it} = y_{it} - \theta_1\bar{y}_{i\cdot} - \theta_2\bar{y}_{\cdot t} + \theta_3\bar{y}_{\cdot\cdot}$$

$$\tilde{x}_{it} = x_{it} - \theta_1\bar{x}_{i\cdot} - \theta_2\bar{x}_{\cdot t} + \theta_3\bar{x}_{\cdot\cdot}$$

The θ estimates are obtained from

$$\theta_1 = 1 - \frac{\sigma_\epsilon}{\sqrt{T\sigma_\nu^2 + \sigma_\epsilon^2}}$$

$$\theta_2 = 1 - \frac{\sigma_\epsilon}{\sqrt{N\sigma_e^2 + \sigma_\epsilon^2}}$$

$$\theta_3 = \theta_1 + \theta_2 + \frac{\sigma_\epsilon}{\sqrt{T\sigma_\nu^2 + N\sigma_e^2 + \sigma_\epsilon^2}} - 1$$

With these partial deviations, PROC HPPANEL uses OLS on the transformed series (including an intercept if you want).

The case of an unbalanced panel is somewhat more complicated. Wansbeek and Kapteyn show that the inverse of Ω can be written as

$$\sigma_\epsilon^2\Omega^{-1} = \mathbf{V} - \mathbf{V}\mathbf{Z}_2\tilde{\mathbf{P}}^{-1}\mathbf{Z}_2^{'}\mathbf{V}$$

with the following:

$$\begin{aligned}
\mathbf{V} &= \mathbf{I}_M - \mathbf{Z}_1\tilde{\Delta}_N^{-1}\mathbf{Z}_1^{'} \\
\tilde{\mathbf{P}} &= \tilde{\Delta}_T - \mathbf{A}\tilde{\Delta}_N^{-1}\mathbf{A}^{'} \\
\tilde{\Delta}_N &= \Delta_N + \left(\frac{\sigma_\epsilon^2}{\sigma_\nu^2}\right)\mathbf{I}_N \\
\tilde{\Delta}_T &= \Delta_T + \left(\frac{\sigma_\epsilon^2}{\sigma_e^2}\right)\mathbf{I}_T
\end{aligned}$$

By using the inverse of the covariance matrix of the error, it becomes possible to complete GLS on the unbalanced panel.

Between Estimators

The between-groups estimator is the regression of the cross section means of $\mathbf{y}$ on the cross section means of $\tilde{\mathbf{X}}_s$. In other words, you fit the following regression:

$$\bar{y}_{i\cdot} = \bar{\mathbf{x}}_{i\cdot}\beta^{BG} + \eta_i$$

The between-time-periods estimator is the regression of the time means of $\mathbf{y}$ on the time means of $\tilde{\mathbf{X}}_s$. In other words, you fit the following regression:

$$\bar{y}_{\cdot t} = \bar{\mathbf{x}}_{\cdot t}\beta^{BT} + \zeta_t$$

In both cases, the error is assumed to be normally distributed with mean zero and a constant variance.

Pooled Estimator

The pooled estimator is simply linear regression that is run on all the data, without regard to cross section or time:

$$y_{it} = \mathbf{x}_{it}\beta^{P} + u_{it}$$

The error is assumed to be normally distributed with mean zero and a constant variance.

Linear Hypothesis Testing

For a linear hypothesis of the form $\mathbf{R}\,\beta = \mathbf{r}$, where $\mathbf{R}$ is $J \times K$ and $\mathbf{r}$ is $J \times 1$, the F-statistic with $J, M - K$ degrees of freedom is computed as

$$(\mathbf{R}\beta - \mathbf{r})'[\mathbf{R}\hat{\mathbf{V}}\mathbf{R}']^{-1}(\mathbf{R}\beta - \mathbf{r})$$

However, it is also possible to write the F statistic as

$$F = \frac{(\hat{\mathbf{u}}_*'\hat{\mathbf{u}}_* - \hat{\mathbf{u}}'\hat{\mathbf{u}})/J}{\hat{\mathbf{u}}'\hat{\mathbf{u}}/(M-K)}$$

where

- $\hat{\mathbf{u}}_*$is the residual vector from the restricted regression
- $\hat{\mathbf{u}}$ is the residual vector from the unrestricted regression
- J is the number of restrictions

- $M - K$ are the degrees of freedom, M is the number of observations, and K is the number of parameters in the model

The Wald, likelihood ratio (LR), and LaGrange multiplier (LM) tests are all related to the F test. You use this relationship of the F test to the likelihood ratio and LaGrange multiplier tests. The Wald test is calculated from its definition.

The Wald test statistic is

$$W = (\mathbf{R}\beta - \mathbf{r})'[\mathbf{R}\hat{\mathbf{V}}\mathbf{R}']^{-1}(\mathbf{R}\beta - \mathbf{r})$$

The likelihood ratio is

$$\mathrm{LR} = M \ln\left[1 + \frac{1}{M-K} JF\right]$$

The LaGrange multiplier test statistic is

$$\mathrm{LM} = M\left[\frac{JF}{M - K + JF}\right]$$

where JF represents the number of restrictions multiplied by the result of the F test.

The distribution of these test statistics is the χ^2 distribution whose degrees of freedom equal the number of restrictions imposed (J). The three tests are asymptotically equivalent, but they have differing small-sample properties. Greene (2000, p. 392) and Davidson and MacKinnon (1993, pp. 456–458) discuss the small-sample properties of these statistics.

Specification Tests

The HPPANEL procedure outputs one specification test for random effects: the Hausman (1978) specification test (m statistic) can be used to test hypotheses in terms of bias or inconsistency of an estimator. This test was also proposed by Wu (1973) and further extended in Hausman and Taylor (1982). Hausman's m statistic is as follows.

Consider two estimators, $\hat{\beta}_a$ and $\hat{\beta}_b$, which under the null hypothesis are both consistent, but only $\hat{\beta}_a$ is asymptotically efficient. Under the alternative hypothesis, only $\hat{\beta}_b$ is consistent. The m statistic is

$$m = (\hat{\beta}_b - \hat{\beta}_a)'(\hat{\mathbf{S}}_b - \hat{\mathbf{S}}_a)^{-1}(\hat{\beta}_b - \hat{\beta}_a)$$

where $\hat{\mathbf{S}}_b$ and $\hat{\mathbf{S}}_a$ are consistent estimates of the asymptotic covariance matrices of $\hat{\beta}_b$ and $\hat{\beta}_a$. Then m is distributed as χ^2 with k degrees of freedom, where k is the dimension of $\hat{\beta}_a$ and $\hat{\beta}_b$.

In the random-effects specification, the null hypothesis of no correlation between effects and regressors implies that the OLS estimates of the slope parameters are consistent and inefficient but the GLS estimates of the slope parameters are consistent and efficient. This facilitates a Hausman specification test. The reported degrees of freedom for the χ^2 statistic are equal to the number of slope parameters. If the null hypothesis holds, the random-effects specification should be used.

OUTPUT OUT= Data Set

PROC HPPANEL writes the initial data of the estimated model, predicted values, and residuals to an output data set when the OUT= option is specified in the OUTPUT statement. The OUT= data set contains the following variables:

CSID is the value of the cross section ID. The variable name is the one specified in the id statement.

TSID is the value of the time period in the dynamic model. The variable name is the one specified in the id statement.

Regressors are the values of regressor variables that are specified in the COPYVAR option.

Pred is the predicted value of dependent variable. This column is output only if the PRED option is specified.

Resid is the residual from the regression. This column is output only if the RESIDUAL option is specified.

OUTEST= Data Set

PROC HPPANEL writes the parameter estimates to an output data set when the OUTEST= option is specified in the PROC HPPANEL statement. The OUTEST= data set contains the following variables in the PROC statement:

METHOD is a character variable that identifies the estimation method.

TYPE is a character variable that identifies the type of observation. Values of the _TYPE_ variable are CORRB, COVB, CSPARMS, STD, and the type of model estimated. The CORRB observation contains correlations of the parameter estimates; the COVB observation contains covariances of the parameter estimates; the STD observation indicates the row of standard deviations of the corresponding coefficients; and the type of model estimated observation contains the parameter estimates.

NAME is a character variable that contains the name of a regressor variable for COVB and CORRB observations and is left blank for other observations. The _NAME_ variable is used in conjunction with the _TYPE_ values COVB and CORRB to identify rows of the correlation or covariance matrix.

DEPVAR is a character variable that contains the name of the response variable.

MSE is the mean square error of the transformed model.

VARCS is the variance component estimate due to cross sections. The _VARCS_ variable is included in the OUTEST= data set when the RANONE option is specified in the MODEL or PROC HPPANEL statement.

VARTS is the variance component estimate due to time series. The _VARTS_ variable is included in the OUTEST= data set when the RANTWO option is specified in the MODEL or PROC HPPANEL statement.

VARERR is the variance component estimate due to error. The _VARERR_ variable is included in the OUTEST= data set when the RANONE or RANTWO option is specified in the MODEL or PROC HPPANEL statement.

Intercept is the intercept parameter estimate. (The intercept is missing for models when the NOINT option is specified in the MODEL statement.)

Regressors are the regressor variables that are specified in the MODEL statement. The regressor variables in the OUTEST= data set contain the corresponding parameter estimates, and the corresponding covariance or correlation matrix elements for _TYPE_=COVB and _TYPE_=CORRB observations.

Printed Output

The printed output from PROC HPPANEL includes the following:

- the model information, which includes the data source, the dependent variable name, the estimation method used, the execution mode, and for random-effects model analysis, the variance component estimation method.
- the number of observations
- the fit statistics, which include the sum of squared error (SSE), the degree of freedom for error (DFE), the mean square error (MSE), the root mean square error (RMSE), and the R-square
- the error components estimates for random-effects model
- the Hausman test statistics, which include the degree of freedom (DF), the test statistics, and the *p*-value.
- the regression parameter estimates and analysis, which include for each regressor the name of the regressor, the degrees of freedom, the parameter estimate, the standard error of the estimate, a *t* statistic for testing whether the estimate is significantly different from 0, and the significance probability of the *t* statistic

Optionally, PROC HPPANEL prints the following:

- the covariance and correlation of the resulting regression parameter estimates
- the WALD, LR, and LM test statistics for linear equality restrictions that are specified in the TEST statements
- the timing breakdown of the procedure steps

ODS Table Names

PROC HPPANEL assigns a name to each table it creates. You can use these names to refer to the table when you use the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in Table 21.2.

Table 21.2 ODS Tables Produced in PROC HPPANEL

ODS Table Name	Description	Option
ODS Tables Created by the MODEL Statement		
ModelInfo	Model information	Default
PerformanceInfo	Performance information	Default
Nobs	Number of observations	Default
FitStatistics	Fit statistics	Default
ParameterEstimates	Parameter estimates	Default
CovB	Covariance of parameter estimates	COVB
CorrB	Correlations of parameter estimates	CORRB
RandomEffectsTest	Hausman test for random effects	RANONE, RANTWO
ODS Tables Created by the TEST Statement		
TestResults	Test results	
ODS Tables Created by the PERFORMANCE Statement		
Timing	Timing Table	

Example: HPPANEL Procedure

Example 21.1: One-Way Random-Effects High-Performance Model

This example shows the use of the one-way random-effects model that is available in the HPPANEL procedure; the example emphasizes processing a large data set and the performance improvements that are achieved by executing in a high-performance distributed environment.

The following DATA step generates five million observations from one-way panel data that includes 50,000 cross sections and 100 time periods:

```
data hppan_ex01 (keep = cs ts y x1-x10);
   retain seed 55371;
   array x[10];
   label y  = 'Dependent Variable';
   do cs = 1 to 50000;
      dummy = 10 * rannor(seed);
      do ts = 1 to 100;
      /*- generate regressors and compute the structural */
      /*- part of the dependent variable                 */
         y = 5;
         do k = 1 to 10;
            x[k] = -1 + 2 * ranuni(seed);
            y = y + x[k] * k;
         end;

         /*- add an error term, such that e - N(0,100)   -------*/
         y = y + 10 * rannor(seed);
         /*- add a random effect, such that v - N(0,100) -------*/
         y = y + dummy;
         output;
      end;
   end;
run;
```

The estimation is executed in distributed mode on a grid with ten nodes, with one thread per node. To run the following statements successfully, you need to set the macro variables GRIDHOST and GRIDINSTALLLOC to resolve to appropriate values, or you can replace the references to the macro variables in the example with the appropriate values.

```
%let GRIDHOST      = <<your grid host>>;
%let GRIDINSTALLOC = <<your grid install location>>;

option set = GRIDHOST       = "&GRIDHOST";
option set = GRIDINSTALLLOC = "&GRIDINSTALLLOC";


proc hppanel data=hppan_ex01;
   id  cs ts;
   model y = x1-x10 / ranone;
   performance nodes = 10 threads = 1 details
               host="&GRIDHOST" install="&GRIDINSTALLLOC";
run;
```

In Output 21.1.1, the "Performance Information" table shows that the model was estimated on the grid that is defined in the macro variable named GRIDHOST in a distributed environment with ten nodes, and one thread per node. The grid installation location is defined in the macro variable named GRIDINSTALLLOC.

Output 21.1.1 Grid Information with Ten Nodes and One Thread per Node

Performance Information	
Host Node	<< your grid host >>
Install Location	<< your grid install location >>
Execution Mode	Distributed
Number of Compute Nodes	10
Number of Threads per Node	1

Output 21.1.2 shows the results for the one-way random-effects model. The "Model Information" table shows detailed information about the model. The "Number of Observations" table indicates that all five million observations were used to fit the model. All parameter estimates in the "Parameter Estimates" table are highly significant and correspond to the theoretical values that were set for them during the data generating process. In the "Procedure Task Timing" table, you can see that for five million observations, computing the moments took 101.53 seconds, and the time taken for cross-product accumulation was negligible.

Output 21.1.2 One-Way Random-Effects Model

Model Information	
Data Source	HPPAN_EX01
Response Variable	y
Model	RANONE
Variance Component	WANSBEEK
Execution Mode	Distributed

Number of Observations	
Number of Observations Read	5000000
Number of Observations Used	5000000
Number of Cross Sections	50000
Number of Time Series	100

Fit Statistics	
Sum of Squared Error	4.9976E8
Degrees of Freedom	4999989
Mean Squared Error	99.952
Root Mean Squared Error	9.9976
R-Square	0.559771

Variance Component Estimates	
Variance Component for Cross Sections	99.9117
Variance Component for Error	99.9520

Hausman Test for Random Effects			
Coefficients	**DF**	**m Value**	**Pr > m**
10	10	14.04	0.1713

Output 21.1.2 *continued*

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	4.96955	0.04492	110.62	<.0001
x1	1	1.00902	0.00778	129.69	<.0001
x2	1	1.99743	0.00778	256.66	<.0001
x3	1	3.00116	0.00778	385.64	<.0001
x4	1	3.99847	0.00778	513.68	<.0001
x5	1	4.99497	0.00778	641.81	<.0001
x6	1	6.01034	0.00778	772.12	<.0001
x7	1	6.99770	0.00778	899.39	<.0001
x8	1	7.98897	0.00778	1026.61	<.0001
x9	1	9.00692	0.00778	1157.12	<.0001
x10	1	10.00563	0.00778	1285.47	<.0001

Procedure Task Timing

Task	Seconds	Percent
Data Read and Variable Levelization	0.30	0.29%
Communication to Client	0.00	0.00%
Computing Moments	101.53	99.33%
Cross-Product Accumulation	0.38	0.37%

For comparison, you now fit a pooled regression estimation on the same data, again using a grid of 10 nodes with one thread each. The following SAS statements perform the estimation on the grid:

```
proc hppanel data=hppan_ex01;
   id  cs ts;
   model y = x1-x10 / pooled;
   performance nodes = 10 threads = 1 details
               host="&GRIDHOST" install="&GRIDINSTALLLOC";
run;
```

Based on Output 21.1.3, you find that the parameter estimates are similar to those from the random-effects estimator. You also find that the timings are similar, indicating that the bulk of the computational effort is due to tasks common to both random-effects estimation and standard OLS regression. In both cases, estimation is dominated by the calculation of sums of squares and other moment terms, over the whole data set.

Output 21.1.3 Pooled Regression Model

The HPPANEL Procedure

Model Information

Data Source	HPPAN_EX01
Response Variable	y
Model	POOLED
Execution Mode	Distributed

Output 21.1.3 *continued*

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	4.96957	0.00632	786.03	<.0001
x1	1	1.01251	0.01095	92.49	<.0001
x2	1	1.98374	0.01095	181.17	<.0001
x3	1	3.00294	0.01095	274.23	<.0001
x4	1	3.99649	0.01095	364.90	<.0001
x5	1	5.00187	0.01095	456.77	<.0001
x6	1	5.99952	0.01095	547.77	<.0001
x7	1	7.00478	0.01095	639.88	<.0001
x8	1	7.97232	0.01095	728.13	<.0001
x9	1	9.01244	0.01095	822.90	<.0001
x10	1	10.01578	0.01095	914.52	<.0001

Procedure Task Timing

Task	Seconds	Percent
Data Read and Variable Levelization	0.28	0.27%
Communication to Client	0.00	0.00%
Computing Moments	103.68	99.59%
Cross-Product Accumulation	0.15	0.14%

References

Davidson, R., and MacKinnon, J. G. (1993). *Estimation and Inference in Econometrics*. New York: Oxford University Press.

Fuller, W. A., and Battese, G. E. (1974). "Estimation of Linear Models with Crossed-Error Structure." *Journal of Econometrics* 2:67–78.

Greene, W. H. (1990). *Econometric Analysis*. New York: Macmillan.

Greene, W. H. (2000). *Econometric Analysis*. 4th ed. Upper Saddle River, NJ: Prentice-Hall.

Hausman, J. A. (1978). "Specification Tests in Econometrics." *Econometrica* 46:1251–1271.

Hausman, J. A., and Taylor, W. E. (1982). "A Generalized Specification Test." *Economics Letters* 8:239–245.

Nerlove, M. (1971). "Further Evidence on the Estimation of Dynamic Relations from a Time Series of Cross Sections." *Econometrica* 39:359–382.

Wallace, T., and Hussain, A. (1969). "The Use of Error Components Model in Combining Cross Section with Time Series Data." *Econometrica* 37:55–72.

Wansbeek, T., and Kapteyn, A. (1989). "Estimation of the Error-Components Model with Incomplete Panels." *Journal of Econometrics* 41:341–361.

Wu, D. M. (1973). "Alternative Tests of Independence between Stochastic Regressors and Disturbances." *Econometrica* 41:733–750.

Chapter 22
The HPQLIM Procedure

Contents

Overview: HPQLIM Procedure **1144**
PROC HPQLIM Features 1145
Getting Started: HPQLIM Procedure **1145**
Syntax: HPQLIM Procedure **1147**
Functional Summary 1147
PROC HPQLIM Statement 1150
BAYES Statement 1154
BOUNDS Statement 1158
BY Statement 1159
ENDOGENOUS Statement 1159
FREQ Statement 1161
HETERO Statement 1161
INIT Statement 1162
MODEL Statement 1162
OUTPUT Statement 1165
PERFORMANCE Statement 1166
PRIOR Statement 1166
RESTRICT Statement 1167
TEST Statement 1168
WEIGHT Statement 1169
Details: HPQLIM Procedure **1169**
Ordinal Discrete Choice Modeling 1169
Limited Dependent Variable Models 1170
Stochastic Frontier Production and Cost Models 1171
Heteroscedasticity 1172
Tests on Parameters 1173
Bayesian Analysis 1174
Prior Distributions 1175
Output to SAS Data Set 1178
OUTEST= Data Set 1181
Naming 1182
ODS Table Names 1183
ODS Graphics 1184
Examples: The HPQLIM Procedure **1184**
Example 22.1: High-Performance Model with Censoring 1184
Example 22.2: Bayesian High-Performance Model with Censoring 1188
References **1192**

Overview: HPQLIM Procedure

The HPQLIM (high-performance qualitative and limited dependent variable model) procedure is a high-performance version of the QLIM procedure in SAS/ETS software, which analyzes univariate limited dependent variable models in which dependent variables are observed only in a limited range of values. Unlike the QLIM procedure, which can be run only on an individual workstation, the HPQLIM procedure takes advantage of a computing environment that enables it to distribute the optimization task to one or more nodes. In addition, each node can use one or more threads to perform the optimization on its subset of the data. When several nodes are used and each node uses several threads to carry out its part of the work, the result is a highly parallel computation that provides a dramatic gain in performance.

With the HPQLIM procedure you can read and write data in distributed form and perform analyses in distributed mode and single-machine mode. For more information about how to affect the execution mode of SAS high-performance analytical procedures, see the section "Processing Modes" on page 62 in Chapter 3, "Shared Concepts and Topics."

The HPQLIM procedure is specifically designed to operate in the high-performance distributed environment. It can use maximum likelihood or Bayesian methods. In both cases, the likelihood evaluation is performed in a distributed environment. By default, PROC HPQLIM uses multiple threads to perform computations.

The HPQLIM procedure is similar in use to the other SAS procedures that support regression or simultaneous equations models. For example, the standard model with censoring or truncation is estimated by specifying the endogenous variable to be truncated or censored. When the data are limited by specific values or variables, the limits of the dependent variable can be specified with the CENSORED or TRUNCATED option in the ENDOGENOUS or MODEL statement. For example, the two-limit censored model requires two variables: one that contains the lower (bottom) bound and one that contains the upper (top) bound. The following statements execute the model in the distributed computing environment with two threads and four nodes:

```
proc hpqlim data=a;
   model y = x1 x2 x3;
   endogenous y ~ censored(lb=bottom ub=top);
   performance nthreads=2 nodes=4 details;
run;
```

The bounds can be numbers if they are fixed for all observations in the data set. For example, the standard Tobit model can be specified as follows:

```
proc hpqlim data=a;
   model y = x1 x2 x3;
   endogenous y ~ censored(lb=0);
   performance nthreads=2 nodes=4 details;
run;
```

PROC HPQLIM Features

The HPQLIM procedure supports the following models:

- linear regression models with heteroscedasticity
- Tobit models (censored and truncated) with heteroscedasticity
- stochastic frontier production and cost models

In linear regression models with heteroscedasticity, the assumption that error variance is constant across observations is relaxed. The HPQLIM procedure allows for a number of different linear and nonlinear variance specifications.

The HPQLIM procedure also offers a class of models in which the dependent variable is censored or truncated from below or above or both. When a continuous dependent variable is observed only within a certain range, and values outside this range are not available, the HPQLIM procedure offers a class of models that adjust for truncation. In some cases, the dependent variable is continuous only in a certain range, and all values outside this range are reported as being on its boundary. For example, if it is not possible to observe negative values, the value of the dependent variable is reported as equal to 0. Because the data are censored, ordinary least squares (OLS) results are inconsistent, and it cannot be guaranteed that the predicted values from the model will fall in the appropriate region.

Stochastic frontier production and cost models allow for random shocks of the production or cost. They include a systematic positive component in the error term that adjusts for technical or cost inefficiency.

The HPQLIM procedure can use maximum likelihood or Bayesian methods. Initial starting values for the nonlinear optimizations are typically calculated by OLS. Initial values for the Bayesian sampling are typically calculated by maximum likelihood.

Getting Started: HPQLIM Procedure

This example illustrates the use of the HPQLIM procedure. The data were originally published by Mroz (1987), and the following statements show a subset of the Mroz (1987) data set:

```
title1 'Estimating a Tobit model';

data subset;
   input Hours Yrs_Ed Yrs_Exp @@;
   if Hours eq 0 then Lower=.;
      else            Lower=Hours;
datalines;
0 8 9 0 8 12 0 9 10 0 10 15 0 11 4 0 11 6
1000 12 1 1960 12 29 0 13 3 2100 13 36
3686 14 11 1920 14 38 0 15 14 1728 16 3
1568 16 19 1316 17 7 0 17 15
;
```

In these data, Hours is the number of hours that the wife worked outside the household in a given year, Yrs_Ed is the years of education, and Yrs_Exp is the years of work experience.

By the nature of the data it is clear that there are a number of women who committed some positive number of hours to outside work ($y_i > 0$ is observed). There are also a number of women who did not work outside the home at all ($y_i = 0$ is observed). This yields the following model:

$$y_i^* = \mathbf{x}_i'\boldsymbol{\beta} + \epsilon_i$$

$$y_i = \begin{cases} y_i^* & \text{if } y_i^* > 0 \\ 0 & \text{if } y_i^* \leq 0 \end{cases}$$

where $\epsilon_i \sim iidN(0, \sigma^2)$ and the set of explanatory variables is denoted by $\mathbf{x}_i$. The following statements fit a Tobit model to the hours worked with years of education and years of work experience as covariates:

```
/*-- Tobit Model --*/
proc hpqlim data=subset;
   model hours = yrs_ed yrs_exp;
   endogenous hours ~ censored(lb=0);
   performance nthreads=2 nodes=4 details;
run;
```

The output of the HPQLIM procedure is shown in Output 22.1.

Figure 22.1 Tobit Analysis Results

Estimating a Tobit model

The HPQLIM Procedure

Model Fit Summary	
Number of Endogenous Variables	1
Endogenous Variable	Hours
Number of Observations	17
Log Likelihood	-74.93700
Maximum Absolute Gradient	1.18953E-6
Number of Iterations	23
Optimization Method	Quasi-Newton
AIC	157.87400
Schwarz Criterion	161.20685

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	-5598.295129	27.692220	-202.16	<.0001
Yrs_Ed	1	373.123254	53.988877	6.91	<.0001
Yrs_Exp	1	63.336247	36.551299	1.73	0.0831
_Sigma	1	1582.859635	390.076480	4.06	<.0001

The "Parameter Estimates" table contains four rows. The first three rows correspond to the vector estimate of the regression coefficients $\boldsymbol{\beta}$. The last row is called _Sigma, which corresponds to the estimate of the error variance σ.

Syntax: HPQLIM Procedure

The following statements are available in the HPQLIM procedure:

```
PROC HPQLIM options ;
  BAYES < options > ;
  BOUNDS bound1 < , bound2 . . . > ;
  BY variables ;
  FREQ variable ;
  ENDOGENOUS variables ~ options ;
  HETERO dependent variables ~ exogenous variables / options ;
  INIT initvalue1 < , initvalue2 . . . > ;
  MODEL dependent variables = regressors / options ;
  OUTPUT options ;
  PRIOR variables ~ distributions ;
  RESTRICT restriction1 < , restriction2 . . . > ;
  TEST options ;
  WEIGHT variable ;
```

One MODEL statement is required. If a FREQ or WEIGHT statement is specified more than once, the variable that is specified in the first instance is used.

Functional Summary

Table 22.1 summarizes the statements and options used with the HPQLIM procedure.

Table 22.1 PROC HPQLIM Functional Summary

Description	Statement	Option
Data Set Options		
Specifies the input data set	PROC HPQLIM	DATA=
Writes parameter estimates to an output data set	PROC HPQLIM	OUTEST=
Writes predictions to an output data set	OUTPUT	OUT=
Declaring the Role of Variables		
Specifies BY-group processing	BY	
Specifies a frequency variable	FREQ	
Specifies a weight variable	WEIGHT	NONORMALIZE
Printing Control Options		
Requests all printing options	PROC HPQLIM	PRINTALL
Prints the correlation matrix of the estimates	PROC HPQLIM	CORRB
Prints the covariance matrix of the estimates	PROC HPQLIM	COVB
Suppresses the normal printed output	PROC HPQLIM	NOPRINT

Table 22.1 *continued*

Description	Statement	Option
Plotting Options		
Displays plots	PROC HPQLIM	PLOTS=
Optimization Process Control Options		
Selects the iterative minimization method to use	PROC HPQLIM	METHOD=
Specifies the maximum number of iterations allowed	PROC HPQLIM	MAXITER=
Specifies the maximum number of function calls	PROC HPQLIM	MAXFUNC=
Specifies the upper limit of CPU time in seconds	PROC HPQLIM	MAXTIME=
Specifies an absolute convergence criterion	PROC HPQLIM	ABSCONV=
Specifies an absolute function convergence criterion	PROC HPQLIM	ABSFCONV=
Specifies an absolute gradient convergence criterion	PROC HPQLIM	ABSGCONV=
Specifies a relative function convergence criterion	PROC HPQLIM	FCONV=
Specifies a relative gradient convergence criterion	PROC HPQLIM	GCONV=
Specifies an absolute parameter convergence criterion	PROC HPQLIM	ABSXCONV=
Specifies a matrix singularity criterion	PROC HPQLIM	SINGULAR=
Sets boundary restrictions on parameters	BOUNDS	
Sets initial values for parameters	INIT	
Sets linear restrictions on parameters	RESTRICT	
Model Estimation Options		
Suppresses the intercept parameter	MODEL	NOINT
Specifies the method to calculate parameter covariance	PROC HPQLIM	COVEST=
Bayesian MCMC Options		
Specifies the initial values of the MCMC	INIT	
Specifies the maximum number of tuning phases	BAYES	MAXTUNE=
Specifies the minimum number of tuning phases	BAYES	MINTUNE=
Specifies the number of burn-in iterations	BAYES	NBI=
Specifies the number of iterations during the sampling phase	BAYES	NMC=
Specifies the number of iterations during the tuning phase	BAYES	NTU=
Controls options for constructing the initial proposal covariance matrix	BAYES	PROPCOV
Specifies the sampling scheme	BAYES	SAMPLING=
Specifies the random number generator seed	BAYES	SEED=
Controls the thinning of the Markov chain	BAYES	THIN=
Bayesian Summary Statistics and Convergence Diagnostic Options		
Displays convergence diagnostics	BAYES	DIAGNOSTICS=
Displays summary statistics of the posterior samples	BAYES	STATISTICS=

Table 22.1 *continued*

Description	Statement	Option
Bayesian Prior and Posterior Sample Options		
Specifies a SAS data set for the posterior samples	BAYES	OUTPOST=
Bayesian Analysis Options		
Specifies the normal prior distribution	PRIOR	NORMAL(MEAN=, VAR=)
Specifies the gamma prior distribution	PRIOR	GAMMA(SHAPE=, SCALE=)
Specifies the inverse gamma prior distribution	PRIOR	IGAMMA(SHAPE=, SCALE=)
Specifies the uniform prior distribution	PRIOR	UNIFORM(MIN=, MAX=)
Specifies the beta prior distribution	PRIOR	BETA(SHAPE1=, SHAPE2=, MIN=, MAX=)
Specifies the *t* prior distribution	PRIOR	T(LOCATION=, DF=)
Endogenous Variable Options		
Specifies a discrete variable	ENDOGENOUS	DISCRETE()
Specifies a censored variable	ENDOGENOUS	CENSORED()
Specifies a truncated variable	ENDOGENOUS	TRUNCATED()
Specifies a stochastic frontier variable	ENDOGENOUS	FRONTIER()
Heteroscedasticity Model Options		
Specifies the function for heteroscedasticity models	HETERO	LINK=
Squares the function for heteroscedasticity models	HETERO	SQUARE
Specifies no constant for heteroscedasticity models	HETERO	NOCONST
Output Control Options		
Outputs predicted values	OUTPUT	PREDICTED
Outputs the structured part	OUTPUT	XBETA
Outputs residuals	OUTPUT	RESIDUAL
Outputs the error standard deviation	OUTPUT	ERRSTD
Outputs marginal effects	OUTPUT	MARGINAL
Outputs probability for the current response	OUTPUT	PROB
Outputs probability for all responses	OUTPUT	PROBALL
Outputs the expected value	OUTPUT	EXPECTED
Outputs the conditional expected value	OUTPUT	CONDITIONAL
Outputs inverse Mills ratio	OUTPUT	MILLS
Outputs technical efficiency measures	OUTPUT	TE1
	OUTPUT	TE2
Includes covariances in the OUTEST= data set	PROC HPQLIM	COVOUT
Includes correlations in the OUTEST= data set	PROC HPQLIM	CORROUT

Table 22.1 *continued*

Description	Statement	Option
Test Request Options		
Requests Wald, Lagrange multiplier, and likelihood ratio tests	TEST	ALL
Requests the Wald test	TEST	WALD
Requests the Lagrange multiplier test	TEST	LM
Requests the likelihood ratio test	TEST	LR

PROC HPQLIM Statement

PROC HPQLIM *options* **;**

The PROC HPQLIM statement invokes the HPQLIM procedure. You can specify the following *options*.

Data Set Options

DATA=*SAS-data-set*
specifies the input SAS data set. If this option is not specified, PROC HPQLIM uses the most recently created SAS data set.

Output Data Set Options

OUTEST=*SAS-data-set*
writes the parameter estimates to an output data set.

COVOUT
writes the covariance matrix for the parameter estimates to the OUTEST= data set. This option is valid only if the OUTEST= option is specified.

CORROUT
writes the correlation matrix for the parameter estimates to the OUTEST= data set. This option is valid only if the OUTEST= option is specified.

Printing Options

NOPRINT
suppresses the normal printed output but does not suppress error listings. If this option is specified, then any other print option is turned off.

PRINTALL
turns on all the printing options. The options that are set by PRINTALL are COVB and CORRB.

CORRB
prints the correlation matrix of the parameter estimates.

COVB
prints the covariance matrix of the parameter estimates.

Model Estimation Options

COVEST=*covariance-option*
specifies the method for calculating the covariance matrix of parameter estimates. You can specify the following *covariance-options*.

OP	specifies the covariance from the outer product matrix.
HESSIAN	specifies the covariance from the inverse Hessian matrix.
QML	specifies the covariance from the outer product and Hessian matrices (the quasi-maximum likelihood estimates).

The default is COVEST=HESSIAN.

Optimization Control Options

PROC HPQLIM uses the nonlinear optimization (NLO) subsystem to perform nonlinear optimization tasks. You can specify the following *options*:

ABSCONV=*r*
ABSTOL=*r*
specifies an absolute function value convergence criterion by which minimization stops when $f(\theta^{(k)}) \leq r$. The default value of r is the negative square root of the largest double-precision value, which serves only as a protection against overflows.

ABSFCONV=*r*
ABSFTOL=*r*
specifies an absolute function difference convergence criterion by which minimization stops when the function value has a small change in successive iterations:

$$|f(\theta^{(k-1)}) - f(\theta^{(k)})| \leq r$$

The default value is $r = 0$.

ABSGCONV=*r*
ABSGTOL=*r*
specifies an absolute gradient convergence criterion. Optimization stops when the maximum absolute gradient element is small:

$$\max_j |g_j(\theta^{(k)})| \leq r$$

The default value is *r*=1E–5.

ABSXCONV=*r*

ABSXTOL=*r*

specifies an absolute parameter convergence criterion. Optimization stops when the Euclidean distance between successive parameter vectors is small:

$$\| \theta^{(k)} - \theta^{(k-1)} \|_2 \leq r$$

The default is 0.

FCONV=*r*

FTOL=*r*

specifies a relative function convergence criterion. Optimization stops when a relative change of the function value in successive iterations is small:

$$\frac{|f(\theta^{(k)}) - f(\theta^{(k-1)})|}{|f(\theta^{(k-1)})|} \leq r$$

The default value is $r = 2\epsilon$, where ϵ denotes the machine precision constant, which is the smallest double-precision floating-point number such that $1 + \epsilon > 1$.

GCONV=*r*

GTOL=*r*

specifies a relative gradient convergence criterion. For all techniques except CONGRA, optimization stops when the normalized predicted function reduction is small:

$$\frac{g(\theta^{(k)})^T [H^{(k)}]^{-1} g(\theta^{(k)})}{|f(\theta^{(k)})|} \leq r$$

For the CONGRA technique (where a reliable Hessian estimate H is not available), the following criterion is used:

$$\frac{\| g(\theta^{(k)}) \|_2^2 \quad \| s(\theta^{(k)}) \|_2}{\| g(\theta^{(k)}) - g(\theta^{(k-1)}) \|_2 \, |f(\theta^{(k)})|} \leq r$$

The default value is $r =$1E–8.

MAXFUNC=*i*

MAXFU=*i*

specifies the maximum number of function calls in the optimization process. The default is 1,000.

The optimization can terminate only after completing a full iteration. Therefore, the number of function calls that are actually performed can exceed the number of calls that are specified by this option.

MAXITER=*i*

MAXIT=*i*

specifies the maximum number of iterations in the optimization process. The default is 200.

MAXTIME=*r*

specifies an upper limit of *r* seconds of CPU time for the optimization process. The default value is the largest floating-point double representation of your computer. The time that is specified by this option is checked only once at the end of each iteration. Therefore, the actual running time can be much longer than *r*. The actual running time includes the remaining time needed to finish the iteration and the time needed to generate the output of the results.

METHOD=*value*

specifies the iterative minimization method to use. The default is METHOD=NEWRAP. You can specify the following *values*:

CONGRA	specifies the conjugate-gradient method.
DBLDOG	specifies the double dogleg method.
NONE	specifies that no optimization be performed beyond using the ordinary least squares method to compute the parameter estimates.
NEWRAP	specifies the Newton-Raphson method (the default).
NRRIDG	specifies the Newton-Raphson Ridge method.
QUANEW	specifies the quasi-Newton method.
TRUREG	specifies the trust region method.

SINGULAR=*r*

specifies the general singularity criterion that is applied by the HPQLIM procedure in sweeps and inversions. The default for the optimization is 1E–8.

Plotting Options

PLOTS< (*global-plot-options***) > = ***plot-request* | *(plot-requests)*

controls the display of plots. By default, the plots are displayed in panels unless the UNPACK *global-plot-option* is specified. When you specify only one *plot-request*, you can omit the parentheses around it.

Global Plot Options

You can specify the following *global-plot-options*:

ONLY

displays only the requested plot.

UNPACKPANEL

UNPACK

specifies that all paneled plots be unpacked, meaning that each plot in a panel is displayed separately.

Plot Requests

You can specify the following *plot-requests*:

ALL

specifies all types of available plots.

AUTOCORR< (LAGS=*n***) >**

displays the autocorrelation function plots for the parameters. The optional LAGS= suboption specifies the number (up to lag *n*) of autocorrelations to be plotted in the autocorrelation function plot. If this suboption is not specified, autocorrelations are plotted up to lag 50. This *plot-request* is available only for Bayesian analysis.

BAYESDIAG
is equivalent to specifying the TRACE, AUTOCORR, and DENSITY *plot-requests*.

DENSITY< (FRINGE) >
displays the kernel density plots for the parameters. If you specify the FRINGE suboption, a fringe plot is created on the X axis of the kernel density plot. This *plot-request* is available only for Bayesian analysis.

NONE
suppresses all diagnostic plots.

TRACE< (SMOOTH) >
displays the trace plots for the parameters. The SMOOTH suboption displays a fitted penalized B-spline curve for each plot. This *plot-request* is available only for Bayesian analysis.

BAYES Statement

BAYES < *options* > **;**

The BAYES statement controls the Metropolis sampling scheme that is used to obtain samples from the posterior distribution of the underlying model and data.

DIAGNOSTICS=ALL | NONE | *(keyword-list)*
DIAG=ALL | NONE | *(keyword-list)*
controls which diagnostics are produced. All the following diagnostics are produced when you specify DIAGNOSTICS=ALL. If you do not want any of these diagnostics, specify DIAGNOSTICS=NONE. If you want some but not all of the diagnostics, or if you want to change certain settings of these diagnostics, specify one or more of the following keywords. The default is DIAGNOSTICS=NONE.

AUTOCORR < (LAGS=*numeric-list***) >**
computes the autocorrelations at lags that are specified in the *numeric-list*. Elements in the *numeric-list* are truncated to integers, and repeated values are removed. If the LAGS= option is not specified, autocorrelations of lags 1, 5, and 10 are computed.

ESS
computes Carlin's estimate of the effective sample size, the correlation time, and the efficiency of the chain for each parameter.

GEWEKE < (*geweke-options***) >**
computes the Geweke spectral density diagnostics, which are essentially a two-sample t test between the first f_1 portion and the last f_2 portion of the chain. The defaults are $f_1 = 0.1$ and $f_2 = 0.5$, but you can choose other fractions by using the following *geweke-options*:

FRAC1=*value*
specifies the fraction f_1 for the first window.

FRAC2=*value*
specifies the fraction f_2 for the second window.

HEIDELBERGER < (*heidel-options*) >
computes for each variable the Heidelberger and Welch diagnostic, which consists of a stationarity test of the null hypothesis that the sample values form a stationary process. If the stationarity test is not rejected, a halfwidth test is then carried out. Optionally, you can specify one or more of the following *heidel-options*:

EPS=*value*
specifies a positive number ϵ such that if the halfwidth is less than ϵ times the sample mean of the retained iterates, the halfwidth test is passed.

HALPHA=*value*
specifies the α level ($0 < \alpha < 1$) for the halfwidth test.

SALPHA=*value*
specifies the α level ($0 < \alpha < 1$) for the stationarity test.

MCSE

MCERROR
computes the Monte Carlo standard error for each parameter. The Monte Carlo standard error, which measures the simulation accuracy, is the standard error of the posterior mean estimate and is calculated as the posterior standard deviation divided by the square root of the effective sample size.

RAFTERY< (*raftery-options*) >
computes the Raftery and Lewis diagnostics, which evaluate the accuracy of the estimated quantile ($\hat{\theta}_Q$ for a given $Q \in (0,1)$) of a chain. $\hat{\theta}_Q$ can achieve any degree of accuracy when the chain is allowed to run for a long time. The computation stops when the estimated probability $\hat{P}_Q = \Pr(\theta \leq \hat{\theta}_Q)$ reaches within $\pm R$ of the value Q with probability S; that is, $\Pr(Q - R \leq \hat{P}_Q \leq Q + R) = S$. The following *raftery-options* enable you to specify Q, R, S, and a precision level ϵ for the test:

QUANTILE | Q=*value*
specifies the order (a value between 0 and 1) of the quantile of interest. The default is 0.025.

ACCURACY | R=*value*
specifies a small positive number as the margin of error for measuring the accuracy of the estimation of the quantile. The default is 0.005.

PROBABILITY | S=*value*
specifies the probability of attaining the accuracy of the estimation of the quantile. The default is 0.95.

EPSILON | EPS=*value*
specifies the tolerance level (a small positive number) for the stationary test. The default is 0.001.

MINTUNE=*number*
specifies the minimum number of tuning phases. The default is 2.

MAXTUNE=*number*
specifies the maximum number of tuning phases. The default is 24.

NBI=*number*
specifies the number of burn-in iterations before the chains are saved. The default is 1,000.

NMC=*number*
specifies the number of iterations after the burn-in. The default is 1,000.

NTU=*number*
specifies the number of samples for each tuning phase. The default is 500.

OUTPOST=*SAS-data-set*
names the SAS data set to contain the posterior samples. Alternatively, you can create the output data set by specifying an ODS OUTPUT statement as follows:

ODS OUTPUT POSTERIORSAMPLE = < *SAS-data-set* > **;**

PROPCOV=*value*
specifies the method that is used in constructing the initial covariance matrix for the Metropolis-Hastings algorithm. The QUANEW and NMSIMP methods find numerically approximated covariance matrices at the optimum of the posterior density function with respect to all continuous parameters. The tuning phase starts at the optimized values; in some problems, this can greatly increase convergence performance. If the approximated covariance matrix is not positive definite, then an identity matrix is used instead. You can specify the following *values*:

CONGRA
performs a conjugate-gradient optimization.

DBLDOG
performs a version of double-dogleg optimization.

NEWRAP
performs a Newton-Raphson optimization that combines a line-search algorithm with ridging.

NMSIMP
performs a Nelder-Mead simplex optimization.

NRRIDG
performs a Newton-Raphson optimization with ridging.

QUANEW
performs a quasi-Newton optimization.

TRUREG
performs a trust-region optimization.

SAMPLING=MULTIMETROPOLIS | UNIMETROPOLIS
specifies how to sample from the posterior distribution. SAMPLING=MULTIMETROPOLIS implements a Metropolis sampling scheme on a single block that contains all the parameters of the model. SAMPLING=UNIMETROPOLIS implements a Metropolis sampling scheme on multiple blocks, one for each parameter of the model. The default is SAMPLING=MULTIMETROPOLIS.

SEED=*number*

specifies an integer seed in the range 1 to $2^{31} - 1$ for the random number generator in the simulation. Specifying a seed enables you to reproduce identical Markov chains for the same specification. If you do not specify the SEED= option, or if you specify a nonpositive seed, a random seed is derived from the time of day.

STATISTICS < **(***global-options***)** > **= ALL** | **NONE** | *keyword* | *(keyword-list)*

STATS < **(***global-options***)** > **= ALL** | **NONE** | *keyword* | *(keyword-list)*

controls the number of posterior statistics that are produced. Specifying STATISTICS=ALL is equivalent to specifying STATISTICS=(CORR COV INTERVAL PRIOR SUMMARY). If you do not want any posterior statistics, specify STATISTICS=NONE. The default is STATISTICS=(SUMMARY INTERVAL). You can specify the following *global-options*:

ALPHA=*value* < *,value* >**...** < *,value* >

controls the probabilities of the credible intervals. The *value*, which must be between 0 and 1, produces a pair of 100(1–*value*)% equal-tail and highest posterior density (HPD) intervals for each parameter. The default is ALPHA=0.05, which yields the 95% credible intervals for each parameter.

PERCENT=*value* < *,value* >**...** < *,value* >

requests the percentile points of the posterior samples. The *value* must be between 0 and 100. The default is PERCENT=25, 50, 75, which yields the 25th, 50th, and 75th percentile points, respectively, for each parameter.

You can specify the following *keywords*:

CORR

produces the posterior correlation matrix.

COV

produces the posterior covariance matrix.

INTERVAL

produces equal-tail credible intervals and HPD intervals. The default is to produce the 95% equal-tail credible intervals and 95% HPD intervals, but you can use the ALPHA= *global-option* to request intervals of any probabilities.

NONE

suppresses printing of all summary statistics.

PRIOR

produces a summary table of the prior distributions that are used in the Bayesian analysis.

SUMMARY

produces the means, standard deviations, and percentile points (25th, 50th, and 75th) for the posterior samples. You can use the PERCENT= *global-option* to request specific percentile points.

THIN=*number*

THINNING=*number*

controls the thinning of the Markov chain. Only one in every k samples is used when THIN=k. If NBI=n_0 and NMC=n, the number of samples that are retained is

$$\left[\frac{n_0 + n}{k}\right] - \left[\frac{n_0}{k}\right]$$

where $[a]$ represents the integer part of the number a. The default is THIN=1.

BOUNDS Statement

BOUNDS *bound1* < *, bound2 . . .* > **;**

The BOUNDS statement imposes simple boundary constraints on the parameter estimates. BOUNDS statement constraints refer to the parameters that are estimated by the HPQLIM procedure. You can specify any number of BOUNDS statements.

Each *bound* is composed of parameters, constants, and inequality operators. Parameters that are associated with regressor variables are referred to by the names of the corresponding regressor variables. Specify each bound as follows:

item operator item < *operator item* < *operator item . . .* > >

Each *item* is a constant, the name of a parameter, or a list of parameter names. For more information about how parameters are named in the HPQLIM procedure, see the section "Naming of Parameters" on page 1182. Each *operator* is <, >, <=, or >=.

You can use both the BOUNDS statement and the RESTRICT statement to impose boundary constraints; however, the BOUNDS statement provides a simpler syntax for specifying these types of constraints. For more information, see the section "RESTRICT Statement" on page 1167.

The following BOUNDS statement constrains the estimates of the parameters that are associated with the variable ttime and the variables x1 through x10 to be between 0 and 1. The following example illustrates the use of parameter lists to specify boundary constraints.

```
bounds 0 < ttime x1-x10 < 1;
```

The following BOUNDS statement constrains the estimates of the correlation (_RHO) and sigma (_SIGMA) in the bivariate model:

```
bounds _rho >= 0, _sigma.y1 > 1, _sigma.y2 < 5;
```

BY Statement

BY *variables* **;**

A BY statement can be used with PROC HPQLIM to obtain separate analyses on observations in groups defined by the BY variables.

BY statement processing is not supported when the HPQLIM procedure runs alongside the database or alongside the Hadoop Distributed File System (HDFS). These modes are used if the input data are stored in a database or HDFS and the grid host is the appliance that houses the data.

ENDOGENOUS Statement

ENDOGENOUS *variables* ∼ *options* **;**

The ENDOGENOUS statement specifies the type of dependent variables that appear on the left-hand side of the equation. The listed endogenous variables refer to the dependent variables that appear on the left-hand side of the equation. Currently, no right-hand-side endogeneity is handled in PROC HPQLIM. All variables that appear on the right-hand side of the equation are treated as exogenous.

Discrete Variable Options

DISCRETE < (*discrete-options* **) >**
specifies that the endogenous variables in this statement be discrete. You can specify the following *discrete-options*:

DISTRIBUTION=*distribution-type*

DIST=*distribution-type*

D=*distribution-type*
specifies the cumulative distribution function that is used to model the response probabilities. You can specify the following *distribution-types*:

LOGISTIC	specifies the logistic distribution for the logit model.
NORMAL	specifies the normal distribution for the probit model.

By default, DISTRIBUTION=NORMAL.

ORDER=DATA | FORMATTED | FREQ | INTERNAL
specifies the sort order for the levels of the discrete variables that are specified in the ENDOGENOUS statement. This ordering determines which parameters in the model correspond to each level in the data. You can specify the following sort orders:

DATA	sorts levels by order of appearance in the input data set.
FORMATTED	sorts levels by formatted value. The sort order is machine-dependent.
FREQ	sorts levels by descending frequency count; levels that have the most observations come first in the order.
INTERNAL	sorts levels by unformatted value. The sort order is machine-dependent.

By default, ORDER=FORMATTED. For more information about sort order, see the chapter on the SORT procedure in the *Base SAS Procedures Guide*.

Censored Variable Options

CENSORED (*censored-options* **)**
specifies that the endogenous variables in this statement be censored. You can specify the following *censored-options*:

LB=*value* | *variable*

LOWERBOUND=*value* | *variable*
specifies the lower bound of the censored variables. If *value* is missing or the value in *variable* is missing, no lower bound is set. By default, no lower bound is set.

UB=*value* | *variable*

UPPERBOUND=*value* | *variable*
specifies the upper bound of the censored variables. If *value* is missing or the value in *variable* is missing, no upper bound is set. By default, no upper bound is set.

Truncated Variable Options

TRUNCATED (*truncated-options* **)**
You can specify the following *truncated-options*:

LB=*value* | *variable*

LOWERBOUND=*value* | *variable*
specifies the lower bound of the truncated variables. If *value* is missing or the value in *variable* is missing, no lower bound is set. By default, no lower bound is set.

UB=*value* | *variable*

UPPERBOUND=*value* | *variable*
specifies the upper bound of the truncated variables. If *value* is missing or the value in *variable* is missing, no upper bound is set. By default, no upper bound is set.

Stochastic Frontier Variable Options

FRONTIER < (*frontier-options* **) >**
You can specify the following *frontier-options*:

TYPE=HALF | EXPONENTIAL | TRUNCATED
specifies the model type.

HALF
specifies half-normal model.

EXPONENTIAL
specifies exponential model.

TRUNCATED
specifies truncated normal model.

PRODUCTION
specifies that the estimated model be a production function.

COST
specifies that the estimated model be a cost function.

If neither PRODUCTION nor COST is specified, a production function is estimated by default.

FREQ Statement

FREQ *variable* **;**

The FREQ statement identifies a variable that contains the frequency of occurrence of each observation. PROC HPQLIM treats each observation as if it appeared n times, where is the value of the FREQ variable for the observation. If the frequency value is not an integer, it is truncated to an integer. If the frequency value is less than 1 or missing, the observation is not used in the model fitting. When the FREQ statement is not specified, each observation is assigned a frequency of 1. If you specify more than one FREQ statement, then the first FREQ statement is used.

HETERO Statement

HETERO *dependent variables* $\sim$ *exogenous variables* < / *options* > **;**

The HETERO statement specifies variables that are related to the heteroscedasticity of the residuals and the way that these variables are used to model the error variance. PROC HPQLIM supports the following heteroscedastic regression model:

$$y_i = \mathbf{x}_i'\boldsymbol{\beta} + \epsilon_i$$

$$\epsilon_i \sim \mathrm{N}(0, \sigma_i^2)$$

For more information about the specification of functional forms, see the section "Heteroscedasticity" on page 1172. The following *options* specify the functional forms of heteroscedasticity:

LINK=EXP | LINEAR
specifies the functional form.

EXP
specifies the exponential link function:

$$\sigma_i^2 = \sigma^2(1 + \exp(\mathbf{z}_i'\boldsymbol{\gamma}))$$

LINEAR
specifies the linear link function:

$$\sigma_i^2 = \sigma^2(1 + \mathbf{z}_i'\boldsymbol{\gamma})$$

The default is LINK=EXP.

NOCONST
specifies that there be no constant in the linear or exponential heteroscedasticity model:

$$\sigma_i^2 = \sigma^2(\mathbf{z}_i'\boldsymbol{\gamma})$$
$$\sigma_i^2 = \sigma^2\exp(\mathbf{z}_i'\boldsymbol{\gamma})$$

This option is ignored if you do not specify the LINK= option.

SQUARE
estimates the model by using the square of the linear heteroscedasticity function. For example, you can specify the following heteroscedasticity function:

$$\sigma_i^2 = \sigma^2(1 + (\mathbf{z}_i'\boldsymbol{\gamma})^2)$$

```
model y = x1 x2 / censored(lb=0);
hetero y ~ z1 / link=linear square;
```

The SQUARE option does not apply to the exponential heteroscedasticity function because the square of an exponential function of $\mathbf{z}_i'\boldsymbol{\gamma}$ is the same as the exponential of $2\mathbf{z}_i'\boldsymbol{\gamma}$. Hence, the only difference is that all $\boldsymbol{\gamma}$ estimates are divided by two.

This option is ignored if you do not specify the LINK= option. You cannot use the HETERO statement within a Bayesian framework.

INIT Statement

INIT *initvalue1* < , *initvalue2* . . . > **;**

The INIT statement sets initial values for parameters in the optimization. You can specify any number of INIT statements.

Each *initvalue* is written as a parameter or parameter list, followed by an optional equality operator (=), followed by a number:

parameter **<=>** *number*

MODEL Statement

MODEL *dependent* **=** *regressors* < / *options* > **;**

The MODEL statement specifies the dependent variable and independent regressor variables for the regression model.

You can specify the following *option* after a slash (/).

NOINT
suppresses the intercept parameter.

You can also specify the following endogenous variable options, which are the same as the options that are specified in the ENDOGENOUS statement. If an endogenous variable option is specified in both the MODEL statement and the ENDOGENOUS statement, the option in the ENDOGENOUS statement is used.

Discrete Variable Options

DISCRETE < (*discrete-options* **) >**
specifies that the endogenous variables in this statement be discrete. You can specify the following *discrete-options*:

DISTRIBUTION=*distribution-type*

DIST=*distribution-type*

D=*distribution-type*
specifies the cumulative distribution function that is used to model the response probabilities. You can specify the following *distribution-types*:

LOGISTIC specifies the logistic distribution for the logit model.

NORMAL specifies the normal distribution for the probit model.

By default, DISTRIBUTION=NORMAL.

ORDER=DATA | FORMATTED | FREQ | INTERNAL
specifies the sort order for the levels of the discrete variables that are specified in the ENDOGENOUS statement. This ordering determines which parameters in the model correspond to each level in the data. You can specify the following sort orders:

DATA sorts levels by order of appearance in the input data set.

FORMATTED sorts levels by formatted value. The sort order is machine-dependent.

FREQ sorts levels by descending frequency count; levels that have the most observations come first in the order.

INTERNAL sorts levels by unformatted value. The sort order is machine-dependent.

By default, ORDER=FORMATTED. For more information about sort order, see the chapter on the SORT procedure in the *Base SAS Procedures Guide*.

Censored Variable Options

CENSORED < (*censored-options* **) >**
specifies that the endogenous variables in this statement be censored. You can specify the following *censored-options*:

LB=*value* | *variable*

LOWERBOUND=*value* | *variable*
specifies the lower bound of the censored variables. If *value* is missing or the value in *variable* is missing, no lower bound is set. By default, no lower bound is set.

UB=*value* | *variable*

UPPERBOUND=*value* | *variable*
specifies the upper bound of the censored variables. If *value* is missing or the value in *variable* is missing, no upper bound is set. By default, no upper bound is set.

Truncated Variable Options

TRUNCATED < (*truncated-options* **) >**
You can specify the following *truncated-options*:

LB=*value* | *variable*

LOWERBOUND=*value* | *variable*
specifies the lower bound of the truncated variables. If *value* is missing or the value in *variable* is missing, no lower bound is set. By default, no lower bound is set.

UB=*value* | *variable*

UPPERBOUND=*value* | *variable*
specifies the upper bound of the truncated variables. If *value* is missing or the value in *variable* is missing, no upper bound is set. By default, no upper bound is set.

Stochastic Frontier Variable Options

FRONTIER < (*frontier-options* **) >**
You can specify the following *frontier-options*:

TYPE=HALF | EXPONENTIAL | TRUNCATED
specifies the model type.

HALF
specifies a half-normal model.

EXPONENTIAL
specifies an exponential model.

TRUNCATED
specifies a truncated normal model.

PRODUCTION
specifies that the estimated model be a production function.

COST
specifies that the estimated model be a cost function.

If neither PRODUCTION nor COST is specified, a production function is estimated by default.

OUTPUT Statement

OUTPUT OUT=*SAS-data-set* < *output-options* > **;**

The OUTPUT statement creates a new SAS data set to contain variables that are specified with the COPYVAR option and the following data if they are specified by *output-options*: estimates of $\mathbf{x}'\boldsymbol{\beta}$, predicted value, residual, marginal effects, probability, standard deviation of the error, expected value, conditional expected value, technical efficiency measures, and inverse Mills ratio. When the response values are missing for the observation, all output estimates except the residual are still computed as long as none of the explanatory variables are missing. This enables you to compute these statistics for prediction. You can specify only one OUTPUT statement.

You must specify the OUT= option:

OUT=*SAS-data-set*
names the output data set.

COPYVAR=*SAS-variable-names*

COPYVARS=(*SAS-variable-names***)**
adds SAS variables to the output data set

You can specify one or more of the following *output-options*:

CONDITIONAL
outputs estimates of conditional expected values of continuous endogenous variables.

ERRSTD
outputs estimates of σ_j, the standard deviation of the error term.

EXPECTED
outputs estimates of expected values of continuous endogenous variables.

MARGINAL
outputs marginal effects.

MILLS
outputs estimates of inverse Mills ratios of censored or truncated continuous, binary discrete, and selection endogenous variables.

PREDICTED
outputs estimates of predicted endogenous variables.

PROB
outputs estimates of probability of discrete endogenous variables taking the current observed responses.

PROBALL
outputs estimates of probability of discrete endogenous variables for all possible responses.

RESIDUAL
outputs estimates of residuals of continuous endogenous variables.

XBETA
outputs estimates of $\mathbf{x}'\boldsymbol{\beta}$.

TE1
outputs estimates of technical efficiency for each producer in the stochastic frontier model that is suggested by Battese and Coelli (1988).

TE2
outputs estimates of technical efficiency for each producer in the stochastic frontier model that is suggested by Jondrow et al. (1982).

PERFORMANCE Statement

PERFORMANCE < *performance-options* > **;**

The PERFORMANCE statement specifies *performance-options* to control the multithreaded and distributed computing environment and requests detailed performance results of the HPQLIM procedure. You can also use the PERFORMANCE statement to control whether the HPQLIM procedure executes in single-machine or distributed mode. You can specify the following *performance-options*:

DETAILS
requests a table that shows a timing breakdown of the procedure steps.

NODES=*n*
specifies the number of nodes in the distributed computing environment, provided that the data are not processed alongside the database.

NTHREADS=*n*
specifies the number of threads for analytic computations and overrides the SAS system option THREADS | NOTHREADS. If you do not specify the NTHREADS= option, PROC HPQLIM creates one thread per CPU for the analytic computations.

The PERFORMANCE statement is documented further in the section "PERFORMANCE Statement" on page 87 in Chapter 3, "Shared Concepts and Topics."

PRIOR Statement

PRIOR _REGRESSORS | *parameter-list* ~ *distribution* **;**

The PRIOR statement specifies the prior distribution of the model parameters. You must specify one parameter or a list of parameters, a tilde ~, and then a distribution with its parameters. Multiple PRIOR statements are allowed.

You can specify the following *distributions*:

NORMAL(MEAN=μ, VAR=σ^2)
specifies a normal distribution with the parameters MEAN and VAR.

GAMMA(SHAPE=*a*, SCALE=*b*)
specifies a gamma distribution with the parameters SHAPE and SCALE.

IGAMMA(SHAPE=*a*, SCALE=*b*)
specifies an inverse gamma distribution with the parameters SHAPE and SCALE.

UNIFORM(MIN=*m*, MAX=*M*)
specifies a uniform distribution that is defined between MIN and MAX.

BETA(SHAPE1=*a*, SHAPE2=*b*, MIN=*m*, MAX=*M*)
specifies a beta distribution with the parameters SHAPE1 and SHAPE2 and defined between MIN and MAX.

T(LOCATION=μ, DF=ν)
specifies a noncentral *t* distribution with DF degrees of freedom and a location parameter equal to LOCATION.

For more information about how to specify *distributions*, see the section "Standard Distributions" on page 1175.

You can specify the special keyword REGRESSORS to select all the parameters that are used in the linear regression component of the model.

RESTRICT Statement

RESTRICT *restriction1* < , *restriction2* . . . > **;**

The RESTRICT statement imposes linear restrictions on the parameter estimates. You can specify any number of RESTRICT statements, but the number of restrictions that are imposed is limited by the number of regressors.

Each *restriction* is written as an expression, followed by an equality operator (=) or an inequality operator (<, >, <=, >=), followed by a second expression:

expression operator expression

The *operator* can be =, <, >, <= , or >=. The *operator* and second *expression* are optional.

Restriction expressions can be composed of parameter names; multiplication ($*$), addition ($+$), and substitution ($-$) operators; and constants. Parameters that are named in restriction expressions must be among the parameters that are estimated by the model. Parameters that are associated with a regressor variable are referred to by the name of the corresponding regressor variable. The restriction expressions must be a linear function of the parameters.

The following statements illustrate the use of the RESTRICT statement:

```
proc hpqlim data=one;
   model y = x1-x10 / censored(lb=0);
   restrict x1*2 <= x2 + x3;
run;
```

TEST Statement

<'label':> **TEST** *<'string':> equation < ,equation. . . > / options* **;**

The TEST statement performs Wald, Lagrange multiplier, and likelihood ratio tests of linear hypotheses about the regression parameters in the preceding MODEL statement. Each equation specifies a linear hypothesis to be tested. All hypotheses in one TEST statement are tested jointly. Variable names in the equations must correspond to regressors in the preceding MODEL statement, and each name represents the coefficient of the corresponding regressor. Use the keyword INTERCEPT for a test that includes a constant.

You can specify the following *options* after the slash (/):

ALL
requests Wald, Lagrange multiplier, and likelihood ratio tests.

LM
requests the Lagrange multiplier test.

LR
requests the likelihood ratio test.

WALD
requests the Wald test.

The following statements illustrate the use of the TEST statement (note the use of the INTERCEPT keyword in the second TEST statement):

```
proc hpqlim;
   model y = x1 x2 x3;
   test x1 = 0, x2 * .5 + 2 * x3 = 0;
   test _int: test intercept = 0, x3 = 0;
run;
```

The first TEST statement investigates the joint hypothesis that

$$\beta_1 = 0$$

and

$$0.5\beta_2 + 2\beta_3 = 0$$

Only linear equality restrictions and tests are permitted in PROC HPQLIM. Test expressions can be composed only of algebraic operations that involve the addition symbol (+), subtraction symbol (–), and multiplication symbol (*).

The TEST statement accepts labels that are reproduced in the printed output. You can label a TEST statement in two ways: you can specify a label followed by a colon before the TEST keyword, or you can specify a quoted string after the TEST keyword. If you specify both a label before the TEST keyword and a quoted string after the keyword, PROC HPQLIM uses the label that precedes the colon. If no label or quoted string is specified, PROC HPQLIM labels the test automatically.

WEIGHT Statement

WEIGHT *variable* < / *option* > **;**

The WEIGHT statement specifies a variable that supplies weighting values to use for each observation in estimating parameters. The log likelihood for each observation is multiplied by the corresponding weight variable value.

If the weight of an observation is nonpositive, that observation is not used in the estimation.

You can add the following *option* after a slash (/):

NONORMALIZE
specifies that the weights must be used as is. When this option is not specified, the weights are normalized so that they add up to the actual sample size. Weights w_i are normalized by multiplying them by $\frac{n}{\sum_{i=1}^{n} w_i}$, where n is the sample size.

Details: HPQLIM Procedure

Ordinal Discrete Choice Modeling

Binary Probit and Logit Model

The binary choice model is

$$y_i^* = \mathbf{x}_i'\boldsymbol{\beta} + \epsilon_i$$

where the value of the latent dependent variable, y_i^*, is observed only as follows:

$$\begin{aligned} y_i &= 1 \quad \text{if } y_i^* > 0 \\ &= 0 \quad \text{otherwise} \end{aligned}$$

The disturbance, ϵ_i, of the probit model has a standard normal distribution with the distribution function (CDF)

$$\Phi(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} \exp(-t^2/2)dt$$

The disturbance of the logit model has a standard logistic distribution with the distribution function (CDF)

$$\Lambda(x) = \frac{\exp(x)}{1+\exp(x)} = \frac{1}{1+\exp(-x)}$$

The binary discrete choice model has the following probability that the event $\{y_i = 1\}$ occurs:

$$P(y_i = 1) = F(\mathbf{x}_i'\boldsymbol{\beta}) = \begin{cases} \Phi(\mathbf{x}_i'\boldsymbol{\beta}) & \text{(probit)} \\ \Lambda(\mathbf{x}_i'\boldsymbol{\beta}) & \text{(logit)} \end{cases}$$

For more information, see the section "Ordinal Discrete Choice Modeling" on page 1985.

Ordinal Probit/Logit

When the dependent variable is observed in sequence with M categories, binary discrete choice modeling is not appropriate for data analysis. McKelvey and Zavoina (1975) propose the ordinal (or ordered) probit model.

Consider the regression equation

$$y_i^* = \mathbf{x}_i'\boldsymbol{\beta} + \epsilon_i$$

where error disturbances, ϵ_i, have the distribution function F. The unobserved continuous random variable, y_i^*, is identified as M categories. Suppose there are $M+1$ real numbers, $\mu_0, \ldots, \mu_M$, where $\mu_0 = -\infty$, $\mu_1 = 0$, $\mu_M = \infty$, and $\mu_0 \le \mu_1 \le \cdots \le \mu_M$. Define

$$R_{i,j} = \mu_j - \mathbf{x}_i'\boldsymbol{\beta}$$

The probability that the unobserved dependent variable is contained in the jth category can be written as

$$P[\mu_{j-1} < y_i^* \le \mu_j] = F(R_{i,j}) - F(R_{i,j-1})$$

For more information, see the section "Ordinal Discrete Choice Modeling" on page 1985.

Limited Dependent Variable Models

Censored Regression Models

When the dependent variable is censored, values in a certain range are all transformed to a single value. For example, the standard Tobit model can be defined as

$$y_i^* = \mathbf{x}_i'\boldsymbol{\beta} + \epsilon_i$$

$$y_i = \begin{cases} y_i^* & \text{if } y_i^* > 0 \\ 0 & \text{if } y_i^* \le 0 \end{cases}$$

where $\epsilon_i \sim iidN(0, \sigma^2)$.

The Tobit model can be generalized to handle observation-by-observation censoring. The censored model on both the lower and upper limits can be defined as

$$y_i = \begin{cases} R_i & \text{if } y_i^* \ge R_i \\ y_i^* & \text{if } L_i < y_i^* < R_i \\ L_i & \text{if } y_i^* \le L_i \end{cases}$$

You can see Chapter 29.7, "Censored Regression Models," for more details.

Truncated Regression Models

In a truncated model, the observed sample is a subset of the population where the dependent variable falls within a certain range. For example, when neither a dependent variable nor exogenous variables are observed for $y_i^* \le 0$, the truncated regression model can be specified as

$$\ell = \sum_{i \in \{y_i > 0\}} \left\{ -\ln \Phi(\mathbf{x}_i'\boldsymbol{\beta}/\sigma) + \ln\left[\frac{\phi((y_i - \mathbf{x}_i'\boldsymbol{\beta})/\sigma)}{\sigma}\right]\right\}$$

For more information, see the section "Truncated Regression Models" on page 1990.

Stochastic Frontier Production and Cost Models

Stochastic frontier production models were first developed by Aigner, Lovell, and Schmidt (1977); Meeusen and van den Broeck (1977). Specification of these models allow for random shocks of the production or cost but also include a term for technical or cost inefficiency. Assuming that the production function takes a log-linear Cobb-Douglas form, the stochastic frontier production model can be written as

$$ln(y_i) = \beta_0 + \sum_n \boldsymbol{\beta_n} \ln(x_{ni}) + \epsilon_i$$

where $\epsilon_i = v_i - u_i$. The v_i term represents the stochastic error component, and the u_i term represents the nonnegative, technical inefficiency error component. The v_i error component is assumed to be distributed iid normal and independent from u_i. If $u_i > 0$, the error term ϵ_i is negatively skewed and represents technical inefficiency. If $u_i < 0$, the error term ϵ_i is positively skewed and represents cost inefficiency. PROC HPQLIM models the u_i error component as a half-normal, exponential, or truncated normal distribution.

The Normal-Half-Normal Model

When v_i is iid $N(0, \sigma_v^2)$ in a normal-half-normal model, u_i is iid $N^+(0, \sigma_u^2)$, with v_i and u_i independent of each other. Given the independence of error terms, the joint density of v and u can be written as

$$f(u, v) = \frac{2}{2\pi\sigma_u\sigma_v} \exp\left\{-\frac{u^2}{2\sigma_u^2} - \frac{v^2}{2\sigma_v^2}\right\}$$

Substituting $v = \epsilon + u$ into the preceding equation and integrating u out gives

$$f(\epsilon) = \frac{2}{\sigma}\phi\left(\frac{\epsilon}{\sigma}\right)\Phi\left(-\frac{\epsilon\lambda}{\sigma}\right)$$

where $\lambda = \sigma_u/\sigma_v$ and $\sigma = \sqrt{\sigma_u^2 + \sigma_v^2}$.

In the case of a stochastic frontier cost model, $v = \epsilon - u$ and

$$f(\epsilon) = \frac{2}{\sigma}\phi\left(\frac{\epsilon}{\sigma}\right)\Phi\left(\frac{\epsilon\lambda}{\sigma}\right)$$

For more information, see the section "Stochastic Frontier Production and Cost Models" on page 1991.

The Normal-Exponential Model

Under the normal-exponential model, v_i is iid $N(0, \sigma_v^2)$ and u_i is iid exponential. Given the independence of error term components u_i and v_i, the joint density of v and u can be written as

$$f(u, v) = \frac{1}{\sqrt{2\pi}\sigma_u\sigma_v} \exp\left\{-\frac{u}{\sigma_u} - \frac{v^2}{2\sigma_v^2}\right\}$$

The marginal density function of ϵ for the production function is

$$f(\epsilon) = \left(\frac{1}{\sigma_u}\right)\Phi\left(-\frac{\epsilon}{\sigma_v} - \frac{\sigma_v}{\sigma_u}\right)\exp\left\{\frac{\epsilon}{\sigma_u} + \frac{\sigma_v^2}{2\sigma_u^2}\right\}$$

The marginal density function for the cost function is equal to

$$f(\epsilon) = \left(\frac{1}{\sigma_u}\right)\Phi\left(\frac{\epsilon}{\sigma_v} - \frac{\sigma_v}{\sigma_u}\right)\exp\left\{-\frac{\epsilon}{\sigma_u} + \frac{\sigma_v^2}{2\sigma_u^2}\right\}$$

For more information, see the section "Stochastic Frontier Production and Cost Models" on page 1991.

The Normal–Truncated Normal Model

The normal–truncated normal model is a generalization of the normal-half-normal model that allows the mean of u_i to differ from zero. Under the normal–truncated normal model, the error term component v_i is iid $N^+(0, \sigma_v^2)$ and u_i is iid $N(\mu, \sigma_u^2)$. The joint density of v_i and u_i can be written as

$$f(u, v) = \frac{1}{\sqrt{2\pi}\sigma_u\sigma_v\Phi\left(\mu/\sigma_u\right)}\exp\left\{-\frac{(u-\mu)^2}{2\sigma_u^2} - \frac{v^2}{2\sigma_v^2}\right\}$$

The marginal density function of ϵ for the production function is

$$f(\epsilon) = \frac{1}{\sigma}\phi\left(\frac{\epsilon+\mu}{\sigma}\right)\Phi\left(\frac{\mu}{\sigma\lambda} - \frac{\epsilon\lambda}{\sigma}\right)\left[\Phi\left(\frac{\mu}{\sigma_u}\right)\right]^{-1}$$

The marginal density function for the cost function is

$$f(\epsilon) = \frac{1}{\sigma}\phi\left(\frac{\epsilon-\mu}{\sigma}\right)\Phi\left(\frac{\mu}{\sigma\lambda} + \frac{\epsilon\lambda}{\sigma}\right)\left[\Phi\left(\frac{\mu}{\sigma_u}\right)\right]^{-1}$$

For more information, see the section "Stochastic Frontier Production and Cost Models" on page 1991.

For more information about normal-half-normal, normal-exponential, and normal–truncated normal models, see Kumbhakar and Lovell (2000); Coelli, Prasada Rao, and Battese (1998).

Heteroscedasticity

If the variance of regression disturbance, (ϵ_i), is heteroscedastic, the variance can be specified as a function of variables

$$E(\epsilon_i^2) = \sigma_i^2 = f(\mathbf{z}_i'\boldsymbol{\gamma})$$

Table 22.2 shows various functional forms of heteroscedasticity and the corresponding options to request each model.

Table 22.2 Specification Summary for Modeling Heteroscedasticity

Number	Model	Options
1	$f(\mathbf{z}_i'\boldsymbol{\gamma}) = \sigma^2(1 + \exp(\mathbf{z}_i'\gamma))$	LINK=EXP (default)
2	$f(\mathbf{z}_i'\boldsymbol{\gamma}) = \sigma^2\exp(\mathbf{z}_i'\gamma)$	LINK=EXP NOCONST
3	$f(\mathbf{z}_i'\boldsymbol{\gamma}) = \sigma^2(1 + \sum_{l=1}^{L}\gamma_l z_{li})$	LINK=LINEAR
4	$f(\mathbf{z}_i'\boldsymbol{\gamma}) = \sigma^2(1 + (\sum_{l=1}^{L}\gamma_l z_{li})^2)$	LINK=LINEAR SQUARE
5	$f(\mathbf{z}_i'\boldsymbol{\gamma}) = \sigma^2(\sum_{l=1}^{L}\gamma_l z_{li})$	LINK=LINEAR NOCONST
6	$f(\mathbf{z}_i'\boldsymbol{\gamma}) = \sigma^2((\sum_{l=1}^{L}\gamma_l z_{li})^2)$	LINK=LINEAR SQUARE NOCONST

In models 3 and 5, variances of some observations might be negative. Although the HPQLIM procedure assigns a large penalty to move the optimization away from such a region, the optimization might not be able to improve the objective function value and might become locked in the region. Signs of such an outcome include extremely small likelihood values or missing standard errors in the estimates. In models 2 and 6, variances are guaranteed to be greater than or equal to zero, but variances of some observations might be very close to 0. In these scenarios, standard errors might be missing. Models 1 and 4 do not have such problems. Variances in these models are always positive and never close to 0.

For more information, see the section "Heteroscedasticity and Box-Cox Transformation" on page 1993.

Tests on Parameters

In general, the tested hypothesis can be written as

$$H_0 : \mathbf{h}(\theta) = 0$$

where $\mathbf{h}(\theta)$ is an $r \times 1$ vector-valued function of the parameters θ given by the r expressions that are specified in the TEST statement.

Let $\hat{V}$ be the estimate of the covariance matrix of $\hat{\theta}$. Let $\hat{\theta}$ be the unconstrained estimate of θ and $\tilde{\theta}$ be the constrained estimate of θ such that $h(\tilde{\theta}) = 0$. Let

$$A(\theta) = \partial h(\theta)/\partial\theta \mid_{\hat{\theta}}$$

Using this notation, the test statistics for the three types of tests are computed as follows.

- The Wald test statistic is defined as

  $$W = h'(\hat{\theta}) \left(A(\hat{\theta}) \hat{V} A'(\hat{\theta}) \right)^{-1} h(\hat{\theta})$$

- The Lagrange multiplier test statistic is

  $$LM = \lambda' A(\tilde{\theta}) \tilde{V} A'(\tilde{\theta}) \lambda$$

  where λ is the vector of Lagrange multipliers from the computation of the restricted estimate $\tilde{\theta}$.

- The likelihood ratio test statistic is

  $$LR = 2 \left(L(\hat{\theta}) - L(\tilde{\theta}) \right)$$

  where $\tilde{\theta}$ represents the constrained estimate of θ and L is the concentrated log-likelihood value.

The following statements use the TEST statement to perform a likelihood ratio test:

```
proc hpqlim;
   model y = x1 x2 x3;
   test x1 = 0, x2 * .5 + 2 * x3 = 0 /lr;
run;
```

For more information, see the section "Tests on Parameters" on page 2000.

Bayesian Analysis

To perform Bayesian analysis, you must specify a BAYES statement. Unless otherwise stated, all options that are described in this section are options in the BAYES statement.

By default, PROC HPQLIM uses the random walk Metropolis algorithm to obtain posterior samples. For the implementation details of the Metropolis algorithm in PROC HPQLIM, such as the blocking of the parameters and tuning of the covariance matrices, see the sections "Blocking of Parameters" on page 1174 and "Tuning the Proposal Distribution" on page 1174.

The Bayes theorem states that

$$p(\theta|\mathbf{y}) \propto \pi(\theta)L(y|\theta)$$

where θ is a parameter or a vector of parameters and $\pi(\theta)$ is the product of the prior densities that are specified in the PRIOR statement. The term $L(y|\theta)$ is the likelihood that is associated with the MODEL statement.

Blocking of Parameters

In a multivariate parameter model, all the parameters are updated in one single block (by default or when you specify the SAMPLING=MULTIMETROPOLIS option). This can be inefficient, especially when parameters have vastly different scales. As an alternative, you can update the parameters one at a time (by specifying SAMPLING=UNIMETROPOLIS).

Tuning the Proposal Distribution

One key factor in achieving high efficiency of a Metropolis-based Markov chain is finding a good proposal distribution for each block of parameters. This process is called tuning. The tuning phase consists of a number of loops that are controlled by the options MINTUNE and MAXTUNE. The MINTUNE= option controls the minimum number of tuning loops and has a default value of 2. The MAXTUNE= option controls the maximum number of tuning loops and has a default value of 24. Each loop repeats the number of times specified by the NTU= option, which has a default of 500. At the end of every loop, PROC HPQLIM examines the acceptance probability for each block. The acceptance probability is the percentage of NTU proposed values that have been accepted. If this probability does not fall within the acceptance tolerance range (see the following section), the proposal distribution is modified before the next tuning loop.

A good proposal distribution should resemble the actual posterior distribution of the parameters. Large sample theory states that the posterior distribution of the parameters approaches a multivariate normal distribution (Gelman et al. 2004, Appendix B; Schervish 1995, Section 7.4). That is why a normal proposal distribution often works well in practice. The default proposal distribution in PROC HPQLIM is the normal distribution.

You can see Chapter 29.7, "Bayesian Analysis," for more details.

Initial Values of the Markov Chains

You can assign initial values to any parameters. For more information, see the INIT statement. If you use the optimization PROPCOV= option, PROC HPQLIM starts the tuning at the optimized values. This option overwrites the provided initial values.

Prior Distributions

The PRIOR statement specifies the prior distribution of the model parameters. You must specify one parameter or a list of parameters, a tilde ∼, and then a distribution with its parameters. You can specify multiple PRIOR statements to define independent priors. Parameters that are associated with a regressor variable are referred to by the name of the corresponding regressor variable.

You can specify the special keyword _REGRESSORS to consider all the regressors of a model. If multiple PRIOR statements affect the same parameter, the last PRIOR statement prevails. For example, in a regression with two regressors (X1, X2), the following statements imply that the prior on X1 is NORMAL(MEAN=0, VAR=1), the prior on X2 is GAMMA(SHAPE=3, SCALE=4).

```
...
prior _Regressors ~ uniform(min=0, max=1);
prior X1 X2 ~ gamma(shape=3, scale=4);
prior X1 ~ normal(mean=0, var=1);
...
```

If a parameter is not associated with a PRIOR statement or if some of the prior hyperparameters are missing, then the default choices in Table Table 22.3 are considered.

Table 22.3 Default Values for Prior Distributions

PRIOR Distribution	Hyperparameter$_1$	Hyperparameter$_2$	Min	Max	Parameters Default Choice
NORMAL	MEAN=0	VAR=1E6	$-\infty$	∞	Regression-Location-Threshold
IGAMMA	SHAPE=2.000001	SCALE=1	> 0	∞	Scale
GAMMA	SHAPE=1	SCALE=1	0	∞	
UNIFORM			$-\infty$	∞	
BETA	SHAPE1=1	SHAPE2=1	$-\infty$	∞	
T	LOCATION=0	DF=3	$-\infty$	∞	

For density specification, see the section "Standard Distributions" on page 1175.

Standard Distributions

Table 22.4 through Table 22.9 show all the distribution density functions that PROC HPQLIM recognizes. You specify these distribution densities in the PRIOR statement.

Table 22.4 Beta Distribution

PRIOR statement	BETA(SHAPE1=a, SHAPE2=b, MIN=m, MAX=M) Note: Commonly $m = 0$ and $M = 1$.
Density	$\frac{(\theta-m)^{a-1}(M-\theta)^{b-1}}{B(a,b)(M-m)^{a+b-1}}$
Parameter restriction	$a > 0, \quad b > 0, \quad -\infty < m < M < \infty$

Range	$\begin{cases} [m, M] & \text{when } a = 1, b = 1 \\ [m, M) & \text{when } a = 1, b \neq 1 \\ (m, M] & \text{when } a \neq 1, b = 1 \\ (m, M) & \text{otherwise} \end{cases}$
Mean	$\frac{a}{a+b} \times (M - m) + m$
Variance	$\frac{ab}{(a+b)^2(a+b+1)} \times (M - m)^2$
Mode	$\begin{cases} \frac{a-1}{a+b-2} \times M + \frac{b-1}{a+b-2} \times m & a > 1, b > 1 \\ m \text{ and } M & a < 1, b < 1 \\ m & \begin{cases} a < 1, b \geq 1 \\ a = 1, b > 1 \end{cases} \\ M & \begin{cases} a \geq 1, b < 1 \\ a > 1, b = 1 \end{cases} \\ \text{not unique} & a = b = 1 \end{cases}$
Defaults	SHAPE1=SHAPE2=1, MIN $\rightarrow -\infty$, MAX $\rightarrow \infty$

Table 22.5 Gamma Distribution

PRIOR statement	GAMMA(SHAPE=a, SCALE=b)
Density	$\frac{1}{b^a \Gamma(a)} \theta^{a-1} e^{-\theta/b}$
Parameter restriction	$a > 0, b > 0$
Range	$[0, \infty)$
Mean	ab
Variance	ab^2
Mode	$(a - 1)b$
Defaults	SHAPE=SCALE=1

Table 22.6 Inverse Gamma Distribution

PRIOR statement	IGAMMA(SHAPE=a, SCALE=b)
Density	$\frac{b^a}{\Gamma(a)}\theta^{-(a+1)}e^{-b/\theta}$
Parameter restriction	$a > 0, b > 0$
Range	$0 < \theta < \infty$
Mean	$\frac{b}{a-1}, \quad a > 1$
Variance	$\frac{b^2}{(a-1)^2(a-2)}, \quad a > 2$
Mode	$\frac{b}{a+1}$
Defaults	SHAPE=2.000001, SCALE=1

Table 22.7 Normal Distribution

PRIOR statement	NORMAL(MEAN=μ, VAR=σ^2)
Density	$\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(\theta-\mu)^2}{2\sigma^2}\right)$
Parameter restriction	$\sigma^2 > 0$
Range	$-\infty < \theta < \infty$
Mean	μ
Variance	σ^2
Mode	μ
Defaults	MEAN=0, VAR=1000000

Table 22.8 *t* Distribution

PRIOR statement	T(LOCATION=μ, DF=ν)
Density	$\frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)\sqrt{\pi\nu}}\left[1+\frac{(\theta-\mu)^2}{\nu}\right]^{-\frac{\nu+1}{2}}$
Parameter restriction	$\nu > 0$
Range	$-\infty < \theta < \infty$
Mean	μ, for $\nu > 1$
Variance	$\frac{\nu}{\nu-2}$, for $\nu > 2$
Mode	μ
Defaults	LOCATION=0, DF=3

Table 22.9 Uniform Distribution

PRIOR statement	UNIFORM(MIN=m, MAX=M)
Density	$\frac{1}{M-m}$
Parameter restriction	$-\infty < m < M < \infty$
Range	$\theta \in [m, M]$
Mean	$\frac{m+M}{2}$
Variance	$\frac{(M-m)^2}{12}$
Mode	Not unique
Defaults	MIN$\rightarrow -\infty$, MAX$\rightarrow \infty$

Output to SAS Data Set

XBeta, Predicted, and Residual

Xbeta is the structural part on the right-hand side of the model. The predicted value is the predicted dependent variable value. For censored variables, if the predicted value is outside the boundaries, it is reported as the closest boundary. The residual is defined only for continuous variables and is defined as

$$\text{Residual} = \text{Observed} - \text{Predicted}$$

Error Standard Deviation

The error standard deviation is σ_i in the model. It varies only when the HETERO statement is used.

Marginal Effects

A marginal effect is defined as a contribution of one control variable to the response variable. For a binary choice model with two response categories, $\mu_0 = -\infty$ and $\mu_1 = 0$, $\mu_2 = \infty$. For an ordinal response model with M response categories $(\mu_0, \cdots, \mu_M)$, define

$$R_{i,j} = \mu_j - \mathbf{x}_i'\boldsymbol{\beta}$$

The probability that the unobserved dependent variable is contained in the jth category can be written as

$$P[\mu_{j-1} < y_i^* \le \mu_j] = F(R_{i,j}) - F(R_{i,j-1})$$

The marginal effect of changes in the regressors on the probability of $y_i = j$ is then

$$\frac{\partial \text{Prob}[y_i = j]}{\partial \mathbf{x}} = [f(\mu_{j-1} - \mathbf{x}_i'\boldsymbol{\beta}) - f(\mu_j - \mathbf{x}_i'\boldsymbol{\beta})]\boldsymbol{\beta}$$

where $f(x) = \frac{dF(x)}{dx}$. In particular,

$$f(x) = \frac{dF(x)}{dx} = \begin{cases} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} & \text{(probit)} \\ \frac{e^{-x}}{[1+e^{(-x)}]^2} & \text{(logit)} \end{cases}$$

The marginal effects in the truncated regression model are

$$\frac{\partial E[y_i|L_i < y_i^* < R_i]}{\partial \mathbf{x}} = \boldsymbol{\beta}\left[1 - \frac{(\phi(a_i) - \phi(b_i))^2}{(\Phi(b_i) - \Phi(a_i))^2} + \frac{a_i\phi(a_i) - b_i\phi(b_i)}{\Phi(b_i) - \Phi(a_i)}\right]$$

where $a_i = \frac{L_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma_i}$ and $b_i = \frac{R_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma_i}$.

The marginal effects in the censored regression model are

$$\frac{\partial E[y|\mathbf{x}_i]}{\partial \mathbf{x}} = \boldsymbol{\beta} \times \text{Prob}[L_i < y_i^* < R_i]$$

Expected and Conditionally Expected Values

The expected value is the unconditional expectation of the dependent variable. For a censored variable, it is

$$E[y_i] = \Phi(a_i)L_i + (\mathbf{x}_i'\boldsymbol{\beta} + \lambda\sigma_i)(\Phi(b_i) - \Phi(a_i)) + (1 - \Phi(b_i))R_i$$

For a left-censored variable ($R_i = \infty$), this formula is

$$E[y_i] = \Phi(a_i)L_i + (\mathbf{x}_i'\boldsymbol{\beta} + \lambda\sigma_i)(1 - \Phi(a_i))$$

where $\lambda = \frac{\phi(a_i)}{1-\Phi(a_i)}$.

For a right-censored variable ($L_i = -\infty$), this formula is

$$E[y_i] = (\mathbf{x}_i'\boldsymbol{\beta} + \lambda\sigma_i)\Phi(b_i) + (1 - \Phi(b_i))R_i$$

where $\lambda = -\frac{\phi(b_i)}{\Phi(b_i)}$.

For a noncensored variable, this formula is

$$E[y_i] = \mathbf{x}_i'\boldsymbol{\beta}$$

The conditional expected value is the expectation when the variable is inside the boundaries:

$$E[y_i|L_i < y_i < R_i] = \mathbf{x}_i'\boldsymbol{\beta} + \lambda\sigma_i$$

Technical Efficiency

Technical efficiency for each producer is computed only for stochastic frontier models.

In general, the stochastic production frontier can be written as

$$y_i = f(x_i;\beta)\exp\{v_i\}TE_i$$

where y_i denotes producer i's actual output, $f(\cdot)$ is the deterministic part of the production frontier, $\exp\{v_i\}$ is a producer-specific error term, and TE_i is the technical efficiency coefficient, which can be written as

$$TE_i = \frac{y_i}{f(x_i;\beta)\exp\{v_i\}}$$

For a Cobb-Douglas production function, $TE_i = \exp\{-u_i\}$. For more information, see the section "Stochastic Frontier Production and Cost Models" on page 1171.

The cost frontier can be written in general as

$$E_i = c(y_i, w_i; \beta) \exp\{v_i\}/CE_i$$

where w_i denotes producer i's input prices, $c(\cdot)$ is the deterministic part of the cost frontier, $\exp\{v_i\}$ is a producer-specific error term, and CE_i is the cost efficiency coefficient, which can be written as

$$CE_i = \frac{c(x_i, w_i; \beta) \exp\{v_i\}}{E_i}$$

For a Cobb-Douglas cost function, $CE_i = \exp\{-u_i\}$. For more information, see the section "Stochastic Frontier Production and Cost Models" on page 1171. Hence, both technical and cost efficiency coefficients are the same. The estimates of technical efficiency are provided in the following subsections.

Normal-Half-Normal Model

Define $\mu_* = -\epsilon\sigma_u^2/\sigma^2$ and $\sigma_*^2 = \sigma_u^2\sigma_v^2/\sigma^2$. Then, as shown by Jondrow et al. (1982), conditional density is as follows:

$$f(u|\epsilon) = \frac{f(u,\epsilon)}{f(\epsilon)} = \frac{1}{\sqrt{2\pi}\sigma_*} \exp\left\{-\frac{(u-\mu_*)^2}{2\sigma_*^2}\right\} \Big/ \left[1 - \Phi\left(-\frac{\mu_*}{\sigma_*}\right)\right]$$

Hence, $f(u|\epsilon)$ is the density for $N^+(\mu_*, \sigma_*^2)$.

From this result, it follows that the estimate of technical efficiency (Battese and Coelli 1988) is

$$TE1_i = E(\exp\{-u_i\}|\epsilon_i) = \left[\frac{1 - \Phi(\sigma_* - \mu_{*i}/\sigma_*)}{1 - \Phi(-\mu_{*i}/\sigma_*)}\right] \exp\left\{-\mu_{*i} + \frac{1}{2}\sigma_*^2\right\}$$

The second version of the estimate (Jondrow et al. 1982) is

$$TE2_i = \exp\{-E(u_i|\epsilon_i)\}$$

where

$$E(u_i|\epsilon_i) = \mu_{*i} + \sigma_* \left[\frac{\phi(-\mu_{*i}/\sigma_*)}{1 - \Phi(-\mu_{*i}/\sigma_*)}\right] = \sigma_* \left[\frac{\phi(\epsilon_i\lambda/\sigma)}{1 - \Phi(\epsilon_i\lambda/\sigma)} - \left(\frac{\epsilon_i\lambda}{\sigma}\right)\right]$$

Normal-Exponential Model

Define $A = -\tilde{\mu}/\sigma_v$ and $\tilde{\mu} = -\epsilon - \sigma_v^2/\sigma_u$. Then, as shown by Kumbhakar and Lovell (2000), conditional density is as follows:

$$f(u|\epsilon) = \frac{1}{\sqrt{2\pi}\sigma_v \Phi(-\tilde{\mu}/\sigma_v)} \exp\left\{-\frac{(u-\tilde{\mu})^2}{2\sigma^2}\right\}$$

Hence, $f(u|\epsilon)$ is the density for $N^+(\tilde{\mu}, \sigma_v^2)$.

From this result, it follows that the estimate of technical efficiency is

$$TE1_i = E(\exp\{-u_i\}|\epsilon_i) = \left[\frac{1 - \Phi(\sigma_v - \tilde{\mu}_i/\sigma_v)}{1 - \Phi(-\tilde{\mu}_i/\sigma_v)}\right] \exp\left\{-\tilde{\mu}_i + \frac{1}{2}\sigma_v^2\right\}$$

The second version of the estimate is

$$TE2_i = \exp\{-E(u_i|\epsilon_i)\}$$

where

$$E(u_i|\epsilon_i) = \tilde{\mu}_i + \sigma_v \left[\frac{\phi(-\tilde{\mu}_i/\sigma_v)}{1 - \Phi(-\tilde{\mu}_i/\sigma_v)} \right] = \sigma_v \left[\frac{\phi(A)}{\Phi(-A)} - A \right]$$

Normal–Truncated Normal Model

Define $\tilde{\mu} = (-\sigma_u^2\epsilon_i + \mu\sigma_v^2)/\sigma^2$ and $\sigma_*^2 = \sigma_u^2\sigma_v^2/\sigma^2$. Then, as shown by Kumbhakar and Lovell (2000), conditional density is as follows:

$$f(u|\epsilon) = \frac{1}{\sqrt{2\pi}\sigma_*[1 - \Phi(-\tilde{\mu}/\sigma_*)]} \exp\left\{ -\frac{(u - \tilde{\mu})^2}{2\sigma_*^2} \right\}$$

Hence, $f(u|\epsilon)$ is the density for $N^+(\tilde{\mu}, \sigma_*^2)$.

From this result, it follows that the estimate of technical efficiency is

$$TE1_i = E(\exp\{-u_i\}|\epsilon_i) = \frac{1 - \Phi(\sigma_* - \tilde{\mu}_i/\sigma_*)}{1 - \Phi(-\tilde{\mu}_i/\sigma_*)} \exp\left\{ -\tilde{\mu}_i + \frac{1}{2}\sigma_*^2 \right\}$$

The second version of the estimate is

$$TE2_i = \exp\{-E(u_i|\epsilon_i)\}$$

where

$$E(u_i|\epsilon_i) = \tilde{\mu}_i + \sigma_* \left[\frac{\phi(\tilde{\mu}_i/\sigma_*)}{1 - \Phi(-\tilde{\mu}_i/\sigma_*)} \right]$$

OUTEST= Data Set

The OUTEST= data set contains all the parameters that are estimated by a MODEL statement. Each parameter contains the estimate for the corresponding parameter in the corresponding model. In addition, the OUTEST= data set contains the following variables:

NAME	indicates the name of the independent variable.
TYPE	indicates the type of observation. PARM indicates the row of coefficients; STD indicates the row of standard deviations of the corresponding coefficients.
STATUS	indicates the convergence status for optimization.

The rest of the columns correspond to the explanatory variables.

The OUTEST= data set contains one observation for the MODEL statement, which shows the parameter estimates for that model. If you specify the COVOUT option in the PROC HPQLIM statement, the OUTEST= data set includes additional observations for the MODEL statement, which show the rows of the covariance matrix of parameter estimates. For covariance observations, the value of the _TYPE_ variable is COV, and the _NAME_ variable identifies the parameter that is associated with that row of the covariance matrix. If you specify the CORROUT option in the PROC HPQLIM statement, the OUTEST= data set includes additional observations for the MODEL statement, which show the rows of the correlation matrix of parameter estimates. For correlation observations, the value of the _TYPE_ variable is CORR, and the _NAME_ variable identifies the parameter that is associated with that row of the correlation matrix.

Naming

Naming of Parameters

The parameters are named in the same way as in other SAS procedures such as the REG and PROBIT procedures. The constant in the regression equation is called Intercept. The coefficients of independent variables are named by the independent variables. The standard deviation of the errors is called _Sigma. If the HETERO statement is included, the coefficients of the independent variables in the HETERO statement are called _H.*x*, where *x* is the name of the independent variable.

Naming of Output Variables

Table 22.10 shows the *options* in the OUTPUT statement, with the corresponding variable names and their explanations.

Table 22.10 OUTPUT Statement Options

output-option	**Variable Name**	**Explanation**
CONDITIONAL	CEXPCT_y	Conditional expected value of *y*, conditioned on the truncation
ERRSTD	ERRSTD_y	Standard deviation of error term
EXPECTED	EXPCT_y	Unconditional expected value of *y*
MARGINAL	MEFF_x	Marginal effect of *x* on *y* ($\frac{\partial y}{\partial x}$) with single equation
PREDICTED	P_y	Predicted value of *y*
RESIDUAL	RESID_y	Residual of *y*, (y – PredictedY)
PROB	PROB_y	Probability that *y* is taking the observed value in this observation (discrete *y* only)
PROBALL	PROB*i*_y	Probability that *y* is taking the *i*th value (discrete *y* only)
MILLS	MILLS_y	Inverse Mills ratio for *y*
TE1	TE1	Technical efficiency estimate for each producer proposed by Battese and Coelli (1988)
TE2	TE2	Technical efficiency estimate for each producer proposed by Jondrow et al. (1982)
XBETA	XBETA_y	Structure part ($\mathbf{x}'\boldsymbol{\beta}$) of *y* equation

If you prefer to name the output variables differently, you can use the RENAME option in the data set. For example, the following statements rename the residual of *y* as *Resid*:

```
proc hpqlim data=one;
   model y = x1-x10 / censored;
   output out=outds(rename=(resid_y=resid)) residual;
run;
```

ODS Table Names

PROC HPQLIM assigns a name to each table that it creates. You can use these names to refer to the table when you use the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in Table 22.11.

Table 22.11 ODS Tables Produced in PROC HPQLIM

ODS Table Name	Description	Option
ODS Tables Created by the MODEL Statement and TEST Statement		
ResponseProfile	Response profile	Default
FitSummary	Summary of nonlinear estimation	Default
ParameterEstimates	Parameter estimates	Default
SummaryContResponse	Summary of continuous response	Default
CovB	Covariance of parameter estimates	COVB
CorrB	Correlation of parameter estimates	CORRB
ODS Tables Created by the BAYES Statement		
AutoCorr	Autocorrelation statistics for each parameter	Default
Corr	Correlation matrix of the posterior samples	STATS=COR
Cov	Covariance matrix of the posterior samples	STATS=COV
ESS	Effective sample size for each parameter	Default
MCSE	Monte Carlo standard error for each parameter	Default
Geweke	Geweke diagnostics for each parameter	Default
Heidelberger	Heidelberger-Welch diagnostics for each parameter	DIAGNOSTICS=HEIDEL
PostIntervals	Equal-tail and HPD intervals for each parameter	Default
PosteriorSample	Posterior samples	(ODS output data set only)
PostSummaries	Posterior summaries	Default
PriorSummaries	Prior summaries	STATS=PRIOR
Raftery	Raftery-Lewis diagnostics for each parameter	DIAGNOSTICS=RAFTERY
ODS Tables Created by the TEST Statement		
TestResults	Test results	Default

ODS Graphics

You can use a name to reference every graph that is produced through ODS Graphics. The names of the graphs that PROC HPQLIM generates are listed in Table 22.12.

Table 22.12 Graphs Produced by PROC HPQLIM When a BAYES Statement Is Included

ODS Graph Name	Plot Description	Statement and Option
Bayesian Diagnostic Plots		
ADPanel	Autocorrelation function and density panel	PLOTS=(AUTOCORR DENSITY)
AutocorrPanel	Autocorrelation function panel	PLOTS=AUTOCORR
AutocorrPlot	Autocorrelation function plot	PLOTS(UNPACK)=AUTOCORR
DensityPanel	Density panel	PLOTS=DENSITY
DensityPlot	Density plot	PLOTS(UNPACK)=DENSITY
TAPanel	Trace and autocorrelation function panel	PLOTS=(TRACE AUTOCORR)
TADPanel	Trace, density, and autocorrelation function panel	PLOTS=(TRACE AUTOCORR DENSITY)
TDPanel	Trace and density panel	PLOTS=(TRACE DENSITY)
TracePanel	Trace panel	PLOTS=TRACE
TracePlot	Trace plot	PLOTS(UNPACK)=TRACE

Examples: The HPQLIM Procedure

Example 22.1: High-Performance Model with Censoring

This example shows the use of the HPQLIM procedure with an emphasis on processing a large data set and on the performance improvements that are achieved by executing in the high-performance distributed environment.

The following DATA step generates 5 million replicates from a censored model. The model contains seven variables.

```
data simulate;
     call streaminit(12345);
     array vars x1-x7;
     array parms{7}  (3 4 2 4 -3 -5 -3);

     intercept=2;

    do i=1 to 5000000;
        sum_xb=0;
```

```
      do j=1 to 7;
         vars[j]=rand('NORMAL',0,1);
         sum_xb=sum_xb+parms[j]*vars[j];
      end;
      y=intercept+sum_xb+400*rand('NORMAL',0,1);
      if y>400 then y=400;
      if y<0 then y=0;
      output;
   end;
keep y x1-x7;
run;
```

The following statements estimate a censored model. The model is executed in the distributed computing environment with two threads and only one node. These settings are used to obtain a hypothetical environment that might resemble running the HPQLIM procedure on a desktop workstation with a dual-core CPU. To run these statements successfully, you need to set the macro variables GRIDHOST and GRIDINSTALLLOC to resolve to appropriate values, or you can replace the references to the macro variables in the example with the appropriate values.

```
option set=GRIDHOST="&GRIDHOST";
option set=GRIDINSTALLLOC="&GRIDINSTALLLOC";


proc hpqlim data=simulate ;
   performance nthreads=2 nodes=1 details
               host="&GRIDHOST" install="&GRIDINSTALLLOC";
   model y=x1-x7 /censored(lb=0 ub=400);
run;
```

Output 22.1.1 shows that the censored model was estimated on the grid, defined in a macro variable named GRIDHOST, in a distributed environment on only one node with two threads.

Output 22.1.1 Censored Model with One Node and Two Threads: Performance Table

Estimating a Tobit model

Performance Information	
Host Node	<< your grid host >>
Install Location	<< your grid install location >>
Execution Mode	Distributed
Number of Compute Nodes	1
Number of Threads per Node	2

Output 22.1.2 shows the estimation results for the censored model. The "Model Fit Summary" table shows detailed information about the model and indicates that all 5 million observations were used to fit the model. All parameter estimates in the "Parameter Estimates" table are highly significant and correspond to their theoretical values that were set during the data generating process. The optimization of the model with 5 million observations took 45.4 seconds.

Output 22.1.2 Censored Model with One Node and Two Threads: Summary

Model Information	
Data Source	SIMULATE
Response Variable	y
Optimization Technique	Quasi-Newton

Number of Observations	
Number of Observations Read	5000000
Number of Observations Used	5000000

Summary Statistics of Continuous Responses

Variable	Mean	Standard Error	Type	Lower Bound	Upper Bound	N Obs Lower Bound	N Obs Upper Bound
y	127.0	159.491090	Censored	0	400.0	249E4	8E5

Convergence criterion (FCONV=2.220446E-16) satisfied.

Model Fit Summary	
Number of Endogenous Variables	1
Endogenous Variable	y
Number of Observations	5000000
Log Likelihood	-15268972
Maximum Absolute Gradient	0.0003291
Number of Iterations	11
Optimization Method	Quasi-Newton
AIC	30537962
Schwarz Criterion	30538083

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	2.220379	0.222201	9.99	<.0001
x1	1	3.055533	0.201620	15.15	<.0001
x2	1	4.000176	0.201570	19.85	<.0001
x3	1	1.852740	0.201555	9.19	<.0001
x4	1	4.170266	0.201533	20.69	<.0001
x5	1	-3.010679	0.201458	-14.94	<.0001
x6	1	-5.176016	0.201541	-25.68	<.0001
x7	1	-2.695948	0.201671	-13.37	<.0001
_Sigma	1	399.997845	0.261930	1527.12	<.0001

Procedure Task Timing		
Task	Seconds	Percent
Reading and Levelizing Data	1.46	3.12%
Communication to Client	0.09	0.19%
Optimization	45.39	96.69%
Post-optimization	0.00	0.00%

In the following statements, the PERFORMANCE statement is modified to use a grid with 10 nodes, with each node capable of spawning eight threads:

```
proc hpqlim data=simulate ;
   performance nthreads=8 nodes=10 details
               host="&GRIDHOST" install="&GRIDINSTALLLOC";
   model y=x1-x7 /censored(lb=0 ub=400);
run;
```

The second model which was run on a grid with 10 nodes and eight threads each (Output 22.1.3) took only 1.4 seconds instead of 45.4 seconds to optimize.

Output 22.1.3 Censored Model on Ten Nodes with Eight Threads Each: Performance Table

Estimating a Tobit model

Performance Information	
Host Node	<< your grid host >>
Install Location	<< your grid install location >>
Execution Mode	Distributed
Number of Compute Nodes	10
Number of Threads per Node	8

Because the two models being estimated are identical, it is reasonable to expect that Output 22.1.2 and Output 22.1.4 would show the same results except for the performance. However, in certain circumstances, you might observe slight numerical differences in the results (depending on the number of nodes and threads) because the order in which partial results are accumulated, the limits of numerical precision, and the propagation of error in numerical computations can make a difference in the final result.

Output 22.1.4 Censored Model on Ten Nodes with Eight Threads Each: Summary

Model Information	
Data Source	SIMULATE
Response Variable	y
Optimization Technique	Quasi-Newton

Number of Observations	
Number of Observations Read	5000000
Number of Observations Used	5000000

Summary Statistics of Continuous Responses

Variable	Mean	Standard Error	Type	Lower Bound	Upper Bound	N Obs Lower Bound	N Obs Upper Bound
y	127.0	159.491090	Censored	0	400.0	249E4	8E5

Convergence criterion (FCONV=2.220446E-16) satisfied.

Output 22.1.4 *continued*

Model Fit Summary	
Number of Endogenous Variables	1
Endogenous Variable	y
Number of Observations	5000000
Log Likelihood	-15268972
Maximum Absolute Gradient	0.0008332
Number of Iterations	10
Optimization Method	Quasi-Newton
AIC	30537962
Schwarz Criterion	30538083

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	2.220358	0.222201	9.99	<.0001
x1	1	3.055491	0.201620	15.15	<.0001
x2	1	4.000196	0.201570	19.85	<.0001
x3	1	1.852735	0.201555	9.19	<.0001
x4	1	4.170323	0.201533	20.69	<.0001
x5	1	-3.010670	0.201458	-14.94	<.0001
x6	1	-5.176019	0.201541	-25.68	<.0001
x7	1	-2.695886	0.201671	-13.37	<.0001
_Sigma	1	399.997846	0.261930	1527.12	<.0001

Procedure Task Timing

Task	Seconds	Percent
Reading and Levelizing Data	0.09	5.77%
Communication to Client	0.12	7.67%
Optimization	1.38	86.56%
Post-optimization	0.00	0.00%

As this example suggests, increasing the number of nodes and the number of threads per node improves performance significantly. When you use the parallelism that a high-performance distributed environment affords, you can see an even more dramatic reduction in the time required for the optimization as the number of observations in the data set increases. When the data set is extremely large, the computations might not even be possible with the typical memory resources and computational constraints of a desktop computer. Under such circumstances the high-performance distributed environment becomes a necessity.

Example 22.2: Bayesian High-Performance Model with Censoring

This example shows the use of the Bayesian analysis available in the HPQLIM procedure with an emphasis on processing a large data set and on the performance improvements that are achieved by executing in a high-performance distributed environment.

The model and the data set are the same as in Example 22.1, and the priors are set to the defaults.

The model is executed in the distributed computing environment with two threads and only one node. These settings are used to obtain a hypothetical environment that might resemble running the HPQLIM procedure on a desktop workstation with a dual-core CPU. To run the following statements successfully, you need to set the macro variables GRIDHOST and GRIDINSTALLLOC to resolve to appropriate values, or you can replace the references to the macro variables in the example with the appropriate values.

```
 option set=GRIDHOST="&GRIDHOST";
 option set=GRIDINSTALLLOC="&GRIDINSTALLLOC";


proc hpqlim data=simulate ;
bayes nbi=10000 nmc=30000;
   performance nthreads=2 nodes=1 details
               host="&GRIDHOST" install="&GRIDINSTALLLOC";
   model y=x1-x7 /censored(lb=0 ub=400);
   %*;     ods output PerformanceInfo=perfInfo;
   %*;     ods output Timing=time;
run;
```

Output 22.2.1 shows a summary of the posterior distribution that is associated with the censored model when you use diffuse prior distributions.

Output 22.2.1 Posterior Summary for Bayesian Censored Model

Estimating a Tobit model

The HPQLIM Procedure

Posterior Summaries

Parameter	N	Mean	Standard Deviation	Percentiles 25%	50%	75%
Intercept	30000	2.2168	0.2166	2.0682	2.2171	2.3674
x1	30000	3.0656	0.1953	2.9356	3.0638	3.1974
x2	30000	3.9936	0.2052	3.8572	3.9951	4.1308
x3	30000	1.8496	0.2019	1.7201	1.8475	1.9813
x4	30000	4.1637	0.1986	4.0295	4.1574	4.2996
x5	30000	-3.0226	0.1981	-3.1603	-3.0249	-2.8848
x6	30000	-5.1776	0.1985	-5.3099	-5.1720	-5.0456
x7	30000	-2.6889	0.1993	-2.8235	-2.6882	-2.5523
_Sigma	30000	400.0	0.2623	399.8	400.0	400.2

Output 22.2.2 show a summary of the performance when you use a distributed computing environment with one node and two threads.

Output 22.2.2 Performance Analysis for Bayesian Censored Model on One Node with Two Threads

Estimating a Tobit model

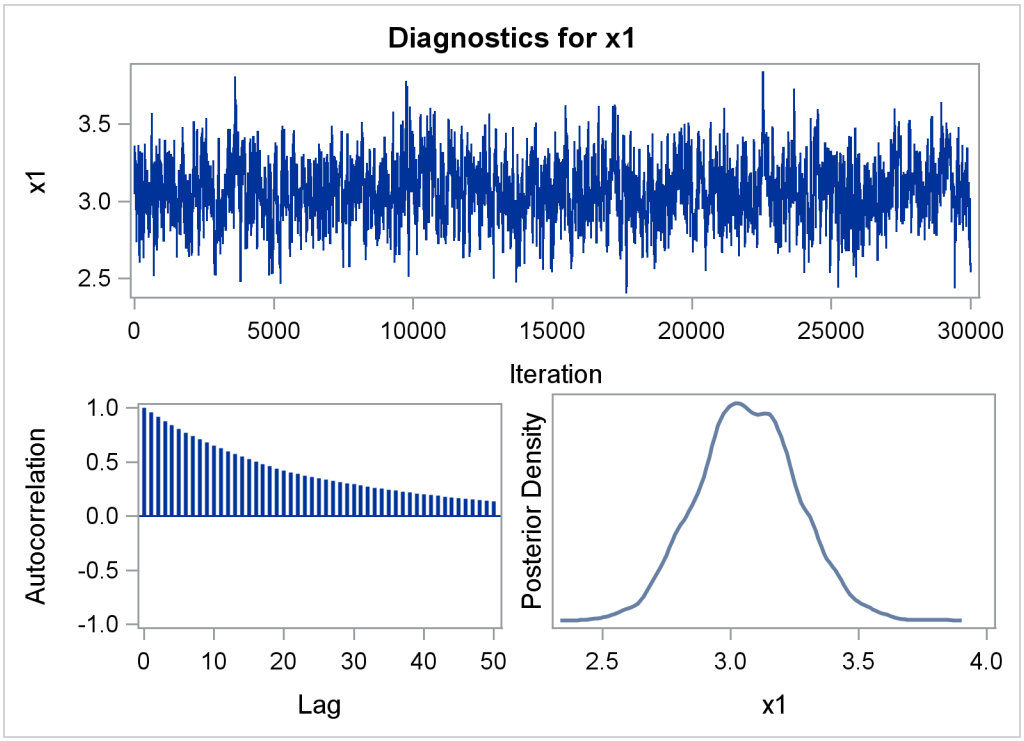
Performance Information	
Host Node	<< your grid host >>
Install Location	<< your grid install location >>
Execution Mode	Distributed
Number of Compute Nodes	1
Number of Threads per Node	2

Estimating a Tobit model

Procedure Task Timing		
Task	**Seconds**	**Percent**
Reading and Levelizing Data	1.59	0.00%
Communication to Client	0.06	0.00%
Bayesian Analysis: Likelihood for MCMC	46088.82	99.88%
Bayesian Analysis: MCMC	1.13	0.00%
Optimization	50.91	0.11%
Post-optimization	0.00	0.00%

Finally, Output 22.2.3 shows the diagnostic and summary plots that are associated with *X1*.

Output 22.2.3 Bayesian Diagnostic and Summary Plots for x1



In the following statements, the PERFORMANCE statement is modified to use a grid with 10 nodes, where each node spawns eight threads:

```
 option set=GRIDHOST="&GRIDHOST";
 option set=GRIDINSTALLLOC="&GRIDINSTALLLOC";


proc hpqlim data=simulate ;
bayes nbi=10000 nmc=30000;
   performance nthreads=8 nodes=10 details
               host="&GRIDHOST" install="&GRIDINSTALLLOC";
   model y=x1-x7 /censored(lb=0 ub=400);
   %*;     ods output PerformanceInfo=perfInfo;
   %*;     ods output Timing=time;
run;
```

The two models are identical, but the second implementation, which was run on a grid that used 10 nodes with eight threads each, took only 15.7 minutes instead of 12.8 hours to sample from the same posterior distribution.

Output 22.2.4 Performance Analysis for Bayesian Censored Model on Ten Nodes with Eight Threads Each

Estimating a Tobit model

Performance Information	
Host Node	<< your grid host >>
Install Location	<< your grid install location >>
Execution Mode	Distributed
Number of Compute Nodes	10
Number of Threads per Node	8

Estimating a Tobit model

Procedure Task Timing		
Task	**Seconds**	**Percent**
Reading and Levelizing Data	0.09	0.01%
Communication to Client	0.21	0.02%
Bayesian Analysis: Likelihood for MCMC	942.12	99.82%
Bayesian Analysis: MCMC	0.21	0.02%
Optimization	1.24	0.13%
Post-optimization	0.00	0.00%

References

Aigner, C., Lovell, C. A. K., and Schmidt, P. (1977). "Formulation and Estimation of Stochastic Frontier Production Function Models." *Journal of Econometrics* 6:21–37.

Battese, G. E., and Coelli, T. J. (1988). "Prediction of Firm-Level Technical Efficiencies with a Generalized Frontier Production Function and Panel Data." *Journal of Econometrics* 38:387–399.

Christensen, L. R., and Greene, W. H. (1976). "Economies of Scale in U.S. Electric Power Generation." *Journal of Political Economy* 84:655–676.

Coelli, T. J., Prasada Rao, D. S., and Battese, G. E. (1998). *An Introduction to Efficiency and Productivity Analysis*. London: Kluwer Academic.

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (2004). *Bayesian Data Analysis*. 2nd ed. London: Chapman & Hall.

Jondrow, J., Lovell, C. A. K., Materov, I. S., and Schmidt, P. (1982). "On the Estimation of Technical Efficiency in the Stochastic Frontier Production Function Model." *Journal of Econometrics* 19:233–238.

Kumbhakar, S. C., and Lovell, C. A. K. (2000). *Stochastic Frontier Analysis*. New York: Cambridge University Press.

McKelvey, R. D., and Zavoina, W. (1975). "A Statistical Model for the Analysis of Ordinal Level Dependent Variables." *Journal of Mathematical Sociology* 4:103–120.

Meeusen, W., and van den Broeck, J. (1977). "Efficiency Estimation from Cobb-Douglas Production Functions with Composed Error." *International Economic Review* 18:435–444.

Mroz, T. A. (1987). "The Sensitivity of an Empirical Model of Married Women's Work to Economic and Statistical Assumptions." *Econometrica* 55:765–799.

Schervish, M. J. (1995). *Theory of Statistics*. New York: Springer-Verlag.

Wooldridge, J. M. (2002). *Econometric Analysis of Cross Section and Panel Data*. Cambridge, MA: MIT Press.

Chapter 23
The HPSEVERITY Procedure

Contents

Overview: HPSEVERITY Procedure **1194**
Getting Started: HPSEVERITY Procedure **1196**
A Simple Example of Fitting Predefined Distributions 1196
An Example with Left-Truncation and Right-Censoring 1199
An Example of Modeling Regression Effects 1202
Syntax: HPSEVERITY Procedure **1206**
Functional Summary 1206
PROC HPSEVERITY Statement 1209
BY Statement 1217
CLASS Statement 1218
DIST Statement 1220
LOSS Statement 1222
NLOPTIONS Statement 1224
OUTSCORELIB Statement 1225
PERFORMANCE Statement 1227
SCALEMODEL Statement 1227
WEIGHT Statement 1229
Programming Statements 1229
Details: HPSEVERITY Procedure **1230**
Predefined Distributions 1230
Censoring and Truncation 1240
Parameter Estimation Method 1242
Parameter Initialization 1244
Estimating Regression Effects 1245
Levelization of Classification Variables 1251
Specification and Parameterization of Model Effects 1253
Empirical Distribution Function Estimation Methods 1260
Statistics of Fit 1266
Distributed and Multithreaded Computation 1271
Defining a Severity Distribution Model with the FCMP Procedure 1273
Predefined Utility Functions 1285
Scoring Functions 1290
Custom Objective Functions 1298
Input Data Sets 1301
Output Data Sets 1302
Displayed Output 1306

ODS Graphics 1309
Examples: HPSEVERITY Procedure **1312**
Example 23.1: Defining a Model for Gaussian Distribution 1312
Example 23.2: Defining a Model for the Gaussian Distribution with a Scale Parameter 1316
Example 23.3: Defining a Model for Mixed-Tail Distributions 1320
Example 23.4: Fitting a Scaled Tweedie Model with Regressors 1327
Example 23.5: Fitting Distributions to Interval-Censored Data 1330
Example 23.6: Benefits of Distributed and Multithreaded Computing 1332
Example 23.7: Estimating Parameters Using Cramér-von Mises Estimator 1338
Example 23.8: Defining a Finite Mixture Model That Has a Scale Parameter 1339
Example 23.9: Predicting Mean and Value-at-Risk by Using Scoring Functions 1345
Example 23.10: Scale Regression with Rich Regression Effects 1350
References **1353**

Overview: HPSEVERITY Procedure

The HPSEVERITY procedure estimates parameters of any arbitrary continuous probability distribution that is used to model the magnitude (severity) of a continuous-valued event of interest. Some examples of such events are loss amounts paid by an insurance company and demand of a product as depicted by its sales. PROC HPSEVERITY is especially useful when the severity of an event does not follow typical distributions (such as the normal distribution) that are often assumed by standard statistical methods.

PROC HPSEVERITY runs in either single-machine mode or distributed mode. **NOTE:** Distributed mode requires SAS High-Performance Econometrics.

PROC HPSEVERITY provides a default set of probability distribution models that includes the Burr, exponential, gamma, generalized Pareto, inverse Gaussian (Wald), lognormal, Pareto, Tweedie, and Weibull distributions. In the simplest form, you can estimate the parameters of any of these distributions by using a list of severity values that are recorded in a SAS data set. You can optionally group the values by a set of BY variables. PROC HPSEVERITY computes the estimates of the model parameters, their standard errors, and their covariance structure by using the maximum likelihood method for each of the BY groups.

PROC HPSEVERITY can fit multiple distributions at the same time and choose the best distribution according to a selection criterion that you specify. You can use seven different statistics of fit as selection criteria. They are log likelihood, Akaike's information criterion (AIC), corrected Akaike's information criterion (AICC), Schwarz Bayesian information criterion (BIC), Kolmogorov-Smirnov statistic (KS), Anderson-Darling statistic (AD), and Cramér-von Mises statistic (CvM).

You can request the procedure to output the status of the estimation process, the parameter estimates and their standard errors, the estimated covariance structure of the parameters, the statistics of fit, estimated cumulative distribution function (CDF) for each of the specified distributions, and the empirical distribution function (EDF) estimate (which is used to compute the KS, AD, and CvM statistics of fit).

The following key features make PROC HPSEVERITY unique among SAS procedures that can estimate continuous probability distributions:

- It enables you to fit a distribution model when the severity values are truncated or censored or both. You can specify any combination of the following types of censoring and truncation effects: left-censoring, right-censoring, left-truncation, or right-truncation. This is especially useful in applications with an insurance-type model where a severity (loss) is reported and recorded only if it is greater than the deductible amount (left-truncation) and where a severity value greater than or equal to the policy limit is recorded at the limit (right-censoring). Another useful application is that of interval-censored data, where you know both the lower limit (right-censoring) and upper limit (left-censoring) on the severity, but you do not know the exact value.

  PROC HPSEVERITY also enables you to specify a *probability of observability* for the left-truncated data, which is a probability of observing values greater than the left-truncation threshold. This additional information can be useful in certain applications to more correctly model the distribution of the severity of events.

  It uses an appropriate estimator of the empirical distribution function (EDF). EDF is required to compute the KS, AD, and CvM statistics-of-fit. The procedure also provides the EDF estimates to your custom parameter initialization method. When you specify truncation or censoring, the EDF is estimated by using either Kaplan-Meier's product-limit estimator or Turnbull's estimator. The former is used by default when you specify only one form of censoring effect (right-censoring or left-censoring), whereas the latter is used by default when you specify both left-censoring and right-censoring effects. The procedure computes the standard errors for all EDF estimators.

- It enables you to define any arbitrary continuous parametric distribution model and to estimate its parameters. You just need to define the key components of the distribution, such as its probability density function (PDF) and cumulative distribution function (CDF), as a set of functions and subroutines written with the FCMP procedure, which is part of Base SAS software. As long as the functions and subroutines follow certain rules, the HPSEVERITY procedure can fit the distribution model defined by them.

- It can model the influence of exogenous or regressor variables on a probability distribution, as long as the distribution has a scale parameter. A linear combination of regression effects is assumed to affect the scale parameter via an exponential link function.

  If a distribution does not have a scale parameter, then either it needs to have another parameter that can be derived from a scale parameter by using a supported transformation or it needs to be reparameterized to have a scale parameter. If neither of these is possible, then regression effects cannot be modeled.

  You can easily construct many types of regression effects by using various operators on a set of classification and continuous variables. You can specify classification variables in the CLASS statement.

- It enables you to specify your own objective function to be optimized for estimating the parameters of a model. You can write SAS programming statements to specify the contribution of each observation to the objective function. You can use keyword functions such as _PDF_ and _CDF_ to generalize the objective function to any distribution. If you do not specify your own objective function, then the parameters of a model are estimated by maximizing the likelihood function of the data.

- It enables you to create scoring functions that offer a convenient way to evaluate any distribution function, such as PDF, CDF, QUANTILE, or your custom distribution function, for a fitted model on new observations.

Because the HPSEVERITY procedure is a high-performance analytical procedure, it also does the following:

- enables you to run in distributed mode on a cluster of machines that distribute the data and the computations
- enables you to run in single-machine mode on the server where SAS is installed
- exploits all the available cores and concurrent threads, regardless of execution mode

For more information, see the section "Processing Modes" on page 62 in Chapter 3, "Shared Concepts and Topics."

Getting Started: HPSEVERITY Procedure

This section outlines the use of the HPSEVERITY procedure to fit continuous probability distribution models. Three examples illustrate different features of the procedure.

A Simple Example of Fitting Predefined Distributions

The simplest way to use PROC HPSEVERITY is to fit all the predefined distributions to a set of values and let the procedure identify the best fitting distribution.

Consider a lognormal distribution, whose probability density function (PDF) f and cumulative distribution function (CDF) F are as follows, respectively, where Φ denotes the CDF of the standard normal distribution:

$$f(x;\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\log(x)-\mu}{\sigma}\right)^2} \quad \text{and} \quad F(x;\mu,\sigma) = \Phi\left(\frac{\log(x)-\mu}{\sigma}\right)$$

The following DATA step statements simulate a sample from a lognormal distribution with population parameters $\mu = 1.5$ and $\sigma = 0.25$, and store the sample in the variable Y of a data set Work.Test_sev1:

```
/*------------- Simple Lognormal Example -------------*/
data test_sev1(keep=y label='Simple Lognormal Sample');
   call streaminit(45678);
   label y='Response Variable';
   Mu = 1.5;
   Sigma = 0.25;
   do n = 1 to 100;
      y = exp(Mu) * rand('LOGNORMAL')**Sigma;
      output;
   end;
run;
```

The following statements fit all the predefined distribution models to the values of Y and identify the best distribution according to the corrected Akaike's information criterion (AICC):

```
proc hpseverity data=test_sev1 crit=aicc;
   loss y;
   dist _predefined_;
run;
```

The PROC HPSEVERITY statement specifies the input data set along with the model selection criterion, the LOSS statement specifies the variable to be modeled, and the DIST statement with the _PREDEFINED_ keyword specifies that all the predefined distribution models be fitted.

Some of the default output displayed by this step is shown in Figure 23.1 through Figure 23.3. First, information about the input data set is displayed followed by the "Model Selection" table, as shown in Figure 23.1. The model selection table displays the convergence status, the value of the selection criterion, and the selection status for each of the candidate models. The Converged column indicates whether the estimation process for a given distribution model has converged, might have converged, or failed. The Selected column indicates whether a given distribution has the best fit for the data according to the selection criterion. For this example, the lognormal distribution model is selected, because it has the lowest value for the selection criterion.

Figure 23.1 Data Set Information and Model Selection Table

The HPSEVERITY Procedure

Input Data Set	
Name	WORK.TEST_SEV1
Label	Simple Lognormal Sample

Model Selection			
Distribution	**Converged**	**AICC**	**Selected**
Burr	Yes	322.50845	No
Exp	Yes	508.12287	No
Gamma	Yes	320.50264	No
Igauss	Yes	319.61652	No
Logn	Yes	319.56579	Yes
Pareto	Yes	510.28172	No
Gpd	Yes	510.20576	No
Weibull	Yes	334.82373	No

Next, the estimation information for each of the candidate models is displayed. The information for the lognormal model, which is the best fitting model, is shown in Figure 23.2. The first table displays a summary of the distribution. The second table displays the convergence status. This is followed by a summary of the optimization process which indicates the technique used, the number of iterations, the number of times the objective function was evaluated, and the log likelihood attained at the end of the optimization. Since the model with lognormal distribution has converged, PROC HPSEVERITY displays its statistics of fit and parameter estimates. The estimates of *Mu*=1.49605 and *Sigma*=0.26243 are quite close to the population parameters of *Mu*=1.5 and *Sigma*=0.25 from which the sample was generated. The *p*-value for each estimate indicates the rejection of the null hypothesis that the estimate is 0, implying that both the estimates are significantly different from 0.

Figure 23.2 Estimation Details for the Lognormal Model

The HPSEVERITY Procedure
Logn Distribution

Distribution Information	
Name	Logn
Description	Lognormal Distribution
Distribution Parameters	2

Convergence Status
Convergence criterion (GCONV=1E-8) satisfied.

Optimization Summary	
Optimization Technique	Trust Region
Iterations	2
Function Calls	8
Log Likelihood	-157.72104

Fit Statistics	
-2 Log Likelihood	315.44208
AIC	319.44208
AICC	319.56579
BIC	324.65242
Kolmogorov-Smirnov	0.50641
Anderson-Darling	0.31240
Cramer-von Mises	0.04353

Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|
Mu	1.49605	0.02651	56.43	<.0001
Sigma	0.26243	0.01874	14.00	<.0001

The parameter estimates of the Burr distribution are shown in Figure 23.3. These estimates are used in the next example.

Figure 23.3 Parameter Estimates for the Burr Model

Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|
Theta	4.62348	0.46181	10.01	<.0001
Alpha	1.15706	0.47493	2.44	0.0167
Gamma	6.41227	0.99039	6.47	<.0001

An Example with Left-Truncation and Right-Censoring

PROC HPSEVERITY enables you to specify that the response variable values are left-truncated or right-censored. The following DATA step expands the data set of the previous example to simulate a scenario that is typically encountered by an automobile insurance company. The values of the variable Y represent the loss values on claims that are reported to an auto insurance company. The variable THRESHOLD records the deductible on the insurance policy. If the actual value of Y is less than or equal to the deductible, then it is unobservable and does not get recorded. In other words, THRESHOLD specifies the left-truncation of Y. LIMIT records the policy limit. If the value of Y is equal to or greater than the recorded value, then the observation is right-censored.

```
/*----- Lognormal Model with left-truncation and censoring -----*/
data test_sev2(keep=y threshold limit
        label='A Lognormal Sample With Censoring and Truncation');
   set test_sev1;
   label y='Censored & Truncated Response';
   if _n_ = 1 then call streaminit(45679);

   /* make about 20% of the observations left-truncated */
   if (rand('UNIFORM') < 0.2) then
      threshold = y * (1 - rand('UNIFORM'));
   else
      threshold = .;
   /* make about 15% of the observations right-censored */
   iscens = (rand('UNIFORM') < 0.15);
   if (iscens) then
      limit = y;
   else
      limit = .;
run;
```

The following statements use the AICC criterion to analyze which of the four predefined distributions (lognormal, Burr, gamma, and Weibull) has the best fit for the data:

```
proc hpseverity data=test_sev2 crit=aicc print=all ;
   loss y / lt=threshold rc=limit;

   dist logn burr gamma weibull;
   performance nthreads=2;
run;
```

The LOSS statement specifies the left-truncation and right-censoring variables. The DIST statement specifies the candidate distributions. The PRINT= option in the PROC HPSEVERITY statement requests that all the displayed output be prepared. The NTHREADS option in the PERFORMANCE statement specifies that two threads of computation be used. The option is shown here just for illustration. You should use it only when you want to restrict the procedure to use a different number of threads than the value of the CPUCOUNT= system option, which usually defaults to the number of physical CPU cores available on your machine, thereby allowing the procedure to fully utilize the computational power of your machine.

Some of the key results prepared by PROC HPSEVERITY are shown in Figure 23.4 through Figure 23.7. In addition to the estimates of the range, mean, and standard deviation of Y, the "Descriptive Statistics for y" table shown in Figure 23.4 also indicates the number of observations that are left-truncated or right-censored. The "Model Selection" table in Figure 23.4 shows that models with all the candidate distributions have converged and that the Logn (lognormal) model has the best fit for the data according to the AICC criterion.

Figure 23.4 Summary Results for the Truncated and Censored Data

The HPSEVERITY Procedure

Input Data Set	
Name	WORK.TEST_SEV2
Label	A Lognormal Sample With Censoring and Truncation

Descriptive Statistics for y	
Observations	100
Observations Used for Estimation	100
Minimum	2.30264
Maximum	8.34116
Mean	4.62007
Standard Deviation	1.23627
Left Truncated Observations	23
Right Censored Observations	14

Model Selection			
Distribution	**Converged**	**AICC**	**Selected**
Logn	Yes	298.92672	Yes
Burr	Yes	302.66229	No
Gamma	Yes	299.45293	No
Weibull	Yes	309.26779	No

PROC HPSEVERITY also prepares a table that shows all the fit statistics for all the candidate models. It is useful to see which model would be the best fit according to each of the criteria. The "All Fit Statistics" table prepared for this example is shown in Figure 23.5. It indicates that the lognormal model is chosen by all the criteria.

Figure 23.5 Comparing All Statistics of Fit for the Truncated and Censored Data

All Fit Statistics												
Distribution	**-2 Log Likelihood**		**AIC**		**AICC**		**BIC**		**KS**		**AD**	
Logn	294.80301	*	298.80301	*	298.92672	*	304.01335	*	0.51824	*	0.34736	*
Burr	296.41229		302.41229		302.66229		310.22780		0.66984		0.36712	
Gamma	295.32921		299.32921		299.45293		304.53955		0.62511		0.42921	
Weibull	305.14408		309.14408		309.26779		314.35442		0.93307		1.40699	

Note: The asterisk (*) marks the best model according to each column's criterion.

All Fit Statistics		
Distribution		**CvM**
Logn	0.05159	*
Burr	0.05726	
Gamma	0.05526	
Weibull	0.17465	

Note: The asterisk (*) marks the best model according to each column's criterion.

Specifying Initial Values for Parameters

All the predefined distributions have parameter initialization functions built into them. For the current example, Figure 23.6 shows the initial values that are obtained by the predefined method for the Burr distribution. It also shows the summary of the optimization process and the final parameter estimates.

Figure 23.6 Burr Model Summary for the Truncated and Censored Data

Initial Parameter Values and Bounds

Parameter	Initial Value	Lower Bound	Upper Bound
Theta	4.78102	1.05367E-8	Infty
Alpha	2.00000	1.05367E-8	Infty
Gamma	2.00000	1.05367E-8	Infty

Optimization Summary

Optimization Technique	Trust Region
Iterations	8
Function Calls	23
Log Likelihood	-148.20614

Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|
Theta	4.76980	0.62492	7.63	<.0001
Alpha	1.16363	0.58859	1.98	0.0509
Gamma	5.94081	1.05004	5.66	<.0001

You can specify a different set of initial values if estimates are available from fitting the distribution to similar data. For this example, the parameters of the Burr distribution can be initialized with the final parameter estimates of the Burr distribution that were obtained in the first example (shown in Figure 23.3). One of the ways in which you can specify the initial values is as follows:

```
/*------ Specifying initial values using INIT= option -------*/
proc hpseverity data=test_sev2 crit=aicc print=all;
   loss y / lt=threshold rc=limit;

   dist burr(init=(theta=4.62348 alpha=1.15706 gamma=6.41227));
   performance nthreads=2;
run;
```

The names of the parameters that are specified in the INIT option must match the parameter names in the definition of the distribution. The results obtained with these initial values are shown in Figure 23.7. These results indicate that new set of initial values causes the optimizer to reach the same solution with fewer iterations and function evaluations as compared to the default initialization.

Figure 23.7 Burr Model Optimization Summary for the Truncated and Censored Data

The HPSEVERITY Procedure
Burr Distribution

Optimization Summary	
Optimization Technique	Trust Region
Iterations	5
Function Calls	16
Log Likelihood	-148.20614

Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|
Theta	4.76980	0.62492	7.63	<.0001
Alpha	1.16363	0.58859	1.98	0.0509
Gamma	5.94081	1.05004	5.66	<.0001

An Example of Modeling Regression Effects

Consider a scenario in which the magnitude of the response variable might be affected by some regressor (exogenous or independent) variables. The HPSEVERITY procedure enables you to model the effect of such variables on the distribution of the response variable via an exponential link function. In particular, if you have k random regressor variables denoted by x_j $(j = 1, \ldots, k)$, then the distribution of the response variable Y is assumed to have the form

$$Y \sim \exp(\sum_{j=1}^{k} \beta_j x_j) \cdot \mathcal{F}(\Theta)$$

where $\mathcal{F}$ denotes the distribution of Y with parameters Θ and $\beta_j (j = 1, \ldots, k)$ denote the regression parameters (coefficients).

For the effective distribution of Y to be a valid distribution from the same parametric family as $\mathcal{F}$, it is necessary for $\mathcal{F}$ to have a scale parameter. The effective distribution of Y can be written as

$$Y \sim \mathcal{F}(\theta, \Omega)$$

where θ denotes the scale parameter and Ω denotes the set of nonscale parameters. The scale θ is affected by the regressors as

$$\theta = \theta_0 \cdot \exp(\sum_{j=1}^{k} \beta_j x_j)$$

where θ_0 denotes a *base* value of the scale parameter.

Given this form of the model, PROC HPSEVERITY allows a distribution to be a candidate for modeling regression effects only if it has an untransformed or a log-transformed scale parameter.

All the predefined distributions, except the lognormal distribution, have a direct scale parameter (that is, a parameter that is a scale parameter without any transformation). For the lognormal distribution, the parameter μ is a log-transformed scale parameter. This can be verified by replacing μ with a parameter $\theta = e^{\mu}$, which results in the following expressions for the PDF f and the CDF F in terms of θ and σ, respectively, where Φ denotes the CDF of the standard normal distribution:

$$f(x;\theta,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\log(x)-\log(\theta)}{\sigma}\right)^2} \quad \text{and} \quad F(x;\theta,\sigma) = \Phi\left(\frac{\log(x)-\log(\theta)}{\sigma}\right)$$

With this parameterization, the PDF satisfies the $f(x;\theta,\sigma) = \frac{1}{\theta} f(\frac{x}{\theta};1,\sigma)$ condition and the CDF satisfies the $F(x;\theta,\sigma) = F(\frac{x}{\theta};1,\sigma)$ condition. This makes θ a scale parameter. Hence, $\mu = \log(\theta)$ is a log-transformed scale parameter and the lognormal distribution is eligible for modeling regression effects.

The following DATA step simulates a lognormal sample whose scale is decided by the values of the three regressors X1, X2, and X3 as follows:

$$\mu = \log(\theta) = 1 + 0.75\,\text{X1} - \text{X2} + 0.25\,\text{X3}$$

```
/*----------- Lognormal Model with Regressors ------------*/
data test_sev3(keep=y x1-x3
               label='A Lognormal Sample Affected by Regressors');
   array x{*} x1-x3;
   array b{4} _TEMPORARY_ (1 0.75 -1 0.25);
   call streaminit(45678);
   label y='Response Influenced by Regressors';
   Sigma = 0.25;
   do n = 1 to 100;
      Mu = b(1); /* log of base value of scale */
      do i = 1 to dim(x);
         x(i) = rand('UNIFORM');
         Mu = Mu + b(i+1) * x(i);
      end;
      y = exp(Mu) * rand('LOGNORMAL')**Sigma;
      output;
   end;
run;
```

The following PROC HPSEVERITY step fits the lognormal, Burr, and gamma distribution models to this data. The regressors are specified in the SCALEMODEL statement.

```
proc hpseverity data=test_sev3 crit=aicc print=all;
   loss y;
   scalemodel x1-x3;

   dist logn burr gamma;
run;
```

Some of the key results prepared by PROC HPSEVERITY are shown in Figure 23.8 through Figure 23.12. The descriptive statistics of all the variables are shown in Figure 23.8.

Figure 23.8 Summary Results for the Regression Example

The HPSEVERITY Procedure

Input Data Set	
Name	WORK.TEST_SEV3
Label	A Lognormal Sample Affected by Regressors

Descriptive Statistics for y	
Observations	100
Observations Used for Estimation	100
Minimum	1.17863
Maximum	6.65269
Mean	2.99859
Standard Deviation	1.12845

Descriptive Statistics for Regressors

Variable	N	Minimum	Maximum	Mean	Standard Deviation
x1	100	0.0005115	0.97971	0.51689	0.28206
x2	100	0.01883	0.99937	0.47345	0.28885
x3	100	0.00255	0.97558	0.48301	0.29709

The comparison of the fit statistics of all the models is shown in Figure 23.9. It indicates that the lognormal model is the best model according to each of the likelihood-based statistics, whereas the gamma model is the best model according to two of the three EDF-based statistics.

Figure 23.9 Comparison of Statistics of Fit for the Regression Example

All Fit Statistics

Distribution	-2 Log Likelihood		AIC		AICC		BIC		KS		AD	
Logn	187.49609	*	197.49609	*	198.13439	*	210.52194	*	1.97544		17.24618	
Burr	190.69154		202.69154		203.59476		218.32256		2.09334		13.93436	*
Gamma	188.91483		198.91483		199.55313		211.94069		1.94472	*	15.84787	

Note: The asterisk (*) marks the best model according to each column's criterion.

All Fit Statistics

Distribution	CvM	
Logn	1.21665	
Burr	1.28529	
Gamma	1.17617	*

Note: The asterisk (*) marks the best model according to each column's criterion.

The distribution information and the convergence results of the lognormal model are shown in Figure 23.10. The iteration history gives you a summary of how the optimizer is traversing the surface of the log-likelihood function in its attempt to reach the optimum. Both the change in the log likelihood and the maximum gradient of the objective function with respect to any of the parameters typically approach 0 if the optimizer converges.

Figure 23.10 Convergence Results for the Lognormal Model with Regressors

The HPSEVERITY Procedure
Logn Distribution

Distribution Information	
Name	Logn
Description	Lognormal Distribution
Distribution Parameters	2
Regression Parameters	3

Convergence Status
Convergence criterion (GCONV=1E-8) satisfied.

Optimization Iteration History

Iter	Function Calls	-Log Likelihood	Change	Maximum Gradient
0	2	93.75285		6.16002
1	4	93.74805	-0.0048055	0.11031
2	6	93.74805	-1.5017E-6	0.00003376
3	10	93.74805	-1.421E-13	3.1051E-12

Optimization Summary	
Optimization Technique	Trust Region
Iterations	3
Function Calls	10
Log Likelihood	-93.74805

The final parameter estimates of the lognormal model are shown in Figure 23.11. All the estimates are significantly different from 0. The estimate that is reported for the parameter *Mu* is the base value for the log-transformed scale parameter μ. Let $x_i (1 \leq i \leq 3)$ denote the observed value for regressor X*i*. If the lognormal distribution is chosen to model *Y*, then the effective value of the parameter μ varies with the observed values of regressors as

$$\mu = 1.04047 + 0.65221\, x_1 - 0.91116\, x_2 + 0.16243\, x_3$$

These estimated coefficients are reasonably close to the population parameters (that is, within one or two standard errors).

Figure 23.11 Parameter Estimates for the Lognormal Model with Regressors

Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|
Mu	1.04047	0.07614	13.66	<.0001
Sigma	0.22177	0.01609	13.78	<.0001
x1	0.65221	0.08167	7.99	<.0001
x2	-0.91116	0.07946	-11.47	<.0001
x3	0.16243	0.07782	2.09	0.0395

The estimates of the gamma distribution model, which is the best model according to a majority of the EDF-based statistics, are shown in Figure 23.12. The estimate that is reported for the parameter *Theta* is the base value for the scale parameter θ. If the gamma distribution is chosen to model Y, then the effective value of the scale parameter is $\theta = 0.14293 \exp(0.64562\, x_1 - 0.89831\, x_2 + 0.14901\, x_3)$.

Figure 23.12 Parameter Estimates for the Gamma Model with Regressors

Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|
Theta	0.14293	0.02329	6.14	<.0001
Alpha	20.37726	2.93277	6.95	<.0001
x1	0.64562	0.08224	7.85	<.0001
x2	-0.89831	0.07962	-11.28	<.0001
x3	0.14901	0.07870	1.89	0.0613

Syntax: HPSEVERITY Procedure

The following statements are available in the HPSEVERITY procedure:

PROC HPSEVERITY *options* ;
BY *variable-list* ;
LOSS < *response-variable* > < / *censoring-truncation-options* > ;
WEIGHT *weight-variable* ;
CLASS *variable* < *(options)* > . . . < *variable* < *(options)* > > < / *global-options* > ;
SCALEMODEL *regression-effect-list* < / *scalemodel-options* > ;
DIST *distribution-name-or-keyword* < *(distribution-option)* < *distribution-name-or-keyword* < *(distribution-option)* > > . . . > < / *preprocess-options* > ;
OUTSCORELIB < *OUTLIB=* > *fcmp-library-name options* ;
NLOPTIONS *options* ;
PERFORMANCE *options* ;
Programming statements ;

Functional Summary

Table 23.1 summarizes the statements and options that control the HPSEVERITY procedure.

Table 23.1 HPSEVERITY Functional Summary

Description	Statement	Option
Statements		
Specifies BY-group processing	BY	
Specifies the response variable to model along with censoring and truncation effects	LOSS	
Specifies the weight variable	WEIGHT	
Specifies the classification variables	CLASS	

Table 23.1 *continued*

Description	Statement	Option
Specifies the regression effects to model	SCALEMODEL	
Specifies distributions to fit	DIST	
Specifies the library to write scoring functions to	OUTSCORELIB	
Specifies optimization options	NLOPTIONS	
Specifies performance options	PERFORMANCE	
Specifies programming statements that define an objective function	Programming statements	
Input and Output Options		
Specifies that the OUTEST= data set contain covariance estimates	PROC HPSEVERITY	COVOUT
Specifies the input data set	PROC HPSEVERITY	DATA=
Specifies the input data set for parameter estimates	PROC HPSEVERITY	INEST=
Specifies the input item store for parameter initialization	PROC HPSEVERITY	INSTORE=
Limits the length of effect names	PROC HPSEVERITY	NAMELEN=
Specifies the output data set for CDF estimates	PROC HPSEVERITY	OUTCDF=
Specifies the output data set for parameter estimates	PROC HPSEVERITY	OUTEST=
Specifies the output data set for model information	PROC HPSEVERITY	OUTMODELINFO=
Specifies the output data set for statistics of fit	PROC HPSEVERITY	OUTSTAT=
Specifies the output item store for context and estimation results	PROC HPSEVERITY	OUTSTORE=
Data Interpretation Options		
Specifies left-censoring	LOSS	LEFTCENSORED=
Specifies left-truncation	LOSS	LEFTTRUNCATED=
Specifies the probability of observability	LOSS	PROBOBSERVED=
Specifies right-censoring	LOSS	RIGHTCENSORED=
Specifies right-truncation	LOSS	RIGHTTRUNCATED=
Model Estimation Options		
Specifies the model selection criterion	PROC HPSEVERITY	CRITERION=
Specifies the method for computing mixture distribution	SCALEMODEL	DFMIXTURE=
Specifies initial values for model parameters	DIST	INIT=
Specifies the objective function symbol	PROC HPSEVERITY	OBJECTIVE=
Specifies the offset variable in the scale regression model	SCALEMODEL	OFFSET=
Specifies the denominator for computing covariance estimates	PROC HPSEVERITY	VARDEF=

Table 23.1 *continued*

Description	Statement	Option
Empirical Distribution Function (EDF) Estimation Options		
Specifies the confidence level for reporting the confidence interval for EDF estimates	PROC HPSEVERITY	EDFALPHA=
Specifies the nonparametric method of CDF estimation	PROC HPSEVERITY	EMPIRICALCDF=
Specifies the sample to be used for computing the EDF estimates	PROC HPSEVERITY	INITSAMPLE
EMPIRICALCDF=MODIFIEDKM Options		
Specifies the α value for the lower bound on risk set size	PROC HPSEVERITY	ALPHA=
Specifies the c value for the lower bound on risk set size	PROC HPSEVERITY	C=
Specifies the absolute lower bound on risk set size	PROC HPSEVERITY	RSLB=
EMPIRICALCDF=TURNBULL Options		
Specifies that the final EDF estimates be maximum likelihood estimates	PROC HPSEVERITY	ENSUREMLE
Specifies the relative convergence criterion	PROC HPSEVERITY	EPS=
Specifies the maximum number of iterations	PROC HPSEVERITY	MAXITER=
Specifies the threshold below which an EDF estimate is deemed to be 0	PROC HPSEVERITY	ZEROPROB=
Scoring Function Generation Options		
Specifies that scoring functions of all models be written to one package	OUTSCORELIB	COMMONPACKAGE
Specifies the output data set for BY-group identifiers	OUTSCORELIB	OUTBYID=
Specifies the output library for scoring functions	OUTSCORELIB	OUTLIB=
Displayed Output and Plotting Options		
Specifies that distributions be listed to the log without estimating any models that use them	DIST	LISTONLY
Limits or suppresses the display of class levels	PROC HPSEVERITY	NOCLPRINT
Suppresses all displayed and graphical output	PROC HPSEVERITY	NOPRINT
Specifies which graphical output to prepare	PROC HPSEVERITY	PLOTS=
Specifies which output to display	PROC HPSEVERITY	PRINT=
Specifies that distributions be validated without estimating any models that use them	DIST	VALIDATEONLY

PROC HPSEVERITY Statement

PROC HPSEVERITY *options* **;**

The PROC HPSEVERITY statement invokes the procedure. You can specify two types of *options* in the PROC HPSEVERITY statement. One set of *options* controls input and output. The other set of *options* controls the model estimation and selection process.

The following *options* control the input data sets used by PROC HPSEVERITY and various forms of output generated by PROC HPSEVERITY. The *options* are listed in alphabetical order.

COVOUT
specifies that the OUTEST= data set contain the estimate of the covariance structure of the parameters. This option has no effect if you do not specify the OUTEST= option. For more information about how the covariance is reported in the OUTEST= data set, see the section "OUTEST= Data Set" on page 1303.

DATA=*SAS-data-set*
names the input data set. If you do not specify the DATA= option, then the most recently created SAS data set is used.

EDFALPHA=*confidence-level*
specifies the confidence level in the (0,1) range that is used for computing the confidence intervals for the EDF estimates. The lower and upper confidence limits that correspond to this level are reported in the OUTCDF= data set, if specified, and are displayed in the plot that is created when you specify the PLOTS=CDFPERDIST option.

If you do not specify the EDFALPHA= option, then PROC HPSEVERITY uses a default value of 0.05.

INEST=*SAS-data-set*
names the input data set that contains the initial values of the parameter estimates to start the optimization process. The initial values that you specify in the INIT= option in the DIST statement take precedence over any initial values that you specify in the INEST= data set. For more information about the variables in this data set, see the section "INEST= Data Set" on page 1301.

If you specify the SCALEMODEL statement, then PROC HPSEVERITY reads the INEST= data set only if the SCALEMODEL statement contains singleton continuous effects. For more generic regression effects, you should save the estimates by specifying the OUTSTORE= item store in a step and then use the INSTORE= option to read those estimates. The INSTORE= option is the newer and more flexible method of specifying initial values for distribution and regression parameters.

INITSAMPLE (*initsample-option***)**

INITSAMPLE (*initsample-option . . . initsample-option***)**
specifies that a sample of the input data be used for initializing the distribution parameters. If you specify more than one *initsample-option*, then separate them with spaces.

When you do not specify initial values for the distribution parameters, PROC HPSEVERITY needs to compute the empirical distribution function (EDF) estimates as part of the default method for parameter initialization. The EDF estimation process can be expensive, especially when you specify censoring or truncation effects for the loss variable. Furthermore, it is not amenable to parallelism due to the sequential nature of the algorithm for truncation effects. You can use the INITSAMPLE option to

specify that only a fraction of the input data be used in order to reduce the time taken to compute the EDF estimates. PROC HPSEVERITY uses the uniform random sampling method to select the sample, the size and randomness of which is controlled by the following *initsample-options*:

FRACTION=*number*
specifies the fraction, between 0 and 1, of the input data to be used for sampling.

SEED=*number*
specifies the seed to be used for the uniform random number generator. This option enables you to select the same sample from the same input data across different runs of PROC HPSEVERITY, which can be useful for replicating the results across different runs. If you do not specify the seed value, PROC HPSEVERITY generates a seed that is based on the system clock.

SIZE=*number*
specifies the size of the sample. If the data are distributed across different nodes, then this size applies to the sample that is prepared at each node. For example, let the input data set of size 100,000 observations be distributed across 10 nodes such that each node has 10,000 observations. If you specify SIZE=1000, then each node computes a local EDF estimate by using a sample of size 1,000 selected randomly from its 10,000 observations. If you specify both of the SIZE= and FRACTION= options, then the value that you specify in the SIZE= option is used and the FRACTION= option is ignored.

If you do not specify the INITSAMPLE option, then a uniform random sample of at most 10,000 observations is used for EDF estimation.

INSTORE=*store-name* (Experimental)
names the item store that contains the context and results of the severity model estimation process. An item store has a binary file format that cannot be modified. You must specify an item store that you have created in another PROC HPSEVERITY step by using the OUTSTORE= option.

The *store-name* is a usual one- or two-level SAS name, as for SAS data sets. If you specify a one-level name, then PROC HPSEVERITY reads the item store from the WORK library. If you specify a two-level name of the form *libname.membername*, then PROC HPSEVERITY reads the item store from the *libname* library.

This option is more flexible than the INEST= option, because it can read estimates of any type of scale regression model; the INEST= option can read only scale regression models that contain singleton continuous effects.

For more information about how the input item store is used for parameter initialization, see the sections "Parameter Initialization" on page 1244 and "Parameter Initialization for Regression Models" on page 1247.

NAMELEN=*number*
specifies the length to which long regression effect names are shortened. The default and minimum value is 20.

This option does not apply to the names of singleton continuous effects if you have not specified any CLASS variables.

NOCLPRINT<=*number*>
suppresses the display of the "Class Level Information" table if you do not specify *number*. If you specify *number*, the values of the classification variables are displayed for only those variables whose number of levels is less than *number*. Specifying a *number* helps to reduce the size of the "Class Level Information" table if some classification variables have a large number of levels. This option has no effect if you do not specify the CLASS statement.

NOPRINT
turns off all displayed and graphical output. If you specify this option, then any value that you specify for the PRINT= and PLOTS= options is ignored.

OUTCDF=*SAS-data-set*
names the output data set to contain estimates of the cumulative distribution function (CDF) value at each of the observations. This data set is created only when you run PROC HPSEVERITY in single-machine mode.

The information is output for each specified model whose parameter estimation process converges. The data set also contains the estimates of the empirical distribution function (EDF). For more information about the variables in this data set, see the section "OUTCDF= Data Set" on page 1302.

OUTEST=*SAS-data-set*
names the output data set to contain estimates of the parameter values and their standard errors for each model whose parameter estimation process converges. For more information about the variables in this data set, see the section "OUTEST= Data Set" on page 1303.

If you specify the SCALEMODEL statement such that it contains at least one effect that is not a singleton continuous effect, then the OUTEST= data set that this option creates cannot be used as an INEST= data set in a subsequent PROC HPSEVERITY step. In such cases, it is recommended that you use the newer OUTSTORE= option to save the estimates and specify those estimates in a subsequent PROC HPSEVERITY step by using the INSTORE= option.

OUTMODELINFO=*SAS-data-set*
names the output data set to contain the information about each candidate distribution. For more information about the variables in this data set, see the section "OUTMODELINFO= Data Set" on page 1304.

OUTSTAT=*SAS-data-set*
names the output data set to contain the values of statistics of fit for each model whose parameter estimation process converges. For more information about the variables in this data set, see the section "OUTSTAT= Data Set" on page 1305.

OUTSTORE=*store-name* (Experimental)
names the item store to contain the context and results of the severity model estimation process. The resulting item store has a binary file format that cannot be modified. You can specify this item store in a subsequent PROC HPSEVERITY step by using the INSTORE= option.

The *store-name* is a usual one- or two-level SAS name, as for SAS data sets. If you specify a one-level name, then the item store resides in the WORK library and is deleted at the end of the SAS session. Because item stores are meant to be consumed by a subsequent PROC HPSEVERITY step for parameter initialization, typical usage specifies a two-level name of the form *libname.membername*.

This option is more useful than the OUTEST= option, especially when you specify a scale regression model that contains interaction effects or effects that have CLASS variables. You can initialize such scale regression models in a subsequent PROC HPSEVERITY step only by specifying the item store that this option creates as an INSTORE= item store in that step.

PLOTS < (*global-plot-options*) > < =*plot-request-option* >

PLOTS < (*global-plot-options*) > < =(*plot-request-option . . . plot-request-option*) >

specifies the desired graphical output. The graphical output is created only when you run PROC HPSEVERITY in single-machine mode. If you specify more than one *global-plot-option*, then separate them with spaces and enclose them in parentheses. If you specify more than one *plot-request-option*, then separate them with spaces and enclose them in parentheses.

You can specify the following *global-plot-options*:

HISTOGRAM
plots the histogram of the response variable on the PDF plots.

KERNEL
plots the kernel estimate of the probability density of the response variable on the PDF plots.

ONLY
turns off the default graphical output and creates only the requested plots.

You can specify the following *plot-request-options*:

ALL
creates all the graphical output.

CDF
creates a plot that compares the cumulative distribution function (CDF) estimates of all the candidate distribution models to the empirical distribution function (EDF) estimate. The plot does not contain CDF estimates for models whose parameter estimation process does not converge.

CDFPERDIST
creates a plot of the CDF estimates of each candidate distribution model. A plot is not created for models whose parameter estimation process does not converge.

NONE
creates none of the graphical output. If you specify this option, then it overrides all the other *plot-request-options*. The default graphical output is also suppressed.

PDF
creates a plot that compares the probability density function (PDF) estimates of all the candidate distribution models. The plot does not contain PDF estimates for models whose parameter estimation process does not converge.

PDFPERDIST
creates a plot of the PDF estimates of each candidate distribution model. A plot is not created for models whose parameter estimation process does not converge.

PP
creates the probability-probability plot (known as the P-P plot), which compares the CDF estimate of each candidate distribution model to the empirical distribution function (EDF). The data that are shown in this plot are used for computing the EDF-based statistics of fit.

QQ
creates the quantile-quantile plot (known as the Q-Q plot), which compares the empirical quantiles to the quantiles of each candidate distribution model.

If you do not specify the PLOTS= option or if you do not specify the ONLY *global-plot-option*, then the default graphical output is equivalent to specifying PLOTS(HISTOGRAM KERNEL)=(CDF PDF).

PRINT <(*global-display-option***)> <=***display-option***>**

PRINT <(*global-display-option***)> <= (***display-option . . . display-option***) >**
specifies the desired displayed output. If you specify more than one *display-option*, then separate them with spaces and enclose them in parentheses.

You can specify the following *global-display-option*:

ONLY
turns off the default displayed output and displays only the requested output.

You can specify the following *display-options*:

ALL
displays all the output.

ALLFITSTATS
displays the comparison of all the statistics of fit for all the models in one table. The table does not include the models whose parameter estimation process does not converge.

CONVSTATUS
displays the convergence status of the parameter estimation process.

DESCSTATS
displays the descriptive statistics for the response variable. If you specify the SCALEMODEL statement, then this option also displays the descriptive statistics for the regression effects that do not contain a CLASS variable.

DISTINFO
displays the information about each specified distribution. For each distribution, the information includes the name, description, validity status, and number of distribution parameters.

ESTIMATES | PARMEST
displays the final estimates of parameters. The estimates are not displayed for models whose parameter estimation process does not converge.

ESTIMATIONDETAILS
displays the details of the estimation process for all the models in one table.

INITIALVALUES
displays the initial values and bounds used for estimating each model.

NLOHISTORY
displays the iteration history of the nonlinear optimization process used for estimating the parameters.

NLOSUMMARY
displays the summary of the nonlinear optimization process used for estimating the parameters.

NONE
displays none of the output. If you specify this option, then it overrides all other display options. The default displayed output is also suppressed.

SELECTION | SELECT
displays the model selection table.

STATISTICS | FITSTATS
displays the statistics of fit for each model. The statistics of fit are not displayed for models whose parameter estimation process does not converge.

If you do not specify the PRINT= option or if you do not specify the ONLY *global-display-option*, then the default displayed output is equivalent to specifying PRINT=(SELECTION CONVSTATUS NLOSUMMARY STATISTICS ESTIMATES).

VARDEF=DF | N
specifies the denominator to use for computing the covariance estimates. You can specify one of the following values:

DF
specifies that the number of nonmissing observations minus the model degrees of freedom (number of parameters) be used.

N
specifies that the number of nonmissing observations be used.

For more information about the covariance estimation, see the section "Estimating Covariance and Standard Errors" on page 1243.

The following *options* control the model estimation and selection process:

CRITERION | CRITERIA | CRIT=*criterion-option*
specifies the model selection criterion.

If you specify two or more candidate models for estimation, then the one with the best value for the selection criterion is chosen as the best model. If you specify the OUTSTAT= data set, then the best model's observation has a value of 1 for the _SELECTED_ variable.

You can specify one of the following *criterion-options*:

AD
specifies the Anderson-Darling (AD) statistic value, which is computed by using the empirical distribution function (EDF) estimate, as the selection criterion. A lower value is deemed better.

AIC
specifies Akaike's information criterion (AIC) as the selection criterion. A lower value is deemed better.

AICC
specifies the finite-sample corrected Akaike's information criterion (AICC) as the selection criterion. A lower value is deemed better.

BIC
specifies the Schwarz Bayesian information criterion (BIC) as the selection criterion. A lower value is deemed better.

CUSTOM
specifies the custom objective function as the selection criterion. You can specify this only if you also specify the OBJECTIVE= option. A lower value is deemed better.

CVM
specifies the Cramér-von Mises (CvM) statistic value, which is computed by using the empirical distribution function (EDF) estimate, as the selection criterion. A lower value is deemed better.

KS
specifies the Kolmogorov-Smirnov (KS) statistic value, which is computed by using the empirical distribution function (EDF) estimate, as the selection criterion. A lower value is deemed better.

LOGLIKELIHOOD | LL
specifies $-2 * \log(L)$ as the selection criterion, where L is the likelihood of the data. A lower value is deemed better. This is the default.

For more information about these *criterion-options*, see the section "Statistics of Fit" on page 1266.

EMPIRICALCDF | EDF=*method*
specifies the method to use for computing the nonparametric or empirical estimate of the cumulative distribution function of the data. You can specify one of the following values for *method*:

AUTOMATIC | AUTO
specifies that the method be chosen automatically based on the data specification.

If you do not specify any censoring or truncation, then the standard empirical estimation method (STANDARD) is chosen. If you specify both right-censoring and left-censoring, then Turnbull's estimation method (TURNBULL) is chosen. For all other combinations of censoring and truncation, the Kaplan-Meier method (KAPLANMEIER) is chosen.

KAPLANMEIER | KM
specifies that the product limit estimator proposed by Kaplan and Meier (1958) be used. Specification of this method has no effect when you specify both right-censoring and left-censoring.

MODIFIEDKM | MKM <(*options*)>
specifies that the modified product limit estimator be used. Specification of this method has no effect when you specify both right-censoring and left-censoring.

This method allows Kaplan-Meier's product limit estimates to be more robust by ignoring the contributions to the estimate due to small risk-set sizes. The risk set is the set of observations at the risk of failing, where an observation is said to fail if it has not been processed yet and might experience censoring or truncation. You can specify the minimum risk-set size that makes it eligible to be included in the estimation either as an absolute lower bound on the size (RSLB= option) or a relative lower bound determined by the formula cn^{α} proposed by Lai and Ying (1991). You can specify the values of c and α by using the C= and ALPHA= options, respectively. By default, the relative lower bound is used with values of $c = 1$ and $\alpha = 0.5$. However, you can modify the default by using the following *options*:

ALPHA | A=*number*
specifies the value to use for α when the lower bound on the risk set size is defined as cn^{α}. This value must satisfy $0 < \alpha < 1$.

C=*number*
specifies the value to use for c when the lower bound on the risk set size is defined as cn^{α}. This value must satisfy $c > 0$.

RSLB=*number*
specifies the absolute lower bound on the risk set size to be included in the estimate.

NOTURNBULL
specifies that the method be chosen automatically based on the data specification and that Turnbull's method not be used. This option is the default.

This method first replaces each left-censored or interval-censored observation with an uncensored observation. If the resulting set of observations has any truncated or right-censored observations, then the Kaplan-Meier method (KAPLANMEIER) is chosen. Otherwise, the standard empirical estimation method (STANDARD) is chosen. The observations are modified only for the purpose of computing the EDF estimates; the modification does not affect the parameter estimation process.

STANDARD | STD
specifies that the standard empirical estimation method be used. If you specify both right-censoring and left-censoring, then the specification of this method has no effect. If you specify any other combination of censoring or truncation effects, then this method ignores such effects, and can thus result in estimates that are more biased than those obtained with other methods that are more suitable for censored or truncated data.

TURNBULL | EM <(*options*)>
specifies that the Turnbull's method be used. This method is used when you specify both right-censoring and left-censoring. An iterative expectation-maximization (EM) algorithm proposed by Turnbull (1976) is used to compute the empirical estimates. If you also specify truncation, then the modification suggested by Frydman (1994) is used.

This method is used if you specify both right-censoring and left-censoring and if you explicitly specify the EMPIRICALCDF=TURNBULL option.

You can modify the default behavior of the EM algorithm by using the following *options*:

ENSUREMLE
specifies that the final EDF estimates be maximum likelihood estimates. The Kuhn-Tucker conditions are computed for the likelihood maximization problem and checked to ensure that EM algorithm converges to maximum likelihood estimates. The method generalizes the method proposed by Gentleman and Geyer (1994) by taking into account any truncation information that you might specify.

EPS=*number*
specifies the maximum relative error to be allowed between estimates of two consecutive iterations. This criterion is used to check the convergence of the algorithm. If you do not specify this option, then PROC HPSEVERITY uses a default value of 1.0E–8.

MAXITER=*number*
specifies the maximum number of iterations to attempt to find the empirical estimates. If you do not specify this option, then PROC HPSEVERITY uses a default value of 500.

ZEROPROB=*number*
specifies the threshold below which an empirical estimate of the probability is considered zero. This option is used to decide if the final estimate is a maximum likelihood estimate. This option does not have an effect if you do not specify the ENSUREMLE option. If you specify the ENSUREMLE option, but do not specify this option, then PROC HPSEVERITY uses a default value of 1.0E–8.

For more information about each of the methods, see the section "Empirical Distribution Function Estimation Methods" on page 1260.

OBJECTIVE=*symbol-name*
names the symbol that represents the objective function in the SAS programming statements that you specify. For each model to be estimated, PROC HPSEVERITY executes the programming statements to compute the value of this symbol for each observation. The values are added across all observations to obtain the value of the objective function. The optimization algorithm estimates the model parameters such that the objective function value is *minimized*. A separate optimization problem is solved for each candidate distribution. If you specify a BY statement, then a separate optimization problem is solved for each candidate distribution within each BY group.

For more information about writing SAS programming statements to define your own objective function, see the section "Custom Objective Functions" on page 1298.

BY Statement

BY *variable-list* **;**

A BY statement can be used in the HPSEVERITY procedure to process the input data set in groups of observations defined by the BY variables. If you specify the BY statement, then PROC HPSEVERITY expects the input data set to be sorted in the order of the BY variables unless you specify the NOTSORTED option.

The BY statement is always supported in the single-machine mode of execution. For the distributed mode, it is supported only when the DATA= data set resides on the client machine. In other words, the BY statement is supported only in the client-data (or local-data) mode of the distributed computing model and not for any of the alongside modes, such as the alongside-the-database or alongside-HDFS mode.

CLASS Statement

CLASS *variable* < *(options)* > . . . < *variable* < *(options)* > > < / *global-options* > **;**

The CLASS statement names the classification variables to be used in the scale regression model. These variables enter the analysis not through their values, but through levels to which the unique values are mapped. For more information about these mappings, see the section "Levelization of Classification Variables" on page 1251.

If you specify a CLASS statement, then it must precede the SCALEMODEL statement.

You can specify options either as individual variable *options* or as *global-options*. You can specify *options* for each variable by enclosing the options in parentheses after the variable name. You can also specify *global-options* for the CLASS statement by placing them after a slash (/). *Global-options* are applied to all the variables that you specify in the CLASS statement. If you specify more than one CLASS statement, the *global-options* that are specified in any one CLASS statement apply to all CLASS statements. However, individual CLASS variable *options* override the *global-options*.

You can specify the following values for either an *option* or a *global-option*:

DESCENDING

DESC

reverses the sort order of the classification variable. If you specify both the DESCENDING and ORDER= options, the HPSEVERITY procedure orders the levels of classification variables according to the ORDER= option and then reverses that order.

ORDER=DATA | FORMATTED | INTERNAL

ORDER=FREQ | FREQDATA | FREQFORMATTED | FREQINTERNAL

specifies the sort order for the levels of classification variables. This order is used by the parameterization method to create the parameters in the model. By default, ORDER=FORMATTED. For ORDER=FORMATTED and ORDER=INTERNAL, the sort order is machine-dependent. When ORDER=FORMATTED is in effect for numeric variables for which you have supplied no explicit format, the levels are ordered by their internal values.

The following table shows how the HPSEVERITY procedure interprets values of the ORDER= option.

Value of ORDER=	Levels Sorted By
DATA	Order of appearance in the input data set
FORMATTED	External formatted values, except for numeric variables that have no explicit format, which are sorted by their unformatted (internal) values
FREQ	Descending frequency count (levels that have more observations come earlier in the order)
FREQDATA	Order of descending frequency count, and within counts by order of appearance in the input data set when counts are tied
FREQFORMATTED	Order of descending frequency count, and within counts by formatted value when counts are tied
FREQINTERNAL	Order of descending frequency count, and within counts by unformatted (internal) value when counts are tied
INTERNAL	Unformatted value

For more information about sort order, see the chapter about the SORT procedure in *Base SAS Procedures Guide* and the discussion of BY-group processing in *SAS Language Reference: Concepts*.

REF=*'level'* | *keyword*

REFERENCE=*'level'* | *keyword*

specifies the reference level that is used when you specify PARAM=REFERENCE. For an individual (but not a global) variable REF= *option*, you can specify the *level* of the variable to use as the reference level. Specify the formatted value of the variable if a format is assigned. For a REF= *option* or *global-option*, you can use one of the following *keywords*.

FIRST designates the first-ordered level as reference.

LAST designates the last-ordered level as reference.

By default, REF=LAST.

If you choose a reference level for any CLASS variable, all variables are parameterized in the reference parameterization for computational efficiency. In other words, the HPSEVERITY procedure applies a single parameterization method to all classification variables.

Suppose that the variable temp has three levels (`'hot'`, `'warm'`, and `'cold'`) and that the variable gender has two levels (`'M'` and `'F'`). The following statements fit a scale regression model:

```
proc hpseverity;
   loss y;
   class gender(ref='F') temp;
   scalemodel gender*temp gender;
run;
```

Both CLASS variables are in reference parameterization in this model. The reference levels are `'F'` for the variable gender and `'warm'` for the variable temp, because the statements are equivalent to the following statements:

```
proc hpseverity;
   loss y;
   class gender(ref='F') temp(ref=last);
   scalemodel gender*temp gender;
run;
```

You can specify the following *global-options*:

MISSING

treats missing values (“.”, “.A”, . . . , “.Z” for numeric variables and blanks for character variables) as valid values for the CLASS variable.

If you do not specify the MISSING option, observations that have missing values for CLASS variables are removed from the analysis, even if the CLASS variables are not used in the model formulation.

PARAM=*keyword*

specifies the parameterization method for the classification variable or variables. You can specify the following *keywords*:

GLM specifies a less-than-full-rank reference cell coding.

REFERENCE specifies a reference cell encoding. You can choose the reference value by specifying an option for a specific *variable* or set of *variables* in the CLASS statement, or you can designate the first- or last-ordered value by specifying a *global-option*. By default, REFERENCE=LAST.

The GLM parameterization is the default. For more information about how parameterization of classification variables affects the construction and interpretation of model effects, see the section "Specification and Parameterization of Model Effects" on page 1253.

TRUNCATE< =*n* **>**

specifies the truncation width of formatted values of CLASS variables when the optional *n* is specified.

If *n* is not specified, the TRUNCATE option requests that classification levels be determined by using no more than the first 16 characters of the formatted values of CLASS variables.

DIST Statement

DIST *distribution-name-or-keyword* < *(distribution-option)* < *distribution-name-or-keyword* < *(distribution-option)* > > . . . > < / *preprocess-options* > **;**

The DIST statement specifies candidate distributions to be estimated by the HPSEVERITY procedure. You can specify multiple DIST statements, and each statement can contain one or more distribution specifications.

For your convenience, PROC HPSEVERITY provides the following 10 different predefined distributions (the name in the parentheses is the name to use in the DIST statement): Burr (BURR), exponential (EXP), gamma (GAMMA), generalized Pareto (GPD), inverse Gaussian or Wald (IGAUSS), lognormal (LOGN), Pareto (PARETO), Tweedie (TWEEDIE), scaled Tweedie (STWEEDIE), and Weibull (WEIBULL). These are described in detail in the section "Predefined Distributions" on page 1230.

You can specify any of the predefined distributions or any distribution that you have defined. If a distribution that you specify is not a predefined distribution, then you must submit the CMPLIB= system option with appropriate libraries before you submit the PROC HPSEVERITY step to enable the procedure to find the functions associated with your distribution. The predefined distributions are defined in the Sashelp.Svrtdist library. However, you are not required to specify this library in the CMPLIB= system option. For more information about defining your own distributions, see the section "Defining a Severity Distribution Model with the FCMP Procedure" on page 1273.

As a convenience, you can also use a shortcut keyword to indicate a list of distributions. You can specify one or more of the following keywords:

ALL

specifies all the predefined distributions and the distributions that you have defined in the libraries that you specify in the CMPLIB= system option. In addition to the eight predefined distributions included by the _PREDEFINED_ keyword, this list also includes the Tweedie and scaled Tweedie distributions that are defined in the Sashelp.Svrtdist library.

PREDEFINED
specifies the list of eight predefined distributions: BURR, EXP, GAMMA, GPD, IGAUSS, LOGN, PARETO, and WEIBULL. Although the TWEEDIE and STWEEDIE distributions are available in the Sashelp.Svrtdist library along with these eight distributions, they are not included by this keyword. If you want to fit the TWEEDIE and STWEEDIE distributions, then you must specify them explicitly or use the _ALL_ keyword.

USER
specifies the list of all the distributions that you have defined in the libraries that you specify in the CMPLIB= system option. This list does not include the distributions defined in the Sashelp.Svrtdist library, even if you specify the Sashelp.Svrtdist library in the CMPLIB= option.

The use of these keywords, especially _ALL_, can result in a large list of distributions, which might take a longer time to estimate. A warning is printed to the SAS log if the number of total distribution models to estimate exceeds 10.

If you specify the OUTCDF= option or request a CDF plot and you do not specify any DIST statement, then PROC HPSEVERITY does not fit any distributions and produces the empirical estimates of the cumulative distribution function.

The following *distribution-option* values can be used in the DIST statement for a distribution name that is not a shortcut keyword:

INIT=*(name=value . . . name=value)*
specifies the initial values to be used for the distribution parameters to start the parameter estimation process. You must specify the values by parameter names and the parameter names must match the names used in the model definition. For example, let a model M's definition contain a M_PDF function with following signature:

```
function M_PDF(x, alpha, beta);
```

For this model, the names **alpha** and **beta** must be used for the INIT option. The names are case-insensitive. If you do not specify initial values for some parameters in the INIT statement, then a default value of 0.001 is assumed for those parameters. If you specify an incorrect parameter, PROC HPSEVERITY prints a warning to the SAS log and does not fit the model. All specified values must be nonmissing.

If you are modeling regression effects, then the initial value of the first distribution parameter (**alpha** in the preceding example) should be the initial *base* value of the scale parameter or log-transformed scale parameter. For more information, see the section "Estimating Regression Effects" on page 1245.

The use of INIT= option is one of the three methods available for initializing the parameters. For more information, see the section "Parameter Initialization" on page 1244. If none of the initialization methods is used, then PROC HPSEVERITY initializes all parameters to 0.001.

You can specify the following *preprocess-options* in the DIST statement:

LISTONLY
specifies that the list of all candidate distributions be printed to the SAS log without doing any further processing on them. This option is especially useful when you use a shortcut keyword to include a list of distributions. It enables you to find out which distributions are included by the keyword.

VALIDATEONLY
specifies that all candidate distributions be checked for validity without doing any further processing on them. If a distribution is invalid, the reason for invalidity is written to the SAS log. If all distributions are valid, then the distribution information is written to the SAS log. The information includes name, description, validity status (valid or invalid), and number of distribution parameters. The information is not written to the SAS log if you specify an OUTMODELINFO= data set or the PRINT=DISTINFO or PRINT=ALL option in the PROC HPSEVERITY statement. This option is especially useful when you specify your own distributions or when you specify the _USER_ or _ALL_ keywords in the DIST statement. It enables you to check whether your custom distribution definitions satisfy PROC HPSEVERITY's requirements for the specified modeling task. It is recommended that you specify the SCALEMODEL statement if you intend to fit a model with regression effects, because the SCALEMODEL statement instructs PROC HPSEVERITY to perform additional checks to validate whether regression effects can be modeled on each candidate distribution.

LOSS Statement

LOSS < *response-variable-name* > < / *censoring-truncation-options* > **;**

The LOSS statement specifies the name of the response or loss variable whose distribution needs to be modeled. You can also specify additional options to indicate any truncation or censoring of the response. The specification of response variable is optional if you specify at least one type of censoring. You must specify a response variable if you do not specify any censoring. If you specify more than one LOSS statement, then the first statement is used.

All the analysis variables that you specify in this statement must be present in the input data set that you specify by using the DATA= option in the PROC HPSEVERITY statement. The response variable is expected to have nonmissing values. If the variable has a missing value in an observation, then a warning is written to the SAS log and that observation is ignored.

The following *censoring-truncation-options* can be used in the LOSS statement:

LEFTCENSORED | LC=*variable-name*

LEFTCENSORED | LC=*number*
specifies the left-censoring variable or a global left-censoring limit.

You can use the *variable-name* argument to specify a data set variable that contains the left-censoring limit. If the value of this variable is missing, then PROC HPSEVERITY assumes that such observations are not left-censored.

Alternatively, you can use the *number* argument to specify a left-censoring limit value that applies to all the observations in the data set. This limit must be a nonzero positive number.

By the definition of left-censoring, an exact value of the response is not known when it is less than or equal to the left-censoring limit. If you specify the response variable and the value of that variable is less than or equal to the value of the left-censoring limit for some observations, then PROC HPSEVERITY treats such observations as left-censored and the value of the response variable is ignored. If you specify the response variable and the value of that variable is greater than the value of the left-censoring limit for some observations, then PROC HPSEVERITY assumes that such observations are not left-censored and the value of the left-censoring limit is ignored.

If you specify both right-censoring and left-censoring limits, then the left-censoring limit must be greater than or equal to the right-censoring limit. If both limits are identical, then the observation is assumed to be uncensored.

For more information about left-censoring, see the section "Censoring and Truncation" on page 1240.

LEFTTRUNCATED | LT=*variable-name* < *(left-truncation-option)* >

LEFTTRUNCATED | LT=*number* < *(left-truncation-option)* >

specifies the left-truncation variable or a global left-truncation threshold.

You can use the *variable-name* argument to specify a data set variable that contains the left-truncation threshold. If the value of this variable is missing or 0 for some observations, then PROC HPSEVERITY assumes that such observations are not left-truncated.

Alternatively, you can use the *number* argument to specify a left-truncation threshold that applies to all the observations in the data set. This threshold must be a nonzero positive number.

It is assumed that the response variable contains the observed values. By the definition of left-truncation, you can observe only a value that is greater than the left-truncation threshold. If a response variable value is less than or equal to the left-truncation threshold, a warning is printed to the SAS log, and the observation is ignored. For more information about left-truncation, see the section "Censoring and Truncation" on page 1240.

You can specify the following *left-truncation-option* for an alternative interpretation of the left-truncation threshold:

PROBOBSERVED | POBS=*number*

specifies the probability of observability, which is defined as the probability that the underlying severity event is observed (and recorded) for the specified left-threshold value.

The specified *number* must lie in the (0.0, 1.0] interval. A value of 1.0 is equivalent to specifying that there is no left-truncation, because it means that no severity events can occur with a value less than or equal to the threshold. If you specify value of 1.0, PROC HPSEVERITY prints a warning to the SAS log and proceeds by assuming that LEFTTRUNCATED= option is not specified.

For more information, see the section "Probability of Observability" on page 1241.

RIGHTCENSORED | RC=*variable-name*

RIGHTCENSORED | RC=*number*

specifies the right-censoring variable or a global right-censoring limit.

You can use the *variable-name* argument to specify a data set variable that contains the right-censoring limit. If the value of this variable is missing, then PROC HPSEVERITY assumes that such observations are not right-censored.

Alternatively, you can use the *number* argument to specify a right-censoring limit value that applies to all the observations in the data set. This limit must be a nonzero positive number.

By the definition of right-censoring, an exact value of the response is not known when it is greater than or equal to the right-censoring limit. If you specify the response variable and the value of that variable is greater than or equal to the value of the right-censoring limit for some observations, then PROC HPSEVERITY treats such observations as right-censored and the value of the response variable is ignored. If you specify the response variable and the value of that variable is less than the value of the

right-censoring limit for some observations, then PROC HPSEVERITY assumes that such observations are not right-censored and the value of the right-censoring limit is ignored.

If you specify both right-censoring and left-censoring limits, then the left-censoring limit must be greater than or equal to the right-censoring limit. If both limits are identical, then the observation is assumed to be uncensored.

For more information about right-censoring, see the section "Censoring and Truncation" on page 1240.

RIGHTTRUNCATED | RT=*variable-name*

RIGHTTRUNCATED | RT=*number*

specifies the right-truncation variable or a global right-truncation threshold.

You can use the *variable-name* argument to specify a data set variable that contains the right-truncation threshold. If the value of this variable is missing for some observations, then PROC HPSEVERITY assumes that such observations are not right-truncated.

Alternatively, you can use the *number* argument to specify a right-truncation threshold that applies to all the observations in the data set. This threshold must be a nonzero positive number.

It is assumed that the response variable contains the observed values. By the definition of right-truncation, you can observe only a value that is less than or equal to the right-truncation threshold. If a response variable value is greater than the right-truncation threshold, a warning is printed to the SAS log, and the observation is ignored. For more information about right-truncation, see the section "Censoring and Truncation" on page 1240.

NLOPTIONS Statement

NLOPTIONS *options* **;**

The HPSEVERITY procedure uses the nonlinear optimization (NLO) subsystem to perform the nonlinear optimization of the likelihood function to obtain the estimates of distribution and regression parameters. You can use the NLOPTIONS statement to control different aspects of this optimization process. For most problems, the default settings of the optimization process are adequate. However, in some cases it might be useful to change the optimization technique or to change the maximum number of iterations. The following statement uses the MAXITER= option to set the maximum number of iterations to 200 and uses the TECH= option to change the optimization technique to the double-dogleg optimization (DBLDOG) rather than the default technique, the trust region optimization (TRUREG), that is used in the HPSEVERITY procedure:

```
nloptions tech=dbldog maxiter=200;
```

A discussion of the full range of *options* that can be used in the NLOPTIONS statement is given in Chapter 7, "Nonlinear Optimization Methods." The HPSEVERITY procedure supports all those options except the options that are related to displaying the optimization information. You can use the PRINT= option in the PROC HPSEVERITY statement to request the optimization summary and iteration history. If you specify more than one NLOPTIONS statement, then the first statement is used.

OUTSCORELIB Statement

OUTSCORELIB < **OUTLIB**=> *fcmp-library-name options* **;**

The OUTSCORELIB statement specifies the library to write scoring functions to. Scoring functions enable you to easily compute a distribution function on the fitted parameters of the distribution without going through a potentially complex process of extracting the fitted parameter estimates from other output such as the OUTEST= data set that is created by PROC HPSEVERITY.

If you specify the SCALEMODEL statement and if you specify interaction or classification effects, then PROC HPSEVERITY ignores the OUTSCORELIB statement and does not generate scoring functions. In other words, if you specify the SCALEMODEL statement, then PROC HPSEVERITY generates scoring functions if you specify only singleton continuous effects in the SCALEMODEL statement.

You must specify the following option as the first option in the statement:

OUTLIB=*fcmp-library-name*
names the FCMP library to contain the scoring functions. PROC HPSEVERITY writes the scoring functions to the FCMP library named *fcmp-library-name*. If a library or data set named *fcmp-library-name* already exists, PROC HPSEVERITY deletes it before proceeding.

This option is similar to the OUTLIB= option that you would specify in a PROC FCMP statement, except that *fcmp-library-name* must be a two-level name whereas the OUTLIB= option in the PROC FCMP statement requires a three-level name. The third level of a three-level name specifies the package to which the functions belong. You do not need to specify the package name in the *fcmp-library-name*, because PROC HPSEVERITY automatically creates the package for you. By default, a separate package is created for each distribution that has not failed to converge. Each package is named for a distribution. For example, if you define and fit a distribution named *mydist*, and if *mydist* does not fail to converge, then PROC HPSEVERITY creates a package named *mydist* in the OUTLIB= library that you specify. Further, let the definition of the *mydist* distribution contain three distribution functions, *mydist_PDF(x,Parm1,Parm2)*, *mydist_LOGCDF(x,Parm1,Parm2)*, and *mydist_XYZ(x,Parm1,Parm2)*. If you specify the OUTSCORELIB statement

```
outscorelib outlib=sasuser.scorefunc;
```

then the Sasuser.Scorefunc library contains the following three functions in a package named *mydist*: *SEV_PDF(x)*, *SEV_LOGCDF(x)*, and *SEV_XYZ(x)*.

The key feature of scoring functions is that they do not require the parameter arguments (*Parm1* and *Parm2* in this example). The fitted parameter estimates are encoded inside the scoring function so that you can compute or score the value of each function for a given value of the loss variable without having to know or extract the parameter estimates through some other means.

For convenience, you can omit the OUTLIB= portion of the specification and just specify the name, as in the following example:

```
outscorelib sasuser.scorefunc;
```

When the HPSEVERITY procedure runs successfully, the *fcmp-library-name* is appended to the CMPLIB system option, so you can immediately start using the scoring functions in a DATA step or PROC FCMP step.

You can specify the following *options* in the OUTSCORELIB statement:

COMMONPACKAGE

ONEPACKAGE

requests that only one common package be created to contain all the scoring functions.

If you specify this option, then all the scoring functions are created in a package called *sevfit*. For each distribution function that has the name *distribution_suffix*, the name of the corresponding scoring function is formed as *SEV_suffix_distribution*. For example, the scoring function of the distribution function 'MYDIST_BAR' is named 'SEV_BAR_MYDIST'.

If you do not specify this option, then all scoring functions for a distribution are created in a package that has the same name as the distribution, and for each distribution function that has the name *distribution_suffix*, the name of the corresponding scoring function is formed as *SEV_suffix*. For example, the scoring function of the distribution function 'MYDIST_BAR' is named 'SEV_BAR'.

OUTBYID=*SAS-data-set*

names the output data set to contain the unique identifier for each BY group. This unique identifier is used as part of the name of the package or scoring function for each distribution. This is a required option when you specify a BY statement in PROC HPSEVERITY.

The OUTBYID= data set contains one observation per BY group and a variable named _ID_ in addition to the BY variables that you specify in the BY statement. The _ID_ variable contains the unique identifier for each BY group. The identifier of the BY group is the decimal representation of the sequence number of the BY group. The first BY group has an identifier of 1, the second BY group has an identifier of 2, the tenth BY group has an identifier of 10, and so on.

If you do not specify the COMMONPACKAGE option in the OUTSCORELIB statement, then for each distribution, PROC HPSEVERITY creates as many packages as the number of BY groups. The unique BY-group identifier is used as a suffix for the package name. For example, if your DATA= data set has three BY groups and if you specify the OUTSCORELIB statement

```
outscorelib outlib=sasuser.byscorefunc outbyid=sasuser.byid;
```

then for the distribution 'MYDIST', the Sasuser.Byscorefunc library contains the three packages 'MYDIST1', 'MYDIST2', and 'MYDIST3', and each package contains one scoring function named 'SEV_BAR' for each distribution function named 'MYDIST_BAR'.

If you specify the COMMONPACKAGE option in the OUTSCORELIB statement, PROC HPSEVERITY creates as many versions of the distribution function as the number of BY groups. The unique BY-group identifier is used as a suffix for the function name. Extending the previous example, if you specify the OUTSCORELIB statement with the COMMONPACKAGE option,

```
outscorelib outlib=sasuser.byscorefunc outbyid=sasuser.byid commonpackage;
```

then for the distribution function 'MYDIST_BAR' of the distribution 'MYDIST', the Sasuser.Byscorefunc library contains the following three scoring functions: 'SEV_BAR_MYDIST1',

'SEV_BAR_MYDIST2', and 'SEV_BAR_MYDIST3'. All the scoring functions are created in one common package named *sevfit*.

For both the preceding examples, the Sasuser.Byid data set contains three observations, one for each BY group. The value of the _ID_ variable is 1 for the first BY group, 2 for the second BY group, and 3 for the third BY group.

For more information about scoring functions, see the section "Scoring Functions" on page 1290.

PERFORMANCE Statement

PERFORMANCE *options* **;**

The PERFORMANCE statement defines performance parameters for distributed and multithreaded computing, passes variables that describe the distributed computing environment, and requests detailed results about the performance characteristics of PROC HPSEVERITY.

You can also use the PERFORMANCE statement to control whether a high-performance analytical procedure runs in single-machine or distributed mode.

The PERFORMANCE statement is documented further in the section "PERFORMANCE Statement" on page 87 of Chapter 3, "Shared Concepts and Topics."

SCALEMODEL Statement

SCALEMODEL *regression-effect-list* < / *scalemodel-options* > **;**

The SCALEMODEL statement specifies regression effects. A regression effect is formed from one or more regressor variables according to effect construction rules. Each regression effect forms one element of $\mathbf{X}$ in the linear model structure $\mathbf{X}\boldsymbol{\beta}$ that affects the scale parameter of the distribution. The SCALEMODEL statement in conjunction with the CLASS statement supports a rich set of effects. Effects are specified by a special notation that uses regressor variable names and operators. There are two types of regressor variables: classification (or CLASS) variables and continuous variables. Classification variables can be either numeric or character and are specified in a CLASS statement. To include CLASS variables in regression effects, you must specify the CLASS statement so that it appears before the SCALEMODEL statement. A regressor variable that is not declared in the CLASS statement is assumed to be continuous. For more information about effect construction rules, see the section "Specification and Parameterization of Model Effects" on page 1253.

All the regressor variables must be present in the input data set that you specify by using the DATA= option in the PROC HPSEVERITY statement. The scale parameter of each candidate distribution is linked to the linear predictor $\mathbf{X}\boldsymbol{\beta}$ that includes an intercept. If a distribution does not have a scale parameter, then a model based on that distribution is not estimated. If you specify more than one SCALEMODEL statement, then the first statement is used.

The regressor variables are expected to have nonmissing values. If any of the variables has a missing value in an observation, then a warning is written to the SAS log and that observation is ignored.

For more information about modeling regression effects, see the section "Estimating Regression Effects" on page 1245.

You can specify the following *scalemodel-options* in the SCALEMODEL statement:

DFMIXTURE=*method-name* **<** *(method-options)* **>**
specifies the method for computing representative estimates of the cumulative distribution function (CDF) and the probability density function (PDF).

When you specify regression effects, the scale of the distribution depends on the values of the regressors. For a given distribution family, each observation in the input data set implies a different scaled version of the distribution. To compute estimates of CDF and PDF that are comparable across different distribution families, PROC HPSEVERITY needs to construct a single representative distribution from all such distributions. You can specify one of the following *method-name* values to specify the method that is used to construct the representative distribution. For more information about each of the methods, see the section "CDF and PDF Estimates with Regression Effects" on page 1249.

FULL
specifies that the representative distribution be the mixture of N distributions such that each distribution has a scale value that is implied by each of the N observations that are used for estimation. This method is the slowest.

MEAN
specifies that the representative distribution be the one-point mixture of the distribution whose scale value is computed by using the mean of the N values of the linear predictor that are implied by the N observations that are used for estimation. If you do not specify the DFMIXTURE= option, then this method is used by default. This is also the fastest method.

QUANTILE < (K=*q***) >**
specifies that the representative distribution be the mixture of a fixed number of distributions whose scale values are computed by using the quantiles from the sample of N values of the linear predictor that are implied by the N observations that are used for estimation.

You can use the K= option to specify the number of distributions in the mixture. If you specify K=q, then the mixture contains $(q-1)$ distributions such that each distribution has as its scale one of the $(q-1)$-quantiles.

If you do not specify the K= option, then PROC HPSEVERITY uses the default of 2, which implies the use of a one-point mixture with a distribution whose scale value is the median of all scale values.

RANDOM < (*random-method-options***) >**
specifies that the representative distribution be the mixture of a fixed number of distributions whose scale values are computed by using the values of the linear predictor that are implied by a randomly chosen subset of the set of all observations that are used for estimation. The same subset of observations is used for each distribution family.

You can specify the following *random-method-options* to specify how the subset is chosen:

K=*r*
specifies the number of distributions to include in the mixture. If you do not specify this option, then PROC HPSEVERITY uses the default of 15.

SEED=*number*
specifies the seed that is used to generate the uniform random sample of observation indices. If you do not specify this option, then PROC HPSEVERITY generates a seed internally that is based on the current value of the system clock.

OFFSET=*offset-variable-name*
specifies the name of the offset variable in the scale regression model. An offset variable is a regressor variable whose regression coefficient is known to be 1. For more information, see the section "Offset Variable" on page 1246.

WEIGHT Statement

WEIGHT *variable-name* **;**

The WEIGHT statement specifies the name of a variable whose values represent the weight of each observation. PROC HPSEVERITY associates a weight of w to each observation, where w is the value of the WEIGHT variable for the observation. If the weight value is missing or less than or equal to 0, then the observation is ignored and a warning is written to the SAS log. When you do not specify the WEIGHT statement, each observation is assigned a weight of 1. If you specify more than one WEIGHT statement, then the last statement is used.

The weights are normalized so that they add up to the actual sample size. In particular, the weight of each observation is multiplied by $\frac{N}{\sum_{i=1}^{N} w_i}$, where N is the sample size.

Programming Statements

You can use a series of programming statements that use variables in the input data set that you specify in the DATA= option in the PROC HPSEVERITY statement to assign a value to an objective function symbol. You must specify the objective function symbol by using the OBJECTIVE= option in the PROC HPSEVERITY statement. If you do not specify the OBJECTIVE= option in the PROC HPSEVERITY statement, then the programming statements are ignored and models are estimated using the maximum likelihood method.

You can use most DATA step statements and functions in your program. Any additional functions, restrictions, and differences are listed in the section "Custom Objective Functions" on page 1298.

Details: HPSEVERITY Procedure

Predefined Distributions

PROC HPSEVERITY assumes the following model for the response variable Y

$$Y \sim \mathcal{F}(\Theta)$$

where $\mathcal{F}$ is a continuous probability distribution with parameters Θ. The model hypothesizes that the observed response is generated from a stochastic process that is governed by the distribution $\mathcal{F}$. This model is usually referred to as the error model. Given a representative input sample of response variable values, PROC HPSEVERITY estimates the model parameters for any distribution $\mathcal{F}$ and computes the statistics of fit for each model. This enables you to find the distribution that is most likely to generate the observed sample.

A set of predefined distributions is provided with the HPSEVERITY procedure. A summary of the distributions is provided in Table 23.2. For each distribution, the table lists the name of the distribution that should be used in the DIST statement, the parameters of the distribution along with their bounds, and the mathematical expressions for the probability density function (PDF) and cumulative distribution function (CDF) of the distribution.

All the predefined distributions, except LOGN and TWEEDIE, are parameterized such that their first parameter is the scale parameter. For LOGN, the first parameter μ is a log-transformed scale parameter. TWEEDIE does not have a scale parameter. The presence of scale parameter or a log-transformed scale parameter enables you to use all of the predefined distributions, except TWEEDIE, as a candidate for estimating regression effects.

A distribution model is associated with each predefined distribution. You can also define your own distribution model, which is a set of functions and subroutines that you define by using the FCMP procedure. For more information, see the section "Defining a Severity Distribution Model with the FCMP Procedure" on page 1273.

Table 23.2 Predefined HPSEVERITY Distributions

Name	Distribution	Parameters	PDF (f) and CDF (F)
BURR	Burr	$\theta > 0, \alpha > 0,$ $\gamma > 0$	$f(x) = \frac{\alpha\gamma z^{\gamma}}{x(1+z^{\gamma})^{(\alpha+1)}}$ $F(x) = 1 - \left(\frac{1}{1+z^{\gamma}}\right)^{\alpha}$
EXP	Exponential	$\theta > 0$	$f(x) = \frac{1}{\theta}e^{-z}$ $F(x) = 1 - e^{-z}$
GAMMA	Gamma	$\theta > 0, \alpha > 0$	$f(x) = \frac{z^{\alpha}e^{-z}}{x\Gamma(\alpha)}$ $F(x) = \frac{\gamma(\alpha,z)}{\Gamma(\alpha)}$
GPD	Generalized Pareto	$\theta > 0, \xi > 0$	$f(x) = \frac{1}{\theta}(1+\xi z)^{-1-1/\xi}$ $F(x) = 1 - (1+\xi z)^{-1/\xi}$
IGAUSS	Inverse Gaussian (Wald)	$\theta > 0, \alpha > 0$	$f(x) = \frac{1}{\theta}\sqrt{\frac{\alpha}{2\pi z^3}}\, e^{\frac{-\alpha(z-1)^2}{2z}}$ $F(x) = \Phi\left((z-1)\sqrt{\frac{\alpha}{z}}\right) +$ $\Phi\left(-(z+1)\sqrt{\frac{\alpha}{z}}\right) e^{2\alpha}$
LOGN	Lognormal	μ (no bounds), $\sigma > 0$	$f(x) = \frac{1}{x\sigma\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{\log(x)-\mu}{\sigma}\right)^2}$ $F(x) = \Phi\left(\frac{\log(x)-\mu}{\sigma}\right)$
PARETO	Pareto	$\theta > 0, \alpha > 0$	$f(x) = \frac{\alpha\theta^{\alpha}}{(x+\theta)^{\alpha+1}}$ $F(x) = 1 - \left(\frac{\theta}{x+\theta}\right)^{\alpha}$
TWEEDIE	Tweedie[6]	$p > 1, \mu > 0,$ $\phi > 0$	$f(x) = a(x,\phi)\exp\left[\frac{1}{\phi}\left(\frac{x\mu^{1-p}}{1-p} - \kappa(\mu,p)\right)\right]$ $F(x) = \int_0^x f(t)dt$
STWEEDIE	Scaled Tweedie[6]	$\theta > 0, \lambda > 0,$ $1 < p < 2$	$f(x) = a(x,\theta,\lambda,p)\exp\left(-\frac{x}{\theta} - \lambda\right)$ $F(x) = \int_0^x f(t)dt$
WEIBULL	Weibull	$\theta > 0, \tau > 0$	$f(x) = \frac{1}{x}\tau z^{\tau}e^{-z^{\tau}}$ $F(x) = 1 - e^{-z^{\tau}}$

Notes:
1. $z = x/\theta$, wherever z is used.
2. θ denotes the scale parameter for all the distributions. For LOGN, $\log(\theta) = \mu$.
3. Parameters are listed in the order in which they are defined in the distribution model.
4. $\gamma(a,b) = \int_0^b t^{a-1}e^{-t}dt$ is the lower incomplete gamma function.
5. $\Phi(y) = \frac{1}{2}\left(1 + \text{erf}\left(\frac{y}{\sqrt{2}}\right)\right)$ is the standard normal CDF.
6. For more information, see the section "Tweedie Distributions" on page 1232.

Tweedie Distributions

Tweedie distributions are a special case of the exponential dispersion family (Jørgensen 1987) with a property that the variance of the distribution is equal to $\phi\mu^p$, where μ is the mean of the distribution, ϕ is a dispersion parameter, and p is an index parameter as discovered by Tweedie (1984). The distribution is defined for all values of p except for values of p in the open interval $(0, 1)$. Many important known distributions are a special case of Tweedie distributions including normal (p=0), Poisson (p=1), gamma (p=2), and the inverse Gaussian (p=3). Apart from these special cases, the probability density function (PDF) of the Tweedie distribution does not have an analytic expression. For $p > 1$, it has the form (Dunn and Smyth 2005),

$$f(x; \mu, \phi, p) = a(x, \phi) \exp\left[\frac{1}{\phi}\left(\frac{x\mu^{1-p}}{1-p} - \kappa(\mu, p)\right)\right]$$

where $\kappa(\mu, p) = \mu^{2-p}/(2-p)$ for $p \neq 2$ and $\kappa(\mu, p) = \log(\mu)$ for $p = 2$. The function $a(x, \phi)$ does not have an analytical expression. It is typically evaluated using series expansion methods described in Dunn and Smyth (2005).

For $1 < p < 2$, the Tweedie distribution is a compound Poisson-gamma mixture distribution, which is the distribution of S defined as

$$S = \sum_{i=1}^{N} X_i$$

where $N \sim \text{Poisson}(\lambda)$ and $X_i \sim \text{gamma}(\alpha, \theta)$ are independent and identically distributed gamma random variables with shape parameter α and scale parameter θ. At $X = 0$, the density is a probability mass that is governed by the Poisson distribution, and for values of $X > 0$, it is a mixture of gamma variates with Poisson mixing probability. The parameters λ, α, and θ are related to the natural parameters μ, ϕ, and p of the Tweedie distribution as

$$\lambda = \frac{\mu^{2-p}}{\phi(2-p)}$$
$$\alpha = \frac{2-p}{p-1}$$
$$\theta = \phi(p-1)\mu^{p-1}$$

The mean of a Tweedie distribution is positive for $p > 1$.

Two predefined versions of the Tweedie distribution are provided with the HPSEVERITY procedure. The first version, named TWEEDIE and defined for $p > 1$, has the natural parameterization with parameters μ, ϕ, and p. The second version, named STWEEDIE and defined for $1 < p < 2$, is the version with a scale parameter. It corresponds to the compound Poisson-gamma distribution with gamma scale parameter θ, Poisson mean parameter λ, and the index parameter p. The index parameter decides the shape parameter α of the gamma distribution as

$$\alpha = \frac{2-p}{p-1}$$

The parameters θ and λ of the STWEEDIE distribution are related to the parameters μ and ϕ of the TWEEDIE distribution as

$$\mu = \lambda\theta\alpha$$
$$\phi = \frac{(\lambda\theta\alpha)^{2-p}}{\lambda(2-p)} = \frac{\theta}{(p-1)(\lambda\theta\alpha)^{p-1}}$$

You can fit either version when there are no regression variables. Each version has its own merits. If you fit the TWEEDIE version, you have the direct estimate of the overall mean of the distribution. If you are interested in the most practical range of the index parameter $1 < p < 2$, then you can fit the STWEEDIE version, which provides you direct estimates of the Poisson and gamma components that comprise the distribution (an estimate of the gamma shape parameter α is easily obtained from the estimate of p).

If you want to estimate the effect of exogenous (regression) variables on the distribution, then you must use the STWEEDIE version, because PROC HPSEVERITY requires a distribution to have a scale parameter in order to estimate regression effects. For more information, see the section "Estimating Regression Effects" on page 1245. The gamma scale parameter θ is the scale parameter of the STWEEDIE distribution. If you are interested in determining the effect of regression variables on the mean of the distribution, you can do so by first fitting the STWEEDIE distribution to determine the effect of the regression variables on the scale parameter θ. Then, you can easily estimate how the mean of the distribution μ is affected by the regression variables using the relationship $\mu = c\theta$, where $c = \lambda\alpha = \lambda(2-p)/(p-1)$. The estimates of the regression parameters remain the same, whereas the estimate of the intercept parameter is adjusted by the estimates of the λ and p parameters.

Parameter Initialization for Predefined Distributions

The parameters are initialized by using the method of moments for all the distributions, except for the gamma and the Weibull distributions. For the gamma distribution, approximate maximum likelihood estimates are used. For the Weibull distribution, the method of percentile matching is used.

Given n observations of the severity value y_i $(1 \le i \le n)$, the estimate of kth raw moment is denoted by m_k and computed as

$$m_k = \frac{1}{n}\sum_{i=1}^{n} y_i^k$$

The $100p$th percentile is denoted by π_p $(0 \le p \le 1)$. By definition, π_p satisfies

$$F(\pi_p-) \le p \le F(\pi_p)$$

where $F(\pi_p-) = \lim_{h\downarrow 0} F(\pi_p - h)$. PROC HPSEVERITY uses the following practical method of computing π_p. Let $\hat{F}_n(y)$ denote the empirical distribution function (EDF) estimate at a severity value y. Let y_p^- and y_p^+ denote two consecutive values in the ascending sequence of y values such that $\hat{F}_n(y_p^-) < p$ and $\hat{F}_n(y_p^+) \ge p$. Then, the estimate $\hat{\pi}_p$ is computed as

$$\hat{\pi}_p = y_p^- + \frac{p - \hat{F}_n(y_p^-)}{\hat{F}_n(y_p^+) - \hat{F}_n(y_p^-)}(y_p^+ - y_p^-)$$

Let ϵ denote the smallest double-precision floating-point number such that $1+\epsilon > 1$. This machine precision constant can be obtained by using the CONSTANT function in Base SAS software.

The details of how parameters are initialized for each predefined distribution are as follows:

BURR The parameters are initialized by using the method of moments. The kth raw moment of the Burr distribution is:

$$E[X^k] = \frac{\theta^k \Gamma(1+k/\gamma)\Gamma(\alpha-k/\gamma)}{\Gamma(\alpha)}, \quad -\gamma < k < \alpha\gamma$$

Three moment equations $E[X^k] = m_k$ $(k = 1, 2, 3)$ need to be solved for initializing the three parameters of the distribution. In order to get an approximate closed form solution, the second shape parameter $\hat{\gamma}$ is initialized to a value of 2. If $2m_3 - 3m_1m_2 > 0$, then simplifying and solving the moment equations yields the following feasible set of initial values:

$$\hat{\theta} = \sqrt{\frac{m_2 m_3}{2m_3 - 3m_1 m_2}}, \quad \hat{\alpha} = 1 + \frac{m_3}{2m_3 - 3m_1 m_2}, \quad \hat{\gamma} = 2$$

If $2m_3 - 3m_1m_2 < \epsilon$, then the parameters are initialized as follows:

$$\hat{\theta} = \sqrt{m_2}, \quad \hat{\alpha} = 2, \quad \hat{\gamma} = 2$$

EXP The parameters are initialized by using the method of moments. The kth raw moment of the exponential distribution is:

$$E[X^k] = \theta^k \Gamma(k+1), \quad k > -1$$

Solving $E[X] = m_1$ yields the initial value of $\hat{\theta} = m_1$.

GAMMA The parameter α is initialized by using its *approximate* maximum likelihood (ML) estimate. For a set of n independent and identically distributed observations y_i $(1 \le i \le n)$ drawn from a gamma distribution, the log likelihood l is defined as follows:

$$\begin{aligned} l &= \sum_{i=1}^{n} \log\left(y_i^{\alpha-1} \frac{e^{-y_i/\theta}}{\theta^\alpha \Gamma(\alpha)} \right) \\ &= (\alpha - 1) \sum_{i=1}^{n} \log(y_i) - \frac{1}{\theta} \sum_{i=1}^{n} y_i - n\alpha \log(\theta) - n \log(\Gamma(\alpha)) \end{aligned}$$

Using a shorter notation of $\sum$ to denote $\sum_{i=1}^{n}$ and solving the equation $\partial l / \partial \theta = 0$ yields the following ML estimate of θ:

$$\hat{\theta} = \frac{\sum y_i}{n\alpha} = \frac{m_1}{\alpha}$$

Substituting this estimate in the expression of l and simplifying gives

$$l = (\alpha - 1) \sum \log(y_i) - n\alpha - n\alpha \log(m_1) + n\alpha \log(\alpha) - n \log(\Gamma(\alpha))$$

Let d be defined as follows:

$$d = \log(m_1) - \frac{1}{n} \sum \log(y_i)$$

Solving the equation $\partial l / \partial \alpha = 0$ yields the following expression in terms of the digamma function, $\psi(\alpha)$:

$$\log(\alpha) - \psi(\alpha) = d$$

The digamma function can be approximated as follows:

$$\hat{\psi}(\alpha) \approx \log(\alpha) - \frac{1}{\alpha}\left(0.5 + \frac{1}{12\alpha + 2}\right)$$

This approximation is within 1.4% of the true value for all the values of $\alpha > 0$ except when α is arbitrarily close to the positive root of the digamma function (which is approximately 1.461632). Even for the values of α that are close to the positive root, the absolute error between true and approximate values is still acceptable ($|\hat{\psi}(\alpha) - \psi(\alpha)| < 0.005$ for $\alpha > 1.07$). Solving the equation that arises from this approximation yields the following estimate of α:

$$\hat{\alpha} = \frac{3 - d + \sqrt{(d-3)^2 + 24d}}{12d}$$

If this approximate ML estimate is infeasible, then the method of moments is used. The kth raw moment of the gamma distribution is:

$$E[X^k] = \theta^k \frac{\Gamma(\alpha + k)}{\Gamma(\alpha)}, \quad k > -\alpha$$

Solving $E[X] = m_1$ and $E[X^2] = m_2$ yields the following initial value for α:

$$\hat{\alpha} = \frac{m_1^2}{m_2 - m_1^2}$$

If $m_2 - m_1^2 < \epsilon$ (almost zero sample variance), then α is initialized as follows:

$$\hat{\alpha} = 1$$

After computing the estimate of α, the estimate of θ is computed as follows:

$$\hat{\theta} = \frac{m_1}{\hat{\alpha}}$$

Both the maximum likelihood method and the method of moments arrive at the same relationship between $\hat{\alpha}$ and $\hat{\theta}$.

GPD

The parameters are initialized by using the method of moments. Notice that for $\xi > 0$, the CDF of the generalized Pareto distribution (GPD) is:

$$\begin{aligned} F(x) &= 1 - \left(1 + \frac{\xi x}{\theta}\right)^{-1/\xi} \\ &= 1 - \left(\frac{\theta/\xi}{x + \theta/\xi}\right)^{1/\xi} \end{aligned}$$

This is equivalent to a Pareto distribution with scale parameter $\theta_1 = \theta/\xi$ and shape parameter $\alpha = 1/\xi$. Using this relationship, the parameter initialization method used for the PARETO distribution is used to get the following initial values for the parameters of the GPD distribution:

$$\hat{\theta} = \frac{m_1 m_2}{2(m_2 - m_1^2)}, \quad \hat{\xi} = \frac{m_2 - 2m_1^2}{2(m_2 - m_1^2)}$$

If $m_2 - m_1^2 < \epsilon$ (almost zero sample variance) or $m_2 - 2m_1^2 < \epsilon$, then the parameters are initialized as follows:

$$\hat{\theta} = \frac{m_1}{2}, \quad \hat{\xi} = \frac{1}{2}$$

IGAUSS The parameters are initialized by using the method of moments. The standard parameterization of the inverse Gaussian distribution (also known as the Wald distribution), in terms of the location parameter μ and shape parameter λ, is as follows (Klugman, Panjer, and Willmot 1998, p. 583):

$$f(x) = \sqrt{\frac{\lambda}{2\pi x^3}} \exp\left(\frac{-\lambda(x-\mu)^2}{2\mu^2 x}\right)$$
$$F(x) = \Phi\left(\left(\frac{x}{\mu}-1\right)\sqrt{\frac{\lambda}{x}}\right) + \Phi\left(-\left(\frac{x}{\mu}+1\right)\sqrt{\frac{\lambda}{x}}\right) \exp\left(\frac{2\lambda}{\mu}\right)$$

For this parameterization, it is known that the mean is $E[X] = \mu$ and the variance is $Var[X] = \mu^3/\lambda$, which yields the second raw moment as $E[X^2] = \mu^2(1 + \mu/\lambda)$ (computed by using $E[X^2] = Var[X] + (E[X])^2$).

The predefined IGAUSS distribution in PROC HPSEVERITY uses the following alternate parameterization to allow the distribution to have a scale parameter, θ:

$$f(x) = \sqrt{\frac{\alpha\theta}{2\pi x^3}} \exp\left(\frac{-\alpha(x-\theta)^2}{2x\theta}\right)$$
$$F(x) = \Phi\left(\left(\frac{x}{\theta}-1\right)\sqrt{\frac{\alpha\theta}{x}}\right) + \Phi\left(-\left(\frac{x}{\theta}+1\right)\sqrt{\frac{\alpha\theta}{x}}\right) \exp(2\alpha)$$

The parameters θ (scale) and α (shape) of this alternate form are related to the parameters μ and λ of the preceding form such that $\theta = \mu$ and $\alpha = \lambda/\mu$. Using this relationship, the first and second raw moments of the IGAUSS distribution are:

$$E[X] = \theta$$
$$E[X^2] = \theta^2\left(1 + \frac{1}{\alpha}\right)$$

Solving $E[X] = m_1$ and $E[X^2] = m_2$ yields the following initial values:

$$\hat{\theta} = m_1, \quad \hat{\alpha} = \frac{m_1^2}{m_2 - m_1^2}$$

If $m_2 - m_1^2 < \epsilon$ (almost zero sample variance), then the parameters are initialized as follows:

$$\hat{\theta} = m_1, \quad \hat{\alpha} = 1$$

LOGN The parameters are initialized by using the method of moments. The *k*th raw moment of the lognormal distribution is:

$$E[X^k] = \exp\left(k\mu + \frac{k^2\sigma^2}{2}\right)$$

Solving $E[X] = m_1$ and $E[X^2] = m_2$ yields the following initial values:

$$\hat{\mu} = 2\log(m1) - \frac{\log(m2)}{2}, \quad \hat{\sigma} = \sqrt{\log(m2) - 2\log(m1)}$$

PARETO The parameters are initialized by using the method of moments. The *k*th raw moment of the Pareto distribution is:

$$E[X^k] = \frac{\theta^k \Gamma(k+1)\Gamma(\alpha - k)}{\Gamma(\alpha)}, -1 < k < \alpha$$

Solving $E[X] = m_1$ and $E[X^2] = m_2$ yields the following initial values:

$$\hat{\theta} = \frac{m_1 m_2}{m_2 - 2m_1^2}, \quad \hat{\alpha} = \frac{2(m_2 - m_1^2)}{m_2 - 2m_1^2}$$

If $m_2 - m_1^2 < \epsilon$ (almost zero sample variance) or $m_2 - 2m_1^2 < \epsilon$, then the parameters are initialized as follows:

$$\hat{\theta} = m_1, \quad \hat{\alpha} = 2$$

TWEEDIE The parameter p is initialized by assuming that the sample is generated from a gamma distribution with shape parameter α and by computing $\hat{p} = \frac{\hat{\alpha}+2}{\hat{\alpha}+1}$. The initial value $\hat{\alpha}$ is obtained from using the method previously described for the GAMMA distribution. The parameter μ is the mean of the distribution. Hence, it is initialized to the sample mean as

$$\hat{\mu} = m_1$$

Variance of a Tweedie distribution is equal to $\phi\mu^p$. Thus, the sample variance is used to initialize the value of ϕ as

$$\hat{\phi} = \frac{m_2 - m_1^2}{\hat{\mu}^{\hat{p}}}$$

STWEEDIE STWEEDIE is a compound Poisson-gamma mixture distribution with mean $\mu = \lambda\theta\alpha$, where α is the shape parameter of the gamma random variables in the mixture and the parameter p is determined solely by α. First, the parameter p is initialized by assuming that the sample is generated from a gamma distribution with shape parameter α and by computing $\hat{p} = \frac{\hat{\alpha}+2}{\hat{\alpha}+1}$. The initial value $\hat{\alpha}$ is obtained from using the method previously described for the GAMMA distribution. As done for initializing the parameters of the TWEEDIE distribution, the sample mean and variance are used to compute the values $\hat{\mu}$ and $\hat{\phi}$ as

$$\hat{\mu} = m_1$$
$$\hat{\phi} = \frac{m_2 - m_1^2}{\hat{\mu}^{\hat{p}}}$$

Based on the relationship between the parameters of TWEEDIE and STWEEDIE distributions described in the section "Tweedie Distributions" on page 1232, values of θ and λ are initialized as

$$\hat{\theta} = \hat{\phi}(\hat{p} - 1)\hat{\mu}^{p-1}$$
$$\hat{\lambda} = \frac{\hat{\mu}}{\hat{\theta}\hat{\alpha}}$$

WEIBULL The parameters are initialized by using the percentile matching method. Let $q1$ and $q3$ denote the estimates of the 25th and 75th percentiles, respectively. Using the formula for the CDF of Weibull distribution, they can be written as

$$1 - \exp(-(q1/\theta)^{\tau}) = 0.25$$
$$1 - \exp(-(q3/\theta)^{\tau}) = 0.75$$

Simplifying and solving these two equations yields the following initial values:

$$\hat{\theta} = \exp\left(\frac{r\log(q1) - \log(q3)}{r-1}\right), \quad \hat{\tau} = \frac{\log(\log(4))}{\log(q3) - \log(\hat{\theta})}$$

where $r = \log(\log(4))/\log(\log(4/3))$. These initial values agree with those suggested in Klugman, Panjer, and Willmot (1998).

A summary of the initial values of all the parameters for all the predefined distributions is given in Table 23.3. The table also provides the names of the parameters to use in the INIT= option in the DIST statement if you want to provide a different initial value.

Table 23.3 Parameter Initialization for Predefined Distributions

Distribution	Parameter	Name for INIT option	Default Initial Value
BURR	θ	theta	$\sqrt{\frac{m_2 m_3}{2m_3 - 3m_1 m_2}}$
	α	alpha	$1 + \frac{m_3}{2m_3 - 3m_1 m_2}$
	γ	gamma	2
EXP	θ	theta	m_1
GAMMA	θ	theta	m_1/α
	α	alpha	$\frac{3-d+\sqrt{(d-3)^2+24d}}{12d}$
GPD	θ	theta	$m_1 m_2/(2(m_2 - m_1^2))$
	ξ	xi	$(m_2 - 2m_1^2)/(2(m_2 - m_1^2))$
IGAUSS	θ	theta	m_1
	α	alpha	$m_1^2/(m_2 - m_1^2)$
LOGN	μ	mu	$2\log(m1) - \log(m2)/2$
	σ	sigma	$\sqrt{\log(m2) - 2\log(m1)}$
PARETO	θ	theta	$m_1 m_2/(m_2 - 2m_1^2)$
	α	alpha	$2(m_2 - m_1^2)/(m_2 - 2m_1^2)$
TWEEDIE	μ	mu	m_1
	ϕ	phi	$(m_2 - m_1^2)/m_1^p$
	p	p	$(\alpha + 2)/(\alpha + 1)$ where $\alpha = \frac{3-d+\sqrt{(d-3)^2+24d}}{12d}$
STWEEDIE	θ	theta	$(m_2 - m_1^2)(p-1)/m_1$
	λ	lambda	$m_1^2/(\alpha(m_2 - m_1^2)(p-1))$
	p	p	$(\alpha + 2)/(\alpha + 1)$ where $\alpha = \frac{3-d+\sqrt{(d-3)^2+24d}}{12d}$
WEIBULL	θ	theta	$\exp\left(\frac{r\log(q1) - \log(q3)}{r-1}\right)$
	τ	tau	$\log(\log(4))/(\log(q3) - \log(\hat{\theta}))$

Notes:

- m_k denotes the kth raw moment
- $d = \log(m_1) - (\sum \log(y_i))/n$
- $q1$ and $q3$ denote the 25th and 75th percentiles, respectively
- $r = \log(\log(4))/\log(\log(4/3))$

Censoring and Truncation

One of the key features of PROC HPSEVERITY is that it enables you to specify whether the severity event's magnitude is observable and if it is observable, then whether the exact value of the magnitude is known. If an event is unobservable when the magnitude is in certain intervals, then it is referred to as a truncation effect. If the exact magnitude of the event is not known, but it is known to have a value in a certain interval, then it is referred to as a censoring effect.

PROC HPSEVERITY allows a severity event to be subject to any combination of the following four censoring and truncation effects:

- Left-truncation: An event is said to be left-truncated if it is observed only when $Y > T^l$, where Y denotes the random variable for the magnitude and T^l denotes a random variable for the truncation threshold. You can specify left-truncation using the LEFTTRUNCATED= option in the LOSS statement.
- Right-truncation: An event is said to be right-truncated if it is observed only when $Y \leq T^r$, where Y denotes the random variable for the magnitude and T^r denotes a random variable for the truncation threshold. You can specify right-truncation using the RIGHTTRUNCATED= option in the LOSS statement.
- Left-censoring: An event is said to be left-censored if it is known that the magnitude is $Y \leq C^l$, but the exact value of Y is not known. C^l is a random variable for the censoring limit. You can specify left-censoring using the LEFTCENSORED= option in the LOSS statement.
- Right-censoring: An event is said to be right-censored if it is known that the magnitude is $Y > C^r$, but the exact value of Y is not known. C^r is a random variable for the censoring limit. You can specify right-censoring using the RIGHTCENSORED= option in the LOSS statement.

For each effect, you can specify a different threshold or limit for each observation or specify a single threshold or limit that applies to all the observations.

If all the four types of effects are present on an event, then the following relationship holds: $T^l < C^r \leq C^l \leq T^r$. PROC HPSEVERITY checks these relationships and write a warning to the SAS log if any is violated.

If you specify the response variable in the LOSS statement, then PROC HPSEVERITY also checks whether each observation satisfies the definitions of the specified censoring and truncation effects. If you specify left-truncation, then PROC HPSEVERITY ignores observations where $Y \leq T^l$, because such observations are not observable by definition. Similarly, if you specify right-truncation, then PROC HPSEVERITY ignores observations where $Y > T^r$. If you specify left-censoring, then PROC HPSEVERITY treats an observation with $Y > C^l$ as uncensored and ignores the value of C^l. The observations with $Y \leq C^l$ are considered as left-censored, and the value of Y is ignored. If you specify right-censoring, then PROC HPSEVERITY treats an observation with $Y \leq C^r$ as uncensored and ignores the value of C^r. The observations with $Y > C^r$ are considered as right-censored, and the value of Y is ignored. If you specify both left-censoring and right-censoring, it is referred to as interval-censoring. If $C^r < C^l$ is satisfied for an observation, then it is considered as interval-censored and the value of the response variable is ignored. If $C^r = C^l$ for an observation, then PROC HPSEVERITY assumes that observation to be uncensored. If all the observations in a data set are censored in some form, then the specification of the response variable in the LOSS statement is optional, because the actual value of the response variable is not required for the purposes of estimating a model.

Specification of censoring and truncation affects the likelihood of the data (see the section "Likelihood Function" on page 1242) and how the empirical distribution function (EDF) is estimated (see the section "Empirical Distribution Function Estimation Methods" on page 1260).

Probability of Observability

For left-truncated data, PROC HPSEVERITY also enables you to provide additional information in the form of *probability of observability* by using the PROBOBSERVED= option. It is defined as the probability that the underlying severity event gets observed (and recorded) for the specified left-truncation threshold value. For example, if you specify a value of 0.75, then for every 75 observations recorded above a specified threshold, 25 more events have happened with a severity value less than or equal to the specified threshold. Although the exact severity value of those 25 events is not known, PROC HPSEVERITY can use the information about the number of those events.

In particular, for each left-truncated observation, PROC HPSEVERITY assumes a presence of $(1-p)/p$ additional observations with $y_i = t_i$. These additional observations are then used for computing the likelihood (see the section "Probability of Observability and Likelihood" on page 1243) and an unconditional estimate of the empirical distribution function (see the section "EDF Estimates and Truncation" on page 1265).

Truncation and Conditional CDF Estimates

If you specify left-truncation without the probability of observability or if you specify right-truncation, then the EDF estimates that are computed by all methods except the STANDARD method are conditional on the truncation information. See the section "EDF Estimates and Truncation" on page 1265 for more information. In such cases, PROC HPSEVERITY uses conditional estimates of the CDF for computational or visual comparison to the EDF estimates.

Let $t^l_{\min} = \min_i\{t^l_i\}$ be the smallest value of the left-truncation threshold (t^l_i is the left-truncation threshold for observation i) and $t^r_{\max} = \max_i\{t^r_i\}$ be the largest value of the right-truncation threshold (t^r_i is the right-truncation threshold for observation i). If $\hat{F}(y)$ denotes the unconditional estimate of the CDF at y, then the conditional estimate $\hat{F}^c(y)$ is computed as follows:

- If you do not specify the probability of observability, then the EDF estimates are conditional on the left-truncation information. If an observation is both left-truncated and right-truncated, then

$$\hat{F}^c(y) = \frac{\hat{F}(y) - \hat{F}(t^l_{\min})}{\hat{F}(t^r_{\max}) - \hat{F}(t^l_{\min})}$$

  If an observation is left-truncated but not right-truncated, then

$$\hat{F}^c(y) = \frac{\hat{F}(y) - \hat{F}(t^l_{\min})}{1 - \hat{F}(t^l_{\min})}$$

  If an observation is right-truncated but not left-truncated, then

$$\hat{F}^c(y) = \frac{\hat{F}(y)}{\hat{F}(t^r_{\max})}$$

- If you specify the probability of observability, then EDF estimates are not conditional on the left-truncation information. If an observation is not right-truncated, then the conditional estimate is the

same as the unconditional estimate. If an observation is right-truncated, then the conditional estimate is computed as

$$\hat{F}^c(y) = \frac{\hat{F}(y)}{\hat{F}(t^r_{\max})}$$

If you specify regression effects, then $\hat{F}(y)$, $\hat{F}(t^l_{\min})$, and $\hat{F}(t^r_{\max})$ are all computed from a mixture distribution, as described in the section "CDF and PDF Estimates with Regression Effects" on page 1249.

Parameter Estimation Method

If you do not specify a custom objective function by specifying programming statements and the OBJECTIVE= option in the PROC HPSEVERITY statement, then PROC HPSEVERITY uses the maximum likelihood (ML) method to estimate the parameters of each model. A nonlinear optimization process is used to maximize the log of the likelihood function. If you specify a custom objective function, then PROC HPSEVERITY uses a nonlinear optimization algorithm to estimate the parameters of each model that minimize the value of your specified objective function. For more information, see the section "Custom Objective Functions" on page 1298.

Likelihood Function

Let $f_\Theta(x)$ and $F_\Theta(x)$ denote the PDF and CDF, respectively, evaluated at x for a set of parameter values Θ. Let Y denote the random response variable, and let y denote its value recorded in an observation in the input data set. Let T^l and T^r denote the random variables for the left-truncation and right-truncation threshold, respectively, and let t^l and t^r denote their values for an observation, respectively. If there is no left-truncation, then $t^l = \tau^l$, where τ^l is the smallest value in the support of the distribution; so $F(t^l) = 0$. If there is no right-truncation, then $t^r = \tau_h$, where τ_h is the largest value in the support of the distribution; so $F(t^r) = 1$. Let C^l and C^r denote the random variables for the left-censoring and right-censoring limit, respectively, and let c^l and c^r denote their values for an observation, respectively. If there is no left-censoring, then $c^l = \tau_h$; so $F(c^l) = 1$. If there is no right-censoring, then $c^r = \tau^l$; so $F(c^r) = 0$.

The set of input observations can be categorized into the following four subsets within each BY group:

- E is the set of uncensored and untruncated observations. The likelihood of an observation in E is

$$l_E = \Pr(Y = y) = f_\Theta(y)$$

- E_t is the set of uncensored observations that are truncated. The likelihood of an observation in E_t is

$$l_{E_t} = \Pr(Y = y | t^l < Y \le t^r) = \frac{f_\Theta(y)}{F_\Theta(t^r) - F_\Theta(t^l)}$$

- C is the set of censored observations that are not truncated. The likelihood of an observation C is

$$l_C = \Pr(c^r < Y \le c^l) = F_\Theta(c^l) - F_\Theta(c^r)$$

- C_t is the set of censored observations that are truncated. The likelihood of an observation C_t is

$$l_{C_t} = \Pr(c^r < Y \le c^l | t^l < Y \le t^r) = \frac{F_\Theta(c^l) - F_\Theta(c^r)}{F_\Theta(t^r) - F_\Theta(t^l)}$$

Note that $(E \cup E_t) \cap (C \cup C_t) = \emptyset$. Also, the sets E_t and C_t are empty when you do not specify truncation, and the sets C and C_t are empty when you do not specify censoring.

Given this, the likelihood of the data L is as follows:

$$L = \prod_{E} f_\Theta(y) \prod_{E_t} \frac{f_\Theta(y)}{F_\Theta(t^r) - F_\Theta(t^l)} \prod_{C} F_\Theta(c^l) - F_\Theta(c^r) \prod_{C_t} \frac{F_\Theta(c^l) - F_\Theta(c^r)}{F_\Theta(t^r) - F_\Theta(t^l)}$$

The maximum likelihood procedure used by PROC HPSEVERITY finds an optimal set of parameter values $\hat{\Theta}$ that maximizes $\log(L)$ subject to the boundary constraints on parameter values. For a distribution *dist*, you can specify such boundary constraints by using the *dist*_LOWERBOUNDS and *dist*_UPPERBOUNDS subroutines. For more information, see the section "Defining a Severity Distribution Model with the FCMP Procedure" on page 1273. Some aspects of the optimization process can be controlled by using the NLOPTIONS statement.

Probability of Observability and Likelihood

If you specify the probability of observability for the left-truncation, then PROC HPSEVERITY uses a modified likelihood function for each truncated observation. If the probability of observability is $p \in (0.0, 1.0]$, then for each left-truncated observation with truncation threshold t^l, there exist $(1 - p)/p$ observations with a response variable value less than or equal to t^l. Each such observation has a probability of $\Pr(Y \leq t^l) = F_\Theta(t^l)$. The right-truncation and censoring information does not apply to these added observations. Thus, following the notation of the section "Likelihood Function" on page 1242, the likelihood of the data is as follows:

$$\begin{aligned} L = & \prod_{E} f_\Theta(y) \prod_{E_t, t^l = \tau^l} \frac{f_\Theta(y)}{F_\Theta(t^r)} \prod_{E_t, t^l > \tau^l} \frac{f_\Theta(y)}{F_\Theta(t^r)} F_\Theta(t^l)^{\frac{1-p}{p}} \\ & \prod_{C} F_\Theta(c^l) - F_\Theta(c^r) \prod_{C_t, t^l = \tau^l} \frac{F_\Theta(c^l) - F_\Theta(c^r)}{F_\Theta(t^r)} \prod_{C_t, t^l > \tau^l} \frac{F_\Theta(c^l) - F_\Theta(c^r)}{F_\Theta(t^r)} F_\Theta(t^l)^{\frac{1-p}{p}} \end{aligned}$$

Note that the likelihood of the observations that are not left-truncated (observations in sets E and C, and observations in sets E_t and C_t for which $t^l = \tau^l$) is not affected.

If you specify a custom objective function, then PROC HPSEVERITY accounts for the probability of observability only while computing the empirical distribution function estimate. The parameter estimates are affected only by your custom objective function.

Estimating Covariance and Standard Errors

PROC HPSEVERITY computes an estimate of the covariance matrix of the parameters by using the asymptotic theory of the maximum likelihood estimators (MLE). If N denotes the number of observations used for estimating a parameter vector $\boldsymbol{\theta}$, then the theory states that as $N \to \infty$, the distribution of $\hat{\boldsymbol{\theta}}$, the estimate of $\boldsymbol{\theta}$, converges to a normal distribution with mean $\boldsymbol{\theta}$ and covariance $\hat{\mathbf{C}}$ such that $\mathbf{I}(\boldsymbol{\theta}) \cdot \hat{\mathbf{C}} \to 1$, where $\mathbf{I}(\boldsymbol{\theta}) = -E\left[\nabla^2 \log(L(\boldsymbol{\theta}))\right]$ is the information matrix for the likelihood of the data, $L(\boldsymbol{\theta})$. The covariance estimate is obtained by using the inverse of the information matrix.

In particular, if $\mathbf{G} = \nabla^2(-\log(L(\boldsymbol{\theta})))$ denotes the Hessian matrix of the negative of log likelihood, then the covariance estimate is computed as

$$\hat{\mathbf{C}} = \frac{N}{d}\mathbf{G}^{-1}$$

where d is a denominator that is determined by the VARDEF= option. If VARDEF=N, then $d = N$, which yields the asymptotic covariance estimate. If VARDEF=DF, then $d = N - k$, where k is number of parameters (the model's degrees of freedom). The VARDEF=DF option is the default, because it attempts to correct the potential bias introduced by the finite sample.

The standard error s_i of the parameter θ_i is computed as the square root of the ith diagonal element of the estimated covariance matrix; that is, $s_i = \sqrt{\hat{C}_{ii}}$.

If you specify a custom objective function, then the covariance matrix of the parameters is still computed by inverting the information matrix, except that the Hessian matrix $\mathbf{G}$ is computed as $\mathbf{G} = \nabla^2 \log(U(\boldsymbol{\theta}))$, where U denotes your custom objective function that is minimized by the optimizer.

Covariance and standard error estimates might not be available if the Hessian matrix is found to be singular at the end of the optimization process. This can especially happen if the optimization process stops without converging.

Parameter Initialization

PROC HPSEVERITY enables you to initialize parameters of a model in different ways. A model can have two kinds of parameters: distribution parameters and regression parameters.

The distribution parameters can be initialized by using one of the following three methods:

INIT= option	You can use the INIT= option in the DIST statement.
INEST= or INSTORE= option	You can use either the INEST= data set or the INSTORE= item store, but not both.
PARMINIT subroutine	You can define a *dist*_PARMINIT subroutine in the distribution model. For more information, see the section "Defining a Severity Distribution Model with the FCMP Procedure" on page 1273.

Note that only one of the initialization methods is used. You cannot combine them. They are used in the following order:

- The method that uses the INIT= option takes the highest precedence. If you use the INIT= option to provide an initial value for at least one parameter, then other initialization methods (INEST=, INSTORE=, or PARMINIT) are not used. If you specify initial values for some but not all the parameters by using the INIT= option, then the uninitialized parameters are initialized to the default value of 0.001.

  If you use this option and if you specify the regression effects, then the value of the first distribution parameter must be related to the initial value for the *base* value of the scale or log-transformed scale parameter. For more information, see the section "Estimating Regression Effects" on page 1245.

- The method that uses the INEST= data set or INSTORE= item store takes second precedence. If the INEST= data set or INSTORE= item store contains a nonmissing value for even one distribution parameter, then the PARMINIT method is not used and any uninitialized parameters are initialized to the default value of 0.001.
- If none of the distribution parameters are initialized by using the INIT= option, the INEST= data set, or the INSTORE= item store, but the distribution model defines a PARMINIT subroutine, then PROC HPSEVERITY invokes that subroutine with appropriate inputs to initialize the parameters. If the PARMINIT subroutine returns missing values for some parameters, then those parameters are initialized to the default value of 0.001.
- If none of the initialization methods are used, each distribution parameter is initialized to the default value of 0.001.

For more information about regression models and initialization of regression parameters, see the section "Estimating Regression Effects" on page 1245.

PARMINIT-Based Parameter Initialization Method and Distributed Data

If you specify a distributed mode of execution for the procedure, then the input data are distributed across the computational nodes. For more information about the distributed computing model, see the section "Distributed and Multithreaded Computation" on page 1271. If the PARMINIT subroutine is used for initializing the distribution parameters, then PROC HPSEVERITY invokes that subroutine on each computational node with the data that are local to that node. The EDF estimates that are supplied to the PARMINIT subroutine are also computed using the local data. The initial values of the parameters that are supplied to the optimizer are the average of the local estimates that are computed on each node. This approach works well if the data are distributed randomly across nodes. If you distribute the data on the appliance before you run the procedure (alongside-the-database model), then you should try to make the distribution as random as possible in order to increase the chances of computing good initial values. If you specify a data set that is not distributed before you run the procedure, then PROC HPSEVERITY distributes the data for you by sending the first observation to the first node, the second observation to the second node, and so on. If the order of observations is random, then this method ensures random distribution of data across the computational nodes.

Estimating Regression Effects

The HPSEVERITY procedure enables you to estimate the influence of regression (exogenous) effects while fitting a distribution if the distribution has a scale parameter or a log-transformed scale parameter.

Let x_j, $j = 1, \ldots, k$, denote the k regression effects. Let β_j denote the regression parameter that corresponds to the effect x_j. If you do not specify regression effects, then the model for the response variable Y is of the form

$$Y \sim \mathcal{F}(\Theta)$$

where $\mathcal{F}$ is the distribution of Y with parameters Θ. This model is usually referred to as the error model. The regression effects are modeled by extending the error model to the following form:

$$Y \sim \exp\left(\sum_{j=1}^{k} \beta_j x_j\right) \cdot \mathcal{F}(\Theta)$$

Under this model, the distribution of Y is valid and belongs to the same parametric family as $\mathcal{F}$ if and only if $\mathcal{F}$ has a scale parameter. Let θ denote the scale parameter and Ω denote the set of nonscale distribution parameters of $\mathcal{F}$. Then the model can be rewritten as

$$Y \sim \mathcal{F}(\theta, \Omega)$$

such that θ is modeled by the regression effects as

$$\theta = \theta_0 \cdot \exp(\sum_{j=1}^{k} \beta_j x_j)$$

where θ_0 is the *base* value of the scale parameter. Thus, the scale regression model consists of the following parameters: θ_0, Ω, and $\beta_j (j = 1, \ldots, k)$.

Given this form of the model, distributions without a scale parameter cannot be considered when regression effects are to be modeled. If a distribution does not have a direct scale parameter, then PROC HPSEVERITY accepts it only if it has a log-transformed scale parameter—that is, if it has a parameter $p = \log(\theta)$.

Offset Variable

You can specify that an offset variable be included in the scale regression model by specifying it in the OFFSET= option of the SCALEMODEL statement. The offset variable is a regressor whose regression coefficient is known to be 1. If x_o denotes the offset variable, then the scale regression model becomes

$$\theta = \theta_0 \cdot \exp(x_o + \sum_{j=1}^{k} \beta_j x_j)$$

The regression coefficient of the offset variable is fixed at 1 and not estimated, so it is not reported in the ParameterEstimates ODS table. However, if you specify the OUTEST= data set, then the regression coefficient is added as a variable to that data set. The value of the offset variable in OUTEST= data set is equal to 1 for the estimates row (_TYPE_='EST') and is equal to a special missing value (.F) for the standard error (_TYPE_='STDERR') and covariance (_TYPE_='COV') rows.

An offset variable is useful to model the scale parameter per unit of some measure of exposure. For example, in the automobile insurance context, measure of exposure can be the number of car-years insured or the total number of miles driven by a fleet of cars at a rental car company. For worker's compensation insurance, if you want to model the expected loss per enterprise, then you can use the number of employees or total employee salary as the measure of exposure. For epidemiological data, measure of exposure can be the number of people who are exposed to a certain pathogen when you are modeling the loss associated with an epidemic. In general, if e denotes the value of the exposure measure and if you specify $x_o = \log(e)$ as the offset variable, then you are modeling the influence of other regression effects (x_j) on the size of the scale of the distribution *per unit of exposure*.

Another use for an offset variable is when you have a priori knowledge of the influence of some exogenous variables that cannot be included in the SCALEMODEL statement. You can model the combined influence of such variables as an offset variable in order to correct for the omitted variable bias.

Parameter Initialization for Regression Models

The regression parameters are initialized either by using the values that you specify or by the default method.

- If you provide initial values for the regression parameters, then you must provide valid, nonmissing initial values for θ_0 and β_j parameters for all j.

  You can specify the initial value for θ_0 by using either the INEST= data set, the INSTORE= item store, or the INIT= option in the DIST statement. If the distribution has a direct scale parameter (no transformation), then the initial value for the first parameter of the distribution is used as an initial value for θ_0. If the distribution has a log-transformed scale parameter, then the initial value for the first parameter of the distribution is used as an initial value for $\log(\theta_0)$.

  You can use only the INEST= data set or the INSTORE= item store, but not both, to specify the initial values for β_j. The requirements for each option are as follows:

  – If you use the INEST= data set, then it must contain nonmissing initial values for all the regressors that you specify in the SCALEMODEL statement. The only missing value that is allowed is the special missing value .R, which indicates that the regressor is linearly dependent on other regressors. If you specify .R for a regressor for one distribution in a BY group, you must specify it the same way for all the distributions in that BY group.

    Note that you cannot specify INEST= data set if the regression model contains effects that have CLASS variables or interaction effects.

  – The parameter estimates in the INSTORE= item store are used to initialize the parameters of a model if the item store contains a model specification that matches the model specification in the current PROC HPSEVERITY step according to the following rules:

    * The distribution name and the number and names of the distribution parameters must match.
    * The model in the item store must include a scale regression model whose regression parameters match as follows:
      · If the regression model in the item store does not contain any redundant parameters, then at least one regression parameter must match. Initial values of the parameters that match are set equal to the estimates that are read from the item store, and initial values of the other regression parameters are set equal to the default value of 0.001.
      · If the regression model in the item store contains any redundant parameters, then all the regression parameters must match, and the initial values of all parameters are set equal to the estimates that are read from the item store.

    Note that a regression parameter is defined by the variables that form the underlying regression effect and by the levels of the CLASS variables if the effect contains any CLASS variables.

- If you do not specify valid initial values for θ_0 or β_j parameters for all j, then PROC HPSEVERITY initializes those parameters by using the following method:

  Let a random variable Y be distributed as $\mathcal{F}(\theta, \Omega)$, where θ is the scale parameter. By the definition of the scale parameter, a random variable $W = Y/\theta$ is distributed as $\mathcal{G}(\Omega)$ such that $\mathcal{G}(\Omega) = \mathcal{F}(1, \Omega)$. Given a random error term e that is generated from a distribution $\mathcal{G}(\Omega)$, a value y from the distribution of Y can be generated as

  $$y = \theta \cdot e$$

Taking the logarithm of both sides and using the relationship of θ with the regression effects yields:

$$\log(y) = \log(\theta_0) + \sum_{j=1}^{k} \beta_j x_j + \log(e)$$

PROC HPSEVERITY makes use of the preceding relationship to initialize parameters of a regression model with distribution *dist* as follows:

1. The following linear regression problem is solved to obtain initial estimates of β_0 and β_j:

   $$\log(y) = \beta_0 + \sum_{j=1}^{k} \beta_j x_j$$

   The estimates of $\beta_j (j = 1, \ldots, k)$ in the solution of this regression problem are used to initialize the respective regression parameters of the model. The estimate of β_0 is later used to initialize the value of θ_0.

   The results of this regression are also used to detect whether any regression parameters are linearly dependent on the other regression parameters. If any such parameters are found, then a warning is written to the SAS log and the corresponding parameter is eliminated from further analysis. The estimates for linearly dependent regression parameters are denoted by a special missing value of .R in the OUTEST= data set and in any displayed output.

2. Let s_0 denote the initial value of the scale parameter.

   If the distribution model of *dist* does not contain the *dist*_PARMINIT subroutine, then s_0 and all the nonscale distribution parameters are initialized to the default value of 0.001.

   However, it is strongly recommended that each distribution's model contain the *dist*_PARMINIT subroutine. For more information, see the section "Defining a Severity Distribution Model with the FCMP Procedure" on page 1273. If that subroutine is defined, then s_0 is initialized as follows:

   Each input value y_i of the response variable is transformed to its scale-normalized version w_i as

   $$w_i = \frac{y_i}{\exp(\beta_0 + \sum_{j=1}^{k} \beta_j x_{ij})}$$

   where x_{ij} denotes the value of *j*th regression effect in the *i*th input observation. These w_i values are used to compute the input arguments for the *dist*_PARMINIT subroutine. The values that are computed by the subroutine for nonscale parameters are used as their respective initial values. If the distribution has an untransformed scale parameter, then s_0 is set to the value of the scale parameter that is computed by the subroutine. If the distribution has a log-transformed scale parameter P, then s_0 is computed as $s_0 = \exp(l_0)$, where l_0 is the value of P computed by the subroutine.

3. The value of θ_0 is initialized as

   $$\theta_0 = s_0 \cdot \exp(\beta_0)$$

Reporting Estimates of Regression Parameters

When you request estimates to be written to the output (either ODS displayed output or in the OUTEST= data set), the estimate of the base value of the first distribution parameter is reported. If the first parameter is the log-transformed scale parameter, then the estimate of $\log(\theta_0)$ is reported; otherwise, the estimate of θ_0 is reported. The transform of the first parameter of a distribution *dist* is controlled by the *dist*_SCALETRANSFORM function that is defined for it.

CDF and PDF Estimates with Regression Effects

When regression effects are estimated, the estimate of the scale parameter depends on the values of the regressors and the estimates of the regression parameters. This dependency results in a potentially different distribution for each observation. To make estimates of the cumulative distribution function (CDF) and probability density function (PDF) comparable across distributions and comparable to the empirical distribution function (EDF), PROC HPSEVERITY computes and reports the CDF and PDF estimates from a representative distribution. The *representative distribution* is a mixture of a certain number of distributions, where each distribution differs only in the value of the scale parameter. You can specify the number of distributions in the mixture and how their scale values are chosen by using the DFMIXTURE= option in the SCALEMODEL statement.

Let N denote the number of observations that are used for estimation, K denote the number of components in the mixture distribution, s_k denote the scale parameter of the kth mixture component, and d_k denote the weight associated with kth mixture component.

Let $f(y; s_k, \hat{\Omega})$ and $F(y; s_k, \hat{\Omega})$ denote the PDF and CDF, respectively, of the kth component distribution, where $\hat{\Omega}$ denotes the set of estimates of all parameters of the distribution other than the scale parameter. Then, the PDF and CDF estimates, $f^*(y)$ and $F^*(y)$, respectively, of the mixture distribution at y are computed as

$$f^*(y) = \frac{1}{D}\sum_{k=1}^{K} d_k f(y; s_k, \hat{\Omega})$$

$$F^*(y) = \frac{1}{D}\sum_{k=1}^{K} d_k F(y; s_k, \hat{\Omega})$$

where D is the normalization factor ($D = \sum_{k=1}^{K} d_k$).

PROC HPSEVERITY uses the $F^*(y)$ values to compute the EDF-based statistics of fit and to create the OUTCDF= data set and the CDF plots. The PDF estimates that it plots in the PDF plots are the $f^*(y)$ values.

The scale values s_k for the K mixture components are derived from the set $\{\hat{\lambda}_i\}$ ($i = 1, \ldots, N$) of N linear predictor values, where $\hat{\lambda}_i$ denotes the estimate of the linear predictor due to observation i. It is computed as

$$\hat{\lambda}_i = \log(\hat{\theta}_0) + \sum_{j=1}^{k} \hat{\beta}_j x_{ij}$$

where $\hat{\theta}_0$ is an estimate of the base value of the scale parameter, $\hat{\beta}_j$ are the estimates of regression coefficients, and x_{ij} is the value of jth regression effect in observation i.

Let w_i denote the weight of observation i. If you specify the WEIGHT statement, then the weight is equal to the value of the specified weight variable for the corresponding observation in the DATA= data set; otherwise, the weight is set to 1.

You can specify one of the following *method-names* in the DFMIXTURE= option in the SCALEMODEL statement to specify the method of choosing K and the corresponding s_k and d_k values:

FULL
: In this method, there are as many mixture components as the number of observations that are used for estimation. In other words, $K = N$, $s_k = \hat{\theta}_k$, and $d_k = w_k$ $(k = 1, \ldots, N)$. This is the slowest method, because it requires $O(N)$ computations to compute the mixture CDF $F^*(y_i)$ or the mixture PDF $f^*(y_i)$ of one observation. For N observations, the computational complexity in terms of number of CDF or PDF evaluations is $O(N^2)$. Even for moderately large values of N, the time that is taken to compute the mixture CDF and PDF can significantly exceed the time that is taken to estimate the model parameters. So it is recommended that you use the FULL method only for small data sets.

MEAN
: In this method, the mixture contains only one distribution, whose scale value is determined by the mean of the linear predictor values that are implied by all the observations. In other words, s_1 is computed as

$$s_1 = \exp\left(\frac{1}{N}\sum_{i=1}^{N}\hat{\lambda}_i\right)$$

The component's weight d_1 is set to 1.

This method is the fastest because it requires only one CDF or PDF evaluation per observation. The computational complexity is $O(N)$ for N observations.

If you do not specify the DFMIXTURE= option in the SCALEMODEL statement, then this is the default method.

QUANTILE
: In this method, a certain number of quantiles are chosen from the set of all linear predictor values. If you specify a value of q for the K= option when specifying this method, then $K = q - 1$ and s_k $(k = 1, \ldots, K)$ is computed as $s_k = \exp(\hat{\lambda}_k)$, where $\hat{\lambda}_k$ is the kth q-quantile from the set $\{\hat{\lambda}_i\}$ $(i = 1, \ldots, N)$. The weight of each of the components (d_k) is assumed to be 1 for this method.

The default value of q is 2, which implies a one-point mixture that has a distribution whose scale value is equal to the median scale value.

For this method, PROC HPSEVERITY needs to sort the N linear predictor values in the set $\{\hat{\lambda}_i\}$; the sorting requires $O(N \log(N))$ computations. Then, computing the mixture estimate of one observation requires $(q - 1)$ CDF or PDF evaluations. Hence, the computational complexity of this method is $O(qN) + O(N \log(N))$ for computing a mixture CDF or PDF of N observations. For $q << N$, the QUANTILE method is significantly faster than the FULL method.

RANDOM
: In this method, a uniform random sample of observations is chosen, and the mixture contains the distributions that are implied by those observations. If you specify a value of r for the K= option when specifying this method, then the size of the sample is r. Hence, $K = r$. If l_j denotes the index of jth observation in the sample $(j = 1, \ldots, r)$, such that $1 \leq l_j \leq N$, then the scale of kth component distribution in the mixture is $s_k = \exp(\hat{\lambda}_{l_k})$. The weight of each of the components (d_k) is assumed to be 1 for this method.

You can also specify the seed to be used for generating the random sample by using the SEED= option for this method. The same sample of observations is used for all models.

Computing a mixture estimate of one observation requires r CDF or PDF evaluations. Hence, the computational complexity of this method is $O(rN)$ for computing a mixture CDF or PDF of N observations. For $r << N$, the RANDOM method is significantly faster than the FULL method.

Levelization of Classification Variables

A classification variable enters the statistical analysis or model not through its values but through its levels. The process of associating values of a variable with levels is called *levelization*.

During the process of levelization, observations that share the same value are assigned to the same level. The manner in which values are grouped can be affected by the inclusion of formats. You can determine the sort order of the levels by specifying the ORDER= option in the CLASS statement. You can also control the sort order separately for each variable in the CLASS statement.

Consider the data on nine observations in Table 23.4. The variable A is integer-valued, and the variable X is a continuous variable that has a missing value for the fourth observation. The fourth and fifth columns of Table 23.4 apply two different formats to the variable X.

Table 23.4 Example Data for Levelization

Obs	A	X	FORMAT X 3.0	FORMAT X 3.1
1	2	1.09	1	1.1
2	2	1.13	1	1.1
3	2	1.27	1	1.3
4	3	.	.	.
5	3	2.26	2	2.3
6	3	2.48	2	2.5
7	4	3.34	3	3.3
8	4	3.34	3	3.3
9	4	3.14	3	3.1

By default, levelization of the variables groups the observations by the formatted value of the variable, except for numerical variables for which no explicit format is provided. Those numerical variables are sorted by their internal value. The levelization of the four columns in Table 23.4 leads to the level assignment in Table 23.5.

Table 23.5 Values and Levels

	A		X		FORMAT X 3.0		FORMAT X 3.1	
Obs	Value	Level	Value	Level	Value	Level	Value	Level
1	2	1	1.09	1	1	1	1.1	1
2	2	1	1.13	2	1	1	1.1	1
3	2	1	1.27	3	1	1	1.3	2
4	3	2	.	.	.	.	.	.
5	3	2	2.26	4	2	2	2.3	3
6	3	2	2.48	5	2	2	2.5	4
7	4	3	3.34	7	3	3	3.3	6
8	4	3	3.34	7	3	3	3.3	6
9	4	3	3.14	6	3	3	3.1	5

You can specify the sort order for the levels of CLASS variables in the ORDER= option in the CLASS statement.

When ORDER=FORMATTED (which is the default) is in effect for numeric variables for which you have supplied no explicit format, the levels are ordered by their internal values. To order numeric class levels that have no explicit format by their BEST12. formatted values, you can specify the BEST12. format explicitly for the CLASS variables.

Table 23.6 shows how values of the ORDER= option are interpreted.

Table 23.6 Interpretation of Values of ORDER= Option

Value of ORDER=	Levels Sorted By
DATA	Order of appearance in the input data set
FORMATTED	External formatted value, except for numeric variables that have no explicit format, which are sorted by their unformatted (internal) value
FREQ	Descending frequency count (levels that have the most observations come first in the order)
INTERNAL	Unformatted value
FREQDATA	Order of descending frequency count, and within counts by order of appearance in the input data set when counts are tied
FREQFORMATTED	Order of descending frequency count, and within counts by formatted value when counts are tied
FREQINTERNAL	Order of descending frequency count, and within counts by unformatted (internal) value when counts are tied

For FORMATTED, FREQFORMATTED, FREQINTERNAL, and INTERNAL values, the sort order is machine-dependent. For more information about sort order, see the chapter about the SORT procedure in the *Base SAS Procedures Guide* and the discussion of BY-group processing in *SAS Language Reference: Concepts*.

When you specify the MISSING option in the CLASS statement, the missing values ('.' for a numeric variable and blanks for a character variable) are included in the levelization and are assigned a level. Table 23.7 displays the results of levelizing the values in Table 23.4 when the MISSING option is in effect.

Table 23.7 Values and Levels with the MISSING Option

	A		X		FORMAT x 3.0		FORMAT x 3.1	
Obs	Value	Level	Value	Level	Value	Level	Value	Level
1	2	1	1.09	2	1	2	1.1	2
2	2	1	1.13	3	1	2	1.1	2
3	2	1	1.27	4	1	2	1.3	3
4	3	2	.	1	.	1	.	1
5	3	2	2.26	5	2	3	2.3	4

Table 23.7 *continued*

Obs	A Value	A Level	X Value	X Level	format x 3.0 Value	format x 3.0 Level	format x 3.1 Value	format x 3.1 Level
6	3	2	2.48	6	2	3	2.5	5
7	4	3	3.34	8	3	4	3.3	7
8	4	3	3.34	8	3	4	3.3	7
9	4	3	3.14	7	3	4	3.1	6

When you do not specify the MISSING option, it is important to understand the implications of missing values for your statistical analysis. When PROC HPSEVERITY levelizes the CLASS variables, any observations for which a CLASS variable has a missing value are excluded from the analysis. This is true regardless of whether the variable is used to form the statistical model. For example, consider the case in which some observations contain missing values for variable A but the records for these observations are otherwise complete with respect to all other variables in the model. The analysis results that come from the following statements do not include any observations for which variable A contains missing values, even though A is not specified in the SCALEMODEL statement:

```
class A B;
scalemodel B x B*x;
```

You can request PROC HPSEVERITY to print the "Descriptive Statistics" table, which shows the number of observations that are read from the data set and the number of observations that are used in the analysis. Pay careful attention to this table—especially when your data set contains missing values—to ensure that no observations are unintentionally excluded from the analysis.

Specification and Parameterization of Model Effects

PROC HPSEVERITY supports formation of regression effects in the SCALEMODEL statement. A *regression effect* is formed from one or more regressor variables according to effect construction rules (*parameterization*). Each regression effect forms one element of $\mathbf{X}$ in the linear model structure $\mathbf{X}\boldsymbol{\beta}$ that affects the scale parameter. The SCALEMODEL statement in conjunction with the CLASS statement supports a rich set of effects. In order to correctly interpret the results, you need to understand the specification and parameterization of effects that are discussed in this section.

Effects are specified by a special notation that uses variable names and operators. There are two types of regressor variables: classification (or CLASS) variables and continuous variables. *Classification variables* can be either numeric or character and are specified in a CLASS statement. For more information, see the section "Levelization of Classification Variables" on page 1251. A regressor variable that is not declared in the CLASS statement is assumed to be *continuous*.

Two primary operators (crossing and nesting) are used for combining the variables, and several additional operators are used to simplify effect specification. Operators are discussed in the section "Effect Operators" on page 1254.

If you specify the CLASS statement, then PROC HPSEVERITY supports a general linear model (GLM) parameterization and a reference parameterization for the classification variables. The GLM parameterization is the default. For more information, see the sections "GLM Parameterization of Classification Variables and Effects" on page 1256 and "Reference Parameterization" on page 1259.

Effect Operators

Table 23.8 summarizes the operators that are available for selecting and constructing effects. These operators are discussed in the following sections.

Table 23.8 Available Effect Operators

Operator	Example	Description
Interaction	A*B	Crosses the levels of the effects
Nesting	A(B)	Nests A levels within B levels
Bar operator	A | B | C	Specifies all interactions
At sign operator	A | B | C@2	Reduces interactions in bar effects
Dash operator	A1-A10	Specifies sequentially numbered variables
Colon operator	A:	Specifies variables that have a common prefix
Double dash operator	A- -C	Specifies sequential variables in data set order

Bar and At Sign Operators

You can shorten the specification of a large factorial model by using the bar operator. For example, two ways of writing the model for a full three-way factorial model follow:

```
scalemodel A B C   A*B A*C B*C   A*B*C;

scalemodel A|B|C;
```

When you use the bar (|), the right and left sides become effects, and the cross of them becomes an effect. Multiple bars are permitted. The expressions are expanded from left to right, using rules 2–4 from Searle (1971, p. 390).

- Multiple bars are evaluated from left to right. For example, A | B | C is evaluated as follows:

$$\begin{aligned} \text{A | B | C} &\rightarrow \{\text{ A | B }\}\text{ | C} \\ &\rightarrow \{\text{ A B A*B }\}\text{ | C} \\ &\rightarrow \text{A B A*B C A*C B*C A*B*C} \end{aligned}$$

- Crossed and nested groups of variables are combined. For example, A(B) | C(D) generates A*C(B D), among other terms.
- Duplicate variables are removed. For example, A(C) | B(C) generates A*B(C C), among other terms, and the extra C is removed.
- Effects are discarded if a variable occurs on both the crossed and nested parts of an effect. For example, A(B) | B(D E) generates A*B(B D E), but this effect is eliminated immediately.

You can also specify the maximum number of variables involved in any effect that results from bar evaluation by specifying that maximum number, preceded by an at sign (@), at the end of the bar effect. For example, the following specification selects only those effects that contain two or fewer variables:

```
scalemodel A|B|C@2;
```

The preceding example is equivalent to the following SCALEMODEL statement:

```
scalemodel A B C   A*B A*C B*C;
```

More examples of using the bar and at sign operators follow:

A | C(B)	is equivalent to	A C(B) A*C(B)
A(B) | C(B)	is equivalent to	A(B) C(B) A*C(B)
A(B) | B(D E)	is equivalent to	A(B) B(D E)
A | B(A) | C	is equivalent to	A B(A) C A*C B*C(A)
A | B(A) | C@2	is equivalent to	A B(A) C A*C
A | B | C | D@2	is equivalent to	A B A*B C A*C B*C D A*D B*D C*D
A*B(C*D)	is equivalent to	A*B(C D)

NOTE: The preceding examples assume the following CLASS statement specification:

```
class A B C D;
```

Colon, Dash, and Double Dash Operators

You can simplify the specification of a large model when some of your variables have a common prefix by using the colon (:) operator and the dash (-) operator. The colon operator selects all variables that have a particular prefix, and the dash operator enables you to list variables that are numbered sequentially. For example, if your data set contains the variables X1 through X9, the following SCALEMODEL statements are equivalent:

```
scalemodel X1 X2 X3 X4 X5 X6 X7 X8 X9;

scalemodel X1-X9;

scalemodel X:;
```

If your data set contains only the three covariates X1, X2, and X9, then the colon operator selects all three variables:

```
scalemodel X:;
```

However, the following specification returns an error because X3 through X8 are not in the data set:

```
scalemodel X1-X9;
```

The double dash (--) operator enables you to select variables that are stored sequentially in the SAS data set, whether or not they have a common prefix. You can use the CONTENTS procedure (see *Base SAS Procedures Guide*) to determine your variable ordering. For example, if you replace the dash in the preceding SCALEMODEL statement with a double dash, as follows, then all three variables are selected:

```
scalemodel X1--X9;
```

If your data set contains the variables A, B, and C, then you can use the double dash operator to select these variables by specifying the following:

```
scalemodel A--C;
```

GLM Parameterization of Classification Variables and Effects

Table 23.9 shows the types of effects that are available in the HPSEVERITY procedure; they are discussed in more detail in the following sections. Let A, B, and C represent classification variables, and let X and Z represent continuous variables.

Table 23.9 Available Types of Effects

Effect	Example	Description
Singleton continuous	X Z	Continuous variables
Polynomial continuous	X*Z	Interaction of continuous variables
Main	A B	CLASS variables
Interaction	A*B	Crossing of CLASS variables
Nested	A(B)	Main effect A nested within CLASS effect B
Continuous-by-class	X*A	Crossing of continuous and CLASS variables
Continuous-nesting-class	X(A)	Continuous variable X nested within CLASS variable A
General	X*Z*A(B)	Combinations of different types of effects

Continuous Effects

Continuous variables or polynomial terms that involve them can be included in the model as continuous effects. An effect that contains a single continuous variable is referred to as a *singleton continuous* effect, and an effect that contains an interaction of only continuous variables is referred to as a *polynomial continuous* effect. The actual values of such terms are included as columns of the relevant model matrices. You can use the bar operator along with a continuous variable to generate polynomial effects. For example, X | X | X expands to X X*X X*X*X, which is a cubic model.

Main Effects

If a classification variable has m levels, the GLM parameterization generates m columns for its main effect in the model matrix. Each column is an indicator variable for a given level. The order of the columns is the sort order of the values of their levels and can be controlled by the ORDER= option in the CLASS statement.

Table 23.10 is an example where β_0 denotes the intercept and A and B are classification variables that have two and three levels, respectively.

Table 23.10 Example of Main Effects

Data		I	A		B		
A	B	β_0	A1	A2	B1	B2	B3
1	1	1	1	0	1	0	0
1	2	1	1	0	0	1	0
1	3	1	1	0	0	0	1
2	1	1	0	1	1	0	0
2	2	1	0	1	0	1	0
2	3	1	0	1	0	0	1

There are usually more columns for these effects than there are degrees of freedom to estimate them. In other words, the GLM parameterization of main effects is *singular*.

Interaction Effects

Often a regression model includes interaction (crossed) effects to account for how the effect of a variable changes along with the values of other variables. In an interaction, the terms are first reordered to correspond to the order of the variables in the CLASS statement. Thus, B*A becomes A*B if A precedes B in the CLASS statement. Then, the GLM parameterization generates columns for all combinations of levels that occur in the data. The order of the columns is such that the rightmost variables in the interaction change faster than the leftmost variables, as illustrated in Table 23.11.

Table 23.11 Example of Interaction Effects

Data		I	A		B			A*B					
A	B	β_0	A1	A2	B1	B2	B3	A1B1	A1B2	A1B3	A2B1	A2B2	A2B3
1	1	1	1	0	1	0	0	1	0	0	0	0	0
1	2	1	1	0	0	1	0	0	1	0	0	0	0
1	3	1	1	0	0	0	1	0	0	1	0	0	0
2	1	1	0	1	1	0	0	0	0	0	1	0	0
2	2	1	0	1	0	1	0	0	0	0	0	1	0
2	3	1	0	1	0	0	1	0	0	0	0	0	1

In the matrix in Table 23.11, main-effects columns are not linearly independent of crossed-effects columns. In fact, the column space for the crossed effects contains the space of the main effect.

When your regression model contains many interaction effects, you might be able to code them more parsimoniously by using the bar operator (|). The bar operator generates all possible interaction effects. For example, A | B | C expands to A B A*B C A*C B*C A*B*C. To eliminate higher-order interaction effects, use the at sign (@) in conjunction with the bar operator. For example, A | B | C | D@2 expands to A B A*B C A*C B*C D A*D B*D C*D.

Nested Effects

Nested effects are generated in the same manner as crossed effects. Hence, the design columns that are generated by the following two statements are the same (but the ordering of the columns is different):

```
scalemodel A B(A);

scalemodel A A*B;
```

The nesting operator in PROC HPSEVERITY is more of a notational convenience than an operation that is distinct from crossing. Nested effects are usually characterized by the property that the nested variables do not appear as main effects. The order of the variables within nesting parentheses is made to correspond to the order of these variables in the CLASS statement. The order of the columns is such that variables outside the parentheses index faster than those inside the parentheses, and the rightmost nested variables index faster than the leftmost variables, as illustrated in Table 23.12.

Table 23.12 Example of Nested Effects

Data		I	A		B(A)					
A	B	β_0	A1	A2	B1A1	B2A1	B3A1	B1A2	B2A2	B3A2
1	1	1	1	0	1	0	0	0	0	0
1	2	1	1	0	0	1	0	0	0	0
1	3	1	1	0	0	0	1	0	0	0
2	1	1	0	1	0	0	0	1	0	0

Table 23.12 *continued*

Data		I	A		B(A)					
2	2	1	0	1	0	0	0	0	1	0
2	3	1	0	1	0	0	0	0	0	1

Continuous-Nesting-Class Effects

When a continuous variable nests or crosses with a classification variable, the design columns are constructed by multiplying the continuous values into the design columns for the classification effect, as illustrated in Table 23.13.

Table 23.13 Example of Continuous-Nesting-Class Effects

Data		I	A		X(A)	
X	A	β_0	A1	A2	X(A1)	X(A2)
21	1	1	1	0	21	0
24	1	1	1	0	24	0
22	1	1	1	0	22	0
28	2	1	0	1	0	28
19	2	1	0	1	0	19
23	2	1	0	1	0	23

Continuous-by-Class Effects

Continuous-by-class effects generate the same design columns as continuous-nesting-class effects. Table 23.14 shows the construction of the X*A effect. The two columns for this effect are the same as the columns for the X(A) effect in Table 23.13.

Table 23.14 Example of Continuous-by-Class Effects

Data		I	X	A		X*A	
X	A	β_0	X	A1	A2	X*A1	X*A2
21	1	1	21	1	0	21	0
24	1	1	24	1	0	24	0
22	1	1	22	1	0	22	0
28	2	1	28	0	1	0	28
19	2	1	19	0	1	0	19
23	2	1	23	0	1	0	23

General Effects

An example that combines all the effects is X1*X2*A*B*C(D E). The continuous list comes first, followed by the crossed list, followed by the nested list in parentheses. PROC HPSEVERITY might rename effects to correspond to ordering rules. For example, B*A(E D) might be renamed A*B(D E) to satisfy the following:

- Classification variables that occur outside parentheses (crossed effects) are sorted in the order in which they appear in the CLASS statement.

- Variables within parentheses (nested effects) are sorted in the order in which they appear in the CLASS statement.

The sequencing of the parameters that are generated by an effect is determined by the variables whose levels are indexed faster:

- Variables in the crossed list index faster than variables in the nested list.
- Within a crossed or nested list, variables to the right index faster than variables to the left.

For example, suppose a model includes four effects—A, B, C, and D—each of which has two levels, 1 and 2. Assume the CLASS statement is

```
class A B C D;
```

Then the order of the parameters for the effect B*A(C D), which is renamed A*B(C D), is

$$\begin{array}{llll} A_1B_1C_1D_1 \rightarrow & A_1B_2C_1D_1 \rightarrow & A_2B_1C_1D_1 \rightarrow & A_2B_2C_1D_1 \rightarrow \\ A_1B_1C_1D_2 \rightarrow & A_1B_2C_1D_2 \rightarrow & A_2B_1C_1D_2 \rightarrow & A_2B_2C_1D_2 \rightarrow \\ A_1B_1C_2D_1 \rightarrow & A_1B_2C_2D_1 \rightarrow & A_2B_1C_2D_1 \rightarrow & A_2B_2C_2D_1 \rightarrow \\ A_1B_1C_2D_2 \rightarrow & A_1B_2C_2D_2 \rightarrow & A_2B_1C_2D_2 \rightarrow & A_2B_2C_2D_2 \end{array}$$

Note that first the crossed effects B and A are sorted in the order in which they appear in the CLASS statement so that A precedes B in the parameter list. Then, for each combination of the nested effects in turn, combinations of A and B appear. The B effect changes fastest because it is rightmost in the cross list. Then A changes next fastest, and D changes next fastest after that. The C effect changes most slowly because it is leftmost in the nested list.

Reference Parameterization

Classification variables can be represented in the reference parameterization. Consider the classification variable A that has four values, 1, 2, 5, and 7. The reference parameterization generates three columns (one less than the number of variable levels). The columns indicate group membership of the nonreference levels. For the reference level, the three dummy variables have a value of 0. If the reference level is 7 (REF='7'), the design columns for variable A are as shown in Table 23.15.

Table 23.15 Reference Coding

	Design Matrix		
A	**A1**	**A2**	**A5**
1	1	0	0
2	0	1	0
5	0	0	1
7	0	0	0

Parameter estimates of CLASS main effects that use the reference coding scheme estimate the difference in the effect of each nonreference level compared to the effect of the reference level.

Empirical Distribution Function Estimation Methods

The empirical distribution function (EDF) is a nonparametric estimate of the cumulative distribution function (CDF) of the distribution. PROC HPSEVERITY computes EDF estimates for two purposes: to send the estimates to a distribution's PARMINIT subroutine in order to initialize the distribution parameters, and to compute the EDF-based statistics of fit.

To reduce the time that it takes to compute the EDF estimates, you can use the INITSAMPLE option to specify that only a fraction of the input data be used. If you do not specify the INITSAMPLE option and the data set has more than 10,000 valid observations, then a uniform random sample of at most 10,000 observations is used for EDF estimation.

In the distributed mode of execution, in which data are distributed across the grid nodes, the EDF estimates are computed on each node by using the portion of the input data that is located on that node. These local EDF estimates are an approximation of the global EDF estimates, which would been computed by using the entire input data set. PROC HPSEVERITY does not compute global EDF estimates. Let X denote a quantity that depends on the EDF estimates. X can be either an EDF-based initial value of a distribution parameter or an EDF-based statistic of fit. PROC HPSEVERITY estimates X as follows: First, each grid node k computes an estimate X_k by using the local EDF estimates that are computed on that node. Then, the estimate $\hat{X}$ of X is computed as an average of all the X_k values; that is, $\hat{X} = \sum_{i=1}^{K} X_k$, where K denotes the total number of nodes where the data reside.

This section describes the methods that are used for computing EDF estimates.

Notation

Let there be a set of N observations, each containing a quintuplet of values $(y_i, t_i^l, t_i^r, c_i^r, c_i^l), i = 1, \ldots, N$, where y_i is the value of the response variable, t_i^l is the value of the left-truncation threshold, t_i^r is the value of the right-truncation threshold, c_i^r is the value of the right-censoring limit, and c_i^l is the value of the left-censoring limit.

If an observation is not left-truncated, then $t_i^l = \tau^l$, where τ^l is the smallest value in the support of the distribution; so $F(t_i^l) = 0$. If an observation is not right-truncated, then $t_i^r = \tau_h$, where τ_h is the largest value in the support of the distribution; so $F(t_i^r) = 1$. If an observation is not right-censored, then $c_i^r = \tau^l$; so $F(c_i^r) = 0$. If an observation is not left-censored, then $c_i^l = \tau_h$; so $F(c_i^l) = 1$.

Let w_i denote the weight associated with ith observation. If you specify the WEIGHT statement, then w_i is the normalized value of the weight variable; otherwise, it is set to 1. The weights are normalized such that they sum up to N.

An indicator function $I[e]$ takes a value of 1 or 0 if the expression e is true or false, respectively.

Estimation Methods

If the response variable is subject to both left-censoring and right-censoring effects and if you explicitly specify the EMPIRICALCDF=TURNBULL option, then PROC HPSEVERITY uses the Turnbull's method. This section describes methods other than Turnbull's method. For Turnbull's method, see the next section "Turnbull's EDF Estimation Method" on page 1263.

The method descriptions assume that all observations are either uncensored or right-censored; that is, each observation is of the form $(y_i, t_i^l, t_i^r, \tau^l, \tau_h)$ or $(y_i, t_i^l, t_i^r, c_i^r, \tau_h)$.

If all observations are either uncensored or left-censored, then each observation is of the form $(y_i, t_i^l, t_i^r, \tau_l, c_i^l)$. It is converted to an observation $(-y_i, -t_i^r, -t_i^l, -c_i^l, \tau_h)$; that is, the signs of all the response variable values are reversed, the new left-truncation threshold is equal to the negative of the original right-truncation threshold, the new right-truncation threshold is equal to the negative of the original left-truncation threshold, and the negative of the original left-censoring limit becomes the new right-censoring limit. With this transformation, each observation is either uncensored or right-censored. The methods described for handling uncensored or right-censored data are now applicable. After the EDF estimates are computed, the observations are transformed back to the original form and EDF estimates are adjusted such $F_n(y_i) = 1 - F_n(-y_i-)$, where $F_n(-y_i-)$ denotes the EDF estimate of the value slightly less than the transformed value $-y_i$.

Further, a set of uncensored or right-censored observations can be converted to a set of observations of the form $(y_i, t_i^l, t_i^r, \delta_i)$, where δ_i is the indicator of right-censoring. $\delta_i = 0$ indicates a right-censored observation, in which case y_i is assumed to record the right-censoring limit c_i^r. $\delta_i = 1$ indicates an uncensored observation, and y_i records the exact observed value. In other words, $\delta_i = I[Y \leq C^r]$ and $y_i = \min(y_i, c_i^r)$.

Given this notation, the EDF is estimated as

$$F_n(y) = \begin{cases} 0 & \text{if } y < y^{(1)} \\ \hat{F}_n(y^{(k)}) & \text{if } y^{(k)} \leq y < y^{(k+1)}, k = 1, \ldots, N-1 \\ \hat{F}_n(y^{(N)}) & \text{if } y^{(N)} \leq y \end{cases}$$

where $y^{(k)}$ denotes the kth order statistic of the set $\{y_i\}$ and $\hat{F}_n(y^{(k)})$ is the estimate computed at that value. The definition of $\hat{F}_n$ depends on the estimation method. You can specify a particular method or let PROC HPSEVERITY choose an appropriate method by using the EMPIRICALCDF= option in the PROC HPSEVERITY statement. Each method computes $\hat{F}_n$ as follows:

NOTURNBULL This is the default method. First, censored observations, if any, are processed as follows:

- An observation that is left-censored but not right-censored is converted to an uncensored observation $(y_i^u, t_i^l, t_i^r, \tau^l, \tau_h)$, where $y_i^u = c_i^l/2$.
- An observation that is both left-censored and right-censored is converted to an uncensored observation $(y_i^u, t_i^l, t_i^r, \tau^l, \tau_h)$, where $y_i^u = (c_i^r + c_i^l)/2$.
- An observation that is right-censored but not left-censored is left unchanged.

If the processed set of observations contains any truncated or right-censored observations, the KAPLANMEIER method is used. Otherwise, the STANDARD method is used.

The observations are modified only for the purpose of computing the EDF estimates. The original censoring information is used by the parameter estimation process.

STANDARD This method is the standard way of computing EDF. The EDF estimate at observation i is computed as follows:

$$\hat{F}_n(y_i) = \frac{1}{N} \sum_{j=1}^{N} w_j \cdot I[y_j \leq y_i]$$

If you do not specify any censoring or truncation information, then this method is chosen. If you explicitly specify this method, then PROC HPSEVERITY ignores any censoring and truncation information that you specify in the LOSS statement.

The standard error of $\hat{F}_n(y_i)$ is computed by using the normal approximation method:

$$\hat{\sigma}_n(y_i) = \sqrt{\hat{F}_n(y_i)(1 - \hat{F}_n(y_i))/N}$$

KAPLANMEIER

The Kaplan-Meier (KM) estimator, also known as the product-limit estimator, was first introduced by Kaplan and Meier (1958) for censored data. Lynden-Bell (1971) derived a similar estimator for left-truncated data. PROC HPSEVERITY uses the definition that combines both censoring and truncation information (Klein and Moeschberger 1997; Lai and Ying 1991).

The EDF estimate at observation i is computed as

$$\hat{F}_n(y_i) = 1 - \prod_{\tau \le y_i} \left(1 - \frac{n(\tau)}{R_n(\tau)}\right)$$

where $n(\tau)$ and $R_n(\tau)$ are defined as follows:

- $n(\tau) = \sum_{k=1}^{N} w_k \cdot I[y_k = \tau \text{ and } \tau \le t_k^r \text{ and } \delta_k = 1]$, which is the number of uncensored observations ($\delta_k = 1$) for which the response variable value is equal to τ and τ is observable according to the right-truncation threshold of that observation ($\tau \le t_k^r$).
- $R_n(\tau) = \sum_{k=1}^{N} w_k \cdot I[y_k \ge \tau > t_k^l]$, which is the size (cardinality) of the risk set at τ. The term *risk set* has its origins in survival analysis; it contains the events that are at risk of failure at a given time, τ. In other words, it contains the events that have survived up to time τ and might fail at or after τ. For PROC HPSEVERITY, *time* is equivalent to the magnitude of the event and *failure* is equivalent to an uncensored and observable event, where observable means it satisfies the truncation thresholds.

This method is chosen when you specify at least one form of censoring or truncation.

The standard error of $\hat{F}_n(y_i)$ is computed by using Greenwood's formula (Greenwood 1926):

$$\hat{\sigma}_n(y_i) = \sqrt{(1 - \hat{F}_n(y_i))^2 \cdot \sum_{\tau \le y_i} \left(\frac{n(\tau)}{R_n(\tau)(R_n(\tau) - n(\tau))}\right)}$$

MODIFIEDKM

The product-limit estimator used by the KAPLANMEIER method does not work well if the risk set size becomes very small. For right-censored data, the size can become small towards the right tail. For left-truncated data, the size can become small at the left tail and can remain so for the entire range of data. This was demonstrated by Lai and Ying (1991). They proposed a modification to the estimator that ignores the effects due to small risk set sizes.

The EDF estimate at observation i is computed as

$$\hat{F}_n(y_i) = 1 - \prod_{\tau \le y_i} \left(1 - \frac{n(\tau)}{R_n(\tau)} \cdot I[R_n(\tau) \ge cN^{\alpha}]\right)$$

where the definitions of $n(\tau)$ and $R_n(\tau)$ are identical to those used for the KAPLAN-MEIER method described previously.

You can specify the values of c and α by using the C= and ALPHA= options. If you do not specify a value for c, the default value used is $c = 1$. If you do not specify a value for α, the default value used is $\alpha = 0.5$.

As an alternative, you can also specify an absolute lower bound, say L, on the risk set size by using the RSLB= option, in which case $I[R_n(\tau) \geq cN^{\alpha}]$ is replaced by $I[R_n(\tau) \geq L]$ in the definition.

The standard error of $\hat{F}_n(y_i)$ is computed by using Greenwood's formula (Greenwood 1926):

$$\hat{\sigma}_n(y_i) = \sqrt{(1 - \hat{F}_n(y_i))^2 \cdot \sum_{\tau \leq y_i} \left(\frac{n(\tau)}{R_n(\tau)(R_n(\tau) - n(\tau))} \cdot I[R_n(\tau) \geq cN^{\alpha}] \right)}$$

Turnbull's EDF Estimation Method

If the response variable is subject to both left-censoring and right-censoring effects and if you explicitly specify the EMPIRICALCDF=TURNBULL option, then the HPSEVERITY procedure uses a method proposed by Turnbull (1976) to compute the nonparametric estimates of the cumulative distribution function. The original Turnbull's method is modified using the suggestions made by Frydman (1994) when truncation effects are present.

Let the input data consist of N observations in the form of quintuplets of values $(y_i, t_i^l, t_i^r, c_i^r, c_i^l), i = 1, \ldots, N$ with notation described in the section "Notation" on page 1260. For each observation, let $A_i = (c_i^r, c_i^l]$ be the censoring interval; that is, the response variable value is known to lie in the interval A_i, but the exact value is not known. If an observation is uncensored, then $A_i = (y_i - \epsilon, y_i]$ for any arbitrarily small value of $\epsilon > 0$. If an observation is censored, then the value y_i is ignored. Similarly, for each observation, let $B_i = (t_i^l, t_i^r]$ be the truncation interval; that is, the observation is drawn from a truncated (conditional) distribution $F(y, B_i) = P(Y \leq y | Y \in B_i)$.

Two sets, L and R, are formed using A_i and B_i as follows:

$$L = \{c_i^r, 1 \leq i \leq N\} \cup \{t_i^r, 1 \leq i \leq N\}$$
$$R = \{c_i^l, 1 \leq i \leq N\} \cup \{t_i^l, 1 \leq i \leq N\}$$

The sets L and R represent the left endpoints and right endpoints, respectively. A set of disjoint intervals $C_j = [q_j, p_j], 1 \leq j \leq M$ is formed such that $q_j \in L$ and $p_j \in R$ and $q_j \leq p_j$ and $p_j < q_{j+1}$. The value of M is dependent on the nature of censoring and truncation intervals in the input data. Turnbull (1976) showed that the maximum likelihood estimate (MLE) of the EDF can increase only inside intervals C_j. In other words, the MLE estimate is constant in the interval (p_j, q_{j+1}). The likelihood is independent of the behavior of F_n inside any of the intervals C_j. Let s_j denote the increase in F_n inside an interval C_j. Then, the EDF estimate is as follows:

$$F_n(y) = \begin{cases} 0 & \text{if } y < q_1 \\ \sum_{k=1}^{j} s_k & \text{if } p_j < y < q_{j+1}, 1 \leq j \leq M-1 \\ 1 & \text{if } y > p_M \end{cases}$$

PROC HPSEVERITY computes the estimates $F_n(p_j+) = F_n(q_{j+1}-) = \sum_{k=1}^{j} s_k$ at points p_j and q_{j+1} and computes $F_n(q_1-) = 0$ at point q_1, where $F_n(x+)$ denotes the limiting estimate at a point that is infinitesimally larger than x when approaching x from values larger than x and where $F_n(x-)$ denotes the limiting estimate at a point that is infinitesimally smaller than x when approaching x from values smaller than x.

PROC HPSEVERITY uses the expectation-maximization (EM) algorithm proposed by Turnbull (1976), who referred to the algorithm as the self-consistency algorithm. By default, the algorithm runs until one of the following criteria is met:

- Relative-error criterion: The maximum relative error between the two consecutive estimates of s_j falls below a threshold ϵ. If l indicates an index of the current iteration, then this can be formally stated as

$$\arg \max_{1 \le j \le M} \left\{ \frac{|s_j^l - s_j^{l-1}|}{s_j^{l-1}} \right\} \le \epsilon$$

  You can control the value of ϵ by specifying the EPS= suboption of the EDF=TURNBULL option in the PROC HPSEVERITY statement. The default value is 1.0E–8.

- Maximum-iteration criterion: The number of iterations exceeds an upper limit that you specify for the MAXITER= suboption of the EDF=TURNBULL option in the PROC HPSEVERITY statement. The default number of maximum iterations is 500.

The self-consistent estimates obtained in this manner might not be maximum likelihood estimates. Gentleman and Geyer (1994) suggested the use of the Kuhn-Tucker conditions for the maximum likelihood problem to ensure that the estimates are MLE. If you specify the ENSUREMLE suboption of the EDF=TURNBULL option in the PROC HPSEVERITY statement, then PROC HPSEVERITY computes the Kuhn-Tucker conditions at the end of each iteration to determine whether the estimates $\{s_j\}$ are MLE. If you do not specify any truncation effects, then the Kuhn-Tucker conditions derived by Gentleman and Geyer (1994) are used. If you specify any truncation effects, then PROC HPSEVERITY uses modified Kuhn-Tucker conditions that account for the truncation effects. An integral part of checking the conditions is to determine whether an estimate s_j is zero or whether an estimate of the Lagrange multiplier or the reduced gradient associated with the estimate s_j is zero. PROC HPSEVERITY declares these values to be zero if they are less than or equal to a threshold δ. You can control the value of δ by specifying the ZEROPROB= suboption of the EDF=TURNBULL option in the PROC HPSEVERITY statement. The default value is 1.0E–8. The algorithm continues until the Kuhn-Tucker conditions are satisfied or the number of iterations exceeds the upper limit. The relative-error criterion stated previously is not used when you specify the ENSUREMLE option.

The standard errors for Turnbull's EDF estimates are computed by using the asymptotic theory of the maximum likelihood estimators (MLE), even though the final estimates might not be MLE. Turnbull's estimator essentially attempts to maximize the likelihood L, which depends on the parameters s_j ($j = 1 \ldots M$). Let $\boldsymbol{s} = \{s_j\}$ denote the set of these parameters. If $\mathbf{G}(\boldsymbol{s}) = \nabla^2(-\log(L(\boldsymbol{s})))$ denotes the Hessian matrix of the negative of log likelihood, then the variance-covariance matrix of $\boldsymbol{s}$ is estimated as $\hat{\mathbf{C}}(\boldsymbol{s}) = \mathbf{G}^{-1}(\boldsymbol{s})$. Given this matrix, the standard error of $F_n(y)$ is computed as

$$\sigma_n(y) = \sqrt{\sum_{k=1}^{j} \left(\hat{C}_{kk} + 2 \cdot \sum_{l=1}^{k-1} \hat{C}_{kl} \right)}, \text{ if } p_j < y < q_{j+1}, 1 \le j \le M-1$$

The standard error is undefined outside of these intervals.

EDF Estimates and Truncation

If you specify truncation, then the estimate $\hat{F}_n(y)$ that is computed by any method other than the STANDARD method is a *conditional* estimate. In other words, $\hat{F}_n(y) = \Pr(Y \le y | \tau_G < Y \le \tau_H)$, where G and H denote the (unknown) distribution functions of the left-truncation threshold variable T^l and the right-truncation threshold variable T^r, respectively, τ_G denotes the smallest left-truncation threshold with a nonzero cumulative probability, and τ_H denotes the largest right-truncation threshold with a nonzero cumulative probability. Formally, $\tau_G = \inf\{s : G(s) > 0\}$ and $\tau_H = \sup\{s : H(s) > 0\}$. For computational purposes, PROC HPSEVERITY estimates τ_G and τ_H by $t^l_{\min}$ and $t^r_{\max}$, respectively, defined as

$$t^l_{\min} = \min\{t^l_k : 1 \le k \le N\}$$
$$t^r_{\max} = \max\{t^r_k : 1 \le k \le N\}$$

These estimates of $t^l_{\min}$ and $t^r_{\max}$ are used to compute the conditional estimates of the CDF as described in the section "Truncation and Conditional CDF Estimates" on page 1241.

If you specify left-truncation *with* the probability of observability p, then PROC HPSEVERITY uses the additional information provided by p to compute an estimate of the EDF that is not conditional on the left-truncation information. In particular, for each left-truncated observation i with response variable value y_i and truncation threshold t^l_i, an observation j is added with *weight* $w_j = (1 - p)/p$ and $y_j = t^l_j$. Each added observation is assumed to be uncensored and untruncated. Then, your specified EDF method is used by assuming no left-truncation. The EDF estimate that is obtained using this method is not conditional on the left-truncation information. For the KAPLANMEIER and MODIFIEDKM methods with uncensored or right-censored data, definitions of $n(\tau)$ and $R_n(\tau)$ are modified to account for the added observations. If N^a denotes the total number of observations including the added observations, then $n(\tau)$ is defined as $n(\tau) = \sum_{k=1}^{N^a} w_k I[y_k = \tau \text{ and } \tau \le t^r_k \text{ and } \delta_k = 1]$, and $R_n(\tau)$ is defined as $R_n(\tau) = \sum_{k=1}^{N^a} w_k I[y_k \ge \tau]$. In the definition of $R_n(\tau)$, the left-truncation information is not used, because it was used along with p to add the observations.

If the original data are a combination of left- and right-censored data and if you specify the EMPIRICALCDF=TURNBULL option, then Turnbull's method is applied to the appended set that contains no left-truncated observations.

Supplying EDF Estimates to Functions and Subroutines

The parameter initialization subroutines in distribution models and some predefined utility functions require EDF estimates. For more information, see the sections "Defining a Severity Distribution Model with the FCMP Procedure" on page 1273 and "Predefined Utility Functions" on page 1285.

PROC HPSEVERITY supplies the EDF estimates to these subroutines and functions by using two arrays, x and F, the dimension of each array, and a type of the EDF estimates. The type identifies how the EDF estimates are computed and stored. They are as follows:

Type 1 specifies that EDF estimates are computed using the STANDARD method; that is, the data that are used for estimation are neither censored nor truncated.

Type 2 specifies that EDF estimates are computed using either the KAPLANMEIER or the MODIFIEDKM method; that is, the data that are used for estimation are subject to truncation and one type of censoring (left or right, but not both).

Type 3 specifies that EDF estimates are computed using the TURNBULL method; that is, the data that are used for estimation are subject to both left- and right-censoring. The data might or might not be truncated.

For Types 1 and 2, the EDF estimates are stored in arrays x and F of dimension N such that the following holds:

$$F_n(y) = \begin{cases} 0 & \text{if } y < x[1] \\ F[k] & \text{if } x[k] \le y < x[k+1], k = 1, \ldots, N-1 \\ F[N] & \text{if } x[N] \le y \end{cases}$$

where $[k]$ denotes kth element of the array ([1] denotes the first element of the array).

For Type 3, the EDF estimates are stored in arrays x and F of dimension N such that the following holds:

$$F_n(y) = \begin{cases} 0 & \text{if } y < x[1] \\ \text{undefined} & \text{if } x[2k-1] \le y < x[2k], k = 1, \ldots, (N-1)/2 \\ F[2k] = F[2k+1] & \text{if } x[2k] \le y < x[2k+1], k = 1, \ldots, (N-1)/2 \\ F[N] & \text{if } x[N] \le y \end{cases}$$

Although the behavior of EDF is theoretically undefined for the interval $[x[2k-1], x[2k])$, for computational purposes, all predefined functions and subroutines assume that the EDF increases linearly from $F[2k-1]$ to $F[2k]$ in that interval if $x[2k-1] < x[2k]$. If $x[2k-1] = x[2k]$, which can happen when the EDF is estimated from a combination of uncensored and interval-censored data, the predefined functions and subroutines assume that $F_n(x[2k-1]) = F_n(x[2k]) = F[2k]$.

Statistics of Fit

PROC HPSEVERITY computes and reports various statistics of fit to indicate how well the estimated model fits the data. The statistics belong to two categories: likelihood-based statistics and EDF-based statistics. Neg2LogLike, AIC, AICC, and BIC are likelihood-based statistics, and KS, AD, and CvM are EDF-based statistics.

In the distributed mode of execution, in which data are distributed across the grid nodes, the EDF estimates are computed by using the local data. The EDF-based statistics are computed by using these local EDF estimates. The reported value of each EDF-based statistic is an average of the values of the statistic that are computed by all the grid nodes where the data reside. Also, for large data sets, in both single-machine and distributed modes of execution, the EDF estimates are computed by using a fraction of the input data that is governed by either the INITSAMPLE option or the default sample size. Because of this nature of computing the EDF estimates, the EDF-based statistics of fit are an approximation of the values that would have been computed if the entire input data set were used for computing the EDF estimates. So the values that are reported for EDF-based statistics should be used only for comparing different models. The reported values should not be interpreted as true estimates of the corresponding statistics.

The likelihood-based statistics are reported for the entire input data in both single-machine and distributed modes of execution.

The following subsections provide definitions of each category of statistics.

Likelihood-Based Statistics of Fit

Let $y_i, i = 1, \ldots, N$, denote the response variable values. Let L be the likelihood as defined in the section "Likelihood Function" on page 1242. Let p denote the number of model parameters that are estimated. Note that $p = p_d + (k - k_r)$, where p_d is the number of distribution parameters, k is the number of all regression parameters, and k_r is the number of regression parameters that are found to be linearly dependent (redundant) on other regression parameters. Given this notation, the likelihood-based statistics are defined as follows:

Neg2LogLike The log likelihood is reported as

$$\text{Neg2LogLike} = -2\log(L)$$

The multiplying factor -2 makes it easy to compare it to the other likelihood-based statistics. A model that has a smaller value of Neg2LogLike is deemed better.

AIC Akaike's information criterion (AIC) is defined as

$$\text{AIC} = -2\log(L) + 2p$$

A model that has a smaller AIC value is deemed better.

AICC The corrected Akaike's information criterion (AICC) is defined as

$$\text{AICC} = -2\log(L) + \frac{2Np}{N-p-1}$$

A model that has a smaller AICC value is deemed better. It corrects the finite-sample bias that AIC has when N is small compared to p. AICC is related to AIC as

$$\text{AICC} = \text{AIC} + \frac{2p(p+1)}{N-p-1}$$

As N becomes large compared to p, AICC converges to AIC. AICC is usually recommended over AIC as a model selection criterion.

BIC The Schwarz Bayesian information criterion (BIC) is defined as

$$\text{BIC} = -2\log(L) + p\log(N)$$

A model that has a smaller BIC value is deemed better.

EDF-Based Statistics

This class of statistics is based on the difference between the estimate of the cumulative distribution function (CDF) and the estimate of the empirical distribution function (EDF). A model that has a smaller value of the chosen EDF-based statistic is deemed better.

Let $y_i, i = 1, \ldots, N$ denote the sample of N values of the response variable. Let $r_i = \sum_{j=1}^{N} I[y_j \le y_i]$ denote the number of observations with a value less than or equal to y_i, where I is an indicator function. Let $F_n(y_i)$ denote the EDF estimate that is computed by using the method that you specify in the EMPIRICAL-CDF= option. Let $Z_i = \hat{F}(y_i)$ denote the estimate of the CDF. Let $F_n(Z_i)$ denote the EDF estimate of Z_i values that are computed using the same method that is used to compute the EDF of y_i values. Using the probability integral transformation, if $F(y)$ is the true distribution of the random variable Y, then the random variable $Z = F(y)$ is uniformly distributed between 0 and 1 (D'Agostino and Stephens 1986, Ch. 4). Thus, comparing $F_n(y_i)$ with $\hat{F}(y_i)$ is equivalent to comparing $F_n(Z_i)$ with $\hat{F}(Z_i) = Z_i$ (uniform distribution).

Note the following two points regarding which CDF estimates are used for computing the test statistics:

- If you specify regression effects, then the CDF estimates Z_i that are used for computing the EDF test statistics are from a mixture distribution. See the section "CDF and PDF Estimates with Regression Effects" on page 1249 for more information.

- If the EDF estimates are conditional because of the truncation information, then each unconditional estimate Z_i is converted to a conditional estimate using the method described in the section "Truncation and Conditional CDF Estimates" on page 1241.

In the following, it is assumed that Z_i denotes an appropriate estimate of the CDF if you specify any truncation or regression effects. Given this, the EDF-based statistics of fit are defined as follows:

KS The Kolmogorov-Smirnov (KS) statistic computes the largest vertical distance between the CDF and the EDF. It is formally defined as follows:

$$\text{KS} = \sup_y |F_n(y) - F(y)|$$

If the STANDARD method is used to compute the EDF, then the following formula is used:

$$D^+ = \max_i(\frac{r_i}{N} - Z_i)$$
$$D^- = \max_i(Z_i - \frac{r_{i-1}}{N})$$
$$\text{KS} = \sqrt{N}\ \max(D^+, D^-) + \frac{0.19}{\sqrt{N}}$$

Note that r_0 is assumed to be 0.

If the method used to compute the EDF is any method other than the STANDARD method, then the following formula is used:

$$D^+ = \max_i(F_n(Z_i) - Z_i),\ \text{if } F_n(Z_i) \geq Z_i$$
$$D^- = \max_i(Z_i - F_n(Z_i)),\ \text{if } F_n(Z_i) < Z_i$$
$$\text{KS} = \sqrt{N}\ \max(D^+, D^-) + \frac{0.19}{\sqrt{N}}$$

AD The Anderson-Darling (AD) statistic is a quadratic EDF statistic that is proportional to the expected value of the weighted squared difference between the EDF and CDF. It is formally defined as follows:

$$\text{AD} = N \int_{-\infty}^{\infty} \frac{(F_n(y) - F(y))^2}{F(y)(1 - F(y))} dF(y)$$

If the STANDARD method is used to compute the EDF, then the following formula is used:

$$\text{AD} = -N - \frac{1}{N}\sum_{i=1}^{N}[(2r_i - 1)\log(Z_i) + (2N + 1 - 2r_i)\log(1 - Z_i)]$$

If the method used to compute the EDF is any method other than the STANDARD method, then the statistic can be computed by using the following two pieces of information:

- If the EDF estimates are computed using the KAPLANMEIER or MODIFIEDKM methods, then EDF is a step function such that the estimate $F_n(z)$ is a constant equal to $F_n(Z_{i-1})$ in interval $[Z_{i-1}, Z_i]$. If the EDF estimates are computed using the TURNBULL method, then there are two types of intervals: one in which the EDF curve is constant and the other in

which the EDF curve is theoretically undefined. For computational purposes, it is assumed that the EDF curve is linear for the latter type of the interval. For each method, the EDF estimate $F_n(y)$ at y can be written as

$$F_n(z) = F_n(Z_{i-1}) + S_i(z - Z_{i-1}), \text{ for } z \in [Z_{i-1}, Z_i]$$

where S_i is the slope of the line defined as

$$S_i = \frac{F_n(Z_i) - F_n(Z_{i-1})}{Z_i - Z_{i-1}}$$

For the KAPLANMEIER or MODIFIEDKM method, $S_i = 0$ in each interval.

- Using the probability integral transform $z = F(y)$, the formula simplifies to

$$\text{AD} = N \int_{-\infty}^{\infty} \frac{(F_n(z) - z)^2}{z(1-z)} dz$$

The computation formula can then be derived from the following approximation:

$$\begin{aligned}
\text{AD} &= N \sum_{i=1}^{K+1} \int_{Z_{i-1}}^{Z_i} \frac{(F_n(z) - z)^2}{z(1-z)} dz \\
&= N \sum_{i=1}^{K+1} \int_{Z_{i-1}}^{Z_i} \frac{(F_n(Z_{i-1}) + S_i(z - Z_{i-1}) - z)^2}{z(1-z)} dz \\
&= N \sum_{i=1}^{K+1} \int_{Z_{i-1}}^{Z_i} \frac{(P_i - Q_i z)^2}{z(1-z)} dz
\end{aligned}$$

where $P_i = F_n(Z_{i-1}) - S_i Z_{i-1}$, $Q_i = 1 - S_i$, and K is the number of points at which the EDF estimate are computed. For the TURNBULL method, $K = 2k$ for some k.

Assuming $Z_0 = 0$, $Z_{K+1} = 1$, $F_n(0) = 0$, and $F_n(Z_K) = 1$ yields the following computation formula:

$$\begin{aligned}
\text{AD} = &- N(Z_1 + \log(1 - Z_1) + \log(Z_K) + (1 - Z_K)) \\
&+ N \sum_{i=2}^{K} \left[P_i^2 A_i - (Q_i - P_i)^2 B_i - Q_i^2 C_i \right]
\end{aligned}$$

where $A_i = \log(Z_i) - \log(Z_{i-1})$, $B_i = \log(1 - Z_i) - \log(1 - Z_{i-1})$, and $C_i = Z_i - Z_{i-1}$.

If EDF estimates are computed using the KAPLANMEIER or MODIFIEDKM method, then $P_i = F_n(Z_{i-1})$ and $Q_i = 1$, which simplifies the formula as

$$\begin{aligned}
\text{AD} = &- N(1 + \log(1 - Z_1) + \log(Z_K)) \\
&+ N \sum_{i=2}^{K} \left[F_n(Z_{i-1})^2 A_i - (1 - F_n(Z_{i-1}))^2 B_i \right]
\end{aligned}$$

CvM The Cramér-von Mises (CvM) statistic is a quadratic EDF statistic that is proportional to the expected value of the squared difference between the EDF and CDF. It is formally defined as follows:

$$\text{CvM} = N \int_{-\infty}^{\infty} (F_n(y) - F(y))^2 dF(y)$$

If the STANDARD method is used to compute the EDF, then the following formula is used:

$$\text{CvM} = \frac{1}{12N} + \sum_{i=1}^{N}\left(Z_i - \frac{(2r_i - 1)}{2N}\right)^2$$

If the method used to compute the EDF is any method other than the STANDARD method, then the statistic can be computed by using the following two pieces of information:

- As described previously for the AD statistic, the EDF estimates are assumed to be piecewise linear such that the estimate $F_n(y)$ at y is

  $$F_n(z) = F_n(Z_{i-1}) + S_i(z - Z_{i-1}), \text{ for } z \in [Z_{i-1}, Z_i]$$

  where S_i is the slope of the line defined as

  $$S_i = \frac{F_n(Z_i) - F_n(Z_{i-1})}{Z_i - Z_{i-1}}$$

  For the KAPLANMEIER or MODIFIEDKM method, $S_i = 0$ in each interval.
- Using the probability integral transform $z = F(y)$, the formula simplifies to:

  $$\text{CvM} = N\int_{-\infty}^{\infty}(F_n(z) - z)^2 dz$$

The computation formula can then be derived from the following approximation:

$$\begin{aligned}\text{CvM} &= N\sum_{i=1}^{K+1}\int_{Z_{i-1}}^{Z_i}(F_n(z) - z)^2 dz \\ &= N\sum_{i=1}^{K+1}\int_{Z_{i-1}}^{Z_i}(F_n(Z_{i-1}) + S_i(z - Z_{i-1}) - z)^2 dz \\ &= N\sum_{i=1}^{K+1}\int_{Z_{i-1}}^{Z_i}(P_i - Q_i z)^2 dz\end{aligned}$$

where $P_i = F_n(Z_{i-1}) - S_i Z_{i-1}$, $Q_i = 1 - S_i$, and K is the number of points at which the EDF estimate are computed. For the TURNBULL method, $K = 2k$ for some k.

Assuming $Z_0 = 0$, $Z_{K+1} = 1$, and $F_n(0) = 0$ yields the following computation formula:

$$\text{CvM} = N\frac{Z_1^3}{3} + N\sum_{i=2}^{K+1}\left[P_i^2 A_i - P_i Q_i B_i - \frac{Q_i^2}{3}C_i\right]$$

where $A_i = Z_i - Z_{i-1}$, $B_i = Z_i^2 - Z_{i-1}^2$, and $C_i = Z_i^3 - Z_{i-1}^3$.

If EDF estimates are computed using the KAPLANMEIER or MODIFIEDKM method, then $P_i = F_n(Z_{i-1})$ and $Q_i = 1$, which simplifies the formula as

$$\text{CvM} = \frac{N}{3} + N\sum_{i=2}^{K+1}\left[F_n(Z_{i-1})^2(Z_i - Z_{i-1}) - F_n(Z_{i-1})(Z_i^2 - Z_{i-1}^2)\right]$$

which is similar to the formula proposed by Koziol and Green (1976).

Distributed and Multithreaded Computation

PROC HPSEVERITY makes an attempt to use all the computational resources that you specify in the PERFORMANCE statement in order to complete the assigned tasks as fast as possible. This section describes the distributed and multithreading computing methods that PROC HPSEVERITY uses.

Distributed Computing

Distributed computing refers to the organization of computation work into multiple tasks that are processed on different nodes; a node is one of the machines that constitute the grid. The number of nodes that PROC HPSEVERITY uses is determined by the distributed processing execution mode. If you specify the client-data (or local-data) mode of execution, then the number of nodes is determined by the NODES= option in the PERFORMANCE statement. If you are using the alongside-the-database mode of execution, then PROC HPSEVERITY determines the number of nodes internally by using the information that is associated with the DATA= data set and the grid information that you specify either in the PERFORMANCE statement or in the grid environment variables. For more information about distributed processing modes, see the section "Processing Modes" on page 62.

In the client-data model, PROC HPSEVERITY distributes the input data across the number of nodes that you specify by sending the first observation to the first node, the second observation to the second node, and so on.

In the alongside-the-database model, PROC HPSEVERITY uses the existing distributed organization of the data. You do not need to specify the NODES= option.

The number of nodes that are used for distributed computing is displayed in the "Performance Information" table, which is part of the default output.

Multithreading

Threading refers to the organization of computational work into multiple tasks (processing units that can be scheduled by the operating system). A task is associated with a thread. Multithreading refers to the concurrent execution of threads. When multithreading is possible, you can achieve more substantial performance gains than you can with sequential (single-threaded) execution.

The number of threads the HPSEVERITY procedure spawns is determined by the number of CPUs on a machine. You can control the number of CPUs in the following ways:

- You can use the CPUCOUNT= SAS system option to specify the CPU count. For example, if you specify the following statement, then PROC HPSEVERITY schedules threads as if it were executing on a system that had four CPUs, regardless of the actual CPU count:

```
options cpucount=4;
```

  You can use this specification only in single-machine mode, and it does not take effect if the THREADS system option is turned off.

The default value of the CPUCOUNT= system option might not equal the number of all the logical CPU cores available on your machine, such as those available because of hyperthreading. To allow PROC HPSEVERITY to use all the logical cores in single-machine mode, specify the following OPTIONS statement:

```
options cpucount=actual;
```

- You can specify the NTHREADS= option in the PERFORMANCE statement. This specification overrides the THREADS and CPUCOUNT= system options. Specify NTHREADS=1 to force single-threaded execution.

  If you do not specify the NTHREADS= option and the THREADS system option is turned on, then the number of threads that are used in distributed mode is equal to the total number of logical CPU cores available on each node of the grid, and the number of threads used in single-machine mode is determined by the CPUCOUNT= system option.

  If you do not specify the NTHREADS= option and the THREADS system option is turned off, then only one thread of execution is used in both single-machine and distributed modes.

The number of threads per machine is displayed in the "Performance Information" table, which is part of the default output.

Performance improvement is not always guaranteed when you use more threads, for several reasons: the increased cost of communication and synchronization among threads might offset the reduced cost of computation, the hyperthreading feature of the processor might not be very efficient for floating-point computations, and other applications might be running on the machine.

Combining the Power of Distributed and Multithreading Computing

The HPSEVERITY procedure combines the powers of distributed and multithreading paradigms by using a data-parallel model. In particular, the distributed tasks are defined by dividing the data among multiple nodes, and within one node, the multithreading tasks are defined by further dividing the local data among the threads. For example, if the input data set has 10,000 observations and you are running on a grid that has five nodes, then each node processes 2,000 observations (this assumes that if you specify an alongside-the-database model, then you have equally and randomly divided the input data among the nodes). Further, if each node has eight CPUs, then 250 observations are associated with each thread within the node. All computations that require access to the data are then distributed and multithreaded.

Note that in single-machine mode (see the section "Processing Modes" on page 62), only multithreading is available.

When you specify more than one candidate distribution model, for some tasks PROC HPSEVERITY exploits the independence among models by processing multiple models in parallel on a single node such that each model is assigned to one of the threads executing in parallel. When a thread finishes processing the assigned model, it starts processing the next unprocessed model, if one exists.

The computations that take advantage of the distributed and multithreaded model include the following:

- Validation and preparation of data: In this stage, the observations in the input data set are validated and transformed, if necessary. The summary statistics of the data are prepared. Because each observation is independent, the computations can be distributed among nodes and among threads within nodes without significant communication overhead.

- Initialization of distribution parameters: In this stage, the parallelism is achieved by initializing multiple models in parallel. The only computational step that is not fully parallelized in this release is the step of computing empirical distribution function (EDF) estimates, which are required when PROC HPSEVERITY needs to invoke a distribution's PARMINIT subroutine to initialize distribution parameters. The EDF estimation step is not amenable to full-fledged parallelism because it requires sequential access to sorted data, especially when the loss variable is modified by truncation effects. When the data are distributed across nodes, the EDF computations take place on local data and the PARMINIT function is invoked on the local data by using the local EDF estimates. The initial values that are supplied to the nonlinear optimizer are computed by averaging the local estimates of the distribution parameters that are returned by the PARMINIT functions on each node.

- Initialization of regression parameters (if you specify the SCALEMODEL statement): In this stage, if you do not specify initial values for the regression parameters by using the INEST= data set or the INSTORE= item store, then PROC HPSEVERITY initializes those parameters by solving a linear regression problem $\log(y) = \beta_0 + \sum_{i=1}^{k} \beta_j x_j$. For more information, see the section "Parameter Initialization for Regression Models" on page 1247. The most computationally intensive step is the formation of the crossproducts matrix. PROC HPSEVERITY exploits the parallelism by observing the fact that the contribution to the crossproducts matrix due to one observation is independent from the contribution due to another observation. Each node computes the contribution of its local data to each entry of the crossproducts matrix. Within each node, each thread computes the contribution of its chunk of data to each entry of the crossproducts matrix. On each node, the contributions from all the threads are added up to form the contribution due to all of the local data. The partial crossproducts matrices are then gathered from all nodes on a central node, which sums them up to form the final crossproducts matrix.

- Optimization: In this stage, the nonlinear optimizer iterates over the parameter space in search of the optimal set of parameters. In each iteration, it evaluates the objective function along with the gradient and Hessian of the objective function, if needed by the optimization method. Within one iteration, for the current estimates of the parameters, each observation's contribution to the objective function, gradient, and Hessian is independent of another observation. This enables PROC HPSEVERITY to fully exploit the distributed and multithreaded paradigms to efficiently parallelize each iteration of the algorithm.

Defining a Severity Distribution Model with the FCMP Procedure

A *severity distribution model* consists of a set of functions and subroutines that are defined using the FCMP procedure. The FCMP procedure is part of Base SAS software. Each function or subroutine must be named as *<distribution-name>_<keyword>*, where *distribution-name* is the identifying short name of the distribution and *keyword* identifies one of the functions or subroutines. The total length of the name should not exceed 32. Each function or subroutine must have a specific signature, which consists of the number of arguments, sequence and types of arguments, and return value type. The summary of all the recognized function and subroutine names and their expected behavior is given in Table 23.16.

Consider the following points when you define a distribution model:

- When you define a function or subroutine requiring parameter arguments, the names and order of those arguments must be the same. Arguments other than the parameter arguments can have any name, but they must satisfy the requirements on their type and order.

- When the HPSEVERITY procedure invokes any function or subroutine, it provides the necessary input values according to the specified signature, and expects the function or subroutine to prepare the output and return it according to the specification of the return values in the signature.
- You can use most of the SAS programming statements and SAS functions that you can use in a DATA step for defining the FCMP functions and subroutines. However, there are a few differences in the capabilities of the DATA step and the FCMP procedure. To learn more, see the documentation of the FCMP procedure in the *Base SAS Procedures Guide*.
- You must specify either the PDF or the LOGPDF function. Similarly, you must specify either the CDF or the LOGCDF function. All other functions are optional, except when necessary for correct definition of the distribution. It is strongly recommended that you define the PARMINIT subroutine to provide a good set of initial values for the parameters. The information provided by PROC HPSEVERITY to the PARMINIT subroutine enables you to use popular initialization approaches based on the method of moments and the method of percentile matching, but you can implement any algorithm to initialize the parameters by using the values of the response variable and the estimate of its empirical distribution function.
- The LOWERBOUNDS subroutines should be defined if the lower bound on at least one distribution parameter is different from the default lower bound of 0. If you define a LOWERBOUNDS subroutine but do not set a lower bound for some parameter inside the subroutine, then that parameter is assumed to have no lower bound (or a lower bound of $-\infty$). Hence, it is recommended that you explicitly return the lower bound for each parameter when you define the LOWERBOUNDS subroutine.
- The UPPERBOUNDS subroutines should be defined if the upper bound on at least one distribution parameter is different from the default upper bound of ∞. If you define an UPPERBOUNDS subroutine but do not set an upper bound for some parameter inside the subroutine, then that parameter is assumed to have no upper bound (or a upper bound of ∞). Hence, it is recommended that you explicitly return the upper bound for each parameter when you define the UPPERBOUNDS subroutine.
- If you want to use the distribution in a model with regression effects, then make sure that the first parameter of the distribution is the scale parameter itself or a log-transformed scale parameter. If the first parameter is a log-transformed scale parameter, then you must define the SCALETRANSFORM function.
- In general, it is not necessary to define the gradient and Hessian functions, because the HPSEVERITY procedure uses an internal system to evaluate the required derivatives. The internal system typically computes the derivatives analytically. But it might not be able to do so if your function definitions use other functions that it cannot differentiate analytically. In such cases, derivatives are approximated using a finite difference method and a note is written to the SAS log to indicate the components that are differentiated using such approximations. PROC HPSEVERITY does reasonably well with these finite difference approximations. But, if you know of a way to compute the derivatives of such components analytically, then you should define the gradient and Hessian functions.

In order to use your distribution with PROC HPSEVERITY, you need to record the FCMP library that contains the functions and subroutines for your distribution and other FCMP libraries that contain FCMP functions or subroutines used within your distribution's functions and subroutines. Specify all those libraries in the CMPLIB= system option by using the OPTIONS global statement. For more information about the OPTIONS statement, see *SAS Statements: Reference*. For more information about the CMPLIB= system option, see *SAS System Options: Reference*.

Each predefined distribution mentioned in the section "Predefined Distributions" on page 1230 has a distribution model associated with it. The functions and subroutines of all those models are available in the

Sashelp.Svrtdist library. The order of the parameters in the signatures of the functions and subroutines is the same as listed in Table 23.2. You do not need to use the CMPLIB= option in order to use the predefined distributions with PROC HPSEVERITY. However, if you need to use the functions or subroutines of the predefined distributions in SAS statements other than the PROC HPSEVERITY step (such as in a DATA step), then specify the Sashelp.Svrtdist library in the CMPLIB= system option by using the OPTIONS global statement prior to using them.

Table 23.16 shows functions and subroutines that define a distribution model, and subsections after the table provide more detail. The functions are listed in alphabetical order of the keyword suffix.

Table 23.16 List of Functions and Subroutines That Define a Distribution Model

Name	Type	Required	Expected to Return
*dist*_CDF	Function	YES[1]	Cumulative distribution function value
*dist*_CDFGRADIENT	Subroutine	NO	Gradient of the CDF
*dist*_CDFHESSIAN	Subroutine	NO	Hessian of the CDF
*dist*_CONSTANTPARM	Subroutine	NO	Constant parameters
*dist*_DESCRIPTION	Function	NO	Description of the distribution
*dist*_LOGCDF	Function	YES[1]	Log of cumulative distribution function value
*dist*_LOGCDFGRADIENT	Subroutine	NO	Gradient of the LOGCDF
*dist*_LOGCDFHESSIAN	Subroutine	NO	Hessian of the LOGCDF
*dist*_LOGPDF	Function	YES[2]	Log of probability density function value
*dist*_LOGPDFGRADIENT	Subroutine	NO	Gradient of the LOGPDF
*dist*_LOGPDFHESSIAN	Subroutine	NO	Hessian of the LOGPDF
*dist*_LOGSDF	Function	NO	Log of survival function value
*dist*_LOGSDFGRADIENT	Subroutine	NO	Gradient of the LOGSDF
*dist*_LOGSDFHESSIAN	Subroutine	NO	Hessian of the LOGSDF
*dist*_LOWERBOUNDS	Subroutine	NO	Lower bounds on parameters
*dist*_PARMINIT	Subroutine	NO	Initial values for parameters
*dist*_PDF	Function	YES[2]	Probability density function value
*dist*_PDFGRADIENT	Subroutine	NO	Gradient of the PDF
*dist*_PDFHESSIAN	Subroutine	NO	Hessian of the PDF
*dist*_QUANTILE	Function	NO	Quantile for a given CDF value
*dist*_SCALETRANSFORM	Function	NO	Type of relationship between the first distribution parameter and the scale parameter
*dist*_SDF	Function	NO	Survival function value
*dist*_SDFGRADIENT	Subroutine	NO	Gradient of the SDF
*dist*_SDFHESSIAN	Subroutine	NO	Hessian of the SDF
*dist*_UPPERBOUNDS	Subroutine	NO	Upper bounds on parameters

Notes:
1. Either the *dist*_CDF or the *dist*_LOGCDF function must be defined.
2. Either the *dist*_PDF or the *dist*_LOGPDF function must be defined.

The signature syntax and semantics of each function or subroutine are as follows:

*dist*_**CDF**
defines a function that returns the value of the cumulative distribution function (CDF) of the distribution at the specified values of the random variable and distribution parameters.

- *Type*: Function
- *Required*: YES
- *Number of arguments*: $m + 1$, where m is the number of distribution parameters
- *Sequence and type of arguments*:

  x Numeric value of the random variable at which the CDF value should be evaluated

  p1 Numeric value of the first parameter

  p2 Numeric value of the second parameter

  ...

  pm Numeric value of the mth parameter

- *Return value*: Numeric value that contains the CDF value $F(x; p_1, p_2, \ldots, p_m)$

If you want to consider this distribution as a candidate distribution when you estimate a response variable model with regression effects, then the first parameter of this distribution must be a scale parameter or log-transformed scale parameter. In other words, if the distribution has a scale parameter, then the following equation must be satisfied:

$$F(x; p_1, p_2, \ldots, p_m) = F(\frac{x}{p_1}; 1, p_2, \ldots, p_m)$$

If the distribution has a log-transformed scale parameter, then the following equation must be satisfied:

$$F(x; p_1, p_2, \ldots, p_m) = F(\frac{x}{\exp(p_1)}; 0, p_2, \ldots, p_m)$$

Here is a sample structure of the function for a distribution named 'FOO':

```
function FOO_CDF(x, P1, P2);
    /* Code to compute CDF by using x, P1, and P2 */

    F = <computed CDF>;
    return (F);
endsub;
```

*dist*_**CONSTANTPARM**
defines a subroutine that specifies constant parameters. A parameter is *constant* if it is required for defining a distribution but is not subject to optimization in PROC HPSEVERITY. Constant parameters are required to be part of the model in order to compute the PDF or the CDF of the distribution. Typically, values of these parameters are known a priori or estimated using some means other than the maximum likelihood method used by PROC HPSEVERITY. You can estimate them inside the *dist*_PARMINIT subroutine. Once initialized, the parameters remain constant in the context of PROC HPSEVERITY; that is, they retain their initial value. PROC HPSEVERITY estimates only the nonconstant parameters.

- *Type*: Subroutine
- *Required*: NO
- *Number of arguments*: *k*, where *k* is the number of constant parameters
- *Sequence and type of arguments*:

  constant parameter 1 Name of the first constant parameter

  . . .

  constant parameter *k* Name of the *k*th constant parameter

- *Return value*: None

Here is a sample structure of the subroutine for a distribution named 'FOO' that has P3 and P5 as its constant parameters, assuming that distribution has at least three parameters:

```
subroutine FOO_CONSTANTPARM(p5, p3);
endsub;
```

Note the following points when you specify the constant parameters:

- At least one distribution parameter must be free to be optimized; that is, if a distribution has total *m* parameters, then *k* must be strictly less than *m*.
- If you want to use this distribution for modeling regression effects, then the first parameter must not be a constant parameter.
- The order of arguments in the signature of this subroutine does not matter as long as each argument's name matches the name of one of the parameters that are defined in the signature of the *dist*_PDF function.
- The constant parameters must be specified in signatures of all the functions and subroutines that accept distribution parameters as their arguments.
- You must provide a nonmissing initial value for each constant parameter by using one of the supported parameter initialization methods.

*dist*_DESCRIPTION

defines a function that returns a description of the distribution.

- *Type*: Function
- *Required*: NO
- *Number of arguments*: None
- *Sequence and type of arguments*: Not applicable
- *Return value*: Character value containing a description of the distribution

Here is a sample structure of the function for a distribution named 'FOO':

```
function FOO_DESCRIPTION() $48;
    length desc $48;
    desc = "A model for a continuous distribution named foo";
    return (desc);
endsub;
```

There is no restriction on the length of the description (the length of 48 used in the previous example is for illustration purposes only). However, if the length is greater than 256, then only the first 256 characters appear in the displayed output and in the _DESCRIPTION_ variable of the OUTMODELINFO= data set. Hence, the recommended length of the description is less than or equal to 256.

*dist*_**LOG***core*
defines a function that returns the natural logarithm of the specified *core* function of the distribution at the specified values of the random variable and distribution parameters. The *core* keyword can be PDF, CDF, or SDF.

- *Type*: Function
- *Required*: YES only if *core* is PDF or CDF and you have not defined that *core* function; otherwise, NO
- *Number of arguments*: $m + 1$, where m is the number of distribution parameters
- *Sequence and type of arguments*:

  x Numeric value of the random variable at which the natural logarithm of the *core* function should be evaluated

  p1 Numeric value of the first parameter

  p2 Numeric value of the second parameter

  ...

  p*m* Numeric value of the *m*th parameter

- *Return value*: Numeric value that contains the natural logarithm of the *core* function

Here is a sample structure of the function for the core function PDF of a distribution named 'FOO':

```
function FOO_LOGPDF(x, P1, P2);
    /* Code to compute LOGPDF by using x, P1, and P2 */

    l = <computed LOGPDF>;
    return (l);
endsub;
```

*dist*_**LOWERBOUNDS**
defines a subroutine that returns lower bounds for the parameters of the distribution. If this subroutine is not defined for a given distribution, then the HPSEVERITY procedure assumes a lower bound of 0 for each parameter. If a lower bound of l_i is returned for a parameter p_i, then the HPSEVERITY procedure assumes that $l_i < p_i$ (strict inequality). If a missing value is returned for some parameter, then the HPSEVERITY procedure assumes that there is no lower bound for that parameter (equivalent to a lower bound of $-\infty$).

- *Type*: Subroutine
- *Required*: NO
- *Number of arguments*: m, where m is the number of distribution parameters

- *Sequence and type of arguments*:

  p1 Output argument that returns the lower bound on the first parameter. You must specify this in the OUTARGS statement inside the subroutine's definition.

  p2 Output argument that returns the lower bound on the second parameter. You must specify this in the OUTARGS statement inside the subroutine's definition.

  ...

  p*m* Output argument that returns the lower bound on the *m*th parameter. You must specify this in the OUTARGS statement inside the subroutine's definition.

- *Return value*: The results, lower bounds on parameter values, should be returned in the parameter arguments of the subroutine.

Here is a sample structure of the subroutine for a distribution named 'FOO':

```
subroutine FOO_LOWERBOUNDS(p1, p2);
    outargs p1, p2;

    p1 = <lower bound for P1>;
    p2 = <lower bound for P2>;
endsub;
```

***dist*_PARMINIT**

defines a subroutine that returns the initial values for the distribution's parameters given an empirical distribution function (EDF) estimate.

- *Type*: Subroutine
- *Required*: NO
- *Number of arguments*: $m + 4$, where m is the number of distribution parameters
- *Sequence and type of arguments*:

  dim Input numeric value that contains the dimension of the x, nx, and F array arguments.

  x{*} Input numeric array of dimension *dim* that contains values of the random variables at which the EDF estimate is available. It can be assumed that x contains values in an increasing order. In other words, if $i < j$, then x[i] < x[j].

  nx{*} Input numeric array of dimension *dim*. Each nx[i] contains the number of observations in the original data that have the value x[i].

  F{*} Input numeric array of dimension *dim*. Each F[i] contains the EDF estimate for x[i]. This estimate is computed by the HPSEVERITY procedure based on the options that you specify in the LOSS statement and the EMPIRICALCDF= option.

  Ftype Input numeric value that contains the type of the EDF estimate that is stored in x and F. See the section "Supplying EDF Estimates to Functions and Subroutines" on page 1265 for definition of types.

  p1 Output argument that returns the initial value of the first parameter. You must specify this in the OUTARGS statement inside the subroutine's definition.

p2 Output argument that returns the initial value of the second parameter. You must specify this in the OUTARGS statement inside the subroutine's definition.
...
p*m* Output argument that returns the initial value of the *m*th parameter. You must specify this in the OUTARGS statement inside the subroutine's definition.

- *Return value*: The results, initial values of the parameters, should be returned in the parameter arguments of the subroutine.

Here is a sample structure of the subroutine for a distribution named 'FOO':

```
subroutine FOO_PARMINIT(dim, x{*}, nx{*}, F{*}, Ftype, p1, p2);
    outargs p1, p2;

    /* Code to initialize values of P1 and P2 by using
       dim, x, nx, and F */

    p1 = <initial value for p1>;
    p2 = <initial value for p2>;
endsub;
```

*dist*_**PDF**

defines a function that returns the value of the probability density function (PDF) of the distribution at the specified values of the random variable and distribution parameters.

- *Type*: Function
- *Required*: YES
- *Number of arguments*: $m + 1$, where m is the number of distribution parameters
- *Sequence and type of arguments*:

  x Numeric value of the random variable at which the PDF value should be evaluated

  p1 Numeric value of the first parameter

  p2 Numeric value of the second parameter
  ...

  p*m* Numeric value of the *m*th parameter

- *Return value*: Numeric value that contains the PDF value $f(x; p_1, p_2, \ldots, p_m)$

If you want to consider this distribution as a candidate distribution when you estimate a response variable model with regression effects, then the first parameter of this distribution must be a scale parameter or log-transformed scale parameter. In other words, if the distribution has a scale parameter, then the following equation must be satisfied:

$$f(x; p_1, p_2, \ldots, p_m) = \frac{1}{p_1} f(\frac{x}{p_1}; 1, p_2, \ldots, p_m)$$

If the distribution has a log-transformed scale parameter, then the following equation must be satisfied:

$$f(x; p_1, p_2, \ldots, p_m) = \frac{1}{\exp(p_1)} f(\frac{x}{\exp(p_1)}; 0, p_2, \ldots, p_m)$$

Here is a sample structure of the function for a distribution named 'FOO':

```
function FOO_PDF(x, P1, P2);
    /* Code to compute PDF by using x, P1, and P2 */

    f = <computed PDF>;
    return (f);
endsub;
```

*dist*_QUANTILE

defines a function that returns the quantile of the distribution at the specified value of the CDF for the specified values of distribution parameters.

- *Type*: Function
- *Required*: NO
- *Number of arguments*: $m + 1$, where m is the number of distribution parameters
- *Sequence and type of arguments*:

  cdf Numeric value of the cumulative distribution function (CDF) for which the quantile should be evaluated

  p1 Numeric value of the first parameter

  p2 Numeric value of the second parameter

  ...

  pm Numeric value of the mth parameter

- *Return value*: Numeric value that contains the quantile $F^{-1}(\text{cdf}; p_1, p_2, \ldots, p_m)$

Here is a sample structure of the function for a distribution named 'FOO':

```
function FOO_QUANTILE(c, P1, P2);
    /* Code to compute quantile by using c, P1, and P2 */

    Q = <computed quantile>;
    return (Q);
endsub;
```

*dist*_SCALETRANSFORM

defines a function that returns a keyword to identify the transform that needs to be applied to the scale parameter to convert it to the first parameter of the distribution.

If you want to use this distribution for modeling regression effects, then the first parameter of this distribution must be a scale parameter. However, for some distributions, a typical or convenient parameterization might not have a scale parameter, but one of the parameters can be a simple transform of the scale parameter. As an example, consider a typical parameterization of the lognormal distribution with two parameters, location μ and shape σ, for which the PDF is defined as follows:

$$f(x;\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\log(x)-\mu}{\sigma}\right)^2}$$

You can reparameterize this distribution to contain a parameter θ instead of the parameter μ such that $\mu = \log(\theta)$. The parameter θ would then be a scale parameter. However, if you want to specify the distribution in terms of μ and σ (which is a more recognized form of the lognormal distribution) and still allow it as a candidate distribution for estimating regression effects, then instead of writing another distribution with parameters θ and σ, you can simply define the distribution with μ as the first parameter and specify that it is the logarithm of the scale parameter.

- *Type*: Function
- *Required*: NO
- *Number of arguments*: None
- *Sequence and type of arguments*: Not applicable
- *Return value*: Character value that contains one of the following keywords:

LOG	specifies that the first parameter is the logarithm of the scale parameter.
IDENTITY	specifies that the first parameter is a scale parameter without any transformation.

If you do not specify this function, then the IDENTITY transform is assumed.

Here is a sample structure of the function for a distribution named 'FOO':

```
function FOO_SCALETRANSFORM() $8;
    length xform $8;
    xform = "IDENTITY";
    return (xform);
endsub;
```

*dist*_**SDF**

defines a function that returns the value of the survival distribution function (SDF) of the distribution at the specified values of the random variable and distribution parameters.

- *Type*: Function
- *Required*: NO
- *Number of arguments*: $m + 1$, where m is the number of distribution parameters
- *Sequence and type of arguments*:

x	Numeric value of the random variable at which the SDF value should be evaluated
p1	Numeric value of the first parameter
p2	Numeric value of the second parameter
...	
p*m*	Numeric value of the *m*th parameter

- *Return value*: Numeric value that contains the SDF value $S(x; p_1, p_2, \ldots, p_m)$

If you want to consider this distribution as a candidate distribution when estimating a response variable model with regression effects, then the first parameter of this distribution must be a scale parameter

or log-transformed scale parameter. In other words, if the distribution has a scale parameter, then the following equation must be satisfied:

$$S(x; p_1, p_2, \ldots, p_m) = S(\frac{x}{p_1}; 1, p_2, \ldots, p_m)$$

If the distribution has a log-transformed scale parameter, then the following equation must be satisfied:

$$S(x; p_1, p_2, \ldots, p_m) = S(\frac{x}{\exp(p_1)}; 0, p_2, \ldots, p_m)$$

Here is a sample structure of the function for a distribution named 'FOO':

```
function FOO_SDF(x, P1, P2);
    /* Code to compute SDF by using x, P1, and P2 */

    S = <computed SDF>;
    return (S);
endsub;
```

*dist*_UPPERBOUNDS

defines a subroutine that returns upper bounds for the parameters of the distribution. If this subroutine is not defined for a given distribution, then the HPSEVERITY procedure assumes that there is no upper bound for any of the parameters. If an upper bound of u_i is returned for a parameter p_i, then the HPSEVERITY procedure assumes that $p_i < u_i$ (strict inequality). If a missing value is returned for some parameter, then the HPSEVERITY procedure assumes that there is no upper bound for that parameter (equivalent to an upper bound of ∞).

- *Type*: Subroutine
- *Required*: NO
- *Number of arguments*: *m*, where *m* is the number of distribution parameters
- *Sequence and type of arguments*:

  p1 — Output argument that returns the upper bound on the first parameter. You must specify this in the OUTARGS statement inside the subroutine's definition.

  p2 — Output argument that returns the upper bound on the second parameter. You must specify this in the OUTARGS statement inside the subroutine's definition.

  ...

  p*m* — Output argument that returns the upper bound on the *m*th parameter. You must specify this in the OUTARGS statement inside the subroutine's definition.

- *Return value*: The results, upper bounds on parameter values, should be returned in the parameter arguments of the subroutine.

Here is a sample structure of the subroutine for a distribution named 'FOO':

```
subroutine FOO_UPPERBOUNDS(p1, p2);
    outargs p1, p2;

    p1 = <upper bound for P1>;
    p2 = <upper bound for P2>;
endsub;
```

*dist_core*__GRADIENT__

defines a subroutine that returns the gradient vector of the specified *core* function of the distribution at the specified values of the random variable and distribution parameters. The *core* keyword can be PDF, CDF, SDF, LOGPDF, LOGCDF, or LOGSDF.

- *Type*: Subroutine
- *Required*: NO
- *Number of arguments*: $m + 2$, where m is the number of distribution parameters
- *Sequence and type of arguments*:

x	Numeric value of the random variable at which the gradient should be evaluated
p1	Numeric value of the first parameter
p2	Numeric value of the second parameter
	...
p*m*	Numeric value of the *m*th parameter
grad{*}	Output numeric array of size m that contains the gradient vector evaluated at the specified values. If h denotes the value of the *core* function, then the expected order of the values in the array is as follows: $\frac{\partial h}{\partial p_1}\ \frac{\partial h}{\partial p_2} \cdots \frac{\partial h}{\partial p_m}$

- *Return value*: Numeric array that contains the gradient evaluated at x for the parameter values $(p_1, p_2, \ldots, p_m)$

Here is a sample structure of the function for the core function CDF of a distribution named 'FOO':

```
subroutine FOO_CDFGRADIENT(x, P1, P2, grad{*});
    outargs grad;

    /* Code to compute gradient by using x, P1, and P2 */
    grad[1] = <partial derivative of CDF w.r.t. P1
               evaluated at x, P1, P2>;
    grad[2] = <partial derivative of CDF w.r.t. P2
               evaluated at x, P1, P2>;
endsub;
```

*dist_core*__HESSIAN__

defines a subroutine that returns the Hessian matrix of the specified *core* function of the distribution at the specified values of the random variable and distribution parameters. The *core* keyword can be PDF, CDF, SDF, LOGPDF, LOGCDF, or LOGSDF.

- *Type*: Subroutine
- *Required*: NO
- *Number of arguments*: $m + 2$, where m is the number of distribution parameters
- *Sequence and type of arguments*:

x	Numeric value of the random variable at which the Hessian matrix should be evaluated
p1	Numeric value of the first parameter

p2 Numeric value of the second parameter

...

p*m* Numeric value of the *m*th parameter

hess{*} Output numeric array of size $m(m+1)/2$ that contains the lower triangular portion of the Hessian matrix in a packed vector form, evaluated at the specified values. If h denotes the value of the *core* function, then the expected order of the values in the array is as follows: $\frac{\partial^2 h}{\partial p_1^2} \mid \frac{\partial^2 h}{\partial p_1 \partial p_2} \; \frac{\partial^2 h}{\partial p_2^2} \mid \cdots \mid \frac{\partial^2 h}{\partial p_1 \partial p_m} \; \frac{\partial^2 h}{\partial p_2 \partial p_m} \cdots \frac{\partial^2 h}{\partial p_m^2}$

- *Return value*: Numeric array that contains the lower triangular portion of the Hessian matrix evaluated at x for the parameter values $(p_1, p_2, \ldots, p_m)$

Here is a sample structure of the subroutine for the core function LOGSDF of a distribution named 'FOO':

```
subroutine FOO_LOGSDFHESSIAN(x, P1, P2, hess{*});
    outargs hess;

    /* Code to compute Hessian by using x, P1, and P2 */
    hess[1] = <second order partial derivative of LOGSDF
               w.r.t. P1 evaluated at x, P1, P2>;
    hess[2] = <second order partial derivative of LOGSDF
               w.r.t. P1 and P2 evaluated at x, P1, P2>;
    hess[3] = <second order partial derivative of LOGSDF
               w.r.t. P2 evaluated at x, P1, P2>;
endsub;
```

Predefined Utility Functions

The following predefined utility functions are provided with the HPSEVERITY procedure and are available in the Sashelp.Svrtdist library:

SVRTUTIL_EDF:

This function computes the empirical distribution function (EDF) estimate at the specified value of the random variable given the EDF estimate for a sample.

- *Type*: Function
- *Signature*: SVRTUTIL_EDF(y, n, x{*}, F{*}, Ftype)
- *Argument Description*:

  y Value of the random variable at which the EDF estimate is desired.

  n Dimension of the *x* and *F* input arrays.

  x{*} Input numeric array of dimension *n* that contains values of the random variable observed in the sample. These values are sorted in nondecreasing order.

  F{*} Input numeric array of dimension *n* in which each F[*i*] contains the EDF estimate for x[*i*]. These values must be sorted in nondecreasing order.

  Ftype Type of the empirical estimate that is stored in the *x* and *F* arrays. See the section "Supplying EDF Estimates to Functions and Subroutines" on page 1265 for definition of types.

- *Return value*: The EDF estimate at y.

The type of the sample EDF estimate determines how the EDF estimate at y is computed. For more information, see the section "Supplying EDF Estimates to Functions and Subroutines" on page 1265.

SVRTUTIL_EMPLIMMOMENT:

This function computes the empirical estimate of the limited moment of specified order for the specified upper limit, given the EDF estimate for a sample.

- *Type*: Function
- *Signature*: SVRTUTIL_EMPLIMMOMENT(k, u, n, x{*}, F{*}, Ftype)
- *Argument Description*:

  k — Order of the desired empirical limited moment.

  u — Upper limit on the value of the random variable to be used in the computation of the desired empirical limited moment.

  n — Dimension of the x and F input arrays.

  x{*} — Input numeric array of dimension n that contains values of the random variable observed in the sample. These values are sorted in nondecreasing order.

  F{*} — Input numeric array of dimension n in which each F[i] contains the EDF estimate for x[i]. These values must be sorted in nondecreasing order.

  Ftype — Type of the empirical estimate that is stored in the x and F arrays. See the section "Supplying EDF Estimates to Functions and Subroutines" on page 1265 for definition of types.

- *Return value*: The desired empirical limited moment.

The empirical limited moment is computed by using the empirical estimate of the CDF. If $F_n(x)$ denotes the EDF at x, then the empirical limited moment of order k with upper limit u is defined as

$$E_n[(X \wedge u)^k] = k \int_0^u (1 - F_n(x)) x^{k-1} dx$$

The SVRTUTIL_EMPLIMMOMENT function uses the piecewise linear nature of $F_n(x)$ as described in the section "Supplying EDF Estimates to Functions and Subroutines" on page 1265 to compute the integration.

SVRTUTIL_HILLCUTOFF:

This function computes an estimate of the value where the right tail of a distribution is expected to begin. The function implements the algorithm described in Danielsson et al. 2001. The description of the algorithm uses the following notation:

n — Number of observations in the original sample.

B — Number of bootstrap samples to draw.

m_1 — Size of the bootstrap sample in the first step of the algorithm ($m_1 < n$).

$x_{(i)}^{j,m}$ — ith order statistic of jth bootstrap sample of size m ($1 \leq i \leq m, 1 \leq j \leq B$).

$x_{(i)}$ — ith order statistic of the original sample ($1 \leq i \leq n$).

Given the input sample x and values of B and m_1, the steps of the algorithm are as follows:

1. Take B bootstrap samples of size m_1 from the original sample.
2. Find the integer k_1 that minimizes the bootstrap estimate of the mean squared error:

$$k_1 = \arg \min_{1 \le k < m_1} Q(m_1, k)$$

3. Take B bootstrap samples of size $m_2 = m_1^2/n$ from the original sample.
4. Find the integer k_2 that minimizes the bootstrap estimate of the mean squared error:

$$k_2 = \arg \min_{1 \le k < m_2} Q(m_2, k)$$

5. Compute the integer k_{opt}, which is used for computing the cutoff point:

$$k_{\text{opt}} = \frac{k_1^2}{k_2} \left(\frac{\log(k_1)}{2\log(m_1) - \log(k_1)} \right)^{2-2\log(k_1)/\log(m_1)}$$

6. Set the cutoff point equal to $x_{(k_{\text{opt}}+1)}$.

The bootstrap estimate of the mean squared error is computed as

$$Q(m,k) = \frac{1}{B} \sum_{j=1}^{B} \text{MSE}_j(m,k)$$

The mean squared error of jth bootstrap sample is computed as

$$\text{MSE}_j(m,k) = (M_j(m,k) - 2(\gamma_j(m,k))^2)^2$$

where $M_j(m,k)$ is a control variate proposed by Danielsson et al. 2001,

$$M_j(m,k) = \frac{1}{k} \sum_{i=1}^{k} \left(\log(x^{j,m}_{(m-i+1)}) - \log(x^{j,m}_{(m-k)}) \right)^2$$

and $\gamma_j(m,k)$ is the Hill's estimator of the tail index (Hill 1975),

$$\gamma_j(m,k) = \frac{1}{k} \sum_{i=1}^{k} \log(x^{j,m}_{(m-i+1)}) - \log(x^{j,m}_{(m-k)})$$

This algorithm has two tuning parameters, B and m_1. The number of bootstrap samples B is chosen based on the availability of computational resources. The optimal value of m_1 is chosen such that the following ratio, $R(m_1)$, is minimized:

$$R(m_1) = \frac{(Q(m_1, k_1))^2}{Q(m_2, k_2)}$$

The SVRTUTIL_HILLCUTOFF utility function implements the preceding algorithm. It uses the grid search method to compute the optimal value of m_1.

- *Type*: Function
- *Signature*: SVRTUTIL_HILLCUTOFF(n, x{*}, b, s, status)

- *Argument Description*:

  n — Dimension of the array *x*.

  x{*} — Input numeric array of dimension *n* that contains the sample.

  b — Number of bootstrap samples used to estimate the mean squared error. If *b* is less than 10, then a default value of 50 is used.

  s — Approximate number of steps used to search the optimal value of m_1 in the range $[n^{0.75}, n-1]$. If *s* is less than or equal to 1, then a default value of 10 is used.

  status — Output argument that contains the status of the algorithm. If the algorithm succeeds in computing a valid cutoff point, then *status* is set to 0. If the algorithm fails, then *status* is set to 1.

- *Return value*: The cutoff value where the right tail is estimated to start. If the size of the input sample is inadequate ($n \leq 5$), then a missing value is returned and *status* is set to a missing value. If the algorithm fails to estimate a valid cutoff value (*status* = 1), then the fifth largest value in the input sample is returned.

SVRTUTIL_PERCENTILE:

This function computes the specified empirical percentile given the EDF estimates.

- *Type*: Function
- *Signature*: SVRTUTIL_PERCENTILE(p, n, x{*}, F{*}, Ftype)
- *Argument Description*:

  p — Desired percentile. The value must be in the interval (0,1). The function returns the 100*p*th percentile.

  n — Dimension of the *x* and *F* input arrays.

  x{*} — Input numeric array of dimension *n* that contains values of the random variable observed in the sample. These values are sorted in nondecreasing order.

  F{*} — Input numeric array of dimension *n* in which each F[*i*] contains the EDF estimate for x[*i*]. These values must be sorted in nondecreasing order.

  Ftype — Type of the empirical estimate that is stored in the *x* and *F* arrays. See the section "Supplying EDF Estimates to Functions and Subroutines" on page 1265 for definition of types.

- *Return value*: The 100*p*th percentile of the input sample.

The method used to compute the percentile depends on the type of the EDF estimate (Ftype argument).

Ftype = 1 — Smoothed empirical estimates are computed using the method described in Klugman, Panjer, and Willmot (1998). Let $\lfloor x \rfloor$ denote the greatest integer less than or equal to x. Define $g = \lfloor p(n+1) \rfloor$ and $h = p(n+1) - g$. Then the empirical percentile $\hat{\pi}_p$ is defined as

$$\hat{\pi}_p = (1-h)x[g] + hx[g+1]$$

This method does not work if $p < 1/(n+1)$ or $p > n/(n+1)$. If $p < 1/(n+1)$, then the function returns $\hat{\pi}_p = x[1]/2$, which assumes that the EDF is 0 in the interval $[0, x[1])$. If $p > n/(n+1)$, then $\hat{\pi}_p = x[n]$.

Ftype = 2 If $p < F[1]$, then $\hat{\pi}_p = x[1]/2$, which assumes that the EDF is 0 in the interval $[0, x[1])$. If $|p - F[i]| < \epsilon$ for some value of i and $i < n$, then $\hat{\pi}_p$ is computed as

$$\hat{\pi}_p = \frac{x[i] + x[i+1]}{2}$$

where ϵ is a machine-precision constant as returned by the SAS function CONSTANT('MACEPS'). If $F[i-1] < p < F[i]$, then $\hat{\pi}_p$ is computed as

$$\hat{\pi}_p = x[i]$$

If $p \geq F[n]$, then $\hat{\pi}_p = x[n]$.

Ftype = 3 If $p < F[1]$, then $\hat{\pi}_p = x[1]/2$, which assumes that the EDF is 0 in the interval $[0, x[1])$. If $|p - F[i]| < \epsilon$ for some value of i and $i < n$, then $\hat{\pi}_p$ is computed as

$$\hat{\pi}_p = \frac{x[i] + x[i+1]}{2}$$

where ϵ is a machine-precision constant as returned by the SAS function CONSTANT('MACEPS'). If $F[i-1] < p < F[i]$, then $\hat{\pi}_p$ is computed as

$$\hat{\pi}_p = x[i-1] + (p - F[i-1])\frac{x[i] - x[i-1]}{F[i] - F[i-1]}$$

If $p \geq F[n]$, then $\hat{\pi}_p = x[n]$.

SVRTUTIL_RAWMOMENTS:

This subroutine computes the raw moments of a sample.

- *Type*: Subroutine
- *Signature*: SVRTUTIL_RAWMOMENTS(n, x{*}, nx{*}, nRaw, raw{*})
- *Argument Description*:

  n Dimension of the *x* and *nx* input arrays.

  x{*} Input numeric array of dimension *n* that contains distinct values of the random variable that are observed in the sample.

  nx{*} Input numeric array of dimension *n* in which each *nx*[*i*] contains the number of observations in the sample that have the value x[*i*].

  nRaw Desired number of raw moments. The output array *raw* contains the first *nRaw* raw moments.

  raw{*} Output array of raw moments. The *k*th element in the array (raw{*k*}) contains the *k*th raw moment, where $1 \leq k \leq$ nRaw.

- *Return value*: Numeric array *raw* that contains the first *nRaw* raw moments. The array contains missing values if the sample has no observations (that is, if all the values in the *nx* array add up to zero).

SVRTUTIL_SORT:

This function sorts the given array of numeric values in an ascending or descending order.

- *Type*: Subroutine
- *Signature*: SVRTUTIL_SORT(n, x{*}, flag)

- *Argument Description*:

  n — Dimension of the input array *x*.

  x{*} — Numeric array that contains the values to be sorted at input. The subroutine uses the same array to return the sorted values.

  flag — A numeric value that controls the sort order. If *flag* is 0, then the values are sorted in an ascending order. If *flag* has any value other than 0, then the values are sorted in descending order.

- *Return value*: Numeric array *x*, which is sorted in place (that is, the sorted array is stored in the same storage area occupied by the input array *x*).

You can use the following predefined functions when you use the FCMP procedure to define functions and subroutines. They are summarized here for your information. For more information, see the FCMP procedure documentation in *Base SAS Procedures Guide*.

INVCDF:
This function computes the quantile from any continuous probability distribution by numerically inverting the CDF of that distribution. You need to specify the CDF function of the distribution, the values of its parameters, and the cumulative probability to compute the quantile.

LIMMOMENT:
This function computes the limited moment of order k with upper limit u for any continuous probability distribution. The limited moment is defined as

$$\begin{aligned} E[(X \wedge u)^k] &= \int_0^u x^k f(x)dx + \int_u^\infty u^k f(x)dx \\ &= \int_0^u x^k f(x)dx + u^k(1 - F(u)) \end{aligned}$$

where $f(x)$ and $F(x)$ denote the PDF and the CDF of the distribution, respectively. The LIMMOMENT function uses the following alternate definition, which can be derived using integration-by-parts:

$$E[(X \wedge u)^k] = k \int_0^u (1 - F(x))x^{k-1}dx$$

You need to specify the CDF function of the distribution, the values of its parameters, and the values of k and u to compute the limited moment.

Scoring Functions

Scoring refers to the act of evaluating a distribution function, such as LOGPDF, SDF, or QUANTILE, on an observation by using the fitted parameter estimates of that distribution. You can do scoring in a DATA step by using the OUTEST= data set that you create with PROC HPSEVERITY. However, that approach requires some cumbersome programming. In order to simplify the scoring process, you can specify that PROC HPSEVERITY create scoring functions for each fitted distribution.

As an example, assume that you have fitted the Pareto distribution by using PROC HPSEVERITY and that it converges. Further assume that you want to use the fitted distribution to evaluate the probability of observing a loss value greater than some set of regulatory limits {L} that are encoded in a data set. You can simplify this scoring process as follows. First, in the PROC HPSEVERITY step that fits your distributions, you create the scoring functions library by specifying the OUTSCORELIB statement as illustrated in the following steps:

```
proc hpseverity data=input;
   loss lossclaim;
   dist pareto;
   outscorelib outlib=sasuser.fitdist;
run;
```

Upon successful completion, if the Pareto distribution model has converged, then the Sasuser.Fitdist library contains the *SEV_SDF* scoring function in addition to other scoring functions, such as *SEV_PDF*, *SEV_LOGPDF*, and so on. Further, PROC HPSEVERITY also sets the CMPLIB system option to include the Sasuser.Fitdist library. If the set of limits {L} is recorded in the variable Limit in the scoring data set Work.Limits, then you can submit the following DATA step to compute the probability of seeing a loss greater than each limit:

```
data prob;
   set work.limits;
   exceedance_probability = sev_sdf(limit);
run;
```

Without the use of scoring functions, you can still perform this scoring task, but the DATA step that you need to write to accomplish it becomes more complicated and less flexible. For example, you would need to read the parameter estimates from some output created by PROC HPSEVERITY. To do that, you would need to know the parameter names, which are different for different distributions; this in turn would require you to write a specific DATA step for each distribution or to write a SAS macro. With the use of scoring functions, you can accomplish that task much more easily.

If you fit multiple distributions, then you can specify the COMMONPACKAGE option in the OUTSCORELIB statement as follows:

```
proc hpseverity data=input;
   loss lossclaim;
   dist exp pareto weibull;
   outscorelib outlib=sasuser.fitdist commonpackage;
run;
```

The preceding step creates scoring functions such as *SEV_SDF_Exp*, *SEV_SDF_Pareto*, and *SEV_SDF_Weibull*. You can use them to compare the probabilities of exceeding the limit for different distributions by using the following DATA step:

```
data prob;
   set work.limits;
   exceedance_exp = sev_sdf_exp(limit);
   exceedance_pareto = sev_sdf_pareto(limit);
   exceedance_weibull = sev_sdf_weibull(limit);
run;
```

Formal Description

PROC HPSEVERITY creates a scoring function for each distribution function. A distribution function is defined as any function named *dist_suffix*, where *dist* is the name of a distribution that you specify in the DIST statement and the function's signature is identical to the signature of the required distribution function such as *dist*_CDF or *dist*_LOGCDF. For example, for the function 'FOO_BAR' to be a distribution function, you must specify the distribution 'FOO' in the DIST statement and you must define 'FOO_BAR' in the following manner if the distribution 'FOO' has parameters named 'P1' and 'P2':

```
function FOO_BAR(y, P1, P2);
   /* Code to compute BAR by using y, P1, and P2 */
   R = <computed BAR>;
   return (R);
endsub;
```

For more information about the signature that defines a distribution function, see the description of the *dist*_CDF function in the section "Defining a Severity Distribution Model with the FCMP Procedure" on page 1273.

The name and package of the scoring function of a distribution function depend on whether you specify the COMMONPACKAGE option in the OUTSCORELIB statement.

When you do not specify the COMMONPACKAGE option, the scoring function that corresponds to the distribution function *dist_suffix* is named *SEV_suffix*, where *SEV_* is the standard prefix of all scoring functions. The scoring function is created in a package named *dist*. Each scoring function accepts only one argument, the value of the loss variable, and returns the same value as the value returned by the corresponding distribution function for the final estimates of the distribution's parameters. For example, for the preceding 'FOO_BAR' distribution function, the scoring function named 'SEV_BAR' is created in the package named 'FOO' and 'SEV_BAR' has the following signature:

```
function SEV_BAR(y);
   /* returns value of FOO_BAR for the supplied value
      of y and fitted values of P1, P2 */
endsub;
```

If you specify the COMMONPACKAGE option in the OUTSCORELIB statement, then the scoring function that corresponds to the distribution function *dist_suffix* is named *SEV_suffix_dist*, where *SEV_* is the standard prefix of all scoring functions. The scoring function is created in a package named *sevfit*. For example, for the preceding 'FOO_BAR' distribution function, if you specify the COMMONPACKAGE option, the scoring function named 'SEV_BAR_FOO' is created in the *sevfit* package and 'SEV_BAR_FOO' has the following signature:

```
function SEV_BAR_FOO(y);
   /* returns value of FOO_BAR for the supplied value
      of y and fitted values of P1, P2 */
endsub;
```

Scoring Functions for the Scale Regression Model

If you use the SCALEMODEL statement to specify a scale regression model, then PROC HPSEVERITY generates the scoring functions when you specify only singleton continuous effects. If you specify interaction or classification effects, then scoring functions are not generated.

For a scale regression model, the estimate of the scale parameter or the log-transformed scale parameter of the distribution depends on the values of the regressors. So PROC HPSEVERITY creates a scoring function that has the following signature, where *x{*}* represents the array of regressors:

```
function SEV_BAR(y, x{*});
   /* returns value of FOO_BAR for the supplied value of x and fitted values of P1, P2 */
endsub;
```

As an illustration of using this form, assume that you submit the following PROC HPSEVERITY step to create the scoring library Sasuser.Scalescore:

```
proc hpseverity data=input;
   loss lossclaim;
   scalemodel x1-x3;
   dist pareto;
   outscorelib outlib=sasuser.scalescore;
run;
```

Your scoring data set must contain all the regressors that you specify in the SCALEMODEL statement. You can submit the following DATA step to score observations by using the scale regression model:

```
data prob;
   array regvals{*} x1-x3;
   set work.limits;
   exceedance_probability = sev_sdf(limit, regvals);
run;
```

PROC HPSEVERITY creates two utility functions, SEV_NUMREG and SEV_REGNAME, in the OUTLIB= library that return the number of regressors and name of a given regressor, respectively. They are described in detail in the next section. These utility functions are useful when you do not have easy access to the regressor names in the SCALEMODEL statement. You can use the utility functions as follows:

```
data prob;
   array regvals{10} _temporary_;
   set work.limits;
   do i = 1 to sev_numreg();
      regvals(i) = input(vvaluex(sev_regname(i)), best12.);
   end;
   exceedance_probability = sev_sdf(limit, regvals);
run;
```

The dimension of the regressor values array that you supply to the scoring function must be equal to $K + L$, where K is the number of regressors that you specify in the SCALEMODEL statement irrespective of whether PROC HPSEVERITY deems any of those regressors to be redundant. L is 1 if you specify an OFFSET= variable in the SCALEMODEL statement, and 0 otherwise.

Utility Functions and Subroutines in the OUTLIB= Library

In addition to creating the scoring functions for all distribution functions, PROC HPSEVERITY creates the following utility functions and subroutines in the OUTLIB= library.

SEV_NUMPARM | SEV_NUMPARM_*dist*
is a function that returns the number of distribution parameters and has the following signature:

- *Type*: Function
- *Number of arguments*: 0
- *Sequence and type of arguments*: Not applicable
- *Return value*: Numeric value that contains the number of distribution parameters

If you do not specify the COMMONPACKAGE option in the OUTSCORELIB statement, then a function named SEV_NUMPARM is created in the package of each distribution. Here is a sample structure of the code that PROC HPSEVERITY uses to define the function:

```
function SEV_NUMPARM();
    n = <number of distribution parameters>;
    return (n);
endsub;
```

If you specify the COMMONPACKAGE option in the OUTSCORELIB statement, then for each distribution *dist*, the function named SEV_NUMPARM_*dist* is created in the *sevfit* package. SEV_NUMPARM_*dist* has the same structure as the SEV_NUMPARM function that is described previously.

SEV_PARMEST | SEV_PARMEST_*dist*
is a subroutine that returns the estimate and standard error of a specified distribution parameter and has the following signature:

- *Type*: Subroutine
- *Number of arguments*: 3
- *Sequence and type of arguments*:

  index specifies the numeric value of the index of the distribution parameter for which you want the information. The value of *index* must be in the interval [1,*m*], where *m* is the number of parameters in the distribution to which this subroutine belongs.

  est specifies the output argument that returns the estimate of the requested parameter.

  stderr specifies the output argument that returns the standard error of the requested parameter.

- *Return value*: Estimate and standard error of the requested distribution parameter that are returned in the output arguments *est* and *stderr*, respectively

If you do not specify the COMMONPACKAGE option in the OUTSCORELIB statement, then a subroutine named SEV_PARMEST is created in the package of each distribution. Here is a sample structure of the code that PROC HPSEVERITY uses to define the subroutine:

```
subroutine SEV_PARMEST(index, est, stderr);
    outargs est, stderr;
    est = <value of the estimate for the distribution parameter
           at position 'index'>;
    stderr =  <value of the standard error for distribution parameter
               at position 'index'>;
endsub;
```

If you specify the COMMONPACKAGE option in the OUTSCORELIB statement, then for each distribution *dist*, the subroutine named SEV_PARMEST_*dist* is created in the *sevfit* package. SEV_PARMEST_*dist* has the same structure as the SEV_PARMEST subroutine that is described previously.

If you use the SCALEMODEL statement to specify a scale regression model, and if you specify only singleton continuous effects, then for *index*=1, the returned estimates are of θ_0, the base value of the scale parameter, or $\log(\theta_0)$ if the distribution has a log-scale parameter. For more information about θ_0, see the section "Estimating Regression Effects" on page 1245.

SEV_PARMNAME | SEV_PARMNAME_*dist*

is a function that returns the name of a specified distribution parameter and has the following signature:

- *Type*: Function
- *Number of arguments*: 1
- *Sequence and type of arguments*:

  index specifies the numeric value of the index of the distribution parameter for which you want the information. The value of *index* must be in the interval [1,*m*], where *m* is the number of parameters in the distribution to which this function belongs.

- *Return value*: Character value that contains the name of the distribution parameter that appears at the position *index* in the distribution's definition

If you do not specify the COMMONPACKAGE option in the OUTSCORELIB statement, then a function named SEV_PARMNAME is created in the package of each distribution.

Here is a sample structure of the code that PROC HPSEVERITY uses to define the function:

```
function SEV_PARMNAME(index) $32;
    name = <name of the distribution parameter at position 'index'>;
    return (name);
endsub;
```

If you specify the COMMONPACKAGE option in the OUTSCORELIB statement, then for each distribution *dist*, a function named SEV_PARMNAME_*dist* is created in the *sevfit* package. SEV_PARMNAME_*dist* has the same structure as the SEV_PARMNAME function that is described previously.

If you use the SCALEMODEL statement to specify a scale regression model, and if you specify only singleton continuous effects, then the following helper functions and subroutines are also created in the OUTLIB= library.

SEV_NUMREG
is a function that returns the number of regressors and has the following signature:

- *Type*: Function
- *Number of arguments*: 0
- *Sequence and type of arguments*: Not applicable
- *Return value*: Numeric value that contains the number of regressors that you specify in the SCALEMODEL statement. If you specify an OFFSET= variable in the SCALEMODEL statement, then the returned value is equal to 1 plus the number of regressors that you specify in the SCALEMODEL statement.

Here is a sample structure of the code that PROC HPSEVERITY uses to define the function:

```
function SEV_NUMREG();
    m = <number of regressors>;
    if (<offset variable is specified>) then m = m + 1;
    return (m);
endsub;
```

This function does not depend on any distribution, so it is always created in the *sevfit* package.

SEV_REGEST | SEV_REGEST_*dist*
is a subroutine that returns the estimate and standard error of a specified regression parameter and has the following signature:

- *Type*: Subroutine
- *Number of arguments*: 3
- *Sequence and type of arguments*:

  index specifies the numeric value of the index of the regression parameter for which you want the information. The value of *index* must be in the interval [1,*K*], where *K* is the number of regressors as returned by the SEV_NUMREG function. If you specify an OFFSET= variable in the SCALEMODEL statement, then an *index* value of *K* corresponds to the offset variable.

  est specifies the output argument that returns the estimate of the requested regression parameter.

  stderr specifies the output argument that returns the standard error of the requested regression parameter.

- *Return value*: Estimate and standard error of the requested regression parameter that are returned in the output arguments *est* and *stderr*, respectively

If you do not specify the COMMONPACKAGE option in the OUTSCORELIB statement, then a subroutine named SEV_REGEST is created in the package of each distribution. Here is a sample structure of the code that PROC HPSEVERITY uses to define the subroutine:

```
subroutine SEV_REGEST(index, est, stderr);
    outargs est, stderr;
    est = <value of the estimate for the regression parameter
           at position 'index'>;
    stderr =  <value of the standard error for regression parameter
               at position 'index'>;
endsub;
```

If you specify the COMMONPACKAGE option in the OUTSCORELIB statement, then for each distribution *dist*, the subroutine named SEV_REGEST_*dist* is created in the *sevfit* package. SEV_REGEST_*dist* has the same structure as the SEV_REGEST subroutine that is described previously.

If the regressor that corresponds to the specified *index* value is a redundant regressor, the returned values of both *est* and *stderr* are equal to the special missing value of .R. If you specify an OFFSET= variable in the SCALEMODEL statement and if the *index* value corresponds to the offset variable — that is, it is equal to the value that the SEV_NUMREG function returns — then the returned value of *est* is equal to 1 and the returned value of *stderr* is equal to the special missing value of .F.

SEV_REGNAME

is a function that returns the name of a specified regressor and has the following signature:

- *Type*: Function
- *Number of arguments*: 1
- *Sequence and type of arguments*:

  index specifies the numeric value of the index of the regressor for which you want the name. The value of *index* must be in the interval [1,*K*], where *K* is the number of regressors as returned by the SEV_NUMREG function. If you specify an OFFSET= variable in the SCALEMODEL statement, then an *index* value of *K* corresponds to the offset variable.

- *Return value*: Character value that contains the name of the regressor that appears at the position *index* in the SCALEMODEL statement. If you specify an OFFSET= variable in the SCALEMODEL statement, then for an *index* value of *K*, the returned value contains the name of the offset variable.

Here is a sample structure of the code that PROC HPSEVERITY uses to define the function:

```
function SEV_REGNAME(index) $32;
    name = <name of regressor at position 'index'>;
    return (name);
endsub;
```

This function does not depend on any distribution, so it is always created in the *sevfit* package.

Custom Objective Functions

You can use a series of programming statements that use variables in the DATA= data set to assign a value to an objective function symbol. You must specify the objective function symbol by using the OBJECTIVE= option in the PROC HPSEVERITY statement.

The objective function can be programmed such that it is applicable to any distribution that is used in the model. For that purpose, PROC HPSEVERITY recognizes the following *keyword* functions in the programming statements:

PDF(x) returns the probability density function (PDF) of a distribution evaluated at the current value of a data set variable x.

CDF(x) returns the cumulative distribution function (CDF) of a distribution evaluated at the current value of a data set variable x.

SDF(x) returns the survival distribution function (SDF) of a distribution evaluated at the current value of a data set variable x.

LOGPDF(x) returns the natural logarithm of the PDF of a distribution evaluated at the current value of a data set variable x.

LOGCDF(x) returns the natural logarithm of the CDF of a distribution evaluated at the current value of a data set variable x.

LOGSDF(x) returns the natural logarithm of the SDF of a distribution evaluated at the current value of a data set variable x.

EDF(x) returns the empirical distribution function (EDF) estimate evaluated at the current value of a data set variable x. Internally, PROC HPSEVERITY computes the estimate using the SVRTUTIL_EDF function as described in the section "Predefined Utility Functions" on page 1285. The EDF estimate that is required by the SVRTUTIL_EDF function is computed by using the response variable values in the current BY group or in the entire input data set if you do not specify the BY statement.

EMPLIMMOMENT(k, u)
returns the empirical limited moment of order *k* evaluated at the current value of a data set variable u that represents the upper limit of the limited moment. The order *k* can also be a data set variable. Internally, PROC HPSEVERITY computes the moment using the SVRTUTIL_EMPLIMMOMENT function as described in the section "Predefined Utility Functions" on page 1285. The EDF estimate that is required by the SVRTUTIL_EMPLIMMOMENT function is computed by using the response variable values in the current BY group or in the entire input data set if you do not specify the BY statement.

LIMMOMENT(k, u)
returns the limited moment of order *k* evaluated at the current value of a data set variable u that represents the upper limit of the limited moment. The order *k* can be a data set variable or a constant. Internally, for each candidate distribution, PROC HPSEVERITY computes the moment using the LIMMOMENT function as described in the section "Predefined Utility Functions" on page 1285.

All the preceding functions are right-hand side functions. They act as placeholders for distribution-specific functions, with the exception of _EDF_ and _EMPLIMMOMENT_ functions.

As an example, let the data set Work.Test contain a response variable Y and a left-truncation threshold variable T. The following statements use the values in this data set to fit a model with distribution D such that the parameters of the model minimize the value of the objective function symbol MYOBJ:

```
options cmplib=(work.mydist);
proc hpseverity data=work.test objective=myobj;
    loss y / lt=t;

    myobj = -_LOGPDF_(y);
    if (not(missing(t))) then
        myobj = myobj + log(1-_CDF_(t));

    dist d;
run;
```

The symbol MYOBJ is designated as an objective function symbol by using the OBJECTIVE= option in the PROC HPSEVERITY statement. The response variable Y and left-truncation variable T are specified in the LOSS statement. The distribution D is specified in the DIST statement. The remaining statements constitute a program that computes the value of the MYOBJ symbol.

Let the distribution D have parameters P1 and P2. In order to estimate the model for this distribution, PROC HPSEVERITY internally converts the generic program to the following program specific to distribution D:

```
myobj = -D_LOGPDF(y, p1, p2);
if (not(missing(t))) then
    myobj = myobj + log(1-D_CDF(t, p1, p2));
```

Note that the generic keyword functions _LOGPDF_ and _CDF_ have been replaced with distribution-specific functions D_LOGPDF and D_CDF, respectively, with appropriate distribution parameters. The D_LOGPDF and D_CDF functions must have been defined previously and are assumed to be available in the Work.Mydist library that you specify in the CMPLIB= option.

The program is executed for each observation in Work.Test to compute the value of MYOBJ by using the values of variables Y and T in that observation and internally computed values of the model parameters P1 and P2. The values of MYOBJ are then added over all the observations of the data set or over all the observations of the current BY group if you specify the BY statement. The resulting aggregate value is the value of the objective function, and it is supplied to the optimizer. If the optimizer requires derivatives of the objective function, then PROC HPSEVERITY automatically differentiates MYOBJ with respect to the parameters P1 and P2. The optimizer iterates over various combinations of the values of parameters P1 and P2, each time computing a new value of the objective function and the needed derivatives of it, until it finds a combination that minimizes the objective function.

Note the following points when you define your own program to compute the custom objective function:

- The value of the objective function is always minimized by PROC HPSEVERITY. If you want to maximize the value of a certain objective, then add a statement that assigns the negated value of the maximization objective to the objective function symbol that you specify in the OBJECTIVE= option. Minimization of the negated objective is equivalent to the maximization of the original objective.
- The contributions of individual observations are always added to compute the overall objective function in a given iteration of the optimizer. If you specify the WEIGHT statement, then the contribution of each observation is weighted by multiplying it with the normalized value of the weight variable for that observation.

- If you are fitting multiple distributions in one PROC HPSEVERITY step and use any of the keyword functions in your program, then it is recommended that you do not explicitly use the parameters of any of the specified distributions in your programming statements.
- If you use a specific keyword function in your programming statements, then the corresponding distribution functions must be defined in a library that you specify in the CMPLIB= system option or in Sashelp.Svrtdist, the predefined functions library. In the preceding example, it is assumed that the functions D_LOGPDF and D_CDF are defined in the Work.Mydist library that is specified in the CMPLIB= option.
- You can use most DATA step statements and functions in your program. The DATA step file and the data set I/O statements (for example, INPUT, FILE, SET, and MERGE) are not available. However, some functionality of the PUT statement is supported. See the section "PROC FCMP and DATA Step Differences" in *Base SAS Procedures Guide* for more information. In addition to the differences listed in that section, the following differences exist:
  - Only numeric-valued variables can be used in PROC HPSEVERITY programming statements. This restriction also implies that you cannot use SAS functions or call routines that require character-valued arguments, unless you pass those arguments as constant (literal) strings or characters.
  - You cannot use functions that create lagged versions of a variable in PROC HPSEVERITY programming statements. If you need lagged versions, then you can use a DATA step prior to the PROC HPSEVERITY step to add those versions to the input data set.
- When coding your programming statements, avoid defining variables that begin with an underscore (_), because they might conflict with internal variables created by PROC HPSEVERITY.

Custom Objective Functions and Regression Effects

If you specify regression effects by using the SCALEMODEL statement, then PROC HPSEVERITY automatically adds a statement prior to your programming statements to compute the value of the scale parameter or the log-transformed scale parameter of the distribution using the values of the regression variables and internally created regression parameters. For example, if your specification of the SCALEMODEL statement results in three regression effects x1, x2, and x3, then for a model that contains the distribution D with scale parameter S, PROC HPSEVERITY adds a statement that is equivalent to the following statement to the beginning of your program:

```
S = _SEVTHETA0 * exp(_SEVBETA1 * x1 + _SEVBETA2 * x2 + _SEVBETA3 * x3);
```

If a model contains a distribution D1 with a log-transformed scale parameter M, PROC HPSEVERITY adds a statement that is equivalent to the following statement to the beginning of your program:

```
M = _SEVTHETA0 + _SEVBETA1 * x1 + _SEVBETA2 * x2 + _SEVBETA3 * x3;
```

The _SEVTHETA0, _SEVBETA1, _SEVBETA2, and _SEVBETA3 are the internal regression parameters associated with the intercept and the regression effects x1, x2, and x3, respectively.

Since the names of the internal regression parameters start with a prefix _SEV, if you use a variable in your program with a name that begins with _SEV, then PROC HPSEVERITY writes an error message to the SAS log and stops processing.

Input Data Sets

PROC HPSEVERITY accepts DATA= and INEST= data sets as input data sets. This section details the information they are expected to contain.

DATA= Data Set

The DATA= data set is expected to contain the values of the analysis variables that you specify in the LOSS statement and the SCALEMODEL statement.

If you specify the BY statement, then the DATA= data set must contain all the BY variables that you specify in the BY statement and the data set must be sorted by the BY variables unless you specify the NOTSORTED option in the BY statement.

INEST= Data Set

The INEST= data set is expected to contain the initial values of the parameters for the parameter estimation process.

If you specify the SCALEMODEL statement, then you can use the INEST= data set only if the SCALEMODEL statement contains singleton continuous effects.

If you specify the BY statement, then the INEST= data set must contain all the BY variables that you specify in the BY statement. If you do not specify the NOTSORTED option in the BY statement, then the INEST= data set must be sorted by the BY variables. However, it is not required to contain all the BY groups present in the DATA= data set. For the BY groups that are not present in the INEST= data set, the default parameter initialization method is used. If you specify the NOTSORTED option in the BY statement, then the INEST= data set must contain all the BY groups that are present in the DATA= data set and they must appear in the same order as they appear in the DATA= data set.

In addition to any variables that you specify in the BY statement, the data set must contain the following variables:

MODEL identifying name of the distribution for which the estimates are provided.

TYPE type of the estimate. The value of this variable must be EST for an observation to be valid.

<Parameter 1> . . . <Parameter M>
M variables, named after the parameters of all candidate distributions, that contain initial values of the respective parameters. *M* is the cardinality of the union of parameter name sets from all candidate distributions. In an observation, estimates are read only from variables for parameters that correspond to the distribution that is indicated by the _MODEL_ variable.

If you specify a missing value for some parameters, then default initial values are used unless the parameter is initialized by using the INIT= option in the DIST statement. If you want to use the *dist*_PARMINIT subroutine for initializing the parameters of a model, then you should either not specify the model in the INEST= data set or specify missing values for all the distribution parameters in the INEST= data set and not use the INIT= option in the DIST statement.

If you specify regressors, then the initial value that you provide for the first parameter of each distribution must be the base value of the scale or log-transformed scale parameter. For more information, see the section "Estimating Regression Effects" on page 1245.

<Regressor 1> . . . <Regressor K>
If you specify K regressors in the SCALEMODEL statement, then the INEST= data set must contain K variables that are named for each regressor. The variables contain initial values of the respective regression coefficients. If a regressor is linearly dependent on other regressors for a given BY group, then you can indicate this by providing a special missing value of .R for the respective variable. In a given BY group, if you mark a variable as linearly dependent for one model, then you must mark that variable as linearly dependent for all the models. Similarly, in a given BY group, if you do not mark a variable as linearly dependent for one model, then you must not mark that variable as linearly dependent for all the models.

Output Data Sets

PROC HPSEVERITY writes the OUTCDF=, OUTEST=, OUTMODELINFO=, and OUTSTAT= data sets when requested by their respective options. The data sets and their contents are described in the following sections.

OUTCDF= Data Set

The OUTCDF= data set records the estimates of the cumulative distribution function (CDF) of each of the specified model distributions and an estimate of the empirical distribution function (EDF). This data set is created only when you run PROC HPSEVERITY in single-machine mode.

If you specify BY variables, then the data are organized in BY groups and the data set contains variables that you specify in the BY statement. In addition, the data set contains the following variables:

<response variable>
value of the response variable. The values are sorted. If there are multiple BY groups, the values are sorted within each BY group.

OBSNUM observation number in the DATA= data set. This is a sequence number that indicates the order in which the procedure accesses the observation; it does not necessarily reflect the actual observation number in the data set.

EDF estimate of the empirical distribution function (EDF). This estimate is computed by using the EMPIRICALCDF= option that you specify in the PROC HPSEVERITY statement.

_EDF_STD estimate of the standard error of EDF. This estimate is computed by using a method that is appropriate for the EMPIRICALCDF= option that you specify in the PROC HPSEVERITY statement.

_EDF_LOWER estimate of the lower confidence limit of EDF for a pointwise $100(1-\alpha)\%$ confidence interval, where α is the value of the EDFALPHA= option that you specify in the PROC HPSEVERITY statement (default is $\alpha = 0.05$). For an EDF estimate F_n that has standard error σ_n, it is computed as MAX(0, $F_n - z_{(1-\alpha/2)}\sigma_n$), where z_p is the pth quantile from the standard normal distribution.

_EDF_UPPER estimate of the upper confidence limit of EDF for a pointwise $100(1-\alpha)\%$ confidence interval, where α is the value of the EDFALPHA= option that you specify in the PROC HPSEVERITY statement (default is $\alpha = 0.05$). For an EDF estimate F_n that has standard error σ_n, it is computed as MIN(1, $F_n + z_{(1-\alpha/2)}\sigma_n$), where z_p is the pth quantile from the standard normal distribution.

<distribution1>_CDF . . . <distribution*D*>_CDF
estimate of the cumulative distribution function (CDF) for each of the D candidate distributions, computed by using the final parameter estimates for that distribution. This value is missing if the parameter estimation process does not converge for the given distribution.

If you specify regressor variables, then the reported estimates are from a mixture distribution. For more information, see the section "CDF and PDF Estimates with Regression Effects" on page 1249.

If you specify truncation, then the data set contains the following additional variables:

<distribution1>_COND_CDF . . . <distribution*D*>_COND_CDF
estimate of the conditional CDF for each of the D candidate distributions, computed by using the final parameter estimates for that distribution. This value is missing if the parameter estimation process does not converge for the distribution. The conditional estimates are computed by using the method that is described in the section "Truncation and Conditional CDF Estimates" on page 1241.

OUTEST= Data Set

The OUTEST= data set records the estimates of the model parameters. It also contains estimates of their standard errors and optionally their covariance structure. If you specify BY variables, then the data are organized in BY groups and the data set contains variables that you specify in the BY statement.

If you do not specify the COVOUT option, then the data set contains the following variables:

MODEL identifying name of the distribution model. The observation contains information about this distribution.

TYPE type of the estimates reported in this observation. It can take one of the following two values:

EST	point estimates of model parameters
STDERR	standard error estimates of model parameters

STATUS status of the reported estimates. The possible values are listed in the section "_STATUS_ Variable Values" on page 1306.

<Parameter 1> . . . <Parameter M>
M variables, named after the parameters of all candidate distributions, that contain estimates of the respective parameters. M is the cardinality of the union of parameter name sets from all candidate distributions. In an observation, estimates are populated only for parameters that correspond to the distribution that is indicated by the _MODEL_ variable. If _TYPE_ is EST, then the estimates are missing if the model does not converge. If _TYPE_ is STDERR, then the estimates are missing if covariance estimates cannot be obtained.

If you specify regression effects, then the estimate that is reported for the first parameter of each distribution is the estimate of the base value of the scale or log-transformed scale parameter. For more information, see the section "Estimating Regression Effects" on page 1245.

<Regression Effect 1> . . . <Regression Effect K>
If your effect specification in the SCALEMODEL statement results in *K* regression effects, then the OUTEST= data set contains *K* regression variables. The name of each variable is formed by using the name of the effect and the names of the levels of the CLASS variables that the effect might contain. If the effect name or level names are too long, then the variable name is constructed by using partial effect name and integer identifiers for BY groups and CLASS variable levels. The label of the variable is more descriptive than the name of the variable. The variables contain estimates for their respective regression coefficients. If an effect is deemed to be linearly dependent on other effects for a given BY group, then a warning message is written to the SAS log and a special missing value of .R is written in the respective variable. If _TYPE_ is EST, then the estimates are missing if the model does not converge. If _TYPE_ is STDERR, then the estimates are missing if covariance estimates cannot be obtained.

<Offset Variable>
If you specify an OFFSET= variable in the SCALEMODEL statement, then the OUTEST= data set contains a variable that is named after the offset variable. If _TYPE_ is EST, then the value of this variable is 1. If _TYPE_ is STDERR, then the value of this variable is a special missing value of .F.

If you specify the COVOUT option in the PROC HPSEVERITY statement, then the OUTEST= data set contains additional observations that contain the estimates of the covariance structure. Given the symmetric nature of the covariance structure, only the lower triangular portion is reported. In addition to the variables listed and described previously, the data set contains the following variables that are either new or have a modified description:

TYPE type of the estimates reported in this observation. For observations that contain rows of the covariance structure, the value is COV.

STATUS status of the reported estimates. For observations that contain rows of the covariance structure, the status is 0 if covariance estimation was successful. If estimation fails, the status is 1 and a single observation is reported with _TYPE_=COV and missing values for all the parameter variables.

NAME name of the parameter for the row of covariance matrix that is reported in the current observation.

OUTMODELINFO= Data Set

The OUTMODELINFO= data set records the information about each candidate distribution that you specify in the DIST statement. It contains the following variables:

MODEL identifying name of the distribution model. The observation contains information about this distribution.

DEPVAR name of the loss variable.

DESCRIPTION descriptive name of the model. This has a nonmissing value only if the DESCRIPTION function has been defined for this model.

VALID validity of the distribution definition. This has a value of 1 for valid definitions and a value of 0 for invalid definitions. If the definition is invalid, then PROC HPSEVERITY writes the reason for invalidity to the SAS log.

_PARMNAME1 ... _PARMNAME*M* *M* variables that contain names of parameters of the distribution model, where *M* is the maximum number of parameters across all the specified distribution models. For a given distribution with *m* parameters, values of variables _PARMNAME*j* ($j > m$) are missing.

OUTSTAT= Data Set

The OUTSTAT= data set records statistics of fit and model selection information. If you specify BY variables, then the data are organized in BY groups and the data set contains variables that you specify in the BY statement. The data set contains the following variables:

MODEL identifying name of the distribution model. The observation contains information about this distribution.

NMODELPARM number of parameters in the distribution.

NESTPARM number of estimated parameters. This includes the regression parameters, if you specify any regression effects.

NOBS number of nonmissing observations used for parameter estimation.

STATUS status of the parameter estimation process for this model. The possible values are listed in the section "_STATUS_ Variable Values" on page 1306.

SELECTED indicator of the best distribution model. If the value is 1, then this model is the best model for the current BY group according to the specified model selection criterion. This value is missing if the parameter estimation process does not converge for this model.

Neg2LogLike value of the log likelihood, multiplied by –2, that is attained at the end of the parameter estimation process. This value is missing if the parameter estimation process does not converge for this model.

AIC value of the Akaike's information criterion (AIC) that is attained at the end of the parameter estimation process. This value is missing if the parameter estimation process does not converge for this model.

AICC value of the corrected Akaike's information criterion (AICC) that is attained at the end of the parameter estimation process. This value is missing if the parameter estimation process does not converge for this model.

BIC value of the Schwarz Bayesian information criterion (BIC) that is attained at the end of the parameter estimation process. This value is missing if the parameter estimation process does not converge for this model.

KS value of the Kolmogorov-Smirnov (KS) statistic that is attained at the end of the parameter estimation process. This value is missing if the parameter estimation process does not converge for this model.

AD — value of the Anderson-Darling (AD) statistic that is attained at the end of the parameter estimation process. This value is missing if the parameter estimation process does not converge for this model.

CVM — value of the Cramér-von Mises (CvM) statistic that is attained at the end of the parameter estimation process. This value is missing if the parameter estimation process does not converge for this model.

STATUS Variable Values

The _STATUS_ variable in the OUTEST= and OUTSTAT= data sets contains a value that indicates the status of the parameter estimation process for the respective distribution model. The variable can take the following values in the OUTEST= data set for _TYPE_=EST observations and in the OUTSTAT= data set:

0 The parameter estimation process converged for this model.

301 The parameter estimation process might not have converged for this model because there is no improvement in the objective function value. This might indicate that the initial values of the parameters are optimal, or you can try different convergence criteria in the NLOPTIONS statement.

302 The parameter estimation process might not have converged for this model because the number of iterations exceeded the maximum allowed value. You can try setting a larger value for the MAXITER= options in the NLOPTIONS statement.

303 The parameter estimation process might not have converged for this model because the number of objective function evaluations exceeded the maximum allowed value. You can try setting a larger value for the MAXFUNC= options in the NLOPTIONS statement.

304 The parameter estimation process might not have converged for this model because the time taken by the process exceeded the maximum allowed value. You can try setting a larger value for the MAXTIME= option in the NLOPTIONS statement.

400 The parameter estimation process did not converge for this model.

The _STATUS_ variable can take the following values in the OUTEST= data set for _TYPE_=STDERR and _TYPE_=COV observations:

0 The covariance and standard error estimates are available and valid.

1 The covariance and standard error estimates are not available, because the process of computing covariance estimates failed.

Displayed Output

The HPSEVERITY procedure optionally produces displayed output by using the Output Delivery System (ODS). All output is controlled by the PRINT= option in the PROC HPSEVERITY statement. Table 23.17 relates the ODS tables to PRINT= options.

Table 23.17 ODS Tables Produced in PROC HPSEVERITY

ODS Table Name	Description	Option
AllFitStatistics	Statistics of fit for all the distribution models	PRINT=ALLFITSTATS
ConvergenceStatus	Convergence status of parameter estimation process	PRINT=CONVSTATUS
DescStats	Descriptive statistics for the response variable	PRINT=DESCSTATS
DistributionInfo	Distribution information	PRINT=DISTINFO
EstimationDetails	Details of the estimation process for all the distribution models	PRINT=ESTIMATIONDETAILS
InitialValues	Initial parameter values and bounds	PRINT=INITIALVALUES
IterationHistory	Optimization iteration history	PRINT=NLOHISTORY
ModelSelection	Model selection summary	PRINT=SELECTION
OptimizationSummary	Optimization summary	PRINT=NLOSUMMARY
ParameterEstimates	Final parameter estimates	PRINT=ESTIMATES
PerformanceInfo	Execution environment information that pertains to the computational performance	Default
RegDescStats	Descriptive statistics for the regression effects that do not contain a CLASS variable	PRINT=DESCSTATS
StatisticsOfFit	Statistics of fit	PRINT=STATISTICS
Timing	Timing information for various computational stages of the procedure	DETAILS (PERFORMANCE statement)
TurnbullSummary	Turnbull EDF estimation summary	PRINT=ALL

If you do not specify the PRINT= option, then by default PROC HPSEVERITY produces ModelSelection, PerformanceInfo, ConvergenceStatus, OptimizationSummary, StatisticsOfFit, and ParameterEstimates ODS tables.

The following describes the content that each table displays:

AllFitStatistics (PRINT=ALLFITSTATS)
displays the comparison of all the statistics of fit for all the models in one table. The table does not include the models whose parameter estimation process does not converge. If all the models fail to converge, then this table is not produced. If the table contains more than one model, then the best model according to each statistic is indicated with an asterisk (*) in that statistic's column.

ConvergenceStatus (PRINT=CONVSTATUS)
displays the convergence status of the parameter estimation process.

DescStats (PRINT=DESCSTATS)
displays the descriptive statistics for the response variable.

DistributionInfo (PRINT=DISTINFO)
displays the information about all the candidate distribution. It includes the name, the description, the number of distribution parameters, and whether the distribution is valid for the specified modeling task.

EstimationDetails (PRINT=ESTIMATIONDETAILS)
displays the comparative details of the estimation process that is used to fit each candidate distribution. If you specify the DETAILS option in the PERFORMANCE statement, then this table contains a column that indicates the time taken to estimate each candidate model.

InitialValues (PRINT=INITIALVALUES)
displays the initial values and bounds used for estimating each model.

IterationHistory (PRINT=NLOHISTORY)
displays the iteration history of the nonlinear optimization process used for estimating the parameters.

ModelSelection (PRINT=SELECTION)
displays the model selection table. The table shows the convergence status of each candidate model, and the value of the selection criterion along with an indication of the selected model.

OptimizationSummary (PRINT=NLOSUMMARY)
displays the summary of the nonlinear optimization process used for estimating the parameters.

ParameterEstimates (PRINT=ESTIMATES)
displays the final estimates of parameters. The estimates are not displayed for models whose parameter estimation process does not converge.

PerformanceInfo
displays information about the execution mode. For single-machine mode, the table displays the number of threads that are used. For distributed mode, the table displays the grid mode (symmetric or asymmetric), the number of compute nodes, and the number of threads per node. PROC HPSEVERITY produces this table by default.

RegDescStats (PRINT=DESCSTATS)
displays the descriptive statistics for the regression effects in the SCALEMODEL statement that do not contain a CLASS variable.

StatisticsOfFit (PRINT=STATISTICS)
displays the statistics of fit for each model. The statistics of fit are not displayed for models whose parameter estimation process does not converge.

Timing (DETAILS option in the PERFORMANCE statement
displays elapsed times (absolute and relative) for the main tasks of the procedure. PROC HPSEVERITY produces this table when you specify the DETAILS option in the PERFORMANCE statement,

TurnbullSummary (PRINT=ALL)
displays the summary of Turnbull's estimation process if Turnbull's method is used for computing EDF estimates. The summary includes whether the nonlinear optimization converged, the number of iterations, the maximum absolute relative error, the maximum absolute reduced gradient, and whether the final estimates are maximum likelihood estimates. This table is produced only if you specify PRINT=ALL and Turnbull's method is used for computing EDF estimates.

ODS Graphics

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*).

Before you create graphs, ODS Graphics must be enabled (for example, by using the ODS GRAPHICS ON statement). For more information, see the section "Enabling and Disabling ODS Graphics" (Chapter 21, *SAS/STAT User's Guide*).

The overall appearance of graphs is controlled by ODS styles. Styles and other aspects of using ODS Graphics are discussed in the section "A Primer on ODS Statistical Graphics" (Chapter 21, *SAS/STAT User's Guide*).

This section describes how the HPSEVERITY procedure uses ODS to create graphics.

NOTE: The graphics are created only when you run PROC HPSEVERITY in single-machine mode.

ODS Graph Names

PROC HPSEVERITY assigns a name to each graph that it creates by using ODS. You can use these names to selectively reference the graphs. The names are listed in Table 23.18.

Table 23.18 ODS Graphics Produced by PROC HPSEVERITY

ODS Graph Name	Plot Description	PLOTS= Option
CDFPlot	Comparative CDF plot	CDF
CDFDistPlot	CDF plot per distribution	CDFPERDIST
PDFPlot	Comparative PDF plot	PDF
PDFDistPlot	PDF plot per distribution	PDFPERDIST
PPPlot	P-P plot of CDF and EDF	PP
QQPlot	Q-Q plot	QQ

Comparative CDF Plot

The comparative CDF plot helps you visually compare the cumulative distribution function (CDF) estimates of all the candidate distribution models and the empirical distribution function (EDF) estimate. The plot does not contain CDF estimates for models whose parameter estimation process does not converge. The horizontal axis represents the values of the response variable. The vertical axis represents the values of the CDF or EDF estimates.

If you specify truncation, then conditional CDF estimates are plotted. Otherwise, unconditional CDF estimates are plotted. The conditional estimates are computed by using the method that is described in the section "Truncation and Conditional CDF Estimates" on page 1241.

If you specify regression effects, then the plotted CDF estimates are from a mixture distribution. For more information, see the section "CDF and PDF Estimates with Regression Effects" on page 1249.

CDF Plot per Distribution

The CDF plot per distribution shows the CDF estimates of each candidate distribution model unless that model's parameter estimation process does not converge. The plot also contains estimates of the EDF. The horizontal axis represents the values of the response variable. The vertical axis represents the values of the CDF or EDF estimates.

This plot shows the lower and upper pointwise confidence limits for the EDF estimates. For an EDF estimate F_n with standard error σ_n, they are computed as $\text{MAX}(0, F_n - z_{(1-\alpha/2)}\sigma_n)$ and $\text{MIN}(1, F_n + z_{(1-\alpha/2)}\sigma_n)$, respectively, where z_p is the pth quantile from the standard normal distribution and α denotes the confidence level that you specify in the EDFALPHA= option (the default is $\alpha = 0.05$).

If you specify truncation, then conditional CDF estimates are plotted. Otherwise, unconditional CDF estimates are plotted. The conditional estimates are computed by using the method that is described in the section "Truncation and Conditional CDF Estimates" on page 1241.

If you specify regression effects, then the plotted CDF estimates are from a mixture distribution. For more information, see the section "CDF and PDF Estimates with Regression Effects" on page 1249.

Comparative PDF Plot

The comparative PDF plot helps you visually compare the probability density function (PDF) estimates of all the candidate distribution models. The plot does not contain PDF estimates for models whose parameter estimation process does not converge. The horizontal axis represents the values of the response variable. The vertical axis represents the values of the PDF estimates.

If you specify the HISTOGRAM option, then the plot also contains the histogram of response variable values. If you specify the KERNEL option, then the plot also contains the kernel density estimate of the response variable values.

If you specify regression effects, then the plotted PDF estimates are from a mixture distribution. For more information, see the section "CDF and PDF Estimates with Regression Effects" on page 1249.

PDF Plot per Distribution

The PDF plot per distribution shows the PDF estimates of each candidate distribution model unless that model's parameter estimation process does not converge. The horizontal axis represents the values of the response variable. The vertical axis represents the values of the PDF estimates.

If you specify the HISTOGRAM option, then the plot also contains the histogram of response variable values. If you specify the KERNEL option, then the plot also contains the kernel density estimate of the response variable values.

If you specify regression effects, then the plotted PDF estimates are from a mixture distribution. For more information, see the section "CDF and PDF Estimates with Regression Effects" on page 1249.

P-P Plot of CDF and EDF

The P-P plot of CDF and EDF is the probability-probability plot that compares the CDF estimates of a distribution to the EDF estimates. A plot is not prepared for models whose parameter estimation process does not converge. The horizontal axis represents the CDF estimates of a candidate distribution, and the vertical axis represents the EDF estimates.

This plot can be interpreted as displaying the data that are used for computing the EDF-based statistics of fit for the given candidate distribution. As described in the section "EDF-Based Statistics" on page 1267,

these statistics are computed by comparing the EDF, denoted by $F_n(y)$, to the CDF, denoted by $F(y)$, at each of the response variable values y. Using the probability inverse transform $z = F(y)$, this is equivalent to comparing the EDF of the z, denoted by $F_n(z)$, to the CDF of z, denoted by $F(z)$ (D'Agostino and Stephens 1986, Ch. 4). Because the CDF of z is a uniform distribution ($F(z) = z$), the EDF-based statistics can be computed by comparing the EDF estimate of z to the estimate of z. The horizontal axis of the plot represents the estimated CDF $\hat{z} = \hat{F}(y)$. The vertical axis represents the estimated EDF of z, $\hat{F}_n(z)$. The plot contains a scatter plot of $(\hat{z}, \hat{F}_n(z))$ points and a reference line $F_n(z) = z$ that represents the expected uniform distribution of z. Points that are scattered closer to the reference line indicate a better fit than the points that are scattered farther away from the reference line.

If you specify truncation, then the EDF estimates are conditional, as described in the section "EDF Estimates and Truncation" on page 1265. So conditional estimates of CDF are displayed, which are computed by using the method that is described in the section "Truncation and Conditional CDF Estimates" on page 1241.

If you specify regression effects, then the displayed CDF estimates, both unconditional and conditional, are from a mixture distribution. For more information, see the section "CDF and PDF Estimates with Regression Effects" on page 1249.

Q-Q Plot

The Q-Q plot is a quantile-quantile scatter plot that compares the empirical quantiles to the quantiles from a candidate distribution. A plot is not prepared for models whose parameter estimation process does not converge. The horizontal axis represents the quantiles from a candidate distribution, and the vertical axis represents the empirical quantiles.

Each point in the plot corresponds to a specific value of the EDF estimate, F_n. The Y coordinate is the value of the response variable for which F_n is computed. The X coordinate is computed by using one of the two following methods for a candidate distribution named *dist*:

- If you have defined the *dist*_QUANTILE function that satisfies the requirements listed in the section "*dist*_QUANTILE" on page 1281, then that function is invoked by using F_n and estimated distribution parameters as arguments. The QUANTILE function is defined in the Sashelp.Svrtdist library for all the predefined distributions.
- If the *dist*_QUANTILE function is not defined, then PROC HPSEVERITY numerically inverts the *dist*_CDF function at the CDF value of F_n for the estimated distribution parameters. If the *dist*_CDF function is not defined, then the exp(*dist*_LOGCDF) function is inverted. If the inversion fails, the corresponding point is not plotted in the Q-Q plot.

If you specify truncation, then the EDF estimates are conditional, as described in the section "EDF Estimates and Truncation" on page 1265. The CDF inversion process, whether done numerically or by evaluating the *dist*_QUANTILE function, needs to accept an unconditional CDF value. So the F_n value is first transformed to an unconditional estimate F_n^u as

$$F_n^u = F_n \cdot (\hat{F}(t^r_{\max}) - \hat{F}(t^l_{\min})) + \hat{F}(t^l_{\min})$$

where $\hat{F}(t^r_{\max})$ and $\hat{F}(t^l_{\min})$ are as defined in the section "Truncation and Conditional CDF Estimates" on page 1241.

If you specify regression effects, then the value of the first distribution parameter is determined by using the DFMIXTURE=MEAN method that is described in the section "CDF and PDF Estimates with Regression Effects" on page 1249.

Examples: HPSEVERITY Procedure

Example 23.1: Defining a Model for Gaussian Distribution

Suppose you want to fit a distribution model other than one of the predefined ones available to you. Suppose you want to define a model for the Gaussian distribution with the following typical parameterization of the PDF (f) and CDF (F):

$$f(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$
$$F(x;\mu,\sigma) = \frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right)$$

For PROC HPSEVERITY, a *distribution model* consists of a set of functions and subroutines that are defined with the FCMP procedure. Each function and subroutine should be written following certain rules. For more information, see the section "Defining a Severity Distribution Model with the FCMP Procedure" on page 1273.

NOTE: The Gaussian distribution is not a commonly used severity distribution. It is used in this example primarily to illustrate the process of defining your own distribution models. Although the distribution has a support over the entire real line, you can fit the distribution with PROC HPSEVERITY only if the input sample contains nonnegative values.

The following SAS statements define a distribution model named NORMAL for the Gaussian distribution. The OUTLIB= option in the PROC FCMP statement stores the compiled versions of the functions and subroutines in the 'models' package of the Work.Sevexmpl library. The LIBRARY= option in the PROC FCMP statement enables this PROC FCMP step to use the SVRTUTIL_RAWMOMENTS utility subroutine that is available in the Sashelp.Svrtdist library. The subroutine is described in the section "Predefined Utility Functions" on page 1285.

```
/*-------- Define Normal Distribution with PROC FCMP  ----------*/
proc fcmp library=sashelp.svrtdist outlib=work.sevexmpl.models;
   function normal_pdf(x,Mu,Sigma);
      /* Mu    : Location */
      /* Sigma : Standard Deviation */
      return ( exp(-(x-Mu)**2/(2 * Sigma**2)) /
             (Sigma * sqrt(2*constant('PI'))) );
   endsub;

   function normal_cdf(x,Mu,Sigma);
      /* Mu    : Location */
      /* Sigma : Standard Deviation */
      z = (x-Mu)/Sigma;
      return (0.5 + 0.5*erf(z/sqrt(2)));
   endsub;

   subroutine normal_parminit(dim, x[*], nx[*], F[*], Ftype, Mu, Sigma);
      outargs Mu, Sigma;
      array m[2] / nosymbols;
```

```
      /* Compute estimates by using method of moments */
      call svrtutil_rawmoments(dim, x, nx, 2, m);
      Mu    = m[1];
      Sigma = sqrt(m[2] - m[1]**2);
   endsub;

   subroutine normal_lowerbounds(Mu, Sigma);
      outargs Mu, Sigma;
      Mu = .;   /* Mu has no lower bound */
      Sigma = 0; /* Sigma > 0 */
   endsub;
quit;
```

The statements define the two functions required of any distribution model (NORMAL_PDF and NORMAL_CDF) and two optional subroutines (NORMAL_PARMINIT and NORMAL_LOWERBOUNDS). The name of each function or subroutine must follow a specific structure. It should start with the model's short or identifying name, which is 'NORMAL' in this case, followed by an underscore '_', followed by a keyword suffix such as 'PDF'. Each function or subroutine has a specific purpose. For more information about all the functions and subroutines that you can define for a distribution model, see the section "Defining a Severity Distribution Model with the FCMP Procedure" on page 1273. Following is the description of each function and subroutine defined in this example:

- The PDF and CDF suffixes define functions that return the probability density function and cumulative distribution function values, respectively, given the values of the random variable and the distribution parameters.
- The PARMINIT suffix defines a subroutine that returns the initial values for the parameters by using the sample data or the empirical distribution function (EDF) estimate computed from it. In this example, the parameters are initialized by using the method of moments. Hence, you do not need to use the EDF estimates, which are available in the F array. The first two raw moments of the Gaussian distribution are as follows:

  $$E[x] = \mu,\ E[x^2] = \mu^2 + \sigma^2$$

  Given the sample estimates, m_1 and m_2, of these two raw moments, you can solve the equations $E[x] = m_1$ and $E[x^2] = m_2$ to get the following estimates for the parameters: $\hat{\mu} = m_1$ and $\hat{\sigma} = \sqrt{m_2 - m_1^2}$. The NORMAL_PARMINIT subroutine implements this solution. It uses the SVRTUTIL_RAWMOMENTS utility subroutine to compute the first two raw moments.
- The LOWERBOUNDS suffix defines a subroutine that returns the lower bounds on the parameters. PROC HPSEVERITY assumes a default lower bound of 0 for all the parameters when a LOWERBOUNDS subroutine is not defined. For the parameter μ (*Mu*), there is no lower bound, so you need to define the NORMAL_LOWERBOUNDS subroutine. It is recommended that you assign bounds for all the parameters when you define the LOWERBOUNDS subroutine or its counterpart, the UPPERBOUNDS subroutine. Any unassigned value is returned as a missing value, which PROC HPSEVERITY interprets to mean that the parameter is unbounded, and that might not be what you want.

You can now use this distribution model with PROC HPSEVERITY. Let the following DATA step statements simulate a normal sample with $\mu = 10$ and $\sigma = 2.5$:

```
/*-------- Simulate a Normal sample ----------*/
data testnorm(keep=y);
   call streaminit(12345);
   do i=1 to 100;
      y = rand('NORMAL', 10, 2.5);
      output;
   end;
run;
```

Prior to using your distribution with PROC HPSEVERITY, you must communicate the location of the library that contains the definition of the distribution and the locations of libraries that contain any functions and subroutines used by your distribution model. The following OPTIONS statement sets the CMPLIB= system option to include the FCMP library Work.Sevexmpl in the search path used by PROC HPSEVERITY to find FCMP functions and subroutines.

```
/*--- Set the search path for functions defined with PROC FCMP ---*/
options cmplib=(work.sevexmpl);
```

Now, you are ready to fit the NORMAL distribution model with PROC HPSEVERITY. The following statements fit the model to the values of Y in the Work.Testnorm data set:

```
/*--- Fit models with PROC HPSEVERITY ---*/
proc hpseverity data=testnorm print=all;
   loss y;
   dist Normal;
run;
```

The DIST statement specifies the identifying name of the distribution model, which is 'NORMAL'. Neither the INEST= option nor the INSTORE= option is specified in the PROC HPSEVERITY statement, and the INIT= option is not specified in the DIST statement. So PROC HPSEVERITY initializes the parameters by invoking the NORMAL_PARMINIT subroutine.

Some of the results prepared by the preceding PROC HPSEVERITY step are shown in Output 23.1.1 and Output 23.1.2. The descriptive statistics of variable Y and the "Model Selection" table, which includes just the normal distribution, are shown in Output 23.1.1.

Output 23.1.1 Summary of Results for Fitting the Normal Distribution

The HPSEVERITY Procedure

Input Data Set	
Name	WORK.TESTNORM

Descriptive Statistics for y	
Observations	100
Observations Used for Estimation	100
Minimum	3.88249
Maximum	16.00864
Mean	10.02059
Standard Deviation	2.37730

Output 23.1.1 *continued*

Model Selection

Distribution	Converged	-2 Log Likelihood	Selected
Normal	Yes	455.97541	Yes

The initial values for the parameters, the optimization summary, and the final parameter estimates are shown in Output 23.1.2. No iterations are required to arrive at the final parameter estimates, which are identical to the initial values. This confirms the fact that the maximum likelihood estimates for the Gaussian distribution are identical to the estimates obtained by the method of moments that was used to initialize the parameters in the NORMAL_PARMINIT subroutine.

Output 23.1.2 Details of the Fitted Normal Distribution Model

The HPSEVERITY Procedure
Normal Distribution

Distribution Information

Name	Normal
Distribution Parameters	2

Initial Parameter Values and Bounds

Parameter	Initial Value	Lower Bound	Upper Bound
Mu	10.02059	-Infty	Infty
Sigma	2.36538	1.05367E-8	Infty

Optimization Summary

Optimization Technique	Trust Region
Iterations	0
Function Calls	4
Log Likelihood	-227.98770

Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|
Mu	10.02059	0.23894	41.94	<.0001
Sigma	2.36538	0.16896	14.00	<.0001

The NORMAL distribution defined and illustrated here has no scale parameter, because all the following inequalities are true:

$$f(x;\mu,\sigma) \neq \frac{1}{\mu} f(\frac{x}{\mu};1,\sigma)$$
$$f(x;\mu,\sigma) \neq \frac{1}{\sigma} f(\frac{x}{\sigma};\mu,1)$$
$$F(x;\mu,\sigma) \neq F(\frac{x}{\mu};1,\sigma)$$
$$F(x;\mu,\sigma) \neq F(\frac{x}{\sigma};\mu,1)$$

This implies that you cannot estimate the influence of regression effects on a model for the response variable based on this distribution.

Example 23.2: Defining a Model for the Gaussian Distribution with a Scale Parameter

If you want to estimate the influence of regression effects, then the model needs to be parameterized to have a scale parameter. Although this might not be always possible, it is possible for the Gaussian distribution by replacing the location parameter μ with another parameter, $\alpha = \mu/\sigma$, and defining the PDF (f) and the CDF (F) as follows:

$$f(x;\sigma,\alpha) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x}{\sigma} - \alpha\right)^2\right)$$
$$F(x;\sigma,\alpha) = \frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{1}{\sqrt{2}}\left(\frac{x}{\sigma} - \alpha\right)\right)\right)$$

You can verify that σ is the scale parameter, because both of the following equalities are true:

$$f(x;\sigma,\alpha) = \frac{1}{\sigma} f(\frac{x}{\sigma};1,\alpha)$$
$$F(x;\sigma,\alpha) = F(\frac{x}{\sigma};1,\alpha)$$

NOTE: The Gaussian distribution is not a commonly used severity distribution. It is used in this example primarily to illustrate the concept of parameterizing a distribution such that it has a scale parameter. Although the distribution has a support over the entire real line, you can fit the distribution with PROC HPSEVERITY only if the input sample contains nonnegative values.

The following statements use the alternate parameterization to define a new model named NORMAL_S. The definition is stored in the Work.Sevexmpl library.

```
/*-------- Define Normal Distribution With Scale Parameter  ----------*/
proc fcmp library=sashelp.svrtdist outlib=work.sevexmpl.models;
   function normal_s_pdf(x, Sigma, Alpha);
      /* Sigma : Scale & Standard Deviation */
      /* Alpha : Scaled mean */
      return ( exp(-(x/Sigma - Alpha)**2/2) /
             (Sigma * sqrt(2*constant('PI'))) );
   endsub;

   function normal_s_cdf(x, Sigma, Alpha);
      /* Sigma : Scale & Standard Deviation */
      /* Alpha : Scaled mean */
      z = x/Sigma - Alpha;
      return (0.5 + 0.5*erf(z/sqrt(2)));
   endsub;

   subroutine normal_s_parminit(dim, x[*], nx[*], F[*], Ftype, Sigma, Alpha);
      outargs Sigma, Alpha;
      array m[2] / nosymbols;
      /* Compute estimates by using method of moments */
      call svrtutil_rawmoments(dim, x, nx, 2, m);
      Sigma = sqrt(m[2] - m[1]**2);
      Alpha = m[1]/Sigma;
   endsub;
```

```
   subroutine normal_s_lowerbounds(Sigma, Alpha);
      outargs Sigma, Alpha;
      Alpha = .; /* Alpha has no lower bound */
      Sigma = 0; /* Sigma > 0 */
   endsub;
quit;
```

An important point to note is that the scale parameter *Sigma* is the first distribution parameter (after the 'x' argument) listed in the signatures of NORMAL_S_PDF and NORMAL_S_CDF functions. *Sigma* is also the first distribution parameter listed in the signatures of other subroutines. This is required by PROC HPSEVERITY, so that it can identify which is the scale parameter. When you specify regression effects, PROC HPSEVERITY checks whether the first parameter of each candidate distribution is a scale parameter (or a log-transformed scale parameter if *dist*_SCALETRANSFORM subroutine is defined for the distribution with LOG as the transform). If it is not, then an appropriate message is written the SAS log and that distribution is not fitted.

Let the following DATA step statements simulate a sample from the normal distribution where the parameter σ is affected by the regressors as follows:

$$\sigma = \exp(1 + 0.5\ \mathrm{X1} + 0.75\ \mathrm{X3} - 2\ \mathrm{X4} + \mathrm{X5})$$

The sample is simulated such that the regressor X2 is linearly dependent on regressors X1 and X3.

```
/*--- Simulate a Normal sample affected by Regressors ---*/
data testnorm_reg(keep=y x1-x5 Sigma);
   array x{*} x1-x5;
   array b{6} _TEMPORARY_ (1 0.5 . 0.75 -2 1);
   call streaminit(34567);
   label y='Normal Response Influenced by Regressors';

   do n = 1 to 100;
      /* simulate regressors */
      do i = 1 to dim(x);
         x(i) = rand('UNIFORM');
      end;
      /* make x2 linearly dependent on x1 */
      x(2) = 5 * x(1);

      /* compute log of the scale parameter */
      logSigma = b(1);
      do i = 1 to dim(x);
         if (i ne 2) then
            logSigma = logSigma + b(i+1) * x(i);
      end;

      Sigma = exp(logSigma);
      y = rand('NORMAL', 25, Sigma);
      output;
   end;
run;
```

The following statements use PROC HPSEVERITY to fit the NORMAL_S distribution model along with some of the predefined distributions to the simulated sample:

```
/*--- Set the search path for functions defined with PROC FCMP ---*/
options cmplib=(work.sevexmpl);

/*-------- Fit models with PROC HPSEVERITY --------*/
proc hpseverity data=testnorm_reg print=all;
   loss y;
   scalemodel x1-x5;
   dist Normal_s burr logn pareto weibull;
run;
```

The "Model Selection" table in Output 23.2.1 indicates that all the models, except the Burr distribution model, have converged. Also, only three models, Normal_s, Burr, and Weibull, seem to have a good fit for the data. The table that compares all the fit statistics indicates that Normal_s model is the best according to the likelihood-based statistics; however, the Burr model is the best according to the EDF-based statistics.

Output 23.2.1 Summary of Results for Fitting the Normal Distribution with Regressors

The HPSEVERITY Procedure

Input Data Set	
Name	WORK.TESTNORM_REG

Model Selection

Distribution	Converged	-2 Log Likelihood	Selected
Normal_s	Yes	603.95786	Yes
Burr	Maybe	612.81685	No
Logn	Yes	749.20125	No
Pareto	Yes	841.07022	No
Weibull	Yes	612.77496	No

All Fit Statistics

Distribution	-2 Log Likelihood		AIC		AICC		BIC		KS		AD	
Normal_s	603.95786	*	615.95786	*	616.86108	*	631.58888	*	1.52388		4.00152	
Burr	612.81685		626.81685		628.03424		645.05304		1.50448	*	3.90072	*
Logn	749.20125		761.20125		762.10448		776.83227		2.88110		16.20558	
Pareto	841.07022		853.07022		853.97345		868.70124		4.83810		31.60568	
Weibull	612.77496		624.77496		625.67819		640.40598		1.50490		3.90559	

Note: The asterisk (*) marks the best model according to each column's criterion.

All Fit Statistics

Distribution		CvM
Normal_s	0.70769	
Burr	0.63399	*
Logn	3.04825	
Pareto	6.84046	
Weibull	0.63458	

Note: The asterisk (*) marks the best model according to each column's criterion.

This prompts you to further evaluate why the model with Burr distribution has not converged. The initial values, convergence status, and the optimization summary for the Burr distribution are shown in Output 23.2.2. The initial values table indicates that the regressor X2 is redundant, which is expected. More importantly, the convergence status indicates that it requires more than 50 iterations. PROC HPSEVERITY enables you to change several settings of the optimizer by using the NLOPTIONS statement. In this case, you can increase the limit of 50 on the iterations, change the convergence criterion, or change the technique to something other than the default trust-region technique.

Output 23.2.2 Details of the Fitted Burr Distribution Model

The HPSEVERITY Procedure
Burr Distribution

Distribution Information	
Name	Burr
Description	Burr Distribution
Distribution Parameters	3
Regression Parameters	4

Initial Parameter Values and Bounds

Parameter	Initial Value	Lower Bound	Upper Bound
Theta	25.75198	1.05367E-8	Infty
Alpha	2.00000	1.05367E-8	Infty
Gamma	2.00000	1.05367E-8	Infty
x1	0.07345	-709.78271	709.78271
x2	Redundant		
x3	-0.14056	-709.78271	709.78271
x4	0.27064	-709.78271	709.78271
x5	-0.23230	-709.78271	709.78271

Convergence Status
Needs more than 50 iterations.

Optimization Summary	
Optimization Technique	Trust Region
Iterations	50
Function Calls	137
Log Likelihood	-306.40842

The following PROC HPSEVERITY step uses the NLOPTIONS statement to change the convergence criterion and the limits on the iterations and function evaluations, exclude the lognormal and Pareto distributions that have been confirmed previously to fit the data poorly, and exclude the redundant regressor X2 from the model:

```
/*--- Refit and compare models with higher limit on iterations ---*/
proc hpseverity data=testnorm_reg print=all;
   loss y;
   scalemodel x1 x3-x5;
   dist Normal_s burr weibull;
   nloptions absfconv=2.0e-5 maxiter=100 maxfunc=500;
run;
```

The results shown in Output 23.2.3 indicate that the Burr distribution has now converged and that the Burr and Weibull distributions have an almost identical fit for the data. The NORMAL_S distribution is still the best distribution according to the likelihood-based criteria.

Output 23.2.3 Summary of Results after Changing Maximum Number of Iterations

The HPSEVERITY Procedure

Input Data Set	
Name	WORK.TESTNORM_REG

Model Selection

Distribution	Converged	-2 Log Likelihood	Selected
Normal_s	Yes	603.95786	Yes
Burr	Yes	612.79276	No
Weibull	Yes	612.77496	No

All Fit Statistics

Distribution	-2 Log Likelihood		AIC		AICC		BIC		KS		AD	
Normal_s	603.95786	*	615.95786	*	616.86108	*	631.58888	*	1.52388		4.00152	
Burr	612.79276		626.79276		628.01015		645.02895		1.50472	*	3.90351	*
Weibull	612.77496		624.77496		625.67819		640.40598		1.50490		3.90559	

Note: The asterisk (*) marks the best model according to each column's criterion.

All Fit Statistics

Distribution	CvM	
Normal_s	0.70769	
Burr	0.63433	*
Weibull	0.63458	

Note: The asterisk (*) marks the best model according to each column's criterion.

Example 23.3: Defining a Model for Mixed-Tail Distributions

In some applications, a few severity values tend to be extreme as compared to the typical values. The extreme values represent the worst case scenarios and cannot be discarded as outliers. Instead, their distribution must be modeled to prepare for their occurrences. In such cases, it is often useful to fit one distribution to the non-extreme values and another distribution to the extreme values. The *mixed-tail* distribution mixes two distributions: one for the *body* region, which contains the non-extreme values, and another for the *tail* region, which contains the extreme values. The tail distribution is usually a generalized Pareto distribution (GPD), because it is usually good for modeling the conditional excess severity above a threshold. The body distribution can be any distribution. The following definitions are used in describing a generic formulation of a mixed-tail distribution:

$g(x)$ PDF of the body distribution

$G(x)$ CDF of the body distribution

$h(x)$	PDF of the tail distribution
$H(x)$	CDF of the tail distribution
θ	scale parameter for the body distribution
Ω	set of nonscale parameters for the body distribution
ξ	shape parameter for the GPD tail distribution
x_r	normalized value of the response variable at which the tail starts
p_n	mixing probability

Given these notations, the PDF $f(x)$ and the CDF $F(x)$ of the mixed-tail distribution are defined as

$$f(x) = \begin{cases} \frac{p_n}{G(x_b)} g(x) & \text{if } x \le x_b \\ (1 - p_n) h(x - x_b) & \text{if } x > x_b \end{cases}$$

$$F(x) = \begin{cases} \frac{p_n}{G(x_b)} G(x) & \text{if } x \le x_b \\ p_n + (1 - p_n) H(x - x_b) & \text{if } x > x_b \end{cases}$$

where $x_b = \theta x_r$ is the value of the response variable at which the tail starts.

These definitions indicate the following:

- The body distribution is conditional on $X \le x_b$, where X denotes the random response variable.
- The tail distribution is the generalized Pareto distribution of the $(X - x_b)$ values.
- The probability that a response variable value belongs to the body is p_n. Consequently the probability that the value belongs to the tail is $(1 - p_n)$.

The parameters of this distribution are θ, Ω, ξ, x_r, and p_n. The scale of the GPD tail distribution θ_t is computed as

$$\theta_t = \frac{G(x_b; \theta, \Omega)}{g(x_b; \theta, \Omega)} \frac{(1 - p_n)}{p_n} = \theta \frac{G(x_r; \theta = 1, \Omega)}{g(x_r; \theta = 1, \Omega)} \frac{(1 - p_n)}{p_n}$$

The parameter x_r is usually estimated using a tail index estimation algorithm. One such algorithm is the Hill's algorithm (Danielsson et al. 2001), which is implemented by the predefined utility function SVRTUTIL_HILLCUTOFF available to you in the Sashelp.Svrtdist library. The algorithm and the utility function are described in detail in the section "Predefined Utility Functions" on page 1285. The function computes an estimate of x_b, which can be used to compute an estimate of x_r because $x_r = x_b / \hat{\theta}$, where $\hat{\theta}$ is the estimate of the scale parameter of the body distribution.

The parameter p_n is usually determined by the domain expert based on the fraction of losses that are expected to belong to the tail.

The following SAS statements define the LOGNGPD distribution model for a mixed-tail distribution with the lognormal distribution as the body distribution and GPD as the tail distribution:

```
/*------- Define Lognormal Body-GPD Tail Mixed Distribution -------*/
proc fcmp library=sashelp.svrtdist outlib=work.sevexmpl.models;
   function LOGNGPD_DESCRIPTION() $256;
      length desc $256;
      desc1 = "Lognormal Body-GPD Tail Distribution.";
      desc2 = " Mu, Sigma, and Xi are free parameters.";
      desc3 = " Xr and Pn are constant parameters.";
      desc = desc1 || desc2 || desc3;
      return(desc);
   endsub;

   function LOGNGPD_SCALETRANSFORM() $3;
      length xform $3;
      xform = "LOG";
      return (xform);
   endsub;

   subroutine LOGNGPD_CONSTANTPARM(Xr,Pn);
   endsub;

   function LOGNGPD_PDF(x, Mu,Sigma,Xi,Xr,Pn);
      cutoff = exp(Mu) * Xr;
      p = CDF('LOGN',cutoff, Mu, Sigma);
      if (x < cutoff + constant('MACEPS')) then do;
         return ((Pn/p)*PDF('LOGN', x, Mu, Sigma));
      end;
      else do;
         gpd_scale = p*((1-Pn)/Pn)/PDF('LOGN', cutoff, Mu, Sigma);
         h = (1+Xi*(x-cutoff)/gpd_scale)**(-1-(1/Xi))/gpd_scale;
         return ((1-Pn)*h);
      end;
   endsub;

   function LOGNGPD_CDF(x, Mu,Sigma,Xi,Xr,Pn);
      cutoff = exp(Mu) * Xr;
      p = CDF('LOGN',cutoff, Mu, Sigma);
      if (x < cutoff + constant('MACEPS')) then do;
         return ((Pn/p)*CDF('LOGN', x, Mu, Sigma));
      end;
      else do;
         gpd_scale = p*((1-Pn)/Pn)/PDF('LOGN', cutoff, Mu, Sigma);
         H = 1 - (1 + Xi*((x-cutoff)/gpd_scale))**(-1/Xi);
         return (Pn + (1-Pn)*H);
      end;
   endsub;

   subroutine LOGNGPD_PARMINIT(dim,x[*],nx[*],F[*],Ftype,
                        Mu,Sigma,Xi,Xr,Pn);
      outargs Mu,Sigma,Xi,Xr,Pn;
      array xe[1] / nosymbols;
      array nxe[1] / nosymbols;

      eps = constant('MACEPS');
```

```
      Pn = 0.8; /* Set mixing probability */
      _status_ = .;
      call streaminit(56789);
      Xb = svrtutil_hillcutoff(dim, x, 100, 25, _status_);
      if (missing(_status_) or _status_ = 1) then
         Xb = svrtutil_percentile(Pn, dim, x, F, Ftype);

      /* Initialize lognormal parameters */
      call logn_parminit(dim, x, nx, F, Ftype, Mu, Sigma);
      if (not(missing(Mu))) then
         Xr = Xb/exp(Mu);
      else
         Xr = .;

      /* prepare arrays for excess values */
      i = 1;
      do while (i <= dim and x[i] < Xb+eps);
         i = i + 1;
      end;
      dime = dim-i+1;
      if (dime > 0) then do;
         call dynamic_array(xe, dime);
         call dynamic_array(nxe, dime);
         j = 1;
         do while(i <= dim);
            xe[j] = x[i] - Xb;
            nxe[j] = nx[i];
            i = i + 1;
            j = j + 1;
         end;

         /* Initialize GPD's shape parameter using excess values */
         call gpd_parminit(dime, xe, nxe, F, Ftype, theta_gpd, Xi);
      end;
      else do;
         Xi = .;
      end;
   endsub;

   subroutine LOGNGPD_LOWERBOUNDS(Mu,Sigma,Xi,Xr,Pn);
      outargs Mu,Sigma,Xi,Xr,Pn;

      Mu   = .; /* Mu has no lower bound */
      Sigma = 0; /* Sigma > 0 */
      Xi   = 0; /* Xi > 0 */
   endsub;
quit;
```

Note the following points about the LOGNGPD definition:

- The parameters x_r and p_n are not estimated with the maximum likelihood method used by PROC HPSEVERITY, so you need to specify them as *constant* parameters by defining the *dist*_CONSTANTPARM subroutine. The signature of LOGNGPD_CONSTANTPARM subroutine lists only the constant parameters *Xr* and *Pn*.

- The parameter x_r is estimated by first using the SVRTUTIL_HILLCUTOFF utility function to compute an estimate of the cutoff point $\hat{x}_b$ and then computing $x_r = \hat{x}_b / e^{\hat{\mu}}$. If SVRTUTIL_HILLCUTOFF fails to compute a valid estimate, then the SVRTUTIL_PERCENTILE utility function is used to set $\hat{x}_b$ to the p_nth percentile of the data. The parameter p_n is fixed to 0.8.

- The Sashelp.Svrtdist library is specified with the LIBRARY= option in the PROC FCMP statement to enable the LOGNGPD_PARMINIT subroutine to use the predefined utility functions (SVRTUTIL_HILLCUTOFF and SVRTUTIL_PERCENTILE) and parameter initialization subroutines (LOGN_PARMINIT and GPD_PARMINIT).

- The LOGNGPD_LOWERBOUNDS subroutine defines the lower bounds for all parameters. This subroutine is required because the parameter *Mu* has a non-default lower bound. The bounds for *Sigma* and *Xi* must be specified. If they are not specified, they are returned as missing values, which PROC HPSEVERITY interprets as having no lower bound. You need not specify any bounds for the constant parameters *Xr* and *Pn*, because they are not subject to optimization.

The following DATA step statements simulate a sample from a mixed-tail distribution with a lognormal body and GPD tail. The parameter p_n is fixed to 0.8, the same value used in the LOGNGPD_PARMINIT subroutine defined previously.

```
/*----- Simulate a sample for the mixed-tail distribution -----*/
data testmixdist(keep=y label='Lognormal Body-GPD Tail Sample');
   call streaminit(45678);
   label y='Response Variable';
   N = 100;
   Mu = 1.5;
   Sigma = 0.25;
   Xi = 1.5;
   Pn = 0.8;

   /* Generate data comprising the lognormal body */
   Nbody = N*Pn;
   do i=1 to Nbody;
      y = exp(Mu) * rand('LOGNORMAL')**Sigma;
      output;
   end;

   /* Generate data comprising the GPD tail */
   cutoff = quantile('LOGNORMAL', Pn, Mu, Sigma);
   gpd_scale = (1-Pn) / pdf('LOGNORMAL', cutoff, Mu, Sigma);
   do i=Nbody+1 to N;
      y = cutoff + ((1-rand('UNIFORM'))**(-Xi) - 1)*gpd_scale/Xi;
      output;
   end;
run;
```

The following statements use PROC HPSEVERITY to fit the LOGNGPD distribution model to the simulated sample. They also fit three other predefined distributions (BURR, LOGN, and GPD). The final parameter estimates are written to the Work.Parmest data set.

```
/*--- Set the search path for functions defined with PROC FCMP ---*/
options cmplib=(work.sevexmpl);
/*-------- Fit LOGNGPD model with PROC HPSEVERITY --------*/
proc hpseverity data=testmixdist print=all outest=parmest;
   loss y;
   dist logngpd burr logn gpd;
run;
```

Some of the results prepared by PROC HPSEVERITY are shown in Output 23.3.1 and Output 23.3.2. The "Model Selection" table in Output 23.3.1 indicates that all models converged. The last table in Output 23.3.1 shows that the model with LOGNGPD distribution has the best fit according to almost all the statistics of fit. The Burr distribution model is the closest contender to the LOGNGPD model, but the GPD distribution model fits the data very poorly.

Output 23.3.1 Summary of Fitting Mixed-Tail Distribution

The HPSEVERITY Procedure

Input Data Set	
Name	WORK.TESTMIXDIST
Label	Lognormal Body-GPD Tail Sample

Model Selection

Distribution	Converged	-2 Log Likelihood	Selected
logngpd	Yes	418.78232	Yes
Burr	Yes	424.93728	No
Logn	Yes	459.43471	No
Gpd	Yes	558.13444	No

All Fit Statistics

Distribution	-2 Log Likelihood		AIC		AICC		BIC		KS		AD	
logngpd	418.78232	*	428.78232	*	429.42062	*	441.80817		0.62140	*	0.31670	*
Burr	424.93728		430.93728		431.18728		438.75280	*	0.71373		0.57649	
Logn	459.43471		463.43471		463.55842		468.64505		1.55267		3.27122	
Gpd	558.13444		562.13444		562.25815		567.34478		3.43470		16.74156	

Note: The asterisk (*) marks the best model according to each column's criterion.

All Fit Statistics

Distribution		CvM
logngpd	0.04972	*
Burr	0.07860	
Logn	0.48448	
Gpd	3.31860	

Note: The asterisk (*) marks the best model according to each column's criterion.

The detailed results for the LOGNGPD distribution are shown in Output 23.3.2. The initial values table indicates the values computed by LOGNGPD_PARMINIT subroutine for the *Xr* and *Pn* parameters. It also

uses the bounds columns to indicate the constant parameters. The last table in the figure shows the final parameter estimates. The estimates of all free parameters are significantly different from 0. As expected, the final estimates of the constant parameters *Xr* and *Pn* have not changed from their initial values.

Output 23.3.2 Detailed Results for the LOGNGPD Distribution

The HPSEVERITY Procedure
logngpd Distribution

Distribution Information	
Name	logngpd
Description	Lognormal Body-GPD Tail Distribution. Mu, Sigma, and Xi are free parameters. Xr and Pn are constant parameters.
Distribution Parameters	5

Initial Parameter Values and Bounds

Parameter	Initial Value	Lower Bound	Upper Bound
Mu	1.49954	-Infty	Infty
Sigma	0.76306	1.05367E-8	Infty
Xi	0.36661	1.05367E-8	Infty
Xr	1.27395	Constant	Constant
Pn	0.80000	Constant	Constant

Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

Optimization Summary

Optimization Technique	Trust Region
Iterations	11
Function Calls	33
Failed Function Calls	1
Log Likelihood	-209.39116

Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|
Mu	1.57921	0.06426	24.57	<.0001
Sigma	0.31868	0.04459	7.15	<.0001
Xi	1.03771	0.38205	2.72	0.0078
Xr	1.27395	Constant	.	.
Pn	0.80000	Constant	.	.

The following SAS statements use the parameter estimates to compute the value where the tail region is estimated to start ($x_b = e^{\hat{\mu}}\hat{x_r}$) and the scale of the GPD tail distribution ($\theta_t = \frac{G(x_b)}{g(x_b)}\frac{(1-p_n)}{p_n}$):

```
/*-------- Compute tail cutoff and tail distribution's scale --------*/
data xb_thetat(keep=x_b theta_t);
   set parmest(where=(_MODEL_='logngpd' and _TYPE_='EST'));
   x_b = exp(Mu) * Xr;
   theta_t = (CDF('LOGN',x_b,Mu,Sigma)/PDF('LOGN',x_b,Mu,Sigma)) *
             ((1-Pn)/Pn);
run;
```

```
proc print data=xb_thetat noobs;
run;
```

Output 23.3.3 Start of the Tail and Scale of the GPD Tail Distribution

x_b	theta_t
6.18005	1.27865

The computed values of x_b and θ_t are shown as *x_b* and *theta_t* in Output 23.3.3. Equipped with this additional derived information, you can now interpret the results of fitting the mixed-tail distribution as follows:

- The tail starts at $y \approx 6.18$. The primary benefit of using the scale-normalized cutoff (x_r) as the constant parameter instead of using the actual cutoff (x_b) is that the absolute cutoff is optimized by virtue of optimizing the scale of the body region ($\theta = e^{\mu}$).
- The values $y \leq 6.18$ follow the lognormal distribution with parameters $\mu \approx 1.58$ and $\sigma \approx 0.32$. These parameter estimates are reasonably close to the parameters used for simulating the sample.
- The values $y_t = y - 6.18$ ($y_t > 0$) follow the GPD distribution with scale $\theta_t \approx 1.28$ and shape $\xi \approx 1.04$.

Example 23.4: Fitting a Scaled Tweedie Model with Regressors

The Tweedie distribution is often used in the insurance industry to explain the influence of regression effects on the distribution of losses. PROC HPSEVERITY provides a predefined scaled Tweedie distribution (STWEEDIE) that enables you to model the influence of regression effects on the scale parameter. The scale regression model has its own advantages such as the ability to easily account for inflation effects. This example illustrates how that model can be used to evaluate the influence of regression effects on the *mean* of the Tweedie distribution, which is useful in problems such rate-making and pure premium modeling.

Assume a Tweedie process, whose mean μ is affected by k regression effects x_j, $j = 1, \ldots, k$ as follows:

$$\mu = \mu_0 \exp\left(\sum_{j=1}^{k} \beta_j x_j\right)$$

where μ_0 represents the base value of the mean (you can think of μ_0 as $\exp(\beta_0)$, where β_0 is the intercept). This model for the mean is identical to the popular generalized linear model for the mean with a logarithmic link function.

More interestingly, it parallels the model used by PROC HPSEVERITY for the scale parameter θ,

$$\theta = \theta_0 \exp\left(\sum_{j=1}^{k} \beta_j x_j\right)$$

where θ_0 represents the base value of the scale parameter. As described in the section "Tweedie Distributions" on page 1232, for the parameter range $p \in (1, 2)$, the mean of the Tweedie distribution is given by

$$\mu = \theta\lambda\frac{2-p}{p-1}$$

where λ is the Poisson mean parameter of the scaled Tweedie distribution. This relationship enables you to use the scale regression model to infer the influence of regression effects on the mean of the distribution.

Let the data set Work.Test_Sevtw contain a sample generated from a Tweedie distribution with dispersion parameter $\phi = 0.5$, index parameter $p = 1.75$, and the mean parameter that is affected by three regression variables x1, x2, and x3 as follows:

$$\mu = 5 \ \exp(0.25\,\mathrm{x1} - \mathrm{x2} + 3\,\mathrm{x3})$$

Thus, the population values of regression parameters are $\mu_0 = 5$, $\beta_1 = 0.25$, $\beta_2 = -1$, and $\beta_3 = 3$. You can find the code used to generate the sample in the PROC HPSEVERITY sample program *hsevex04.sas*.

The following PROC HPSEVERITY step uses the sample in Work.Test_Sevtw data set to estimate the parameters of the scale regression model for the predefined scaled Tweedie distribution (STWEEDIE) with the dual quasi-Newton (QUANEW) optimization technique:

```
/*--- Fit the scale parameter version of the Tweedie distribution ---*/
proc hpseverity data=test_sevtw outest=estw covout print=all;
   loss y;
   scalemodel x1-x3;

   dist stweedie;
   nloptions tech=quanew;
run;
```

The dual quasi-Newton technique is used because it requires only the first-order derivatives of the objective function, and it is harder to compute reasonably accurate estimates of the second-order derivatives of Tweedie distribution's PDF with respect to the parameters.

Some of the key results prepared by PROC HPSEVERITY are shown in Output 23.4.1 and Output 23.4.2. The distribution information and the convergence results are shown in Output 23.4.1.

Output 23.4.1 Convergence Results for the STWEEDIE Model with Regressors

The HPSEVERITY Procedure
stweedie Distribution

Distribution Information	
Name	stweedie
Description	Tweedie Distribution with Scale Parameter
Distribution Parameters	3
Regression Parameters	3

Convergence Status
Convergence criterion (FCONV=2.220446E-16) satisfied.

Optimization Summary	
Optimization Technique	Dual Quasi-Newton
Iterations	42
Function Calls	218
Log Likelihood	-1044.3

The final parameter estimates of the STWEEDIE regression model are shown in Output 23.4.2. The estimate that is reported for the parameter Theta is the estimate of the base value θ_0. The estimates of regression coefficients β_1, β_2, and β_3 are indicated by the rows of x1, x2, and x3, respectively.

Output 23.4.2 Parameter Estimates for the STWEEDIE Model with Regressors

Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|
Theta	0.82888	0.26657	3.11	0.0021
Lambda	16.57174	13.12083	1.26	0.2076
P	1.75440	0.20187	8.69	<.0001
x1	0.27970	0.09876	2.83	0.0049
x2	-0.76715	0.10313	-7.44	<.0001
x3	3.03225	0.10142	29.90	<.0001

If your goal is to explain the influence of regression effects on the scale parameter, then the output displayed in Output 23.4.2 is sufficient. But, if you want to compute the influence of regression effects on the mean of the distribution, then you need to do some postprocessing. Using the relationship between μ and θ, μ can be written in terms of the parameters of the STWEEDIE model as

$$\mu = \theta_0 \exp\left(\sum_{j=1}^{k} \beta_j x_j\right) \lambda \frac{2-p}{p-1}$$

This shows that the parameters β_j are identical for the mean and the scale model, and the base value μ_0 of the mean model is

$$\mu_0 = \theta_0 \lambda \frac{2-p}{p-1}$$

The estimate of μ_0 and the standard error associated with it can be computed by using the property of the functions of maximum likelihood estimators (MLE). If $g(\Omega)$ represents a totally differentiable function of parameters Ω, then the MLE of g has an asymptotic normal distribution with mean $g(\hat{\Omega})$ and covariance $C = (\partial \mathbf{g})' \Sigma (\partial \mathbf{g})$, where $\hat{\Omega}$ is the MLE of Ω, Σ is the estimate of covariance matrix of Ω, and $\partial \mathbf{g}$ is the gradient vector of g with respect to Ω evaluated at $\hat{\Omega}$. For μ_0, the function is $g(\Omega) = \theta_0 \lambda (2-p)/(p-1)$. The gradient vector is

$$\begin{aligned} \partial \mathbf{g} &= \left(\frac{\partial g}{\partial \theta_0} \quad \frac{\partial g}{\partial \lambda} \quad \frac{\partial g}{\partial p} \quad \frac{\partial g}{\partial \beta_1} \cdots \frac{\partial g}{\partial \beta_k} \right) \\ &= \left(\frac{\mu_0}{\theta_0} \quad \frac{\mu_0}{\lambda} \quad \frac{-\mu_0}{(p-1)(2-p)} \quad 0 \ldots 0 \right) \end{aligned}$$

You can write a DATA step that implements these computations by using the parameter and covariance estimates prepared by PROC HPSEVERITY step. The DATA step program is available in the sample program *hsevex04.sas*. The estimates of μ_0 prepared by that program are shown in Output 23.4.3. These estimates and the estimates of β_j as shown in Output 23.4.2 are reasonably close (that is, within one or two standard errors) to the parameters of the population from which the sample in Work.Test_Sevtw data set was drawn.

Output 23.4.3 Estimate of the Base Value Mu0 of the Mean Parameter

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|
Mu0	4.47179	0.42225	10.5904	0

Another outcome of using the scaled Tweedie distribution to model the influence of regression effects is that the regression effects also influence the variance V of the Tweedie distribution. The variance is related to the mean as $V = \phi\mu^p$, where ϕ is the dispersion parameter. Using the relationship between the parameters TWEEDIE and STWEEDIE distributions as described in the section "Tweedie Distributions" on page 1232, the regression model for the dispersion parameter is

$$\begin{aligned}\log(\phi) &= (2-p)\log(\mu) - \log(\lambda(2-p)) \\ &= ((2-p)\log(\mu_0) - \log(\lambda(2-p))) + (2-p)\sum_{j=1}^{k}\beta_j x_j\end{aligned}$$

Subsequently, the regression model for the variance is

$$\begin{aligned}\log(V) &= 2\log(\mu) - \log(\lambda(2-p)) \\ &= (2\log(\mu_0) - \log(\lambda(2-p))) + 2\sum_{j=1}^{k}\beta_j x_j\end{aligned}$$

In summary, PROC HPSEVERITY enables you to estimate regression effects on various parameters and statistics of the Tweedie model.

Example 23.5: Fitting Distributions to Interval-Censored Data

In some applications, the data available for modeling might not be exact. A commonly encountered scenario is the use of grouped data from an external agency, which for several reasons, including privacy, does not provide information about individual loss events. The losses are grouped into disjoint bins, and you know only the range and number of values in each bin. Each group is essentially interval-censored, because you know that a loss magnitude is in certain interval, but you do not know the exact magnitude. This example illustrates how you can use PROC HPSEVERITY to model such data.

The following DATA step generates sample grouped data for dental insurance claims, which is taken from Klugman, Panjer, and Willmot (1998):

```
/* Grouped dental insurance claims data
    (Klugman, Panjer, and Willmot, 1998) */
data gdental;
   input lowerbd upperbd count @@;
   datalines;
0 25 30  25 50 31  50 100 57  100 150 42  150 250 65  250 500 84
500 1000 45  1000 1500 10  1500 2500 11  2500 4000 3
;
run;
```

The following PROC HPSEVERITY step fits all the predefined distributions to the data in Work.Gdental data set:

```
/* Fit all predefined distributions */
proc hpseverity data=gdental edf=turnbull print=all criterion=aicc;
   loss / rc=lowerbd lc=upperbd;
   weight count;
   dist _predef_;
   performance nthreads=1;
run;
```

The EDF= option in the PROC HPSEVERITY statement specifies that the Turnbull's method be used for EDF estimation. The LOSS statement specifies the left and right boundaries of each group as the right-censoring and left-censoring limits, respectively. The variable count records the number of losses in each group and is specified in the WEIGHT statement. Note that no response variable is specified in the LOSS statement, which is allowed as long as each observation in the input data set is censored. The PERFORMANCE statement specifies that just one thread of execution be used, to minimize the overhead associated with multithreading, because the input data set is very small.

Some of the key results prepared by PROC HPSEVERITY are shown in Output 23.5.1. According to the "Model Selection" table in Output 23.5.1, all distribution models have converged. The "All Fit Statistics" table in Output 23.5.1 indicates that the exponential distribution (EXP) has the best fit for data according to a majority of the likelihood-based statistics.

Output 23.5.1 Statistics of Fit for Interval-Censored Data

The HPSEVERITY Procedure

Input Data Set	
Name	WORK.GDENTAL

Model Selection			
Distribution	**Converged**	**AICC**	**Selected**
Burr	Yes	51.41112	No
Exp	Yes	44.64768	Yes
Gamma	Yes	47.63969	No
Igauss	Yes	48.05874	No
Logn	Yes	47.34027	No
Pareto	Yes	47.16908	No
Gpd	Yes	47.16908	No
Weibull	Yes	47.47700	No

Output 23.5.1 *continued*

All Fit Statistics

Distribution	-2 Log Likelihood		AIC		AICC		BIC		KS		AD
Burr	41.41112	*	47.41112		51.41112		48.31888		0.08974	*	0.00103	*
Exp	42.14768		44.14768	*	44.64768	*	44.45026	*	0.26412		0.09936	
Gamma	41.92541		45.92541		47.63969		46.53058		0.19569		0.04608	
Igauss	42.34445		46.34445		48.05874		46.94962		0.34514		0.12301	
Logn	41.62598		45.62598		47.34027		46.23115		0.16853		0.01884	
Pareto	41.45480		45.45480		47.16908		46.05997		0.11423		0.00739	
Gpd	41.45480		45.45480		47.16908		46.05997		0.11423		0.00739	
Weibull	41.76272		45.76272		47.47700		46.36789		0.17238		0.03293	

Note: The asterisk (*) marks the best model according to each column's criterion.

All Fit Statistics

Distribution	CvM	
Burr	0.0000816	*
Exp	0.01866	
Gamma	0.00759	
Igauss	0.02562	
Logn	0.00333	
Pareto	0.0009084	
Gpd	0.0009084	
Weibull	0.00472	

Note: The asterisk (*) marks the best model according to each column's criterion.

Example 23.6: Benefits of Distributed and Multithreaded Computing

One of the key features of the HPSEVERITY procedure is that is takes advantage of the distributed and multithreaded computing machinery in order to solve a given problem faster. This example illustrates the benefits of using multithreading and distributed computing.

The example uses a simulated data set Work.Largedata, which contains 10,000,000 observations, some of which are right-censored or left-truncated. The losses are affected by three external effects. The DATA step program that generates this data set is available in the accompanying sample program *hsevex06.sas*.

The following PROC HPSEVERITY step fits all the predefined distributions to the data in Work.Largedata data set on the client machine with just one thread of computation:

```
/* Fit all predefined distributions without any multithreading or
   distributed computing */
proc hpseverity data=largedata criterion=aicc initsample(size=20000);
   loss y / lt=threshold rc=limit;
   scalemodel x1-x3;
   dist _predef_;
   performance nthreads=1 bufsize=1000000 details;
run;
```

The NTHREADS=1 option in the PERFORMANCE statement specifies that just one thread of computation be used. The absence of the NODES= option in the PERFORMANCE statement specifies that single-machine mode of execution be used. That is, this step does not use any multithreading or distributed computing. The BUFSIZE= option in the PERFORMANCE statement specifies the number of observations to read at one time. Specifying a larger value tends to decrease the time it takes to load the data. The DETAILS option in the performance statement enables reporting of the timing information. The INITSAMPLE option in the PROC HPSEVERITY statement specifies that a uniform random sample of maximum 20,000 observations be used for parameter initialization.

The "Performance Information" and "Procedure Task Timing" tables that PROC HPSEVERITY creates are shown in Output 23.6.1. The "Performance Information" table contains the information about the execution environment. The "Procedure Task Timing" table indicates the total time and relative time taken by each of the four main steps of PROC HPSEVERITY. As that table shows, it takes around 28.2 minutes for the task of estimating parameters, which is usually the most time-consuming of all the tasks.

Output 23.6.1 Performance for Single-Machine Mode with No Multithreading

The HPSEVERITY Procedure

Performance Information	
Execution Mode	Single-Machine
Number of Threads	1

Procedure Task Timing		
Task	**Seconds**	**Percent**
Load and Prepare Models	3.04	0.18%
Load and Prepare Data	1.30	0.08%
Initialize Parameters	0.84	0.05%
Estimate Parameters	1694.44	99.62%
Compute Fit Statistics	1.31	0.08%

If the grid appliance is not available, you can improve the performance by using multiple threads of computation; this is in fact the default. The following PROC HPSEVERITY step fits all the predefined distributions by using all the logical CPU cores of the machine:

```
/* Specify that all the logical CPU cores on the machine be used */
options cpucount=actual;

/* Fit all predefined distributions with multithreading, but no
   distributed computing */
proc hpseverity data=largedata criterion=aicc initsample(size=20000);
   loss y / lt=threshold rc=limit;
   scalemodel x1-x3;
   dist _predef_;
   performance bufsize=1000000 details;
run;
```

When you do not specify the NTHREADS= option in the PERFORMANCE statement, the HPSEVERITY procedure uses the value of the CPUCOUNT= system option to decide the number of threads to use in single-machine mode. Setting the CPUCOUNT= option to ACTUAL before the PROC HPSEVERITY step enables the procedure to use all the logical cores of the machine. The machine that is used to obtain these results (and the earlier results in Output 23.6.1) has four physical CPU cores, each with a clock speed of 3.4

GHz. Hyperthreading is enabled on the CPUs to yield eight logical CPU cores; this number is confirmed by the "Performance Information" table in Output 23.6.2. The results in the "Procedure Task Timing" table in Output 23.6.2 indicate that the use of multithreading has improved the performance by reducing the time to estimate parameters to around 5.7 minutes.

Output 23.6.2 Performance for Single-Machine Mode with Eight Threads

The HPSEVERITY Procedure

Performance Information	
Execution Mode	Single-Machine
Number of Threads	8

Procedure Task Timing		
Task	**Seconds**	**Percent**
Load and Prepare Models	0.50	0.14%
Load and Prepare Data	1.03	0.29%
Initialize Parameters	0.67	0.19%
Estimate Parameters	343.44	97.74%
Compute Fit Statistics	5.76	1.64%

When a grid appliance is available, performance can be further improved by using more than one node in the distributed mode of execution. Large data sets are usually predistributed on the grid appliance that hosts a distributed database. In other words, large problems are best suited for the alongside-the-database model of execution. However, for the purpose of illustration, this example assumes that the data set is available on the client machine and is then distributed to the grid nodes by the HPSEVERITY procedure according to the options that are specified in the PERFORMANCE statement.

The next few PROC HPSEVERITY steps are run on a grid appliance by varying the number of nodes and the number of threads that are used within each node.

You can specify your distributed computing environment by using SAS environment variables or by specifying options in the PERFORMANCE statement, or by a combination of these methods. For example, you can submit the following statements to specify the appliance host (GRIDHOST= SAS environment variable) and the installation location of shared libraries on the appliance (GRIDINSTALLLOC= SAS environment variable):

```
/* Set the appliance host and installation location that are
   appropriate for your distributed mode setup */
option set=GRIDHOST      ="&GRIDHOST";
option set=GRIDINSTALLLOC="&GRIDINSTALLLOC";
```

To run the preceding statements successfully, you need to set the macro variables GRIDHOST and GRIDINSTALLLOC to resolve to appropriate values, or you can replace the references to macro variables with the appropriate values. Alternatively, you can specify the HOST= and INSTALL= options in the PERFORMANCE statement; this method is used in the PROC HPSEVERITY steps of this example. You can use other SAS environment variables and PERFORMANCE statement options to describe your distributed computing environment. For more information, see the section "PERFORMANCE Statement" on page 87.

To establish a reference point for the performance of one CPU of a grid node, the results of using only one node of the grid appliance without any multithreading are presented first. The particular grid appliance that is used to obtain these results has more than sixteen nodes. Each node has 8 dual-core CPUs with a clock speed of 2.7 GHz. The following PROC HPSEVERITY step fits all the predefined distributions to the data in the Work.Largedata data set:

```
/* Fit all predefined distributions on 1 grid node without
   any multithreading */
proc hpseverity data=largedata criterion=aicc initsample(size=20000);
   loss y / lt=threshold rc=limit;
   scalemodel x1-x3;
   dist _predef_;
   performance nodes=1 nthreads=1 details
      host="&GRIDHOST" install="&GRIDINSTALLLOC";
run;
```

The PERFORMANCE statement specifies that only one node be used to fit the models, with only one thread of computation on that node. The "Performance Information" and "Procedure Task Timing" tables that PROC HPSEVERITY creates are shown in Output 23.6.3. It takes around 33.5 minutes to complete the task of estimating parameters. Note that this time is longer than the time taken for the single-machine mode with one thread of computation, because the CPUs of an individual grid node are slower than the CPUs of the machine that is used in single-machine mode. When the performance is measured, the grid node is shared among multiple users, unlike the machine that is used in single-machine mode.

Output 23.6.3 Performance on One Grid Appliance Node with No Multithreading

Performance Information	
Host Node	<< your grid host >>
Install Location	<< your grid install location >>
Execution Mode	Distributed
Number of Compute Nodes	1
Number of Threads per Node	1

Procedure Task Timing		
Task	**Seconds**	**Percent**
Load and Prepare Models	0.49	0.02%
Load and Prepare Data	0.92	0.05%
Initialize Parameters	1.03	0.05%
Estimate Parameters	2008.81	99.80%
Compute Fit Statistics	1.61	0.08%

The computations and time taken to fit each model are shown in the "Estimation Details" table of Output 23.6.4, which is generated whenever you specify the DETAILS option in the PERFORMANCE statement. This table can be useful for comparing the relative effort required to fit each model and drawing some broader conclusions. For example, even if the Pareto distribution takes a larger number of iterations, function calls, and gradient and Hessian updates than the gamma distribution, it takes less time to complete; this indicates that the individual PDF and CDF computations of the gamma distribution are more expensive than those of the Pareto distribution.

Output 23.6.4 Estimation Details

Estimation Details						
Distribution	Converged	Iterations	Function Calls	Gradient Updates	Hessian Updates	Time (Seconds)
Burr	Yes	11	28	104	90	325.96
Exp	Yes	4	12	27	20	29.56
Gamma	Yes	6	16	44	35	722.06
Igauss	Yes	4	16	27	20	215.40
Logn	Yes	4	12	27	20	112.60
Pareto	Yes	39	113	902	860	397.75
Gpd	Yes	6	17	44	35	132.98
Weibull	Yes	4	12	27	20	72.57

To obtain the next reference point for performance, the following PROC HPSEVERITY step specifies that 16 computation threads be used on one node of the grid appliance:

```
/* Fit all predefined distributions on 1 grid node with multithreading */
proc hpseverity data=largedata criterion=aicc initsample(size=20000);
   loss y / lt=threshold rc=limit;
   scalemodel x1-x3;
   dist _predef_;
   performance nodes=1 nthreads=16 details
      host="&GRIDHOST" install="&GRIDINSTALLLOC";
run;
```

The performance tables that are created by the preceding statements are shown in Output 23.6.5. As the "Procedure Task Timing" table shows, use of multithreading has improved the performance significantly over that of the single-threaded case. Now, it takes around 2.9 minutes to complete the task of estimating parameters.

Output 23.6.5 Performance Information with Multithreading but No Distributed Computing

Performance Information	
Host Node	<< your grid host >>
Install Location	<< your grid install location >>
Execution Mode	Distributed
Number of Compute Nodes	1
Number of Threads per Node	16

Procedure Task Timing		
Task	Seconds	Percent
Load and Prepare Models	0.49	0.28%
Load and Prepare Data	0.51	0.29%
Initialize Parameters	0.91	0.52%
Estimate Parameters	173.34	98.38%
Compute Fit Statistics	0.94	0.53%

You can combine the power of multithreading and distributed computing by specifying that multiple nodes of the grid be used to accomplish the task. The following PROC HPSEVERITY step specifies that 16 nodes of the grid appliance be used:

```
/* Fit all predefined distributions with distributed computing and
   multithreading within each node */
proc hpseverity data=largedata criterion=aicc initsample(size=20000);
   loss y / lt=threshold rc=limit;
   scalemodel x1-x3;
   dist _predef_;
   performance nodes=16 nthreads=16 details
      host="&GRIDHOST" install="&GRIDINSTALLLOC";
run;
```

When the DATA= data set is local to the client machine, as it is in this example, you must specify a nonzero value for the NODES= option in the PERFORMANCE statement in order to enable the distributed mode of execution. In other words, for the distributed mode that is not executing alongside the database, omitting the NODES= option is equivalent to specifying NODES=0, which is single-machine mode.

The performance tables that are created by the preceding statements are shown in Output 23.6.6. If you compare these tables to the tables in Output 23.6.3 and Output 23.6.5, you see that the task that would have taken a long time with a single thread of execution on a single machine (over half an hour) can be performed in a much shorter time (around 15 seconds) by using the computational resources of the grid appliance to combine the power of multithreaded and distributed computing.

Output 23.6.6 Performance Information with Distributed Computing and Multithreading

Performance Information	
Host Node	<< your grid host >>
Install Location	<< your grid install location >>
Execution Mode	Distributed
Number of Compute Nodes	16
Number of Threads per Node	16

Procedure Task Timing		
Task	**Seconds**	**Percent**
Load and Prepare Models	0.81	4.86%
Load and Prepare Data	0.04	0.25%
Initialize Parameters	0.71	4.24%
Estimate Parameters	14.52	86.62%
Compute Fit Statistics	0.68	4.03%

The machines that were used to obtain these performance results are relatively modest machines, and PROC HPSEVERITY was run in a multiuser environment; that is, background processes were running in single-machine mode or other users were using the grid in distributed mode. For time-critical applications, you can use a larger, dedicated grid that consists of more powerful machines to achieve more dramatic performance improvement.

Example 23.7: Estimating Parameters Using Cramér-von Mises Estimator

PROC HPSEVERITY enables you to estimate model parameters by minimizing your own objective function. This example illustrates how you can use PROC HPSEVERITY to implement the Cramér-von Mises estimator. Let $F(y_i;\Theta)$ denote the estimate of CDF at y_i for a distribution with parameters Θ, and let $F_n(y_i)$ denote the empirical estimate of CDF (EDF) at y_i that is computed from a sample y_i, $1 \le i \le N$. Then, the Cramér-von Mises estimator of the parameters is defined as

$$\hat{\Theta} = \arg\min_{\Theta} \sum_{i=1}^{N} (F(y_i;\Theta) - F_n(y_i))^2$$

This estimator belongs to the class of minimum distance estimators. It attempts to estimate the parameters such that the squared distance between the CDF and EDF estimates is minimized.

The following PROC HPSEVERITY step uses the Cramér-von Mises estimator to fit four candidate distribution models, including the LOGNGPD mixed-tail distribution model that was defined in "Example 23.3: Defining a Model for Mixed-Tail Distributions" on page 1320. The input sample is the same as is used in that example.

```
/*--- Set the search path for functions defined with PROC FCMP ---*/
options cmplib=(work.sevexmpl);

/*-------- Fit LOGNGPD model with PROC HPSEVERITY by using -------
  -------- the Cramer-von Mises minimum distance estimator -------*/
proc hpseverity data=testmixdist obj=cvmobj print=all;
   loss y;
   dist logngpd burr logn gpd;

   * Cramer-von Mises estimator (minimizes the distance *
   * between parametric and nonparametric estimates)    *;
   cvmobj = _cdf_(y);
   cvmobj = (cvmobj -_edf_(y))**2;
run;
```

The OBJ= option in the PROC HPSEVERITY statement specifies that the objective function cvmobj should be minimized. The programming statements compute the contribution of each observation in the input data set to the objective function cvmobj. The use of keyword functions _CDF_ and _EDF_ makes the program applicable to all the distributions.

Some of the key results prepared by PROC HPSEVERITY are shown in Output 23.7.1. The "Model Selection" table indicates that all models converged. When you specify a custom objective function, the default selection criterion is the value of the custom objective function. The "All Fit Statistics" table indicates that LOGNGPD is the best distribution according to all the statistics of fit. Comparing the fit statistics of Output 23.7.1 with those of Output 23.3.1 indicates that the use of the Cramér-von Mises estimator has resulted in smaller values for all the EDF-based statistics of fit for all the models, which is expected from a minimum distance estimator.

Output 23.7.1 Summary of Cramér-von Mises Estimation

The HPSEVERITY Procedure

Input Data Set	
Name	WORK.TESTMIXDIST
Label	Lognormal Body-GPD Tail Sample

Model Selection

Distribution	Converged	cvmobj	Selected
logngpd	Yes	0.02694	Yes
Burr	Yes	0.03325	No
Logn	Yes	0.03633	No
Gpd	Yes	2.96090	No

All Fit Statistics

Distribution	cvmobj		-2 Log Likelihood		AIC		AICC		BIC		KS	
logngpd	0.02694	*	419.49635	*	429.49635	*	430.13464	*	442.52220	*	0.51332	*
Burr	0.03325		436.58823		442.58823		442.83823		450.40374		0.53084	
Logn	0.03633		491.88659		495.88659		496.01030		501.09693		0.52469	
Gpd	2.96090		560.35409		564.35409		564.47780		569.56443		2.99095	

Note: The asterisk (*) marks the best model according to each column's criterion.

All Fit Statistics

Distribution	AD		CvM	
logngpd	0.21563	*	0.03030	*
Burr	0.82875		0.03807	
Logn	2.08312		0.04173	
Gpd	15.51378		2.97806	

Note: The asterisk (*) marks the best model according to each column's criterion.

Example 23.8: Defining a Finite Mixture Model That Has a Scale Parameter

A finite mixture model is a stochastic model that postulates that the probability distribution of the data generation process is a mixture of a finite number of probability distributions. For example, when an insurance company analyzes loss data from multiple policies that are underwritten in different geographic regions, some regions might behave similarly, but the distribution that governs some regions might be different from the distribution that governs other regions. Further, it might not be known which regions behave similarly. Also, the larger amounts of losses might follow a different stochastic process from the stochastic process that governs the smaller amounts of losses. It helps to model all policies together in order to pool the data together and exploit any commonalities among the regions, and the use of a finite mixture model can help capture the differences in distributions across regions and ranges of loss amounts.

Formally, if f_i and F_i denote the PDF and CDF, respectively, of component distribution i and p_i represents the mixing probability that is associated with component i, then the PDF and CDF of the finite mixture of K distribution components are

$$f(x;\Theta,\mathbf{p}) = \sum_{i=1}^{K} p_i\, f_i(x;\Theta_i)$$
$$F(x;\Theta,\mathbf{p}) = \sum_{i=1}^{K} p_i\, F_i(x;\Theta_i)$$

where Θ_i denotes the parameters of component distribution i and Θ denotes the parameters of the mixture distribution, which is a union of all the Θ_i parameters. $\mathbf{p}$ denotes the set of mixing probabilities. All mixing probabilities must add up to 1 ($\sum_{i=1}^{K} p_i = 1$).

You can define the finite mixture of a specific number of components and specific distributions for each of the components by defining the FCMP functions for the PDF and CDF. However, in general, it is not possible to fit a scale regression model by using any finite mixture distribution unless you take special care to ensure that the mixture distribution has a scale parameter. This example provides a formulation of a two-component finite mixture model that has a scale parameter.

To start with, each component distribution must have either a scale parameter or a log-transformed scale parameter. Let θ_1 and θ_2 denote the scale parameters of the first and second components, respectively. Let $p_1 = p$ be the mixing probability, which makes $p_2 = 1 - p$ by using the constraint on $\mathbf{p}$. The PDF of the mixture of these two distributions can be written as

$$f(x;\theta_1,\theta_2,\Phi,p) = \frac{p}{\theta_1} f_1(\frac{x}{\theta_1};\Phi_1) + \frac{1-p}{\theta_2} f_2(\frac{x}{\theta_2};\Phi_2)$$

where Φ_1 and Φ_2 denote the sets of nonscale parameters of the first and second components, respectively, and Φ denotes a union of Φ_1 and Φ_2. For the mixture to have the scale parameter θ, the PDF must be of the form

$$f(x;\theta,\Phi',p) = \frac{1}{\theta}\left(p f_1(\frac{x}{\theta};\Phi'_1) + (1-p) f_2(\frac{x}{\theta};\Phi'_2)\right)$$

where Φ', Φ'_1, and Φ'_2 denote the modified sets of nonscale parameters. One simple way to achieve this is to make $\theta_1 = \theta_2 = \theta$ and $\Phi' = \Phi$; that is, you simply equate the scale parameters of both components and keep the set of nonscale parameters unchanged. However, forcing the scale parameters to be equal in both components is restrictive, because the mixture cannot model potential differences in the scales of the two components. A better approach is to tie the scale parameters of the two components by a ratio such that $\theta_1 = \theta$ and $\theta_2 = \rho\theta$. If the ratio parameter ρ is estimated along with the other parameters, then the mixture distribution becomes flexible enough to model the variations across the scale parameters of individual components.

To summarize, the PDF and CDF are of the following form for the two-component mixture that has a scale parameter:

$$f(x;\theta,\rho,\Phi,p) = \frac{1}{\theta}\left(p f_1(\frac{x}{\theta};\Phi_1) + (1-p) f_2(\frac{x}{\theta};\rho,\Phi_2)\right)$$
$$F(x;\theta,\rho,\Phi,p) = p F_1(\frac{x}{\theta};\Phi_1) + (1-p) F_2(\frac{x}{\theta};\rho,\Phi_2)$$

This can be generalized to a mixture of K components by introducing the $K-1$ ratio parameters ρ_i that relate the scale parameters of each of the K components to the scale parameter θ of the mixture distribution as follows:

$$\theta_1 = \theta$$
$$\theta_i = \rho_i \theta;\ i \in [2, K]$$

In order to illustrate this approach, define a mixture of two lognormal distributions by using the following PDF function:

$$f(x; \mu, \sigma_1, p_2, \rho_2, \sigma_2) = \frac{(1-p_2)}{\sigma_1 x\sqrt{2\pi}} \exp\left(\frac{-(\log(x)-\mu)^2}{2\sigma_1^2}\right) + \frac{p_2}{\sigma_2 x\sqrt{2\pi}} \exp\left(\frac{-(\log(x)-\mu-\log(\rho_2))^2}{2\sigma_2^2}\right)$$

You can verify that μ serves as the log of the scale parameter θ ($\mu = \log(\theta)$).

The following PROC FCMP steps encode this formulation in a distribution named SLOGNMIX2 for use with PROC HPSEVERITY:

```
/*- Define Mixture of 2 Lognormal Distributions with a Log-Scale Parameter -*/
proc fcmp library=sashelp.svrtdist outlib=work.sevexmpl.models;
   function slognmix2_description() $128;
      return ("Mixture of two lognormals with a log-scale parameter Mu");
   endsub;

   function slognmix2_scaletransform() $8;
      return ("LOG");
   endsub;

   function slognmix2_pdf(x, Mu, Sigma1, p2, Rho2, Sigma2);
      Mu1 = Mu;
      Mu2 = Mu + log(Rho2);
      pdf1 = logn_pdf(x, Mu1, Sigma1);
      pdf2 = logn_pdf(x, Mu2, Sigma2);
      return ((1-p2)*pdf1 + p2*pdf2);
   endsub;

   function slognmix2_cdf(x, Mu, Sigma1, p2, Rho2, Sigma2);
      Mu1 = Mu;
      Mu2 = Mu + log(Rho2);
      cdf1 = logn_cdf(x, Mu1, Sigma1);
      cdf2 = logn_cdf(x, Mu2, Sigma2);
      return ((1-p2)*cdf1 + p2*cdf2);
   endsub;

   subroutine slognmix2_parminit(dim, x[*], nx[*], F[*], Ftype,
                                 Mu, Sigma1, p2, Rho2, Sigma2);
      outargs Mu, Sigma1, p2, Rho2, Sigma2;
      array m[1] / nosymbols;
      p2 = 0.5;
      Rho2 = 0.5;
```

```
      median = svrtutil_percentile(0.5, dim, x, F, Ftype);
      Mu = log(2*median/1.5);
      call svrtutil_rawmoments(dim, x, nx, 1, m);
      lm1 = log(m[1]);

      /* Search Rho2 that makes log(sample mean) > Mu */
      do while (lm1 <= Mu and Rho2 < 1);
         Rho2 = Rho2 + 0.01;
         Mu = log(2*median/(1+Rho2));
      end;
      if (Rho2 >= 1) then
         /* If Mu cannot be decreased enough to make it less
            than log(sample mean), then revert to Rho2=0.5.
            That will set Sigma1 and possibly Sigma2 to missing.
            PROC HPSEVERITY replaces missing initial values with 0.001. */
         Mu = log(2*median/1.5);

      Sigma1 = sqrt(2.0*(log(m[1])-Mu));
      Sigma2 = sqrt(2.0*(log(m[1])-Mu-log(Rho2)));
   endsub;

   subroutine slognmix2_lowerbounds(Mu, Sigma1, p2, Rho2, Sigma2);
      outargs Mu, Sigma1, p2, Rho2, Sigma2;
      Mu = .; /* Mu has no lower bound */
      Sigma1 = 0; /* Sigma1 > 0 */
      p2 = 0; /* p2 > 0 */
      Rho2 = 0; /* Rho2 > 0 */
      Sigma2 = 0; /* Sigma2 > 0 */
   endsub;

   subroutine slognmix2_upperbounds(Mu, Sigma1, p2, Rho2, Sigma2);
      outargs Mu, Sigma1, p2, Rho2, Sigma2;
      Mu = .; /* Mu has no upper bound */
      Sigma1 = .; /* Sigma1 has no upper bound */
      p2 = 1; /* p2 < 1 */
      Rho2 = 1; /* Rho2 < 1 */
      Sigma2 = .; /* Sigma2 has no upper bound */
   endsub;
quit;
```

As shown in previous examples, an important aspect of defining a distribution for use with PROC HPSEVERITY is the definition of the PARMINIT subroutine that initializes the parameters. For mixture distributions, in general, the parameter initialization is a nontrivial task. For a two-component mixture, some simplifying assumptions make the problem easier to handle. For the initialization of SLOGNMIX2, the initial values of p_2 and ρ_2 are fixed at 0.5, and the following two simplifying assumptions are made:

- The median of the mixture is the average of the medians of the two components:

  $$F^{-1}(0.5) = (\exp(\mu_1) + \exp(\mu_2))/2 = \exp(\mu)(1 + \rho_2)/2$$

  Solution of this equation yields the value of μ in terms of ρ_2 and the sample median.

- Each component has the same mean, which implies the following:

  $$\exp(\mu + \sigma_1^2/2) = \exp(\mu + \log(\rho_2) + \sigma_2^2/2)$$

If X_i represents the random variable of component distribution i and X represents the random variable of the mixture distribution, then the following equation holds for the raw moment of any order k:

$$E[X^k] = \sum_{i=1}^{K} p_i E[X_i^k]$$

This, in conjunction with the assumption on component means, leads to the equations

$$\log(m_1) = \mu + \frac{\sigma_1^2}{2}$$
$$\log(m_1) = \mu + \log(\rho_2) + \frac{\sigma_2^2}{2}$$

where m_1 denotes the first raw moment of the sample. Solving these equations leads to the following values of σ_1 and σ_2:

$$\sigma_1^2 = 2(\log(m_1) - \mu)$$
$$\sigma_2^2 = 2(\log(m_1) - \mu - \log(\rho_2))$$

Note that σ_1 has a valid value only if $\log(m_1) > \mu$. Among the many possible methods of ensuring this condition, the SLOGNMIX2_PARMINIT subroutine uses the method of doing a linear search over ρ_2.

Even when the preceding assumptions are not true for a given problem, they produce reasonable initial values to help guide the nonlinear optimizer to an acceptable optimum if the mixture of two lognormal distributions is indeed a good fit for your input data. This is illustrated by the results of the following steps that fit the SLOGNMIX2 distribution to simulated data, which have different means for the two components (12.18 and 22.76, respectively), and the median of the sample (15.94) is not equal to the average of the medians of the two components (7.39 and 20.09, respectively):

```
/*-------- Simulate a lognormal mixture sample ----------*/
data testlognmix(keep=y);
   call streaminit(12345);
   Mu1 = 2;
   Sigma1 = 1;
   i = 0;
   do j=1 to 2000;
      y = exp(Mu1) * rand('LOGNORMAL')**Sigma1;
      output;
   end;
   Mu2 = 3;
   Sigma2 = 0.5;
   do j=1 to 3000;
      y = exp(Mu2) * rand('LOGNORMAL')**Sigma2;
      output;
   end;
run;

/*-- Fit and compare scale regression models with 2-component  --*/
/*-- lognormal mixture and the standard lognormal distribution --*/
options cmplib=(work.sevexmpl);
```

```
proc hpseverity data=testlognmix print=all;
   loss y;
   dist slognmix2 logn;
run;
```

The comparison of the fit statistics of SLOGNMIX2 and LOGN, as shown in Output 23.8.1, confirms that the two-component mixture is certainly a better fit to these data than the single lognormal distribution.

Output 23.8.1 Comparison of Fitting One versus Two Lognormal Components to Mixture Data

The HPSEVERITY Procedure

All Fit Statistics

Distribution	-2 Log Likelihood		AIC		AICC		BIC		KS		AD		CvM	
slognmix2	38343	*	38353	*	38353	*	38386	*	0.52221	*	0.19843	*	0.02728	*
Logn	39073		39077		39077		39090		5.86522		66.93414		11.72703	

Note: The asterisk (*) marks the best model according to each column's criterion.

The detailed results for the SLOGNMIX2 distribution are shown in Output 23.8.2. According to the "Initial Parameter Values and Bounds" table, the initial value of ρ_2 is not 0.5, indicating that a linear search was conducted to ensure $\log(m_1) > \mu$.

Output 23.8.2 Detailed Estimation Results for the SLOGNMIX2 Distribution

The HPSEVERITY Procedure
slognmix2 Distribution

Distribution Information

Name	slognmix2
Description	Mixture of two lognormals with a log-scale parameter Mu
Distribution Parameters	5

Initial Parameter Values and Bounds

Parameter	Initial Value	Lower Bound	Upper Bound
Mu	2.92006	-Infty	Infty
Sigma1	0.10455	1.05367E-8	Infty
P2	0.50000	1.05367E-8	1.00000
Rho2	0.72000	1.05367E-8	1.00000
Sigma2	0.81728	1.05367E-8	Infty

Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

Optimization Summary

Optimization Technique	Trust Region
Iterations	7
Function Calls	18
Log Likelihood	-19171.5

Output 23.8.2 *continued*

Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|
Mu	3.00922	0.01554	193.68	<.0001
Sigma1	0.49516	0.01451	34.13	<.0001
P2	0.40619	0.02600	15.62	<.0001
Rho2	0.37212	0.02038	18.26	<.0001
Sigma2	1.00019	0.02124	47.09	<.0001

By using the relationship that $\mu_2 = \mu + \log(\rho_2)$, you can see that the final parameter estimates are indeed close to the true parameter values that were used to simulate the input sample.

Example 23.9: Predicting Mean and Value-at-Risk by Using Scoring Functions

If you work in the risk management department of an insurance company or a bank, then one of your primary applications of severity loss distribution models is to predict the value-at-risk (VaR) so that there is a very low probability of experiencing a loss value that is greater than the VaR. The probability level at which VaR is measured is prescribed by industry regulations such as Basel III and Solvency II. The VaR level is usually specified in terms of $(1 - \alpha)$, where $\alpha \in (0, 1)$ is the probability that a loss value exceeds the VaR. Typical VaR levels are 0.95, 0.975, and 0.995.

In addition to predicting the VaR, which is regarded as an estimate of the worst-case loss, businesses are often interested in predicting the average loss by estimating either the mean or median of the distribution.

The estimation of the mean and VaR combined with the scale regression model is very potent tool for analyzing worst-case and average losses for various scenarios. For example, if the regressors that are used in a scale regression model represent some key macroeconomic and operational indicators, which are widely referred to as key risk indicators (KRIs), then you can analyze the VaR and mean loss estimates over various values for the KRIs to get a more comprehensive picture of the risk profile of your organization across various market and internal conditions.

This example illustrates the use of scoring functions to simplify the process of predicting the mean and VaR of scale regression models.

To compute the mean, you need to ensure that the function to compute the mean of a distribution is available in the function library. If you define and fit your own distribution and you want to compute its mean, then you need to use the FCMP procedure to define that function and you need to use the CMPLIB= system option to specify the location of that function. For your convenience, the *dist*_MEAN function (which computes the mean of the *dist* distribution) is already defined in the Sashelp.Svrtdist library for each of the 10 predefined distributions. The following statements display the definitions of MEAN functions of all distributions. Note that the MEAN functions for the Burr, Pareto, and generalized Pareto distributions check the existence of the first moment for specified parameter values.

```
/*--------- Definitions distribution functions that compute the mean ----------*/
proc fcmp library=sashelp.svrtdist outlib=work.means.scalemod;
   function BURR_MEAN(x, Theta, Alpha, Gamma);
      if not(Alpha * Gamma > 1) then
         return (.); /* first moment does not exist */
      return (Theta*gamma(1 + 1/Gamma)*gamma(Alpha - 1/Gamma)/gamma(Alpha));
   endsub;
```

```
   function EXP_MEAN(x, Theta);
      return (Theta);
   endsub;
   function GAMMA_MEAN(x, Theta, Alpha);
      return (Theta*Alpha);
   endsub;
   function GPD_MEAN(x, Theta, Xi);
      if not(Xi < 1) then
         return (.); /* first moment does not exist */
      return (Theta/(1 - Xi));
   endsub;
   function IGAUSS_MEAN(x, Theta, Alpha);
      return (Theta);
   endsub;
   function LOGN_MEAN(x, Mu, Sigma);
      return (exp(Mu + Sigma*Sigma/2.0));
   endsub;


   function PARETO_MEAN(x, Theta, Alpha);
      if not(Alpha > 1) then
         return (.); /* first moment does not exist */
      return (Theta/(Alpha - 1));
   endsub;
   function STWEEDIE_MEAN(x, Theta, Lambda, P);
      return (Theta* Lambda * (2 - P) / (P - 1));
   endsub;
   function TWEEDIE_MEAN(x, P, Mu, Phi);
      return (Mu);
   endsub;
   function WEIBULL_MEAN(x, Theta, Tau);
      return (Theta*gamma(1 + 1/Tau));
   endsub;
quit;
```

For your further convenience, the *dist*_QUANTILE function (which computes the quantile of the *dist* distribution) is also defined in the Sashelp.Svrtdist library for each of the 10 predefined distributions. Because the MEAN and QUANTILE functions satisfy the definition of a distribution function as described in the section "Formal Description" on page 1292, you can submit the following PROC HPSEVERITY step to fit all regression-friendly predefined distributions and generate the scoring functions for the MEAN, QUANTILE, and other distribution functions:

```
/*----- Fit all distributions and generate scoring functions ------*/
proc hpseverity data=test_sev9 outest=est print=all;
   loss y;
   scalemodel x1-x5;
   dist _predefined_ stweedie;
   outscorelib outlib=scorefuncs commonpackage;
run;
```

The SAS statements that simulate the sample in the Work.Test_sev9 data set are available in the PROC HPSEVERITY sample program *hsevex09.sas*. The OUTLIB= option in the OUTSCORELIB statement requests that the scoring functions be written to the Work.Scorefuncs library, and the COMMONPACKAGE option in the OUTSCORELIB statement requests that all the functions be written to the same package. Upon completion, PROC HPSEVERITY sets the CMPLIB system option to the following value:

```
(sashelp.svrtdist work.scorefuncs)
```

The "All Fit Statistics" table in Output 23.9.1 shows that the lognormal distribution's scale model is the best and the inverse Gaussian's scale model is a close second according to the likelihood-based statistics.

You can examine the scoring functions that are written to the Work.Scorefuncs library by using the FCMP Function Editor, which is available in the Display Manager session of Base SAS when you select **Solutions→Analysis** from the main menu. For example, PROC HPSEVERITY automatically generates and submits the following PROC FCMP statements to define the scoring functions SEV_MEAN_LOGN and SEV_QUANTILE_IGAUSS:

```
proc fcmp library=(sashelp.svrtdist) outlib=work.scorefuncs.sevfit;
   function SEV_MEAN_LOGN(y, x{*});
      _logscale_=0;
      _logscale_ = _logscale_ + ( 7.64722278930350E-01 * x{1});
      _logscale_ = _logscale_ + ( 2.99209540369860E+00 * x{2});
      _logscale_ = _logscale_ + (-1.00788916253430E+00 * x{3});
      _logscale_ = _logscale_ + ( 2.58883602184890E-01 * x{4});
      _logscale_ = _logscale_ + ( 5.00927479793970E+00 * x{5});
      _logscale_ = _logscale_ + ( 9.95078833050690E-01);
      return (LOGN_MEAN(y, _logscale_,  2.31592981635590E-01));
   endsub;

   function SEV_QUANTILE_IGAUSS(y, x{*});
      _logscale_=0;
      _logscale_ = _logscale_ + ( 7.64581738373520E-01 * x{1});
      _logscale_ = _logscale_ + ( 2.99159055015310E+00 * x{2});
      _logscale_ = _logscale_ + (-1.00793496641510E+00 * x{3});
      _logscale_ = _logscale_ + ( 2.58870460543840E-01 * x{4});
      _logscale_ = _logscale_ + ( 5.00996884646730E+00 * x{5});
      _scale_ =  2.77854870591020E+00 * exp(_logscale_);
      return (IGAUSS_QUANTILE(y, _scale_,  1.81511227238720E+01));
   endsub;
quit;
```

Output 23.9.1 Comparison of Fitted Scale Models for Mean and VaR Illustration

The HPSEVERITY Procedure

All Fit Statistics

Distribution	-2 Log Likelihood		AIC		AICC		BIC		KS		AD
stweedie	460.65756		476.65756		476.95083		510.37442		10.44549		64571	
Burr	451.42238		467.42238		467.71565		501.13924		10.32782		42254	
Exp	1515		1527		1527		1552		8.85827		29917	
Gamma	448.28222		462.28222		462.50986		491.78448		10.42272		63712	
Igauss	444.44512		458.44512		458.67276		487.94738		10.33028		83195	
Logn	444.43670	*	458.43670	*	458.66434	*	487.93895	*	10.37035		68631	
Pareto	1515		1529		1529		1559		8.85775	*	29916	*
Gpd	1515		1529		1529		1559		8.85827		29917	
Weibull	527.28676		541.28676		541.51440		570.78902		10.48084		72814	

Note: The asterisk (*) marks the best model according to each column's criterion.

All Fit Statistics

Distribution	CvM	
stweedie	37.07708	
Burr	37.19808	
Exp	23.98267	
Gamma	37.19450	
Igauss	37.30880	
Logn	37.18553	
Pareto	23.98149	*
Gpd	23.98267	
Weibull	36.36039	

Note: The asterisk (*) marks the best model according to each column's criterion.

PROC HPSEVERITY detects all the distribution functions that are available in the current CMPLIB= search path (which always includes the Sashelp.Svrtdist library) for the distributions that you specify in the DIST statement, and it creates the corresponding scoring functions. You can define any distribution function that has the desired signature to compute an estimate of your choice, include its library in the CMPLIB= system option, and then specify the OUTSCORELIB statement to generate the corresponding scoring functions. Specifying the COMMONPACKAGE option in the OUTSCORELIB statement causes the name of the scoring function to take the form SEV_*function-suffix_dist*. If you do not specify the COMMONPACKAGE option, PROC HPSEVERITY creates a scoring function named SEV_*function-suffix* in a package named *dist*. You can invoke functions from a specific package only inside the FCMP procedure. If you want to invoke the scoring functions from a DATA step, then it is recommended that you specify the COMMONPACKAGE option when you specify multiple distributions in the DIST statement.

To illustrate the use of scoring functions, let Work.Reginput contain the scoring data, where the values of regressors in each observation define one scenario. Scoring functions make it very easy to compute the mean and VaR of each distribution's scale model for each of the scenarios, as the following steps illustrate for the lognormal and inverse Gaussian distributions:

```
/*--- Set VaR level ---*/
%let varLevel=0.975;

/*--- Compute scores (mean and var) for the       ---
  --- scoring data by using the scoring functions ---*/
data scores;
   array x{*} x1-x5;
   set reginput;

   igauss_mean = sev_mean_igauss(., x);
   igauss_var  = sev_quantile_igauss(&varLevel, x);
   logn_mean   = sev_mean_logn(., x);
   logn_var    = sev_quantile_logn(&varLevel, x);
run;
```

The preceding steps use a VaR level of 97.5%.

The following DATA step accomplishes the same task by reading the parameter estimates that were written to the Work.Est data set by the previous PROC HPSEVERITY step:

```
/*--- Compute scores (mean and var) for the       ---
  --- scoring data by using the OUTEST= data set  ---*/
data scoresWithOutest(keep=x1-x5 igauss_mean igauss_var logn_mean logn_var);
   array _x_{*} x1-x5;
   array _xparmIgauss_{5} _temporary_;
   array _xparmLogn_{5} _temporary_;

   retain _Theta0_ Alpha0;
   retain _Mu0_ Sigma0;
   *--- read parameter estimates for igauss and logn models ---*;
   if (_n_ = 1) then do;
      set est(where=(upcase(_MODEL_)='IGAUSS' and _TYPE_='EST'));
      _Theta0_ = Theta; Alpha0 = Alpha;
      do _i_=1 to dim(_x_);
         if (_x_(_i_) = .R) then _xparmIgauss_(_i_) = 0;
         else _xparmIgauss_(_i_) = _x_(_i_);
      end;

      set est(where=(upcase(_MODEL_)='LOGN' and _TYPE_='EST'));
      _Mu0_ = Mu; Sigma0 = Sigma;
      do _i_=1 to dim(_x_);
         if (_x_(_i_) = .R) then _xparmLogn_(_i_) = 0;
         else _xparmLogn_(_i_) = _x_(_i_);
      end;
   end;

   set reginput;

   *---  predict mean and VaR for inverse Gaussian  ---*;
   * first compute X'*beta for inverse Gaussian *;
   _xbeta_ = 0.0;
   do _i_ = 1 to dim(_x_);
      _xbeta_ = _xbeta_ + _xparmIgauss_(_i_) * _x_(_i_);
   end;
```

```
   * now compute scale for inverse Gaussian *;
   _SCALE_ = _Theta0_ * exp(_xbeta_);
   igauss_mean = igauss_mean(., _SCALE_, Alpha0);
   igauss_var = igauss_quantile(&varLevel, _SCALE_, Alpha0);

   *---  predict mean and VaR for lognormal         ---*;
   * first compute X'*beta for lognormal*;
   _xbeta_ = 0.0;
   do _i_ = 1 to dim(_x_);
      _xbeta_ = _xbeta_ + _xparmLogn_(_i_) * _x_(_i_);
   end;
   * now compute Mu=log(scale) for lognormal *;
   _MU_ = _Mu0_ + _xbeta_;
   logn_mean = logn_mean(., _MU_, Sigma0);
   logn_var = logn_quantile(&varLevel, _MU_, Sigma0);
run;
```

The "Values Comparison Summary" table in Output 23.9.2 shows that the difference between the estimates that are produced by both methods is within the acceptable machine precision. However, the comparison of the DATA step complexity of each method clearly shows that the method that uses the scoring functions is much easier because it saves a lot of programming effort. Further, new distribution functions, such as the *dist*_MEAN functions that are illustrated here, are automatically discovered and converted to scoring functions by PROC HPSEVERITY. That enables you to focus your efforts on writing the distribution function that computes your desired score, which needs to be done only once. Then, you can create and use the corresponding scoring functions multiple times with much less effort.

Output 23.9.2 Comparison of Mean and VaR Estimates of Two Scoring Methods

```
                    The COMPARE Procedure
          Comparison of WORK.SCORESWITHOUTEST with WORK.SCORES
              (Method=RELATIVE(0.0222), Criterion=1.0E-12)

NOTE: All values compared are within the equality criterion used. However, 40
      of the values compared are not exactly equal.
```

Example 23.10: Scale Regression with Rich Regression Effects

This example illustrates the use of regression effects that include CLASS variables and interaction effects.

Consider that you, as an actuary at an automobile insurance company, want to evaluate the effect of certain external factors on the distribution of the severity of the losses that your policyholders incur. Such analysis can help you determine the relative differences in premiums that you should charge to policyholders who have different characteristics. Assume that when you collect and record the information about each claim, you also collect and record some key characteristics of the policyholder and the vehicle that is involved in the claim. This example focuses on the following five factors: type of car, safety rating of the car, gender of the policyholder, education level of the policyholder, and annual household income of the policyholder (which can be thought of as a proxy for the luxury level of the car). Let these regressors be recorded in the variables CarType (1: sedan, 2: sport utility vehicle), CarSafety (scaled to be between 0 and 1, the safest being 1), Gender (1: female, 2: male), Education (1: high school graduate, 2: college graduate, 3: advanced degree holder), and Income (scaled by a factor of 1/100,000), respectively. Let the historical data about the severity of each loss be recorded in the LossAmount variable of the Work.Losses data set. Let the data set

also contain two additional variables, Deductible and Limit, that record the deductible and ground-up loss limit provisions, respectively, of the insurance policy that the policyholder has. The limit on ground-up loss is usually derived from the payment limit that a typical insurance policy states. Deductible serves as the left-truncation variable, and Limit serves as the right-censoring variable. The SAS statements that simulate an example of the Work.Losses data set are available in the PROC HPSEVERITY sample program *hsevex10.sas*.

The variables CarType, Education, and Gender each contain a known, finite set of discrete values. By specifying such variables as classification variables, you can separately identify the effect of each level of the variable on the severity distribution. For example, you might be interested in finding out how the magnitude of loss for a sport utility vehicle (SUV) differs from that for a sedan. This is an example of a main effect. You might also want to evaluate how the distribution of losses that are incurred by a policyholder with a college degree who drives a SUV differs from that of a policyholder with an advanced degree who drives a sedan. This is an example of an interaction effect. You can include various such types of effects in the scale regression model. For more information about the effect types, see the section "Specification and Parameterization of Model Effects" on page 1253. Analyzing such a rich set of regression effects can help you make more accurate predictions about the losses that a new applicant with certain characteristics might incur when he or she requests insurance for a specific vehicle, which can further help you with ratemaking decisions.

The following PROC HPSEVERITY step fits the scale regression model with a lognormal distribution to data in the Work.Losses data set, and stores the model and parameter estimate information in the Work.EstStore item store:

```
/* Fit scale regression model with different types of regression effects */
proc hpseverity data=losses outstore=eststore
     print=all plots=none;
   loss lossAmount / lt=deductible rc=limit;
   class carType gender education;
   scalemodel carType gender carSafety income education*carType
              income*gender carSafety*income;
   dist logn;
run;
```

The SCALEMODEL statement in the preceding PROC HPSEVERITY step includes two main effects (carType and gender), two singleton continuous effects (carSafety and income), one interaction effect (education*carType), one continuous-by-class effect (income*gender), and one polynomial continuous effect (carSafety*income). For more information about effect types, see Table 23.9, "GLM Parameterization of Classification Variables and Effects," on page 1256.

When you specify a CLASS statement, it is recommended that you observe the "Class Level Information" table. For this example, the table is shown in Output 23.10.1. Note that if you specify BY-group processing, then the class level information might change from one BY group to the next, potentially resulting in a different parameterization for each BY group.

Output 23.10.1 Class Level Information Table

The HPSEVERITY Procedure

Class Level Information

Class	Levels	Values
carType	2	SUV Sedan
gender	2	Female Male
education	3	AdvancedDegree College High School

The regression modeling results for the lognormal distribution are shown in Output 23.10.2. The "Initial Parameter Values and Bounds" table is important especially because the preceding PROC HPSEVERITY step uses the default GLM parameterization, which is a singular parameterization—that is, it results in some redundant parameters. As shown in the table, the redundant parameters correspond to the last level of each classification variable; this correspondence is a defining characteristic of a GLM parameterization. An alternative would be to use the reference parameterization by specifying the PARAM=REFERENCE option in the CLASS statement, which does not generate redundant parameters for effects that contain CLASS variables and enables you to specify a reference level for each CLASS variable.

Output 23.10.2 Initial Values for the Scale Regression Model with Class and Interaction Effects

Initial Parameter Values and Bounds

Parameter	Initial Value	Lower Bound	Upper Bound
Mu	4.88526	-709.78271	709.78271
Sigma	0.51283	1.05367E-8	Infty
carType SUV	0.56953	-709.78271	709.78271
carType Sedan	Redundant		
gender Female	0.41154	-709.78271	709.78271
gender Male	Redundant		
carSafety	-0.72742	-709.78271	709.78271
income	-0.33216	-709.78271	709.78271
carType*education SUV AdvancedDegree	0.31686	-709.78271	709.78271
carType*education SUV College	0.66361	-709.78271	709.78271
carType*education SUV High School	Redundant		
carType*education Sedan AdvancedDegree	-0.47841	-709.78271	709.78271
carType*education Sedan College	-0.25968	-709.78271	709.78271
carType*education Sedan High School	Redundant		
income*gender Female	-0.02112	-709.78271	709.78271
income*gender Male	Redundant		
carSafety*income	0.13084	-709.78271	709.78271

The convergence and optimization summary information in Output 23.10.3 indicates that the scale regression model for the lognormal distribution has converged with the default optimization technique in five iterations.

Output 23.10.3 Optimization Summary for the Scale Regression Model with Class and Interaction Effects

Convergence Status
Convergence criterion (GCONV=1E-8) satisfied.

Optimization Summary	
Optimization Technique	Trust Region
Iterations	5
Function Calls	14
Log Likelihood	-8286.8

The "Parameter Estimates" table in Output 23.10.4 shows the distribution parameter estimates and estimates for various regression effects. You can use the estimates for effects that contain CLASS variables to infer the relative influence of various CLASS variable levels. For example, on average, the magnitude of losses that are incurred by the female drivers is $\exp(0.44145) \approx 1.56$ times greater than that of male drivers, and an

SUV driver with an advanced degree incurs a loss that is on average $\exp(0.39393)/\exp(-0.35210) \approx 2.11$ times greater than the loss that a college-educated sedan driver incurs. Neither the continuous-by-class effect income*gender nor the polynomial continuous effect carSafety*income is significant in this example.

Output 23.10.4 Parameter Estimates for the Scale Regression with Class and Interaction Effects

Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|
Mu	5.08874	0.05768	88.23	<.0001
Sigma	0.55774	0.01119	49.86	<.0001
carType SUV	0.62459	0.04452	14.03	<.0001
gender Female	0.44145	0.04885	9.04	<.0001
carSafety	-0.82942	0.08371	-9.91	<.0001
income	-0.35212	0.07657	-4.60	<.0001
carType*education SUV AdvancedDegree	0.39393	0.07351	5.36	<.0001
carType*education SUV College	0.76532	0.05723	13.37	<.0001
carType*education Sedan AdvancedDegree	-0.61064	0.05387	-11.34	<.0001
carType*education Sedan College	-0.35210	0.03942	-8.93	<.0001
income*gender Female	-0.01486	0.06629	-0.22	0.8226
carSafety*income	0.07045	0.11447	0.62	0.5383

If you want to update the model when new claims data arrive, then you can potentially speed up the estimation process by specifying the OUTSTORE= item store that is created by the preceding PROC HPSEVERITY step as an INSTORE= item store in a new PROC HPSEVERITY step as follows:

```
/* Refit scale regression model on new data different types of regression effects */
proc hpseverity data=withNewLosses instore=eststore print=all plots=all;
   loss lossAmount / lt=deductible rc=limit;
   class carType gender education;
   scalemodel carType gender carSafety income education*carType
              income*gender carSafety*income;
   dist logn;
run;
```

PROC HPSEVERITY uses the parameter estimates in the INSTORE= item store to initialize the distribution and regression parameters.

References

D'Agostino, R. B., and Stephens, M., eds. (1986). *Goodness-of-Fit Techniques*. New York: Marcel Dekker.

Danielsson, J., de Haan, L., Peng, L., and de Vries, C. G. (2001). "Using a Bootstrap Method to Choose the Sample Fraction in Tail Index Estimation." *Journal of Multivariate Analysis* 76:226–248.

Dunn, P. K., and Smyth, G. K. (2005). "Series Evaluation of Tweedie Exponential Dispersion Model Densities." *Statistics and Computing* 15:267–280.

Frydman, H. (1994). "A Note on Nonparametric Estimation of the Distribution Function from Interval-Censored and Truncated Observations." *Journal of the Royal Statistical Society, Series B* 56:71–74.

Gentleman, R., and Geyer, C. J. (1994). "Maximum Likelihood for Interval Censored Data: Consistency and Computation." *Biometrika* 81:618–623.

Greenwood, M. (1926). "The Natural Duration of Cancer." In *Reports of Public Health and Related Subjects*, vol. 33, 1–26. London: Her Majesty's Stationery Office.

Hill, B. M. (1975). "A Simple General Approach to Inference about the Tail of a Distribution." *Annals of Statistics* 3:1163–1173.

Jørgensen, B. (1987). "Exponential Dispersion Models (with Discussion)." *Journal of the Royal Statistical Society, Series B* 49:127–162.

Kaplan, E. L., and Meier, P. (1958). "Nonparametric Estimation from Incomplete Observations." *Journal of the American Statistical Association* 53:457–481.

Klein, J. P., and Moeschberger, M. L. (1997). *Survival Analysis: Techniques for Censored and Truncated Data*. New York: Springer-Verlag.

Klugman, S. A., Panjer, H. H., and Willmot, G. E. (1998). *Loss Models: From Data to Decisions*. New York: John Wiley & Sons.

Koziol, J. A., and Green, S. B. (1976). "A Cramér–von Mises Statistic for Randomly Censored Data." *Biometrika* 63:466–474.

Lai, T. L., and Ying, Z. (1991). "Estimating a Distribution Function with Truncated and Censored Data." *Annals of Statistics* 19:417–442.

Lynden-Bell, D. (1971). "A Method of Allowing for Known Observational Selection in Small Samples Applied to 3CR Quasars." *Monthly Notices of the Royal Astronomical Society* 155:95–118.

Searle, S. R. (1971). *Linear Models*. New York: John Wiley & Sons.

Turnbull, B. W. (1976). "The Empirical Distribution Function with Arbitrarily Grouped, Censored, and Truncated Data." *Journal of the Royal Statistical Society, Series B* 38:290–295.

Tweedie, M. C. K. (1984). "An Index Which Distinguishes between Some Important Exponential Families." In *Statistics: Applications and New Directions—Proceedings of the Indian Statistical Institute Golden Jubilee International Conference*, edited by J. K. Ghosh, and J. Roy, 579–604. Calcutta: Indian Statistical Institute.

Chapter 24
The LOAN Procedure

Contents

Overview: LOAN Procedure **1356**
Getting Started: LOAN Procedure **1356**
Analyzing Fixed Rate Loans 1357
Analyzing Balloon Payment Loans 1358
Analyzing Adjustable Rate Loans 1359
Analyzing Buydown Rate Loans 1360
Loan Repayment Schedule 1361
Loan Comparison 1362
Syntax: LOAN Procedure **1365**
Functional Summary 1365
PROC LOAN Statement 1367
FIXED Statement 1367
BALLOON Statement 1371
ARM Statement 1372
BUYDOWN Statement 1374
COMPARE Statement 1374
Details: LOAN Procedure **1376**
Computational Details 1376
Loan Comparison Details 1378
OUT= Data Set 1379
OUTCOMP= Data Set 1379
OUTSUM= Data Set 1380
Printed Output 1381
ODS Table Names 1382
Examples: LOAN Procedure **1383**
Example 24.1: Discount Points for Lower Interest Rates 1383
Example 24.2: Refinancing a Loan 1384
Example 24.3: Prepayments on a Loan 1385
Example 24.4: Output Data Sets 1387
Example 24.5: Piggyback Loans 1388
References **1391**

Overview: LOAN Procedure

The LOAN procedure analyzes and compares fixed rate, adjustable rate, buydown, and balloon payment loans. The LOAN procedure computes the loan parameters and outputs the loan summary information for each loan.

Multiple loan specifications can be processed and compared in terms of economic criteria such as after-tax or before-tax present worth of cost and true interest rate, breakeven of periodic payment and of interest paid, and outstanding balance at different periods in time. PROC LOAN selects the best alternative in terms of the specified economic criterion for each loan comparison period.

The LOAN procedure allows various payment and compounding intervals (including continuous compounding) and uniform or lump sum prepayments for a loan. Down payments, discount points, and other initialization costs can be included in the loan analysis and comparison.

The LOAN procedure does not support an input data set. All loans analyzed are specified with statements in the PROC LOAN step. The SAS DATA step provides a function MORT that can be used for data-driven analysis of many fixed-rate mortgage or installment loans. However, the MORT function supports only simple fixed rate loans.

Getting Started: LOAN Procedure

PROC LOAN supports four types of loans. You specify each type of loan with the corresponding statement: FIXED, BALLOON, ARM, and BUYDOWN.

- FIXED—Fixed rate loans have a constant interest rate and periodic payment throughout the life of the loan.
- BALLOON—Balloon payment loans are fixed rate loans with lump sum payments in certain payment periods in addition to the constant periodic payment.
- ARM—Adjustable rate loans are those in which the interest rate and periodic payment vary over the life of the loan. The future interest rates of an adjustable rate loan are not known with certainty, but they will vary within specified limits according to terms stated in the loan agreement. In practice, the rate adjustment terms vary. PROC LOAN offers a flexible set of options to capture a wide variety of rate adjustment terms.
- BUYDOWN—Buydown rate loans are similar to adjustable rate loans, but the interest rate adjustments are predetermined at the initialization of the loan, usually by paying interest points at the time of loan initialization.

Analyzing Fixed Rate Loans

The most common loan analysis is the calculation of the periodic payment when the loan amount, life, and interest rate are known. The following PROC LOAN statements analyze a 15-year (180 monthly payments) fixed rate loan for $100,000 with an annual nominal interest rate of 7.5%:

```
proc loan;
   fixed amount=100000 rate=7.5 life=180;
run;
```

Another parameter the PROC LOAN statement can compute is the maximum amount you can borrow given the periodic payment you can afford and the rates available in the market. The following SAS statements analyze a loan for 180 monthly payments of $900, with a nominal annual rate of 7.5%, and compute the maximum amount that can be borrowed:

```
proc loan;
   fixed payment=900 rate=7.5 life=180;
run;
```

Assume that you want to borrow $100,000 and can pay $900 a month. You know that the lender charges a 7.5% nominal interest rate compounded monthly. To determine how long it will take you to pay off your debt, use the following statements:

```
proc loan;
   fixed amount=100000 payment=900 rate=7.5;
run;
```

Sometimes, a loan is expressed in terms of the amount borrowed and the amount and number of periodic payments. In this case, you want to calculate the annual nominal rate charged on the loan to compare it to other alternatives. The following statements analyze a loan of $100,000 paid in 180 monthly payments of $800:

```
proc loan;
   fixed amount=100000 payment=800 life=180;
run;
```

There are four basic parameters that define a loan: life (number of periodic payments), principal amount, interest rate, and the periodic payment amount. PROC LOAN calculates the missing parameter among these four. Loan analysis output includes a loan summary table and an amortization schedule.

You can use the START= and LABEL= options to enhance your output. The START= option specifies the date of loan initialization and dates all the output accordingly. The LABEL= specification is used to label all output that corresponds to a particular loan; it is especially useful when multiple loans are analyzed. For example, the preceding statements for the first fixed rate loan are revised to include the START= and LABEL= options as follows:

```
proc loan start=1998:12;
   fixed amount=100000 rate=7.5 life=180
         label='BANK1, Fixed Rate';
run;
```

Loan Summary Table

The loan summary table is produced by default and contains loan analysis information. It shows the principal amount, the costs at the time of loan initialization (down payment, discount points, and other loan initialization costs), the total payment and interest, the initial nominal and effective interest rates, payment and compounding intervals, the length of the loan in the time units specified, the start and end dates (if specified), a list of nominal and effective interest rates, and periodic payments throughout the life of the loan.

Figure 24.1 shows the loan summary table for the fixed rate loan labeled "BANK1, Fixed Rate."

Figure 24.1 Fixed Rate Loan Summary

The LOAN Procedure

Fixed Rate Loan Summary

BANK1, Fixed Rate

Downpayment	0.00	**Principal Amount**	100000.00
Initialization	0.00	**Points**	0.00
Total Interest	66862.61	**Nominal Rate**	7.5000%
Total Payment	166862.61	**Effective Rate**	7.7633%
Pay Interval	MONTHLY	**Compounding**	MONTHLY
No. of Payments	180	**No. of Compoundings**	180
Start Date	DEC1998	**End Date**	DEC2013

Rates and Payments for BANK1, Fixed Rate

Date	Nominal Rate	Effective Rate	Payment
DEC1998	7.5000%	7.7633%	927.01

The loan is initialized in December 1998 and paid off in December 2013. The monthly payment is calculated to be $927.01, and the effective interest rate is 7.7633%. Over the 15 years, $66,862.61 is paid for interest charges on the loan.

Analyzing Balloon Payment Loans

You specify balloon payment loans like fixed rate loans, with the additional specification of the balloon payments. Assume you have an alternative to finance the $100,000 investment with a 15-year balloon payment loan. The annual nominal rate is 7.5%, as in the fixed rate loan. The terms of the loan require two balloon payments of $2000 and $1000 at the 15th and 48th payment periods, respectively. These balloon payments keep the periodic payment lower than that of the fixed rate loan. The balloon payment loan is defined by the following BALLOON statement:

```
proc loan start=1998:12;
   balloon amount=100000 rate=7.5 life=180
           balloonpayment=(15=2000 48=1000)
           label = 'BANK2, with Balloon Payment';
run;
```

List of Balloon Payments

In addition to the information for the fixed rate loan, the "Loan Summary Table" for the balloon payment loan includes a list of balloon payments in the list of rates and payments. For example, the balloon payment loan described previously includes two balloon payments, as shown in Figure 24.2.

Figure 24.2 List of Rates and Payments for a Balloon Payment Loan

The LOAN Procedure

Rates and Payments for BANK2, with Balloon Payment			
Date	**Nominal Rate**	**Effective Rate**	**Payment**
DEC1998	7.5000%	7.7633%	903.25

Balloon Period	**Payment**
MAR2000	2000.00
DEC2002	1000.00

The periodic payment for the balloon payment loan is $23.76 less than that of the fixed rate loan.

Analyzing Adjustable Rate Loans

In addition to specifying the basic loan parameters, you need to specify the terms of the rate adjustments for an adjustable rate loan. There are many ways of stating the rate adjustment terms, and PROC LOAN facilitates all of them. For details, see the section "Rate Adjustment Terms Options" on page 1372.

Assume that you have an alternative to finance the $100,000 investment with a 15-year adjustable rate loan with an initial annual nominal interest rate of 5.5%. The rate adjustment terms specify a 0.5% annual cap, a 2.5% life cap, and a rate adjustment every 12 months. *Annual cap* refers to the maximum increase in interest rate per adjustment period, and *life cap* refers to the maximum increase over the life of the loan. The following ARM statement specifies this adjustable rate loan by assuming the interest rate adjustments will always increase by the maximum allowed by the terms of the loan. These assumptions are specified by the WORSTCASE and CAPS= options, as shown in the following statements:

```
proc loan start=1998:12;
   arm amount=100000 rate=5.5 life=180 worstcase
       caps=(0.5, 2.5)
       label='BANK3, Adjustable Rate';
run;
```

List of Rates and Payments for Adjustable Rate Loans

The list of rates and payments in the loan summary table for the adjustable rate loans reflects the changes in the interest rates and payments and the dates these changes become effective. For the adjustable rate loan described previously, Figure 24.3 shows the list of rates and payments that indicate five annual rate adjustments in addition to the initial rate and payment.

Figure 24.3 List of Rates and Payments for an Adjustable Rate Loan

The LOAN Procedure

Rates and Payments for BANK3, Adjustable Rate			
Date	Nominal Rate	Effective Rate	Payment
DEC1998	5.5000%	5.6408%	817.08
JAN2000	6.0000%	6.1678%	842.33
JAN2001	6.5000%	6.6972%	866.44
JAN2002	7.0000%	7.2290%	889.32
JAN2003	7.5000%	7.7633%	910.88
JAN2004	8.0000%	8.3000%	931.03

Notice that the periodic payment of the adjustable rate loan as of January 2004 ($931.03) exceeds that of the fixed rate loan ($927.01).

Analyzing Buydown Rate Loans

A 15-year buydown rate loan is another alternative to finance the $100,000 investment. The nominal annual interest rate is 6.5% initially and will increase to 8% and 9% as of the 24th and 48th payment periods, respectively. The nominal annual interest rate is lower than that of the fixed rate alternative, at the cost of a 1% discount point ($1000) paid at the initialization of the loan. The following BUYDOWN statement represents this loan alternative:

```
proc loan start=1998:12;
   buydown amount=100000 rate=6.5 life=180
           buydownrates=(24=8 48=9) pointpct=1
           label='BANK4, Buydown';
run;
```

List of Rates and Payments for Buydown Rate Loans

Figure 24.4 shows the list of rates and payments in the loan summary table. It reflects the two rate adjustments and the corresponding monthly payments as well as the initial values for these parameters. As of December 2000, the periodic payment of the buydown loan exceeds the periodic payment for any of the other alternatives.

Figure 24.4 List of Rates and Payments for a Buydown Rate Loan

The LOAN Procedure

Rates and Payments for BANK4, Buydown			
Date	**Nominal Rate**	**Effective Rate**	**Payment**
DEC1998	6.5000%	6.6972%	871.11
DEC2000	8.0000%	8.3000%	946.50
DEC2002	9.0000%	9.3807%	992.01

Loan Repayment Schedule

In addition to the loan summary, you can print a loan repayment (amortization) schedule for each loan. For each payment period, this schedule contains the year and period within the year (or date, if the START= option is specified), the principal balance at the beginning of the period, the total payment, interest payment, principal repayment for the period, and the principal balance at the end of the period.

To print the first year of the amortization schedule for the fixed rate loan shown in Figure 24.5, use the following statements:

```
proc loan start=1998:12;
   fixed amount=100000 rate=7.5 life=180
         schedule=1
         label='BANK1, Fixed Rate';
run;
```

Figure 24.5 Loan Repayment Schedule for the First Year

The LOAN Procedure

Loan Repayment Schedule					
	BANK1, Fixed Rate				
Date	**Beginning Outstanding**	**Payment**	**Interest Payment**	**Principal Repayment**	**Ending Outstanding**
DEC1998	100000.00	0.00	0.00	0.00	100000.00
DEC1998	100000.00	0.00	0.00	0.00	100000.00
JAN1999	100000.00	927.01	625.00	302.01	99697.99
FEB1999	99697.99	927.01	623.11	303.90	99394.09
MAR1999	99394.09	927.01	621.21	305.80	99088.29
APR1999	99088.29	927.01	619.30	307.71	98780.58
MAY1999	98780.58	927.01	617.38	309.63	98470.95
JUN1999	98470.95	927.01	615.44	311.57	98159.38
JUL1999	98159.38	927.01	613.50	313.51	97845.87
AUG1999	97845.87	927.01	611.54	315.47	97530.40
SEP1999	97530.40	927.01	609.57	317.44	97212.96
OCT1999	97212.96	927.01	607.58	319.43	96893.53
NOV1999	96893.53	927.01	605.58	321.43	96572.10
DEC1999	96572.10	927.01	603.58	323.43	96248.67
DEC1999	100000.00	11124.12	7372.79	3751.33	96248.67

The principal balance at the end of one year is $96,248.67. The total payment for the year is $11,124.12, of which $3,751.33 went toward principal repayment.

You can also print the amortization schedule with annual summary information or for a specified number of years. The SCHEDULE=YEARLY option produces an annual summary loan amortization schedule, which is useful for loans with a long life. For example, to print the annual summary loan repayment schedule for the buydown loan shown in Figure 24.6, use the following statements:

```
proc loan start=1998:12;
   buydown amount=100000 rate=6.5 life=180
           buydownrates=(24=8 48=9) pointpct=1
           schedule=yearly
           label='BANK4, Buydown';
run;
```

Figure 24.6 Annual Summary Loan Repayment Schedule

The LOAN Procedure

Loan Repayment Schedule

BANK4, Buydown

Year	Beginning Outstanding	Payment	Interest Payment	Principal Repayment	Ending Outstanding
1998	100000.00	1000.00	0.00	0.00	100000.00
1999	100000.00	10453.32	6380.07	4073.25	95926.75
2000	95926.75	10528.71	6222.21	4306.50	91620.25
2001	91620.25	11358.00	7178.57	4179.43	87440.82
2002	87440.82	11403.51	6901.12	4502.39	82938.43
2003	82938.43	11904.12	7276.64	4627.48	78310.95
2004	78310.95	11904.12	6842.58	5061.54	73249.41
2005	73249.41	11904.12	6367.76	5536.36	67713.05
2006	67713.05	11904.12	5848.43	6055.69	61657.36
2007	61657.36	11904.12	5280.35	6623.77	55033.59
2008	55033.59	11904.12	4659.00	7245.12	47788.47
2009	47788.47	11904.12	3979.34	7924.78	39863.69
2010	39863.69	11904.12	3235.96	8668.16	31195.53
2011	31195.53	11904.12	2422.83	9481.29	21714.24
2012	21714.24	11904.12	1533.41	10370.71	11343.53
2013	11343.53	11904.09	560.56	11343.53	0.00

Loan Comparison

The LOAN procedure can compare alternative loans on the basis of different economic criteria and help select the most desirable loan. You can compare alternative loans through different points in time. The economic criteria offered by PROC LOAN are:

- outstanding principal balance—that is, the unpaid balance of the loan

- present worth of cost—that is, before-tax or after-tax net value of the loan cash flow through the comparison period. The cash flow includes all payments, discount points, initialization costs, down payment, and the outstanding principal balance at the comparison period.
- true interest rate—that is, before-tax or after-tax effective annual interest rate charged on the loan. The cash flow includes all payments, discount points, initialization costs, and the outstanding principal balance at the specified comparison period.
- periodic payment
- the total interest paid on the loan

The figures for present worth of cost, true interest rate, and interest paid are reported on the cash flow through the comparison period. The reported outstanding principal balance and the periodic payment are the values as of the comparison period.

The COMPARE statement specifies the type of comparison and the periods of comparison. For each period specified in the COMPARE statement, a loan comparison report is printed that also indicates the best alternative. Different criteria can lead to selection of different alternatives. Also, the period of comparison might change the desirable alternative. See the section "Loan Comparison Details" on page 1378 for further information.

Comparison of 15-Year versus 30-Year Loan Alternatives

An issue that arises in the purchase of a house is the length of the loan life. Residential home loans are often for 15 or 30 years. Ordinarily, 15-year loans have a lower interest rate but higher periodic payments than 30-year loans. A comparison of both loans might identify the better loan for your means and needs. The following SAS statements compare two such loans:

```
proc loan start=1998:12 amount=120000;
   fixed rate=7.5 life=360 label='30 year loan';
   fixed rate=6.5 life=180 label='15 year loan';
   compare;
run;
```

Default Loan Comparison Report

The default loan comparison report in Figure 24.7 shows the ending outstanding balance, periodic payment, interest paid, and before-tax true rate at the end of 30 years. In the case of the default loan comparison, the selection of the best alternative is based on minimization of the true rate.

Figure 24.7 Default Loan Comparison Report

The LOAN Procedure

Loan Comparison Report

Analysis through DEC2028

Loan Label	Ending Outstanding	Payment	Interest Paid	True Rate
30 year loan	0.00	835.48	182058.02	7.76
15 year loan	0.00	1044.95	68159.02	6.70

Note: "15 year loan" is the best alternative based on true rate analysis through DEC2028.

Based on true rate, the best alternative is the 15-year loan. However, if the objective were to minimize the periodic payment, the 30-year loan would be the more desirable.

Comparison of Fixed Rate and Adjustable Rate Loans

Suppose you want to compare a fixed rate loan to an adjustable rate alternative. The nominal interest rate on the adjustable rate loan is initially 1.5% lower than the fixed rate loan. The future rates of the adjustable rate loan are calculated using the worst case scenario.

The interest paid on a loan might be deductible for tax purposes, depending on the purpose of the loan and applicable laws. In the following example, the TAXRATE=28 (income tax rate) option in the COMPARE statement bases the calculations of true interest rate on the after-tax cash flow. Assume, also, that you are uncertain as to how long you will keep this property. The AT=(60 120) option, as shown in the following example, produces two loan comparison reports through the end of the 5th and the 10th years, respectively:

```
proc loan start=1998:12 amount=120000 life=360;
   fixed rate=7.5 label='BANK1, Fixed Rate';
   arm   rate=6.0 worstcase caps=(0.5 2.5)
         label='BANK3, Adjustable Rate';
   compare taxrate=28 at=(60 120);
run;
```

After-Tax Loan Comparison Reports

The two loan comparison reports in Figure 24.8 and Figure 24.9 show the ending outstanding balance, periodic payment, interest paid, and after-tax true rate at the end of five years and ten years, respectively.

Figure 24.8 Loan Comparison Report as of December 2003

The LOAN Procedure

Loan Comparison Report

Analysis through DEC2003

Loan Label	Ending Outstanding	Payment	Interest Paid	True Rate
BANK1, Fixed Rate	113540.74	839.06	43884.34	5.54
BANK3, Adjustable Rate	112958.49	871.83	40701.93	5.11

Note: "BANK3, Adjustable Rate" is the best alternative based on true rate analysis through DEC2003.

Figure 24.9 Loan Comparison Report as of December 2008

Loan Comparison Report

Analysis through DEC2008

Loan Label	Ending Outstanding	Payment	Interest Paid	True Rate
BANK1, Fixed Rate	104153.49	839.06	84840.69	5.54
BANK3, Adjustable Rate	104810.98	909.57	87128.62	5.60

Note: "BANK1, Fixed Rate" is the best alternative based on true rate analysis through DEC2008.

The loan comparison report through December 2003 picks the adjustable rate loan as the best alternative, whereas the report through December 2008 shows the fixed rate loan as the better alternative. This implies that if you intend to keep the loan for 10 years or longer, the best alternative is the fixed rate alternative. Otherwise, the adjustable rate loan is the better alternative in spite of the worst-case scenario. Further analysis shows that the actual breakeven of true interest rate occurs at August 2008. That is, the desirable alternative switches from the adjustable rate loan to the fixed rate loan in August 2008.

Note that, under the assumption of worst-case scenario for the rate adjustments, the periodic payment for the adjustable rate loan already exceeds that of the fixed rate loan on December 2003 (as of the rate adjustment on January 2003 to be exact). If the objective were to minimize the periodic payment, the fixed rate loan would have been more desirable as of December 2003. However, all of the other criteria at that point still favor the adjustable rate loan.

Syntax: LOAN Procedure

The following statements are used with PROC LOAN:

PROC LOAN *options* ;
  FIXED *options* ;
  BALLOON *options* ;
  ARM *options* ;
  BUYDOWN *options* ;
  COMPARE *options* ;

Functional Summary

Table 24.1 summarizes the statements and options that control the LOAN procedure. Many of the loan specification options can be used on all of the statements except the COMPARE statement. For these options, the statement column is left blank. Options specific to a type of loan indicate the statement name.

Table 24.1 LOAN Functional Summary

Description	Statement	Option
Statements		
specify an adjustable rate loan	ARM	
specify a balloon payment loan	BALLOON	
specify a buydown rate loan	BUYDOWN	
specify loan comparisons	COMPARE	
specify a fixed rate loan	FIXED	
Data Set Options		
specify output data set for loan summary	PROC LOAN	OUTSUM=
specify output data set for repayment schedule		OUT=
specify output data set for loan comparison	COMPARE	OUTCOMP=
Printing Control Options		
suppress printing of loan summary report		NOSUMMARYPRINT

Description	Statement	Option
suppress all printed output		NOPRINT
print amortization schedule		SCHEDULE=
suppress printing of loan comparison report	COMPARE	NOCOMPRINT
Required Specifications		
specify the loan amount		AMOUNT=
specify life of loan as number of payments		LIFE=
specify the periodic payment		PAYMENT=
specify the initial annual nominal interest rate		RATE=
Loan Specifications Options		
specify loan amount as percentage of price		AMOUNTPCT=
specify time interval between compoundings		COMPOUND=
specify down payment at loan initialization		DOWNPAYMENT=
specify down payment as percentage of price		DOWNPAYPCT=
specify amount paid for loan initialization		INITIAL=
specify initialization costs as a percent		INITIALPCT=
specify time interval between payments		INTERVAL=
specify label for the loan		LABEL=
specify amount paid for discount points		POINTS=
specify discount points as a percent		POINTPCT=
specify uniform or lump sum prepayments		PREPAYMENTS=
specify the purchase price		PRICE=
specify number of decimal places for rounding		ROUND=
specify the date of loan initialization		START=
Balloon Payment Loan Specification Option		
specify the list of balloon payments	BALLOON	BALLOONPAYMENT=
Rate Adjustment Terms Options		
specify frequency of rate adjustments	ARM	ADJUSTFREQ=
specify periodic and life cap on rate adjustment	ARM	CAPS=
specify maximum rate adjustment	ARM	MAXADJUST=
specify maximum annual nominal interest rate	ARM	MAXRATE=
specify minimum annual nominal interest rate	ARM	MINRATE=
Rate Adjustment Case Options		
specify best-case (optimistic) scenario	ARM	BESTCASE
specify predicted interest rates	ARM	ESTIMATEDCASE=
specify constant rate	ARM	FIXEDCASE
specify worst case (pessimistic) scenario	ARM	WORSTCASE
Buydown Rate Loan Specification Option		
specify list of nominal interest rates	BUYDOWN	BUYDOWNRATES=

Description	Statement	Option
Loan Comparison Options		
specify all comparison criteria	COMPARE	ALL
specify the loan comparison periods	COMPARE	AT=
specify breakeven analysis of the interest paid	COMPARE	BREAKINTEREST
specify breakeven analysis of periodic payment	COMPARE	BREAKPAYMENT
specify minimum attractive rate of return	COMPARE	MARR=
specify present worth of cost analysis	COMPARE	PWOFCOST
specify the income tax rate	COMPARE	TAXRATE=
specify true interest rate analysis	COMPARE	TRUEINTEREST

PROC LOAN Statement

PROC LOAN *options* **;**

The OUTSUM= option can be used in the PROC LOAN statement. In addition, the following loan specification options can be specified in the PROC LOAN statement to be used as defaults for all loans unless otherwise specified for a given loan:

AMOUNT=	INTERVAL=	POINTPCT=
AMOUNTPCT=	LABEL=	PREPAYMENTS=
COMPOUND=	LIFE=	PRICE=
DOWNPAYMENT=	NOSUMMARYPRINT	RATE=
DOWNPAYPCT=	NOPRINT	ROUND=
INITIAL=	PAYMENT=	START=
INITIALPCT=	POINTS=	SCHEDULE=

Output Option

OUTSUM= *SAS-data-set*

creates an output data set that contains loan summary information for all loans other than those for which a different OUTSUM= output data set is specified.

FIXED Statement

FIXED *options* **;**

The FIXED statement specifies a fixed rate and periodic payment loan. It can be specified using the options that are common to all loan statements. The FIXED statement options are listed in this section.

You must specify three of the following options in each loan statement: AMOUNT=, LIFE=, RATE=, and PAYMENT=. The LOAN procedure calculates the fourth parameter based on the values you give the other three. If you specify all four of the options, the PAYMENT= specification is ignored, and the periodic payment is recalculated for consistency.

As an alternative to specifying the AMOUNT= option, you can specify the PRICE= option along with one of the following options to facilitate the calculation of the loan amount: AMOUNTPCT=, DOWNPAYMENT=, or DOWNPAYPCT=.

Required Specifications

AMOUNT=*amount*

A=*amount*

specifies the loan amount (the outstanding principal balance at the initialization of the loan).

LIFE=*n*

L=*n*

gives the life of the loan in number of payments. (The payment frequency is specified by the INTERVAL= option.) For example, if the life of the loan is 10 years with monthly payments, use LIFE=120 and INTERVAL=MONTH (default) to indicate a 10-year loan in which 120 monthly payments are made.

PAYMENT=*amount*

P=*amount*

specifies the periodic payment. For ARM and BUYDOWN loans where the periodic payment might change, the PAYMENT= option specifies the initial amount of the periodic payment.

RATE=*rate*

R=*rate*

specifies the initial annual (nominal) interest rate in percent notation. The rate specified must be in the range 0% to 120%. For example, use RATE=12.75 for a 12.75% loan. For ARM and BUYDOWN loans, where the rate might change over the life of the loan, the RATE= option specifies the initial annual interest rate.

Specification Options

AMOUNTPCT=*value*

APCT=*value*

specifies the loan amount as a percentage of the purchase price (PRICE= option). The AMOUNTPCT= specification is used to calculate the loan amount if the AMOUNT= option is not specified. The value specified must be in the range 1% to 100%.

If both the AMOUNTPCT= and DOWNPAYPCT= options are specified and the sum of their values is not equal to 100, the value of the downpayment percentage is set equal to 100 minus the value of the amount percentage.

COMPOUND=*time-unit*

specifies the time interval between compoundings. The default is the time unit given by the INTERVAL= option. If the INTERVAL= option is not used, then the default is COMPOUND=MONTH. The following time units are valid COMPOUND= values: CONTINUOUS, DAY, SEMIMONTH, MONTH, QUARTER, SEMIYEAR, and YEAR. The compounding interval is used to calculate the simple interest rate per payment period from the nominal annual interest rate or vice versa.

DOWNPAYMENT=*amount*

DP=*amount*

specifies the down payment at the initialization of the loan. The down payment is included in the calculation of the present worth of cost but not in the calculation of the true interest rate. The after-tax analysis assumes that the down payment is not tax-deductible. (Specify after-tax analysis with the TAXRATE= option in the COMPARE statement.)

DOWNPAYPCT=*value*

DPCT=*value*

specifies the down payment as a percentage of the purchase price (PRICE= option). The DOWNPAYPCT= specification is used to calculate the down payment amount if you do not specify the DOWNPAYMENT= option. The value you specify must be in the range 0% to 99%.

If you specified both the AMOUNTPCT= and DOWNPAYPCT= options and the sum of their values is not equal to 100, the value of the downpayment percentage is set equal to 100 minus the value of the amount percentage.

INITIAL=*amount*

INIT=*amount*

specifies the amount paid for loan initialization other than the discount points and down payment. This amount is included in the calculation of the present worth of cost and the true interest rate. The after-tax analysis assumes that the initial amount is not tax-deductible. (After-tax analysis is specified by the TAXRATE= option in the COMPARE statement.)

INITIALPCT=*value*

INITPCT=*value*

specifies the initialization costs as a percentage of the loan amount (AMOUNT= option). The INITIALPCT= specification is used to calculate the amount paid for loan initialization if you do not specify the INITIAL= option. The value you specify must be in the range of 0% to 100%.

INTERVAL=*time-unit*

gives the time interval between periodic payments. The default is INTERVAL=MONTH. The following time units are valid INTERVAL values: SEMIMONTH, MONTH, QUARTER, SEMIYEAR, and YEAR.

LABEL=*'loan-label'*

specifies a label for the loan. If you specify the LABEL= option, all output related to the loan is labeled accordingly. If you do not specify the LABEL= option, the loan is labeled by sequence number.

POINTS=*amount*

PNT=*amount*

specifies the amount paid for discount points at the initialization of the loan. This amount is included in the calculation of the present worth of cost and true interest rate. The amount paid for discount points is assumed to be tax-deductible in after-tax analysis (that is, if the TAXRATE= option is specified in the COMPARE statement).

POINTPCT=*value*

PNTPCT=*value*

specifies the discount points as a percentage of the loan amount (AMOUNT= option). The POINTPCT= specification is used to calculate the amount paid for discount points if you do not specify the POINTS= option. The value you specify must be in the range of 0% to 100%.

PREPAYMENTS=*amount*

PREPAYMENTS=(*date1=prepayment1 date2=prepayment2 ...* **)**

PREPAYMENTS=(*period1=prepayment1 period2=prepayment2 ...* **)**

PREP=

specifies either a uniform prepayment *p* throughout the life of the loan or lump sum prepayments. A uniform prepayment *p* is assumed to be paid with each periodic payment. Specify lump sum prepayments by pairs of periods (or dates) and respective prepayment amounts.

You can specify the prepayment periods as dates if you specify the START= option. Prepayment periods or dates and the respective prepayment amounts must be in time sequence. The prepayments are treated as principal payments, and the outstanding principal balance is adjusted accordingly. In the adjustable rate and buydown rate loans, if there is a rate adjustment after prepayments, the adjusted periodic payment is calculated based on the outstanding principal balance. The prepayments do not result in periodic payment amount adjustments in fixed rate and balloon payment loans.

PRICE=*amount*

PRC=*amount*

specifies the purchase price, which is the loan amount plus the down payment. If you specify the PRICE= option along with the loan amount (AMOUNT= option) or the down payment (DOWNPAYMENT= option), the value of the other one is calculated.

If you specify the PRICE= option with the AMOUNTPCT= or DOWNPAYPCT= options, the loan amount and the downpayment are calculated.

ROUND=*n*

ROUND=NONE

specifies the number of decimal places to which the monetary amounts are rounded for the loan. Valid values for *n* are integers from 0 to 6. If you specify ROUND=NONE, the values are not rounded off internally, but the printed output is rounded off to two decimal places. The default is ROUND=2.

START=*SAS-date-literal*

START=*yyyy:period*

S=

gives the date of loan initialization. The first payment is assumed to be one payment interval after the start date. For example, you can specify the START= option as **`START='1APR2010'D`** or as **`START=2010:3`**, where 3 is the third payment interval within the year 2010. If INTERVAL=QUARTER, 3 refers to the third quarter. If you specify the START= option, all output for the particular loan is dated accordingly.

Output Options

NOSUMMARYPRINT

NOSUMPR

suppresses the printing of the loan summary report. The NOSUMMARYPRINT option is usually used when an OUTSUM= data set is created to store loan summary information.

NOPRINT

NOP

suppresses all printed output for the loan.

OUT=*SAS-data-set*

writes the loan amortization schedule to an output data set.

OUTSUM=*SAS-data-set*

writes the loan summary for the individual loan to an output data set.

SCHEDULE

SCHEDULE=*nyears*

SCHEDULE=YEARLY

SCHED

prints the amortization schedule for the loan. SCHEDULE=*nyears* specifies the number of years the printed amortization table covers. If you omit the number of years or specify a period longer than the loan life, the schedule is printed for the full term of the loan. SCHEDULE=YEARLY prints yearly summary information in the amortization schedule rather than the full amortization schedule. SCHEDULE=YEARLY is useful for long-term loans.

BALLOON Statement

BALLOON *options* **;**

The BALLOON statement specifies a fixed rate loan with scheduled balloon payments in addition to the periodic payment. The following option is used in the BALLOON statement, in addition to the required options listed under the FIXED statement:

BALLOONPAYMENT=(*date1=payment1 date2=payment2 ...* **)**

BALLOONPAYMENT=(*period1=payment1 period2=payment2 ...* **)**

BPAY=(*date1=payment1 date2=payment2 ...* **)**

BPAY=(*period1=payment1 period2=payment2 ...* **)**

specifies pairs of periods and amounts of balloon (lump sum) payments in excess of the periodic payment during the life of the loan. You can also specify the balloon periods as dates if you specify the START= option. The dates are specified as SAS date literals. For example, `BALLOONPAYMENT=( '1MAR2011'D=1000 )` specifies a payment of 1000 in March of 2011.

If you do not specify this option, the calculations are identical to a loan specified in a FIXED statement. Balloon periods (or dates) and the respective balloon payments must be in time sequence.

ARM Statement

ARM *options* **;**

The ARM statement specifies an adjustable rate loan where the future interest rates are not known with certainty but will vary within specified limits according to the terms stated in the loan agreement. In practice, the adjustment terms vary. Adjustments in the interest rate can be captured using the ARM statement options.

In addition to the required specifications and options listed under the FIXED statement, you can use the following options with the ARM statement.

Rate Adjustment Terms Options

ADJUSTFREQ=*n*

ADF=*n*

specifies the number of periods, in terms of the INTERVAL= specification, between rate adjustments. INTERVAL=MONTH ADJUSTFREQ=6 indicates that the nominal interest rate can be adjusted every six months until the life cap or maximum rate (whichever is specified) is reached. The default is ADJUSTFREQ=12. The periodic payment is adjusted every adjustment period even if there is no rate change; therefore, if prepayments are made (as specified with the PREPAYMENTS= option), the periodic payment might change even if the nominal rate does not.

CAPS=(*periodic-cap, life-cap* **)**

specifies the maximum interest rate adjustment, in percent notation, allowed by the loan agreement. The *periodic cap* specifies the maximum adjustment allowed at each adjustment period. The *life cap* specifies the maximum total adjustment over the life of the loan. For example, a loan specified with CAPS=(0.5, 2) indicates that the nominal interest rate can change by 0.5% each adjustment period, and the annual nominal interest rate throughout the life of the loan will be within a 2% range of the initial annual nominal rate.

MAXADJUST=*rate*

MAXAD=*rate*

specifies the maximum rate adjustment, in percent notation, allowed at each adjustment period. Use the MAXADJUST= option with the MAXRATE= and MINRATE= options. The initial nominal rate plus the maximum adjustment should not exceed the specified MAXRATE= value. The initial nominal rate minus the maximum adjustment should not be less than the specified MINRATE= value.

MAXRATE=*rate*

MAXR=*rate*

specifies the maximum annual nominal rate, in percent notation, that might be charged on the loan. The maximum annual nominal rate should be greater than or equal to the initial annual nominal rate specified with the RATE= option.

MINRATE=*rate*

MINR=*rate*

specifies the minimum annual nominal rate, in percent notation, that might be charged on the loan. The minimum annual nominal rate should be less than or equal to the initial annual nominal rate specified with the RATE= option.

Rate Adjustment Case Options

PROC LOAN supports four rate adjustment scenarios for analysis of adjustable rate loans: pessimistic (WORSTCASE), optimistic (BESTCASE), no-change (FIXEDCASE), and estimated (ESTIMATEDCASE). The estimated case enables you to analyze the adjustable rate loan with your predictions of future interest rates. The default is worst-case analysis. If more than one case is specified, worst-case analysis is performed. You can specify options for adjustable rate loans as follows:

BESTCASE

B

specifies a best-case analysis. The best-case analysis assumes that the interest rate charged on the loan will reach its minimum allowed limits at each adjustment period and over the life of the loan. If you use the BESTCASE option, you must specify either the CAPS= option or the MINRATE= and MAXADJUST= options.

ESTIMATEDCASE=(*date1=rate1 date2=rate2 ...* **)**

ESTIMATEDCASE=(*period1=rate1 period2=rate2 ...* **)**

ESTC=

specifies an estimated case analysis that indicates the rate adjustments will follow the rates you predict. This option specifies pairs of periods and estimated nominal interest rates.

The ESTIMATEDCASE= option can specify adjustments that cannot fit into the BESTCASE, WORSTCASE, or FIXEDCASE specifications, or "what-if" type analysis. If you specify the START= option, you can also specify the estimation periods as dates, in the form of SAS date literals. Estimated rates and the respective periods must be in time sequence.

If the estimated period falls between two adjustment periods (determined by ADJUSTFREQ= option), the rate is adjusted in the next adjustment period. The nominal interest rate charged on the loan is constant between two adjustment periods.

If any of the MAXRATE=, MINRATE=, CAPS=, and MAXADJUST= options are specified to indicate the rate adjustment terms of the loan agreement, these specifications are used to bound the rate adjustments. By using the ESTIMATEDCASE= option, you are predicting what the annual nominal rates in the market will be at different points in time, not necessarily the interest rate on your particular loan. For example, if the initial nominal rate (RATE= option) is 6.0, ADJUSTFREQ=6, MAXADJUST=0.5, and the ESTIMATEDCASE=(6=6.5, 12=7.5), the actual nominal rates charged on the loan would be 6.0% initially, 6.5% for the sixth through the eleventh periods, and 7.5% for the twelfth period onward.

FIXEDCASE

FIXCASE

specifies a fixed case analysis that assumes the rate will stay constant. The FIXEDCASE option calculates the ARM loan values similar to a fixed rate loan, but the payments are updated every adjustment period even if the rate does not change, leading to minor differences between the two methods. One such difference is in the way prepayments are handled. In a fixed rate loan, the rate and the payments are never adjusted; therefore, the payment stays the same over the life of the loan even when prepayments are made (instead, the life of the loan is shortened). In an ARM loan with the FIXEDCASE option, on the other hand, if prepayments are made, the payment is adjusted in the following adjustment period, leaving the life of the loan constant.

WORSTCASE

W

specifies a worst-case analysis. The worst-case analysis assumes that the interest rate charged on the loan will reach its maximum allowed limits at each rate adjustment period and over the life of the loan. If the WORSTCASE option is used, either the CAPS= option or the MAXRATE= and MAXADJUST= options must be specified.

BUYDOWN Statement

BUYDOWN *options* **;**

The BUYDOWN statement specifies a buydown rate loan. The buydown rate loans are similar to ARM loans, but the interest rate adjustments are predetermined at the initialization of the loan, usually by paying interest points at the time of loan initialization.

You must use all the required specifications and options listed under the FIXED statement with the BUYDOWN statement. The following option is specific to the BUYDOWN statement and is required:

BUYDOWNRATES=(*date1=rate1 date2=rate2 ...* **)**

BUYDOWNRATES=(*period1=rate1 period2=rate2 ...* **)**

BDR=

specifies pairs of periods and the predetermined nominal interest rates that will be charged on the loan starting at the corresponding time periods.

You can also specify the buydown periods as dates in the form of SAS date literals if you also specify the date of the initial payment by using a date value in the START= option. Buydown periods (or dates) and the respective buydown rates must be in time sequence.

COMPARE Statement

COMPARE *options* **;**

The COMPARE statement compares multiple loans, or it can be used with a single loan. You can use only one COMPARE statement. COMPARE statement options specify the periods and desired types of analysis for loan comparison. The default analysis reports the outstanding principal balance, breakeven of payment, breakeven of interest paid, and before-tax true interest rate. The default comparison period corresponds to the first LIFE= option specification. If the LIFE= option is not specified for any loan, the loan comparison period defaults to the first calculated life.

You can use the following options with the COMPARE statement. For more detailed information on loan comparison, see the section “Loan Comparison Details” on page 1378.

Analysis Options

ALL

is equivalent to specifying the BREAKINTEREST, BREAKPAYMENT, PWOFCOST, and TRUEINTEREST options. The loan comparison report includes all the criteria. You need to specify the MARR= option for present worth of cost calculation.

AT=(*date1 date2 ...* **)**

AT=(*period1 period2 ...* **)**

specifies the periods for loan comparison reports. If you specify the START= option in the PROC LOAN statement, you can specify the AT= option as a list of dates expressed as SAS date literals instead of periods. The comparison periods do not need to be in time sequence. If you do not specify the AT= option, the comparison period defaults to the first LIFE= option specification. If you do not specify the LIFE= option for any of the loans, the loan comparison period defaults to the first calculated life.

BREAKINTEREST

BI

specifies breakeven analysis of the interest paid. The loan comparison report includes the interest paid for each loan through the specified comparison period (AT= option).

BREAKPAYMENT

BP

specifies breakeven analysis of payment. The periodic payment for each loan is reported for every comparison period specified in the AT=option.

MARR=*rate*

specifies the MARR (minimum attractive rate of return) in percent notation. The MARR reflects the cost of capital or the opportunity cost of money. The MARR= option is used in calculating the present worth of cost.

PWOFCOST

PWC

calculates the present worth of cost (net present value of costs) for each loan based on the cash flow through the specified comparison periods. The calculations account for down payment, initialization costs, and discount points, as well as the payments and outstanding principal balance at the comparison period. If you specify the TAXRATE= option, the present worth of cost is based on after-tax cash flow. Otherwise, before-tax present worth of cost is calculated. You need to specify the MARR= option for present worth of cost calculations.

TAXRATE=*rate*

TAX=*rate*

specifies income tax rate in percent notation for the after-tax calculations of the true interest rate and present worth of cost for those assets that qualify for tax deduction. If you specify this option, the amount specified in the POINTS= option and the interest paid on the loan are assumed to be tax-deductible. Otherwise, it is assumed that the asset does not qualify for tax deductions, and the cash flow is not adjusted for tax savings.

TRUEINTEREST

TI

calculates the true interest rate (effective interest rate based on the cash flow of all payments, initialization costs, discount points, and the outstanding principal balance at the comparison period) for all the specified loans through each comparison period. If you specify the TAXRATE= option, the true interest rate is based on after-tax cash flow. Otherwise, the before-tax true interest rate is calculated.

Output Options

NOCOMPRINT

NOCP

suppresses the printing of the loan comparison report. The NOCOMPRINT option is usually used when an OUTCOMP= data set is created to store loan comparison information.

OUTCOMP=*SAS-data-set*

writes the loan comparison report to an output data set.

Details: LOAN Procedure

Computational Details

These terms are used in the formulas that follow:

p	periodic payment
a	principal amount
r_a	nominal annual rate
f	compounding frequency (per year)
f'	payment frequency (per year)
r	periodic rate
r_e	effective interest rate
n	total number of payments

The periodic rate, or the simple interest applied during a payment period, is given by

$$r = \left(1 + \frac{r_a}{f}\right)^{f/f'} - 1$$

Note that the interest calculation is performed at each payment period rather than at the compound period. This is done by adjusting the nominal rate. See Muksian (1984) for details.

Note that when $f = f'$ (that is, when the payment and compounding frequency coincide), the preceding expression reduces to the familiar form:

$$r = \frac{r_a}{f}$$

The periodic rate for continuous compounding can be obtained from this general expression by taking the limit as the compounding frequency f goes to infinity. The resulting expression is

$$r = \exp\left(\frac{r_a}{f'}\right) - 1$$

The effective interest rate, or annualized percentage rate (APR), is that rate which, if compounded once per year, is equivalent to the nominal annual rate compounded f times per year. Thus,

$$(1 + r_e) = (1 + r)^f = \left(1 + \frac{r_a}{f}\right)^f$$

or

$$r_e = \left(1 + \frac{r_a}{f}\right)^f - 1$$

For continuous compounding, the effective interest rate is given by

$$r_e = \exp(r_a) - 1$$

See Muksian (1984) for details.

The payment is calculated as

$$p = \frac{ar}{1 - \frac{1}{(1+r)^n}}$$

The amount is calculated as

$$a = \frac{p}{r}\left(1 - \frac{1}{(1+r)^n}\right)$$

Both the payment and amount are rounded to the nearest hundredth (cent) unless the ROUND= specification is different than the default, 2.

The total number of payments n is calculated as

$$n = \frac{-\ln\left(1 - \frac{ar}{p}\right)}{\ln(1+r)}$$

The total number of payments is rounded up to the nearest integer.

The nominal annual rate is calculated using the bisection method, with a as the objective and r starting in the interval between $8 * 10^{-6}$ and 0.1 with an initial midpoint 0.01 and successive midpoints bisecting.

Loan Comparison Details

In order to compare the costs of different alternatives, the input cash flow for the alternatives must be represented in equivalent values. The equivalent value of a cash flow accounts for the time-value of money. That is, it is preferable to pay the same amount of money later than to pay it now, since the money can earn interest while you keep it. The MARR (minimum attractive rate of return) reflects the cost of capital or the opportunity cost of money—that is, the interest that would have been earned on the savings that is foregone by making the investment. The MARR is used to discount the cash flow of alternatives into equivalent values at a fixed point in time. The MARR can vary for each investor and for each investment. Therefore, the MARR= option must be specified in the COMPARE statement if present worth of cost (PWOFCOST option) comparison is specified.

Present worth of cost reflects the equivalent amount at loan initialization of the loan cash flow discounted at MARR, not accounting for inflation. Present worth of cost accounts for the down payment, initialization costs, discount points, periodic payments, and the principal balance at the end of the report period. Therefore, it reflects the present worth of cost of the asset, not the loan. It is meaningful to use minimization of present worth of cost as a selection criterion only if the assets (down payment plus loan amount) are of the same value.

Another economic selection criterion is the rate of return (internal rate of return) of the alternatives. If interest is being earned by an alternative, the objective is to maximize the rate of return. If interest is being paid, as in loan alternatives, the best alternative is the one that minimizes the rate of return. The true interest rate reflects the effective annual rate charged on the loan based on the cash flow, including the initialization cost and the discount points.

The effects of taxes on different alternatives must be accounted for when these vary among different alternatives. Since interest costs on certain loans are tax-deductible, the comparisons for those loans are made based on the after-tax cash flows. The cost of the loan is reduced by the tax benefits it offers through the loan life if the TAXRATE= option is specified. The present worth of cost and true interest rate are calculated based on the after-tax cash flow of the loan. The down payment on the loan and initialization costs are assumed to be not tax-deductible in after-tax analysis. Discount points and the interest paid in each periodic payment are assumed to be tax-deductible if the TAXRATE= option is specified. If the TAXRATE= option is not specified, the present worth of cost and the true interest rate are based on before-tax cash flow, assuming that the interest paid on the specified loan does not qualify for tax benefits.

The other two selection criteria are breakeven analysis of periodic payment and interest paid. If the objective is to minimize the periodic payment, the best alternative is the one with the minimum periodic payment. If the objective is to minimize the interest paid on the principal, then the best alternative is the one with the least interest paid.

Another criterion might be the minimization of the outstanding balance of the loan at a particular point in time. For example, if you plan to sell a house before the end of the loan life (which is often the case), you might want to select the loan with the minimum principal balance at the time of the sale, since this balance must be paid at that time. The outstanding balance of the alternative loans is calculated for each loan comparison period by default.

If you specified the START= option in the PROC LOAN statement, the present worth of cost reflects the equivalent amount for each loan at that point in time. Any loan that has a START= specification different from the one in the PROC LOAN statement is not processed in the loan comparison.

The loan comparison report for each comparison period contains for each loan the loan label, outstanding balance, and any of the following measures if requested in the COMPARE statement: periodic payment (BREAKPAYMENT option), total interest paid to date (BREAKINTEREST option), present worth of cost (PWOFCOST option), and true interest rate (TRUEINTEREST option). The best loan is selected on the basis of present worth of cost or true interest rate. If both PWOFCOST and TRUEINTEREST options are specified, present worth of cost is the basis for the selection of the best loan.

You can use the OUTCOMP= option in the COMPARE statement to write the loan comparison report to a data set. The NOCOMPRINT option suppresses the printing of a loan comparison report.

OUT= Data Set

The OUT= option writes the loan amortization schedule to an output data set. The OUT= data set contains one observation for each payment period (or one observation for each year if you specified the SCHEDULE=YEARLY option). If you specified the START= option, the DATE variable denotes the date of the payment. Otherwise, YEAR and period variable (SEMIMONTH, MONTH, QUARTER, or SEMIYEAR) denote the payment year and period within the year.

The OUT= data set contains the following variables:

- DATE, date of the payment. DATE is included in the OUT= data set only when you specify the START= option.
- YEAR, year of the payment period. YEAR is included in the OUT= data set only when you do not specify the START= option.
- PERIOD, period within the year of the payment period. The name of the period variable matches the INTERVAL= specification (SEMIMONTH, MONTH, QUARTER, or SEMIYEAR.) The PERIOD variable is included in the OUT= data set only when you do not specify the START= option.
- BEGPRIN, beginning principal balance
- PAYMENT, payment
- INTEREST, interest payment
- PRIN, principal repayment
- ENDPRIN, ending principal balance

OUTCOMP= Data Set

The OUTCOMP= option in the COMPARE statement writes the loan comparison analysis results to an output data set. If you specified the START= option, the DATE variable identifies the date of the loan comparison. Otherwise, the PERIOD variable identifies the comparison period.

The OUTCOMP= data set contains one observation for each loan and for each loan comparison period. The OUTCOMP= data set contains the following variables.

- DATE, date of loan comparison report. The DATE variable is included in the OUTCOMP= data set only when you specify the START= option.
- PERIOD, period of the loan comparison for the observation. The PERIOD variable is included in the OUTCOMP= data set only when you do not specify the START= option.
- LABEL, label string for the loan
- TYPE, type of the loan
- PAYMENT, periodic payment at the time of report. The PAYMENT is included in the OUTCOMP= data set if you specified the BREAKPAYMENT or ALL option or if you used default criteria.
- INTPAY, interest paid through the time of report. The INTPAY variable is included in the OUTCOMP= data set if you specified the BREAKINTEREST or ALL option or if you used default criteria.
- TRUERATE, true interest rate charged on the loan. The TRUERATE variable is included in the OUTCOMP= data set if you specified the TRUERATE or ALL option or if you used default criteria.
- PWOFCOST, present worth of cost. The PWOFCOST variable is included in the OUTCOMP= data set only if you specified the PWOFCOST or ALL option.
- BALANCE, outstanding principal balance at the time of report

OUTSUM= Data Set

The OUTSUM= option writes the loan summary to an output data set. If you specified this option in the PROC LOAN statement, the loan summary information for all loans is written to the specified data set, except for those loans for which you specified a different OUTSUM= data set in the ARM, BALLOON, BUYDOWN, or FIXED statement.

The OUTSUM= data set contains one observation for each loan and contains the following variables:

- TYPE, type of loan
- LABEL, loan label
- PAYMENT, periodic payment
- AMOUNT, loan principal
- DOWNPAY, down payment. DOWNPAY is included in the OUTSUM= data set only when you specify a down payment.
- INITIAL, loan initialization costs. INITIAL is included in the OUTSUM= data set only when you specify initialization costs.
- POINTS, discount points. POINTS is included in the OUTSUM= data set only when you specify discount points.

- TOTAL, total payment
- INTEREST, total interest paid
- RATE, nominal annual interest rate
- EFFRATE, effective interest rate
- INTERVAL, payment interval
- COMPOUND, compounding interval
- LIFE, loan life (that is, the number of payment intervals)
- NCOMPND, number of compounding intervals
- COMPUTE, computed loan parameter: life, amount, payment, or rate

If you specified the START= option either in the PROC LOAN statement or for the individual loan, the OUTSUM= data set also contains the following variables:

- BEGIN, start date
- END, loan termination date

Printed Output

The output from PROC LOAN consists of the loan summary table, loan amortization schedule, and loan comparison report.

Loan Summary Table

The loan summary table shows the total payment and interest, the initial nominal annual and effective interest rates, payment and compounding intervals, the length of the loan in the time units specified, the start and end dates if specified, a list of nominal and effective interest rates, and periodic payments throughout the life of the loan.

A list of balloon payments for balloon payment loans and a list of prepayments if specified are printed with their respective periods or dates.

The loan summary table is printed for each loan by default. The NOSUMMARYPRINT option specified in the PROC LOAN statement suppresses the printing of the loan summary table for all loans. The NOSUMMARYPRINT option can be specified in individual loan statements to selectively suppress the printing of the loan summary table.

Loan Repayment Schedule

The amortization schedule contains for each payment period: the year and period within the year (or date, if you specified the START= option); principal balance at the beginning of the period; total payment, interest payment and principal payment for the period; and the principal balance at the end of the period. If you specified the SCHEDULE=YEARLY option, the amortization contains a summary for each year instead of for each payment period.

The amortization schedule is not printed by default. The SCHEDULE option in the PROC LOAN statement requests the printing of amortization tables for all loans. You can specify the SCHEDULE option in individual loan statements to selectively request the printing of the amortization schedule.

Loan Comparison Report

The loan comparison report is processed for each report period and contains the results of economic analysis of the loans. The quantities reported can include the outstanding principal balance, after-tax or before-tax present worth of cost and true interest rate, periodic payment, and the interest paid through the report period for each loan. The best alternative is identified if the asset value (down payment plus loan amount) is the same for each alternative.

The loan comparison report is printed by default. The NOCOMPRINT option specified in the COMPARE statement suppresses the printing of the loan comparison report.

ODS Table Names

PROC LOAN assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in Table 24.2.

Table 24.2 ODS Tables Produced in PROC LOAN

ODS Table Name	Description	Option
ODS Tables Created by the PROC LOAN, FIXED, ARM, BALLOON, and BUYDOWN Statements		
Repayment	loan repayment schedule	SCHEDULE
ODS Tables Created by the FIXED, ARM, BALLOON, and BUYDOWN Statements		
LoanSummary	loan summary	default
RateList	rates and payments	default
PrepayList	prepayments and periods	PREPAYMENTS=
ODS Tables Created by the BALLOON Statement		
BalloonList	balloon payments and periods	default
ODS Tables Created by the COMPARE Statement		
Comparison	loan comparison report	default

Examples: LOAN Procedure

Example 24.1: Discount Points for Lower Interest Rates

This example illustrates the comparison of two $100,000 loans. The major difference between the two loans is that the nominal interest rate in the second loan is lower than the first with the added expense of paying discount points at the time of initialization.

Both alternatives are 30-year loans. The first loan is labeled "8.25% - no discount points" and the second one is labeled "8% - 1 discount point."

Assume that the interest paid qualifies for a tax deduction and you are in the 33% tax bracket. Also, your minimum attractive rate of return (MARR) for an alternative investment is 4% (adjusted for tax rate).

You use the following statements to find the breakeven point in the life of the loan for your preference between the loans:

```
proc loan start=1992:1 nosummaryprint amount=100000 life=360;
   fixed rate=8.25 label='8.25% - no discount points';
   fixed rate=8 points=1000 label='8% - 1 discount point';
   compare at=(48 54 60) all taxrate=33 marr=4;
run;
```

Output 24.1.1 shows the loan comparison reports as of January 1996 (48th period), July 1996 (54th period), and January 1997 (60th period).

Output 24.1.1 Loan Comparison Reports for Discount Point Breakeven

The LOAN Procedure

Loan Comparison Report

Analysis through JAN1996

Loan Label	Ending Outstanding	Present Worth of Cost	Payment	Interest Paid	True Rate
8.25% - no discount points	96388.09	105546.17	751.27	32449.05	5.67
8% - 1 discount point	96219.32	105604.05	733.76	31439.80	5.69

Note: "8.25% - no discount points" is the best alternative based on present worth of cost analysis through JAN1996.

Loan Comparison Report

Analysis through JUL1996

Loan Label	Ending Outstanding	Present Worth of Cost	Payment	Interest Paid	True Rate
8.25% - no discount points	95847.27	106164.97	751.27	36415.85	5.67
8% - 1 discount point	95656.22	106153.97	733.76	35279.26	5.67

Note: "8% - 1 discount point" is the best alternative based on present worth of cost analysis through JUL1996.

Output 24.1.1 *continued*

Loan Comparison Report

Analysis through JAN1997

Loan Label	Ending Outstanding	Present Worth of Cost	Payment	Interest Paid	True Rate
8.25% - no discount points	95283.74	106768.07	751.27	40359.94	5.67
8% - 1 discount point	95070.21	106689.80	733.76	39095.81	5.66

Note: "8% - 1 discount point" is the best alternative based on present worth of cost analysis through JAN1997.

Notice that the breakeven point for present worth of cost and true rate both happen on July 1996. This indicates that if you intend to keep the loan for 4.5 years or more, it is better to pay the discount points for the lower rate. If your objective is to minimize the interest paid or the periodic payment, the "8% - 1 discount point" loan is the preferred choice.

Example 24.2: Refinancing a Loan

Assume that you obtained a fixed rate 15-year loan in June 1995 for $78,500 with a nominal annual rate of 9%. By early 1998, the market offers a 6.5% interest rate, and you are considering whether to refinance your loan.

Use the following statements to find out the status of the loan on February 1998. Output 24.2.1 shows the results:

```
proc loan start=1995:6;
   fixed life=180 rate=9 amount=78500 noprint
         label='Original Loan';
   compare at=('10FEB1998'd);
run;
```

Output 24.2.1 Loan Comparison Report for Original Loan

The LOAN Procedure

Loan Comparison Report

Analysis through FEB1998

Loan Label	Ending Outstanding	Payment	Interest Paid	True Rate
Original Loan	71028.75	796.20	18007.15	9.38

The monthly payment on the original loan is $796.20. The ending outstanding principal balance as of February is $71,028.75. At this point, you might want to refinance your loan with another 15-year loan. The alternate loan has a 6.5% nominal annual rate. The initialization costs are $1,419.00. Use the following statements to compare your alternatives:

```
proc loan start=1998:2 amount=71028.75;
   fixed rate=9 payment=796.20
         label='Keep the original loan' noprint;
   fixed life=180 rate=6.5 init=1419
         label='Refinance at 6.5%' noprint;
   compare at=(15 16) taxrate=33 marr=4 all;
run;
```

The comparison reports of May 1999 and June 1999 in Output 24.2.2 illustrate the break even between the two alternatives. If you intend to keep the loan through June 1999 or longer, your initialization costs for the refinancing are justified. The periodic payment of the refinanced loan is $618.74.

Output 24.2.2 Loan Comparison Report for Refinancing Decision

The LOAN Procedure

Loan Comparison Report

Analysis through MAY1999

Loan Label	Ending Outstanding	Present Worth of Cost	Payment	Interest Paid	True Rate
Keep the original loan	66862.10	72737.27	796.20	7776.35	6.20
Refinance at 6.5%	67382.48	72747.51	618.74	5634.83	6.23

Note: "Keep the original loan" is the best alternative based on present worth of cost analysis through MAY1999.

Loan Comparison Report

Analysis through JUN1999

Loan Label	Ending Outstanding	Present Worth of Cost	Payment	Interest Paid	True Rate
Keep the original loan	66567.37	72844.52	796.20	8277.82	6.20
Refinance at 6.5%	67128.73	72766.42	618.74	5999.82	6.12

Note: "Refinance at 6.5%" is the best alternative based on present worth of cost analysis through JUN1999.

Example 24.3: Prepayments on a Loan

This example compares a 30-year loan with and without prepayments. Assume the $240,000 30-year loan has an 8.25% nominal annual rate. Use the following statements to see the effect of making uniform prepayments of $500 with periodic payment:

```
proc loan start=1992:12 rate=8.25 amount=240000 life=360;
   fixed label='No prepayments';
   fixed label='With Prepayments' prepay=500;
   compare at=(120) taxrate=33 marr=4 all;
run;
```

Output 24.3.1 Loan Summary Reports without Prepayments

The LOAN Procedure

Fixed Rate Loan Summary

No prepayments

Downpayment	0.00	**Principal Amount**	240000.00
Initialization	0.00	**Points**	0.00
Total Interest	409094.17	**Nominal Rate**	8.2500%
Total Payment	649094.17	**Effective Rate**	8.5692%
Pay Interval	MONTHLY	**Compounding**	MONTHLY
No. of Payments	360	**No. of Compoundings**	360
Start Date	DEC1992	**End Date**	DEC2022

Rates and Payments for No prepayments

Date	Nominal Rate	Effective Rate	Payment
DEC1992	8.2500%	8.5692%	1803.04

Output 24.3.2 Loan Summary Reports with Prepayments

The LOAN Procedure

Fixed Rate Loan Summary

With Prepayments

Downpayment	0.00	**Principal Amount**	240000.00
Initialization	0.00	**Points**	0.00
Total Interest	183650.70	**Nominal Rate**	8.2500%
Total Payment	423650.70	**Effective Rate**	8.5692%
Pay Interval	MONTHLY	**Compounding**	MONTHLY
No. of Payments	184	**No. of Compoundings**	184
Start Date	DEC1992	**End Date**	APR2008

Rates and Payments for With Prepayments

Date	Nominal Rate	Effective Rate	Payment
DEC1992	8.2500%	8.5692%	2303.04

Output 24.3.3 Loan Comparison Report

The LOAN Procedure

Loan Comparison Report

Analysis through DEC2002

Loan Label	Ending Outstanding	Present Worth of Cost	Payment	Interest Paid	True Rate
No prepayments	211608.05	268762.31	1803.04	187972.85	5.67
With Prepayments	118848.23	264149.25	2303.04	155213.03	5.67

Note: "With Prepayments" is the best alternative based on present worth of cost analysis through DEC2002.

Output 24.3.1 through Output 24.3.3 illustrate the Loan Summary Reports and the Loan Comparison report. Notice that with prepayments you pay off the loan in slightly more than 15 years. Also, the total payments and total interest are considerably lower with the prepayments. If you can afford the prepayments of $500 each month, another alternative you should consider is using a 15-year loan, which is generally offered at a lower nominal interest rate.

Example 24.4: Output Data Sets

This example shows the analysis and comparison of five alternative loans. Initialization cost, discount points, and both lump sum and periodic payments are included in the specification of these loans. Although no printed output is produced, the loan summary and loan comparison information is stored in the OUTSUM= and OUTCOMP= data sets.

```
proc loan start=1998:12 noprint outsum=loans
   amount=150000 life=360;

   fixed rate=7.5 life=180 prepayment=500
         label='BANK1, Fixed Rate';

   arm rate=5.5 estimatedcase=(12=7.5 18=8)
       label='BANK1, Adjustable Rate';

   buydown rate=7 interval=semimonth init=15000
           bdrates=(3=9 10=10) label='BANK2, Buydown';

   arm rate=5.75 worstcase caps=(0.5 2.5)
       adjustfreq=6 label='BANK3, Adjustable Rate'
       prepayments=(12=2000 36=5000);

   balloon rate=7.5 life=480
       points=1100 balloonpayment=(15=2000 48=2000)
       label='BANK4, with Balloon Payment';

   compare at=(120 360) all marr=7 tax=33 outcomp=comp;
run;

proc print data=loans;
run;

proc print data=comp;
run;
```

Output 24.4.1 and Output 24.4.2 illustrate the contents of the output data sets.

Output 24.4.1 OUTSUM= Data Set

Obs	TYPE	LABEL	PAYMENT	AMOUNT	INITIAL	POINTS	TOTAL	INTEREST	RATE
1	FIXED	BANK1, Fixed Rate	1890.52	150000	0	0	207839.44	57839.44	0.0750
2	ARM	BANK1, Adjustable Rate	851.68	150000	0	0	390325.49	240325.49	0.0550
3	BUYDOWN	BANK2, Buydown	673.57	150000	15000	0	288858.08	138858.08	0.0700
4	ARM	BANK3, Adjustable Rate	875.36	150000	0	0	387647.82	237647.82	0.0575
5	BALLOON	BANK4, with Balloon Payment	965.36	150000	0	1100	467372.31	317372.31	0.0750

Obs	EFFRATE	INTERVAL	COMPOUND	LIFE	NCOMPND	COMPUTE	START	END
1	0.077633	MONTHLY	MONTHLY	110	110	PAYMENT	DEC1998	FEB2008
2	0.056408	MONTHLY	MONTHLY	360	360	PAYMENT	DEC1998	DEC2028
3	0.072399	SEMIMONTHLY	SEMIMONTHLY	360	360	PAYMENT	DEC1998	DEC2013
4	0.059040	MONTHLY	MONTHLY	360	360	PAYMENT	DEC1998	DEC2028
5	0.077633	MONTHLY	MONTHLY	480	480	PAYMENT	DEC1998	DEC2038

Output 24.4.2 OUTCOMP= Data Set

Obs	DATE	TYPE	LABEL	PAYMENT	INTEREST	TRUERATE	PWOFCOST	BALANCE
1	DEC2008	FIXED	BANK1, Fixed Rate	1772.76	57839.44	0.051424	137741.07	0.00
2	DEC2008	ARM	BANK1, Adjustable Rate	1093.97	108561.77	0.052212	130397.88	130788.65
3	DEC2008	BUYDOWN	BANK2, Buydown	803.98	118182.19	0.087784	161810.00	75798.19
4	DEC2008	ARM	BANK3, Adjustable Rate	1065.18	107015.58	0.053231	131955.90	125011.88
5	DEC2008	BALLOON	BANK4, with Balloon Payment	965.36	107906.61	0.052107	130242.56	138063.41
6	DEC2028	FIXED	BANK1, Fixed Rate	1772.76	57839.44	0.051424	137741.07	0.00
7	DEC2028	ARM	BANK1, Adjustable Rate	1094.01	240325.49	0.053247	121980.94	0.00
8	DEC2028	BUYDOWN	BANK2, Buydown	800.46	138858.08	0.086079	161536.44	0.00
9	DEC2028	ARM	BANK3, Adjustable Rate	1065.20	237647.82	0.054528	124700.22	0.00
10	DEC2028	BALLOON	BANK4, with Balloon Payment	965.36	282855.86	0.051800	117294.50	81326.26

Example 24.5: Piggyback Loans

The *piggyback* loan is becoming a widely available alternative. Borrowers like to avoid the PMI (private mortgage insurance) required with loans where the borrower has a down payment of less than 20% of the price. The piggyback allows a secondary home equity loan to be packaged with a primary loan with less than 20% down payment. The secondary loan usually has a shorter life and higher interest rate. The interest paid on both loans are tax-deductible whereas PMI does not qualify for a tax deduction.

The following example compares a conventional fixed rate loan with 20% down as opposed to a piggyback loan: one primary fixed rate with 10% down payment and a secondary, home equity loan for 10% of the original price. All loans have monthly payments.

The conventional loan alternative is a 30-year loan with a fixed annual rate of 7.5%. The primary loan in the piggyback loan setup is also a 30-year loan with a fixed annual rate of 7.75%. The secondary loan is a 15-year loan with a fixed annual interest rate of 8.25%.

The comparison output for the two loans comprising the piggyback loan is aggregated using the TIMESERIES procedure with a minimum of specified options:

- The INTERVAL= option requests that the data be aggregated into periods of length 5 years beginning on the 25th month, resulting in appropriately identified periods.
- The ACC=TOTAL option specifies that the output should reflect accumulated totals as opposed to, say, averages.
- The NOTSORTED option indicates that the input data set has not been sorted by the ID variable.

See Chapter 39, "The TIMESERIES Procedure," for more information about this procedure.

Use the following statements to analyze the conventional loan, as well as the piggyback alternative, and compare them on the basis of their present worth of cost, outstanding balance, and interest payment amounts at the end of 5, 10, and 15 years into the loan life.

```
title1 'LOAN: Piggyback loan example';

title2 'LOAN: Conventional loan';

proc loan start=2002:1 noprint;

   fixed price=200000 dp=40000 rate=7.5 life=360
         label='20 percent down: Conventional Fixed Rate' ;

   compare at=(60 120 180) pwofcost taxrate=30 marr=12
           breakpay breakint outcomp=comploans;

run;

title2 'LOAN: Piggyback: Primary Loan';

proc loan start=2002:1 noprint;

   fixed amount=160000 dp=20000 rate=7.75 life=360
         label='Piggyback: Primary loan' out=loan1;

   compare at=(60 120 180 ) pwofcost taxrate=30 marr=12
          breakpay breakint outcomp=cloan1;

run;

title2 'LOAN: Piggyback: Secondary (Home Equity) Loan';

proc loan start=2002:1 noprint;

   fixed amount=20000  rate=8.25 life=180
         label='Piggyback: Secondary (Home Equity) Loan' out=loan2;

   compare at=(60 120 180 ) pwofcost taxrate=30 marr=12
           breakpay breakint  outcomp=cloan2;

run;

data cloan12;
   set cloan1  cloan2;
run;
```

```
proc timeseries data=cloan12 out= totcomp ;
   id date interval=year5.25 acc=total notsorted;
   var payment interest pwofcost balance ;
run;

/*-- LOAN: Piggyback loan --*/
title;
proc print data=totcomp;
   format date monyy7.;
run;

data comploans;
   set comploans;
   drop type label;
run;

/*-- LOAN: Conventional Loan --*/
title;
proc print data=comploans;
run;
```

The loan comparisons in Output 24.5.1 and Output 24.5.2 illustrate the after-tax comparison of the loans. The after-tax present value of cost for the piggyback loan is lower than the 20% down conventional fixed rate loan.

Output 24.5.1 Piggyback Loan

Obs	DATE	PAYMENT	INTEREST	PWOFCOST	BALANCE
1	JAN2007	1340.29	67992.92	157157.41	167575.52
2	JAN2012	1340.29	129973.53	135556.98	149138.73
3	JAN2017	1339.66	183028.58	125285.77	121777.01

Output 24.5.2 Conventional Loan

Obs	DATE	PAYMENT	INTEREST	PWOFCOST	BALANCE
1	JAN2007	1118.74	58512.54	160436.81	151388.14
2	JAN2012	1118.74	113121.41	140081.64	138872.61
3	JAN2017	1118.74	162056.97	130014.97	120683.77

References

DeGarmo, E. P., Sullivan, W. G., and Canada, J. R. (1984). *Engineering Economy*. 7th ed. New York: Macmillan.

Muksian, R. (1984). *Financial Mathematics Handbook*. Englewood Cliffs, NJ: Prentice-Hall.

Newnan, D. G. (1988). *Engineering Economic Analysis*. 3rd ed. San Jose, CA: Engineering Press.

Riggs, J. L., and West, T. M. (1986). *Essentials of Engineering Economics*. 2nd ed. New York: McGraw-Hill.

Chapter 25
The MDC Procedure

Contents

Overview: MDC Procedure **1394**
Getting Started: MDC Procedure **1395**
Conditional Logit: Estimation and Prediction 1395
Nested Logit Modeling 1400
Multivariate Normal Utility Function 1403
HEV and Multinomial Probit: Heteroscedastic Utility Function 1404
Parameter Heterogeneity: Mixed Logit 1409
Syntax: MDC Procedure **1411**
Functional Summary 1411
PROC MDC Statement 1413
MDCDATA Statement 1413
BOUNDS Statement 1414
BY Statement 1415
CLASS Statement 1415
ID Statement 1415
MODEL Statement 1415
NEST Statement 1420
NLOPTIONS Statement 1423
OUTPUT Statement 1424
RESTRICT Statement 1424
TEST Statement 1425
UTILITY Statement 1426
Details: MDC Procedure **1427**
Multinomial Discrete Choice Modeling 1427
Multinomial Logit and Conditional Logit 1428
Heteroscedastic Extreme-Value Model 1429
Mixed Logit Model 1430
Multinomial Probit 1432
Nested Logit 1434
Decision Tree and Nested Logit 1436
Model Fit and Goodness-of-Fit Statistics 1438
Tests on Parameters 1439
OUTEST= Data Set 1440
ODS Table Names 1442
Examples: MDC Procedure **1442**
Example 25.1: Binary Data Modeling 1442

Example 25.2: Conditional Logit and Data Conversion 1446
Example 25.3: Correlated Choice Modeling 1448
Example 25.4: Testing for Homoscedasticity of the Utility Function 1451
Example 25.5: Choice of Time for Work Trips: Nested Logit Analysis 1455
Example 25.6: Hausman's Specification Test 1462
Example 25.7: Likelihood Ratio Test 1465
Acknowledgments: MDC Procedure **1465**
References **1466**

Overview: MDC Procedure

The MDC (multinomial discrete choice) procedure analyzes models in which the choice set consists of multiple alternatives. This procedure supports conditional logit, mixed logit, heteroscedastic extreme value, nested logit, and multinomial probit models. The MDC procedure uses the maximum likelihood (ML) or simulated maximum likelihood method for model estimation. The term *multinomial logit* is often used in the econometrics literature to refer to the *conditional logit* model of McFadden (1974). Here, the term *conditional logit* refers to McFadden's conditional logit model, and the term *multinomial logit* refers to a model that differs slightly. Early applications of the multinomial logit model in the econometrics literature are provided by Schmidt and Strauss (1975); Theil (1969). The main difference between McFadden's conditional logit model and the multinomial logit model is that the multinomial logit model makes the choice probabilities depend on the characteristics of the individuals only, whereas the conditional logit model considers the effects of choice attributes on choice probabilities as well.

Unordered multiple choices are observed in many settings in different areas of application. For example, choices of housing location, occupation, political party affiliation, type of automobile, and mode of transportation are all unordered multiple choices. Economics and psychology models often explain observed choices by using the *random utility* function. The utility of a specific choice can be interpreted as the relative pleasure or happiness that the decision maker derives from that choice with respect to other alternatives in a finite choice set. It is assumed that the individual chooses the alternative for which the associated utility is highest. However, the utilities are not known to the analyst with certainty and are therefore treated by the analyst as random variables. When the utility function contains a random component, the individual choice behavior becomes a probabilistic process.

The random utility function of individual i for choice j can be decomposed into deterministic and stochastic components

$$U_{ij} = V_{ij} + \epsilon_{ij}$$

where V_{ij} is a deterministic utility function, assumed to be linear in the explanatory variables, and ϵ_{ij} is an unobserved random variable that captures the factors that affect utility that are not included in V_{ij}. Different assumptions on the distribution of the errors, ϵ_{ij}, give rise to different classes of models.

The features of discrete choice models available in the MDC procedure are summarized in Table 25.1.

Table 25.1 Summary of Models Supported by PROC MDC

Model Type	Utility Function	Distribution of ϵ_{ij}
Conditional logit	$U_{ij} = \mathbf{x}'_{ij}\boldsymbol{\beta} + \epsilon_{ij}$	IEV, independent and identical
HEV	$U_{ij} = \mathbf{x}'_{ij}\boldsymbol{\beta} + \epsilon_{ij}$	HEV, independent and nonidentical
Nested logit	$U_{ij} = \mathbf{x}'_{ij}\boldsymbol{\beta} + \epsilon_{ij}$	GEV, correlated and identical
Mixed logit	$U_{ij} = \mathbf{x}'_{ij}\boldsymbol{\beta} + \xi_{ij} + \epsilon_{ij}$	IEV, independent and identical
Multinomial probit	$U_{ij} = \mathbf{x}'_{ij}\boldsymbol{\beta} + \epsilon_{ij}$	MVN, correlated and nonidentical

IEV stands for type I extreme-value (or Gumbel) distribution with the probability density function and the cumulative distribution function of the random error given by $f(\epsilon_{ij}) = \exp(-\epsilon_{ij})\exp(-\exp(-\epsilon_{ij}))$ and $F(\epsilon_{ij}) = \exp(-\exp(-\epsilon_{ij}))$. HEV stands for heteroscedastic extreme-value distribution with the probability density function and the cumulative distribution function of the random error given by $f(\epsilon_{ij}) = \frac{1}{\theta_j}\exp(\frac{\epsilon_{ij}}{\theta_j})\exp[-\exp(-\frac{\epsilon_{ij}}{\theta_j})]$ and $F(\epsilon_{ij}) = \exp[-\exp(-\frac{\epsilon_{ij}}{\theta_j})]$, where θ_j is a scale parameter for the random component of the *j*th alternative. GEV stands for generalized extreme-value distribution. MVN represents multivariate normal distribution; and ξ_{ij} is an error component. See the "Mixed Logit Model" on page 1430 section for more information about ξ_{ij}.

Getting Started: MDC Procedure

Conditional Logit: Estimation and Prediction

The MDC procedure is similar in use to the other regression model procedures in the SAS System. However, the MDC procedure requires identification and choice variables. For example, consider a random utility function

$$U_{ij} = x_{1,ij}\beta_1 + x_{2,ij}\beta_2 + \epsilon_{ij} \quad j = 1, \ldots, 3$$

where the cumulative distribution function of the stochastic component is a Type I extreme value, $F(\epsilon_{ij}) = \exp(-\exp(-\epsilon_{ij}))$. You can estimate this conditional logit model with the following statements:

```
proc mdc;
   model decision = x1 x2 / type=clogit
      choice=(mode 1 2 3);
   id pid;
run;
```

Note that the MDC procedure, unlike other regression procedures, does not include the intercept term automatically. The dependent variable decision takes the value 1 when a specific alternative is chosen; otherwise, it takes the value 0. Each individual is allowed to choose one and only one of the possible alternatives. In other words, the variable decision takes the value 1 one time only for each individual. If each individual has three elements (1, 2, and 3) in the choice set, the NCHOICE=3 option can be specified instead of CHOICE=(mode 1 2 3).

Consider the following trinomial data from Daganzo (1979). The original data (origdata) contain travel time (ttime1–ttime3) and choice (choice) variables. The variables ttime1–ttime3 are the travel times for three different modes of transportation, and choice indicates which one of the three modes is chosen. The choice variable must have integer values.

```
data origdata;
   input ttime1 ttime2 ttime3 choice @@;
datalines;
16.481  16.196  23.89   2  15.123  11.373  14.182  2
19.469  8.822   20.819  2  18.847  15.649  21.28   2
12.578  10.671  18.335  2  11.513  20.582  27.838  1
10.651  15.537  17.418  1  8.359   15.675  21.05   1

   ... more lines ...
```

A new data set (newdata) is created because PROC MDC requires that each individual decision maker has one case for each alternative in his choice set. Note that the ID statement is required for all MDC models. In the following example, there are two public transportation modes, 1 and 2, and one private transportation mode, 3, and all individuals share the same choice set.

The first nine observations of the raw data set are shown in Figure 25.1.

Figure 25.1 Initial Choice Data

Obs	ttime1	ttime2	ttime3	choice
1	16.481	16.196	23.890	2
2	15.123	11.373	14.182	2
3	19.469	8.822	20.819	2
4	18.847	15.649	21.280	2
5	12.578	10.671	18.335	2
6	11.513	20.582	27.838	1
7	10.651	15.537	17.418	1
8	8.359	15.675	21.050	1
9	11.679	12.668	23.104	1

The following statements transform the data according to MDC procedure requirements:

```
data newdata(keep=pid decision mode ttime);
   set origdata;
   array tvec{3} ttime1 - ttime3;
   retain pid 0;
   pid + 1;
   do i = 1 to 3;
      mode = i;
      ttime = tvec{i};
      decision = ( choice = i );
      output;
   end;
run;
```

The first nine observations of the transformed data set are shown in Figure 25.2.

Figure 25.2 Transformed Modal Choice Data

Obs	pid	mode	ttime	decision
1	1	1	16.481	0
2	1	2	16.196	1
3	1	3	23.890	0
4	2	1	15.123	0
5	2	2	11.373	1
6	2	3	14.182	0
7	3	1	19.469	0
8	3	2	8.822	1
9	3	3	20.819	0

The decision variable, decision, must have one nonzero value for each decision maker that corresponds to the actual choice. When the RANK option is specified, the decision variable must contain rank data. For more details, see the section "MODEL Statement" on page 1415. The following SAS statements estimate the conditional logit model by using maximum likelihood:

```
proc mdc data=newdata;
   model decision = ttime /
            type=clogit
            nchoice=3
            optmethod=qn
            covest=hess;
   id pid;
run;
```

The MDC procedure enables different individuals to have different choice sets. When all individuals have the same choice set, the NCHOICE= option can be used instead of the CHOICE= option. However, the NCHOICE= option is not allowed when a nested logit model is estimated. When the NCHOICE=*number* option is specified, the choices are generated as $1, \ldots, number$. For more flexible alternatives (for example, 1, 3, 6, 8), you need to use the CHOICE= option. The choice variable must have integer values.

The OPTMETHOD=QN option specifies the quasi-Newton optimization technique. The covariance matrix of the parameter estimates is obtained from the Hessian matrix because COVEST=HESS is specified. You can also specify COVEST=OP or COVEST=QML. See the section "MODEL Statement" on page 1415 for more details.

The MDC procedure produces a summary of model estimation displayed in Figure 25.3. Since there are multiple observations for each individual, the "Number of Cases" (150)—that is, the total number of choices faced by all individuals—is larger than the number of individuals, "Number of Observations" (50).

Figure 25.3 Estimation Summary Table

The MDC Procedure

Conditional Logit Estimates

Model Fit Summary	
Dependent Variable	decision
Number of Observations	50
Number of Cases	150
Log Likelihood	-33.32132
Log Likelihood Null (LogL(0))	-54.93061
Maximum Absolute Gradient	2.97024E-6
Number of Iterations	6
Optimization Method	Dual Quasi-Newton
AIC	68.64265
Schwarz Criterion	70.55467

Figure 25.4 shows the frequency distribution of the three choice alternatives. In this example, mode 2 is most frequently chosen.

Figure 25.4 Choice Frequency

Discrete Response Profile			
Index	**CHOICE**	**Frequency**	**Percent**
0	1	14	28.00
1	2	29	58.00
2	3	7	14.00

The MDC procedure computes nine goodness-of-fit measures for the discrete choice model. Seven of them are pseudo-R-square measures based on the null hypothesis that all coefficients except for an intercept term are zero (Figure 25.5). McFadden's likelihood ratio index (LRI) is the smallest in value. For more details, see the section "Model Fit and Goodness-of-Fit Statistics" on page 1438.

Figure 25.5 Likelihood Ratio Test and R-Square Measures

Goodness-of-Fit Measures		
Measure	**Value**	**Formula**
Likelihood Ratio (R)	43.219	2 * (LogL - LogL0)
Upper Bound of R (U)	109.86	- 2 * LogL0
Aldrich-Nelson	0.4636	R / (R+N)
Cragg-Uhler 1	0.5787	1 - exp(-R/N)
Cragg-Uhler 2	0.651	(1-exp(-R/N)) / (1-exp(-U/N))
Estrella	0.6666	1 - (1-R/U)^(U/N)
Adjusted Estrella	0.6442	1 - ((LogL-K)/LogL0)^(-2/N*LogL0)
McFadden's LRI	0.3934	R / U
Veall-Zimmermann	0.6746	(R * (U+N)) / (U * (R+N))
N = # of observations, K = # of regressors		

Finally, the parameter estimate is displayed in Figure 25.6.

Figure 25.6 Parameter Estimate of Conditional Logit

The MDC Procedure

Conditional Logit Estimates

Parameter Estimates					
Parameter	**DF**	**Estimate**	**Standard Error**	**t Value**	**Approx Pr > |t|**
ttime	1	-0.3572	0.0776	-4.60	<.0001

The predicted choice probabilities are produced using the OUTPUT statement:

```
output out=probdata pred=p;
```

The parameter estimates can be used to forecast the choice probability of individuals that are not in the input data set. To do so, you need to append to the input data set extra observations whose values of the dependent variable decision are missing, since these extra observations are not supposed to be used in the estimation stage. The identification variable pid must have values that are not used in the existing observations. The output data set, probdata, contains a new variable, p, in addition to input variables in the data set, extdata.

The following statements forecast the choice probability of individuals that are not in the input data set:

```
data extra;
   input pid mode decision ttime;
datalines;
51  1  .   5.0
51  2  .  15.0
51  3  .  14.0
;

data extdata;
   set newdata extra;
run;

proc mdc data=extdata;
   model decision = ttime /
            type=clogit
            covest=hess
            nchoice=3;
   id pid;
   output out=probdata pred=p;
run;

proc print data=probdata( where=( pid >= 49 ) );
   var mode decision p ttime;
   id pid;
run;
```

The last nine observations from the forecast data set (probdata) are displayed in Figure 25.7. It is expected that the decision maker will choose mode "1" based on predicted probabilities for all modes.

Figure 25.7 Out-of-Sample Mode Choice Forecast

pid	mode	decision	p	ttime
49	1	0	0.46393	11.852
49	2	1	0.41753	12.147
49	3	0	0.11853	15.672
50	1	0	0.06936	15.557
50	2	1	0.92437	8.307
50	3	0	0.00627	22.286
51	1	.	0.93611	5.000
51	2	.	0.02630	15.000
51	3	.	0.03759	14.000

Nested Logit Modeling

A more general model can be specified using the nested logit model.

Consider, for example, the following random utility function:

$$U_{ij} = x_{ij}\beta + \epsilon_{ij} \quad j = 1, \ldots, 3$$

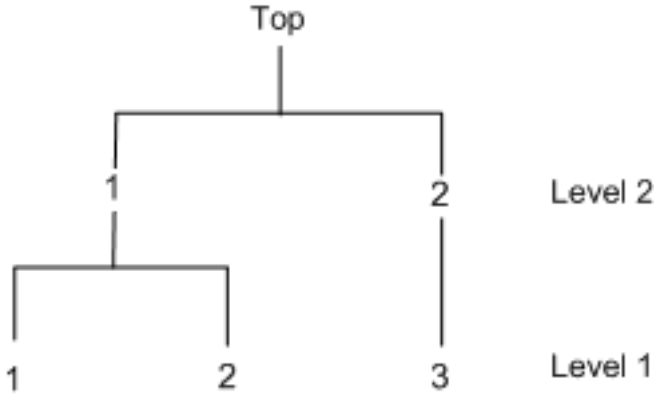
Suppose the set of all alternatives indexed by j is partitioned into K nests, $B_1, \ldots, B_K$. The nested logit model is obtained by assuming that the error term in the utility function has the GEV cumulative distribution function

$$\exp\left(-\sum_{k=1}^{K}\left(\sum_{j\in B_k}\exp\{-\epsilon_{ij}/\lambda_k\}\right)^{\lambda_k}\right)$$

where λ_k is a measure of a degree of independence among the alternatives in nest k. When $\lambda_k = 1$ for all k, the model reduces to the standard logit model.

Since the public transportation modes, 1 and 2, tend to be correlated, these two choices can be grouped together. The decision tree displayed in Figure 25.8 is constructed.

Figure 25.8 Decision Tree for Model Choice



The two-level decision tree is specified in the NEST statement. The NCHOICE= option is not allowed for nested logit estimation. Instead, the CHOICE= option needs to be specified, as in the following statements:

```
/*-- nested logit estimation --*/
proc mdc data=newdata;
   model decision = ttime /
            type=nlogit
            choice=(mode 1 2 3)
            covest=hess;
   id pid;
   utility u(1,) = ttime;
   nest level(1) = (1 2 @ 1, 3 @ 2),
        level(2) = (1 2 @ 1);
run;
```

In Figure 25.9, estimates of the inclusive value parameters, INC_L2G1C1 and INC_L2G1C2, are indicative of a nested model structure. See the section "Nested Logit" on page 1434 and the section "Decision Tree and Nested Logit" on page 1436 for more details about inclusive values.

Figure 25.9 Two-Level Nested Logit Estimates

The MDC Procedure

Nested Logit Estimates

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
ttime_L1	1	-0.4040	0.1241	-3.25	0.0011
INC_L2G1C1	1	0.8016	0.4352	1.84	0.0655
INC_L2G1C2	1	0.8087	0.3591	2.25	0.0243

The nested logit model is estimated with the restriction INC_L2G1C1 = INC_L2G1C2 by specifying the SAMESCALE option, as in the following statements:

```
/*-- nlogit with samescale option --*/
proc mdc data=newdata;
   model decision = ttime /
            type=nlogit
            choice=(mode 1 2 3)
            samescale
            covest=hess;
   id pid;
   utility u(1,) = ttime;
   nest level(1) = (1 2 @ 1, 3 @ 2),
        level(2) = (1 2 @ 1);
run;
```

The estimation result is displayed in Figure 25.10.

Figure 25.10 Nested Logit Estimates with One Dissimilarity Parameter

The MDC Procedure

Nested Logit Estimates

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
ttime_L1	1	-0.4025	0.1217	-3.31	0.0009
INC_L2G1	1	0.8209	0.3019	2.72	0.0066

The nested logit model is equivalent to the conditional logit model if INC_L2G1C1 = INC_L2G1C2 = 1. You can verify this relationship by estimating a constrained nested logit model as shown in the following statements. (See the section "RESTRICT Statement" on page 1424 for details about imposing linear restrictions on parameter estimates.)

```
/*-- constrained nested logit estimation --*/
proc mdc data=newdata;
   model decision = ttime /
            type=nlogit
            choice=(mode 1 2 3)
            covest=hess;
```

```
   id pid;
   utility u(1,) = ttime;
   nest level(1) = (1 2 @ 1, 3 @ 2),
        level(2) = (1 2 @ 1);
   restrict INC_L2G1C1 = 1, INC_L2G1C2 =1;
run;
```

The parameter estimates and the active linear constraints for the constrained nested logit model are displayed in Figure 25.11.

Figure 25.11 Constrained Nested Logit Estimates

The MDC Procedure

Nested Logit Estimates

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|	Parameter Label
ttime_L1	1	-0.3572	0.0776	-4.60	<.0001	
INC_L2G1C1	0	1.0000	0			
INC_L2G1C2	0	1.0000	0			
Restrict1	1	-2.1706	8.4098	-0.26	0.7993*	Linear EC [1]
Restrict2	1	3.6573	10.0001	0.37	0.7186*	Linear EC [2]

*** Probability computed using beta distribution.**

Linearly Independent Active Linear Constraints

1	0	=	-1.0000	+	1.0000	* INC_L2G1C1
2	0	=	-1.0000	+	1.0000	* INC_L2G1C2

Multivariate Normal Utility Function

Consider the random utility function

$$U_{ij} = \text{ttime}_{ij}\beta + \epsilon_{ij}, \quad j = 1, 2, 3$$

where

$$\begin{bmatrix} \epsilon_{i1} \\ \epsilon_{i2} \\ \epsilon_{i3} \end{bmatrix} \sim N\left(\mathbf{0}, \begin{bmatrix} 1 & \rho_{21} & 0 \\ \rho_{21} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right)$$

The correlation coefficient (ρ_{21}) between U_{i1} and U_{i2} represents commonly neglected attributes of public transportation modes, 1 and 2. The following SAS statements estimate this trinomial probit model:

```
/*-- homoscedastic mprobit --*/
proc mdc data=newdata;
   model decision = ttime /
            type=mprobit
            nchoice=3
            unitvariance=(1 2 3)
```

```
            covest=hess;
   id pid;
run;
```

The UNITVARIANCE=(1 2 3) option specifies that the random component of utility for each of these choices has unit variance. If the UNITVARIANCE= option is specified, it needs to include at least two choices. The results of this constrained multinomial probit model estimation are displayed in Figure 25.12 and Figure 25.13. The test for ttime = 0 is rejected at the 1% significance level.

Figure 25.12 Constrained Probit Estimation Summary

The MDC Procedure

Multinomial Probit Estimates

Model Fit Summary	
Dependent Variable	decision
Number of Observations	50
Number of Cases	150
Log Likelihood	-33.88604
Log Likelihood Null (LogL(0))	-54.93061
Maximum Absolute Gradient	0.0002380
Number of Iterations	8
Optimization Method	Dual Quasi-Newton
AIC	71.77209
Schwarz Criterion	75.59613
Number of Simulations	100
Starting Point of Halton Sequence	11

Figure 25.13 Multinomial Probit Estimates with Unit Variances

The MDC Procedure

Multinomial Probit Estimates

Parameter Estimates					
Parameter	**DF**	**Estimate**	**Standard Error**	**t Value**	**Approx Pr > |t|**
ttime	1	-0.2307	0.0472	-4.89	<.0001
RHO_21	1	0.4820	0.3135	1.54	0.1242

HEV and Multinomial Probit: Heteroscedastic Utility Function

When the stochastic components of utility are heteroscedastic and independent, you can model the data by using an HEV or a multinomial probit model. The HEV model assumes that the utility of alternative *j* for each individual *i* has heteroscedastic random components,

$$U_{ij} = V_{ij} + \epsilon_{ij}$$

where the cumulative distribution function of the Gumbel distributed ϵ_{ij} is

$$F(\epsilon_{ij}) = \exp(-\exp(-\epsilon_{ij}/\theta_j))$$

Note that the variance of ϵ_{ij} is $\frac{1}{6}\pi^2\theta_j^2$. Therefore, the error variance is proportional to the square of the scale parameter θ_j. For model identification, at least one of the scale parameters must be normalized to 1. The following SAS statements estimate an HEV model under a unit scale restriction for mode "1" ($\theta_1 = 1$):

```
/*-- hev with gauss-laguerre method --*/
proc mdc data=newdata;
   model decision = ttime /
            type=hev
            nchoice=3
            hev=(unitscale=1, integrate=laguerre)
            covest=hess;
   id pid;
run;
```

The results of computation are presented in Figure 25.14 and Figure 25.15.

Figure 25.14 HEV Estimation Summary

The MDC Procedure

Heteroscedastic Extreme Value Model Estimates

Model Fit Summary	
Dependent Variable	decision
Number of Observations	50
Number of Cases	150
Log Likelihood	-33.41383
Maximum Absolute Gradient	0.0000218
Number of Iterations	11
Optimization Method	Dual Quasi-Newton
AIC	72.82765
Schwarz Criterion	78.56372

Figure 25.15 HEV Parameter Estimates

The MDC Procedure

Heteroscedastic Extreme Value Model Estimates

Parameter Estimates					
Parameter	**DF**	**Estimate**	**Standard Error**	**t Value**	**Approx Pr > |t|**
ttime	1	-0.4407	0.1798	-2.45	0.0143
SCALE2	1	0.7765	0.4348	1.79	0.0741
SCALE3	1	0.5753	0.2752	2.09	0.0366

The parameters SCALE2 and SCALE3 in the output correspond to the estimates of the scale parameters θ_2 and θ_3, respectively.

Note that the estimate of the HEV model is not always stable because computation of the log-likelihood function requires numerical integration. Bhat (1995) proposed the Gauss-Laguerre method. In general, the log-likelihood function value of HEV should be larger than that of conditional logit because HEV models include the conditional logit as a special case. However, in this example the reverse is true (–33.414 for the HEV model, which is less than –33.321 for the conditional logit model). (See Figure 25.14 and Figure 25.3.) This indicates that the Gauss-Laguerre approximation to the true probability is too coarse. You can see how well the Gauss-Laguerre method works by specifying a unit scale restriction for all modes, as in the following statements, since the HEV model with the unit variance for all modes reduces to the conditional logit model:

```
/*-- hev with gauss-laguerre and unit scale --*/
proc mdc data=newdata;
   model decision = ttime /
            type=hev
            nchoice=3
            hev=(unitscale=1 2 3, integrate=laguerre)
            covest=hess;
   id pid;
run;
```

Figure 25.16 shows that the ttime coefficient is not close to that of the conditional logit model.

Figure 25.16 HEV Estimates with All Unit Scale Parameters

The MDC Procedure

Heteroscedastic Extreme Value Model Estimates

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
ttime	1	-0.2926	0.0438	-6.68	<.0001

There is another option of specifying the integration method. The INTEGRATE=HARDY option uses the adaptive Romberg-type integration method. The adaptive integration produces much more accurate probability and log-likelihood function values, but often it is not practical to use this method of analyzing the HEV model because it requires excessive CPU time. The following SAS statements produce the HEV estimates by using the adaptive Romberg-type integration method:

```
/*-- hev with adaptive integration --*/
proc mdc data=newdata;
   model decision = ttime /
             type=hev
             nchoice=3
             hev=(unitscale=1, integrate=hardy)
             covest=hess;
   id pid;
run;
```

The results are displayed in Figure 25.17 and Figure 25.18.

Figure 25.17 HEV Estimation Summary Using Alternative Integration Method

The MDC Procedure

Heteroscedastic Extreme Value Model Estimates

Model Fit Summary	
Dependent Variable	decision
Number of Observations	50
Number of Cases	150
Log Likelihood	-33.02598
Maximum Absolute Gradient	0.0001202
Number of Iterations	8
Optimization Method	Dual Quasi-Newton
AIC	72.05197
Schwarz Criterion	77.78803

Figure 25.18 HEV Estimates Using Alternative Integration Method

The MDC Procedure

Heteroscedastic Extreme Value Model Estimates

Parameter Estimates					
Parameter	**DF**	**Estimate**	**Standard Error**	**t Value**	**Approx Pr > |t|**
ttime	1	-0.4580	0.1861	-2.46	0.0139
SCALE2	1	0.7757	0.4283	1.81	0.0701
SCALE3	1	0.6908	0.3384	2.04	0.0412

With the INTEGRATE=HARDY option, the log-likelihood function value of the HEV model, −33.026, is greater than that of the conditional logit model, −33.321. (See Figure 25.17 and Figure 25.3.)

When you impose unit scale restrictions on all choices, as in the following statements, the HEV model gives the same estimates as the conditional logit model. (See Figure 25.19 and Figure 25.6.)

```
/*-- hev with adaptive integration and unit scale --*/
proc mdc data=newdata;
   model decision = ttime /
            type=hev
            nchoice=3
            hev=(unitscale=1 2 3, integrate=hardy)
            covest=hess;
   id pid;
run;
```

Figure 25.19 Alternative HEV Estimates with Unit Scale Restrictions

The MDC Procedure

Heteroscedastic Extreme Value Model Estimates

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
ttime	1	-0.3572	0.0776	-4.60	<.0001

For comparison, the following statements estimate a heteroscedastic multinomial probit model by imposing a zero restriction on the correlation parameter, $\rho_{31} = 0$. The MDC procedure requires normalization of at least two of the error variances in the multinomial probit model. Also, for identification, the correlation parameters associated with a unit normalized variance are restricted to be zero. When the UNITVARIANCE= option is specified, the zero restriction on correlation coefficients applies to the last choice of the list. In the following statements, the variances of the first and second choices are normalized. The UNITVARIANCE=(1 2) option imposes additional restrictions that $\rho_{32} = \rho_{21} = 0$. The default for the UNITVARIANCE= option is the last two choices (which would have been equivalent to UNITVARIANCE=(2 3) for this example). The result is presented in Figure 25.20.

The utility function can be defined as

$$U_{ij} = V_{ij} + \epsilon_{ij}$$

where

$$\boldsymbol{\epsilon}_i \sim N\left(\mathbf{0}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \sigma_3^2 \end{bmatrix}\right)$$

```
/*-- mprobit estimation --*/
proc mdc data=newdata;
   model decision = ttime /
            type=mprobit
            nchoice=3
            unitvariance=(1 2)
            covest=hess;
   id pid;
   restrict RHO_31 = 0;
run;
```

Figure 25.20 Heteroscedastic Multinomial Probit Estimates

The MDC Procedure

Multinomial Probit Estimates

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|	Parameter Label
ttime	1	-0.3206	0.0920	-3.49	0.0005	
STD_3	1	1.6913	0.6906	2.45	0.0143	
RHO_31	0	0	0			
Restrict1	1	1.1854	1.5490	0.77	0.4499*	Linear EC [1]

*** Probability computed using beta distribution.**

Note that in the output the estimates of standard errors and correlations are denoted by STD_i and RHO_ij, respectively. In this particular case the first two variances (STD_1 and STD_2) are normalized to one, and corresponding correlations (RHO_21 and RHO_32) are set to zero, so they are not listed among parameter estimates.

Parameter Heterogeneity: Mixed Logit

One way of modeling unobserved heterogeneity across individuals in their sensitivity to observed exogenous variables is to use the mixed logit model with a random parameters or random coefficients specification. The probability of choosing alternative j is written as

$$P_i(j) = \frac{\exp(\mathbf{x}'_{ij}\boldsymbol{\beta})}{\sum_{k=1}^{J}\exp(\mathbf{x}'_{ik}\boldsymbol{\beta})}$$

where $\boldsymbol{\beta}$ is a vector of coefficients that varies across individuals and $\mathbf{x}_{ij}$ is a vector of exogenous attributes.

For example, you can specify the distribution of the parameter β to be the normal distribution.

The mixed logit model uses a Monte Carlo simulation method to estimate the probabilities of choice. There are two simulation methods available. If the RANDNUM=PSEUDO option is specified in the MODEL statement, pseudo-random numbers are generated; if the RANDNUM=HALTON option is specified, Halton quasi-random sequences are used. The default value is RANDNUM=HALTON.

You can estimate the model with normally distributed random coefficients of ttime with the following SAS statements:

```
/*-- mixed logit estimation --*/
proc mdc data=newdata type=mixedlogit;
   model decision = ttime /
            nchoice=3
            mixed=(normalparm=ttime);
   id pid;
run;
```

Let β^m and β^s be mean and scale parameters, respectively, for the random coefficient, β. The relevant utility function is

$$U_{ij} = \text{ttime}_{ij}\beta + \epsilon_{ij}$$

where $\beta = \beta^m + \beta^s \eta$ (β^m and β^s are fixed mean and scale parameters, respectively). The stochastic component, η, is assumed to be standard normal since the NORMALPARM= option is given. Alternatively, the UNIFORMPARM= or LOGNORMALPARM= option can be specified. The LOGNORMALPARM= option is useful when nonnegative parameters are being estimated. The NORMALPARM=, UNIFORMPARM=, and LOGNORMALPARM= variables must be included in the right-hand side of the MODEL statement. See the section "Mixed Logit Model" on page 1430 for more details. To estimate a mixed logit model by using the transportation mode choice data, the MDC procedure requires the MIXED= option for random components. Results of the mixed logit estimation are displayed in Figure 25.21.

Figure 25.21 Mixed Logit Model Parameter Estimates

The MDC Procedure

Mixed Multinomial Logit Estimates

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
ttime_M	1	-0.5342	0.2184	-2.45	0.0144
ttime_S	1	0.2843	0.1911	1.49	0.1368

Note that the parameter ttime_M corresponds to the constant mean parameter β^m and the parameter ttime_S corresponds to the constant scale parameter β^s of the random coefficient β.

Syntax: MDC Procedure

The MDC procedure is controlled by the following statements:

```
PROC MDC options ;
  MDCDATA options ;
  BOUNDS bound1 < , bound2 . . . > ;
  BY variables ;
  CLASS options ;
  ID variable ;
  MODEL dependent variables = regressors / options ;
  NEST LEVEL(value) = ((values)@(value),. . . , (values)@(value)) ;
  NLOPTIONS options ;
  OUTPUT options ;
  RESTRICT restriction1 < , restriction2 . . . > ;
  TEST options ;
  UTILITY  U() = variables, . . . , U() = variables  ;
```

Functional Summary

Table 25.2 summarizes the statements and options used with the MDC procedure.

Table 25.2 MDC Functional Summary

Description	Statement	Option
Data Set Options		
Formats the data for use by PROC MDC	MDCDATA	
Specifies the input data set	MDC	DATA=
Specifies the output data set for CLASS STATEMENT	CLASS	OUT =
Writes parameter estimates to an output data set	MDC	OUTEST=
Includes covariances in the OUTEST= data set	MDC	COVOUT
Writes linear predictors and predicted probabilities to an output data set	OUTPUT	OUT=
Declaring the Role of Variables		
Specifies the ID variable	ID	
Specifies BY-group processing variables	BY	
Printing Control Options		
Requests all printing options	MODEL	ALL
Displays correlation matrix of the estimates	MODEL	CORRB
Displays covariance matrix of the estimates	MODEL	COVB
Displays detailed information about optimization iterations	MODEL	ITPRINT
Suppresses all displayed output	MODEL	NOPRINT

Description	Statement	Option
Model Estimation Options		
Specifies the choice variables	MODEL	CHOICE=()
Specifies the convergence criterion	MODEL	CONVERGE=
Specifies the type of covariance matrix	MODEL	COVEST=
Specifies the starting point of the Halton sequence	MODEL	HALTONSTART=
Specifies options specific to the HEV model	MODEL	HEV=()
Sets the initial values of parameters used by the iterative optimization algorithm	MODEL	INITIAL=()
Specifies the maximum number of iterations	MODEL	MAXITER=
Specifies the options specific to mixed logit	MODEL	MIXED=()
Specifies the number of choices for each person	MODEL	NCHOICE=
Specifies the number of simulations	MODEL	NSIMUL=
Specifies the optimization technique	MODEL	OPTMETHOD=
Specifies the type of random number generators	MODEL	RANDNUM=
Specifies that initial values are generated using random numbers	MODEL	RANDINIT
Specifies the rank dependent variable	MODEL	RANK
Specifies optimization restart options	MODEL	RESTART=()
Specifies a restriction on inclusive parameters	MODEL	SAMESCALE
Specifies a seed for pseudo-random number generation	MODEL	SEED=
Specifies a stated preference data restriction on inclusive parameters	MODEL	SPSCALE
Specifies the type of the model	MODEL	TYPE=
Specifies normalization restrictions on multinomial probit error variances	MODEL	UNITVARIANCE=()
Controlling the Optimization Process		
Specifies upper and lower bounds for the parameter estimates	BOUNDS	
Specifies linear restrictions on the parameter estimates	RESTRICT	
Specifies nonlinear optimization options	NLOPTIONS	
Nested Logit Related Options		
Specifies the tree structure	NEST	LEVEL()=
Specifies the type of utility function	UTILITY	U()=
Output Control Options		
Outputs predicted probabilities	OUTPUT	P=
outputs estimated linear predictor	OUTPUT	XBETA=
Test Request Options		
Requests Wald, Lagrange multiplier, and likelihood ratio tests	TEST	ALL

Description	Statement	Option
Requests the Wald test	TEST	WALD
Requests the Lagrange multiplier test	TEST	LM
Requests the likelihood ratio test	TEST	LR

PROC MDC Statement

PROC MDC *options* **;**

The following options can be used in the PROC MDC statement.

DATA=*SAS-data-set*
specifies the input SAS data set. If the DATA= option is not specified, PROC MDC uses the most recently created SAS data set.

OUTEST=*SAS-data-set*
names the SAS data set that the parameter estimates are written to. See "OUTEST= Data Set" later in this chapter for information about the contents of this data set.

COVOUT
writes the covariance matrix for the parameter estimates to the OUTEST= data set. This option is valid only if the OUTEST= option is specified.

In addition, any of the following MODEL statement options can be specified in the PROC MDC statement, which is equivalent to specifying the option for the MODEL statement: ALL, CONVERGE=, CORRB, COVB, COVEST=, HALTONSTART=, ITPRINT, MAXITER=, NOPRINT, NSIMUL=, OPTMETHOD=, RANDINIT, RANK, RESTART=, SAMESCALE, SEED=, SPSCALE, TYPE=, and UNITVARIANCE=.

MDCDATA Statement

MDCDATA *options* < */ OUT= SAS-data-set* > **;**

The MDCDATA statement prepares data for use by PROC MDC when the choice-specific information is stored in multiple variables (for example, see Figure 25.1 in the section "Conditional Logit: Estimation and Prediction" on page 1395).

VARLIST (*name1 = (var1 var2 . . .) name2 = (var1 var2 . . .) . . .***)**
creates *name* variables from a multiple-variable list of choice alternatives in parentheses. The choice-specific dummy variables are created for the first set of multiple variables. At least one set of multiple variables must be specified. The order of (*var1 var2* ...) in the VARLIST option determines the numbering of the alternative; that is, *var1* corresponds to alternative 1, *var2* corresponds to alternative 2, and so on.

SELECT=(*variable***)**
specifies a variable that contains choices for each individual. The SELECT= *variable* needs to be a character-type variable, with values that match variable names in the first VARLIST option: *name1=(var1 var2 ...)*.

ID=(*name***)**
creates a variable that identifies each individual.

ALT=(*name***)**
identifies selection alternatives for each individual.

DECVAR=(*name***)**
creates a 0/1 variable that indicates the choice made for each individual.

OUT=*SAS-data-set*
specifies a SAS data set to which modified data are output.

BOUNDS Statement

BOUNDS *bound1* < , *bound2* . . . > **;**

The BOUNDS statement imposes simple boundary constraints on the parameter estimates. BOUNDS statement constraints refer to the parameters estimated by the MDC procedure. You can specify any number of BOUNDS statements.

Each *bound* is composed of parameters, constants, and inequality operators:

item operator item < *operator item* < *operator item* . . . > > **;**

Each *item* is a constant, parameter, or list of parameters. Parameters associated with a regressor variable are referred to by the name of the corresponding regressor variable. Each *operator* is <, >, <=, or >=.

You can use both the BOUNDS statement and the RESTRICT statement to impose boundary constraints; however, the BOUNDS statement provides a simpler syntax for specifying these kinds of constraints. See also the section "RESTRICT Statement" on page 1424.

Lagrange multipliers are reported for all the active boundary constraints. In the displayed output, the Lagrange multiplier estimates are identified with the names Restrict1, Restrict2, and so on. The probability of the Lagrange multipliers is computed using a beta distribution (LaMotte 1994). Nonactive (nonbinding) bounds have no effect on the estimation results and are not noted in the output.

The following BOUNDS statement constrains the estimates of the coefficient of ttime to be negative and the coefficients of x1 through x10 to be between zero and one. This example illustrates the use of parameter lists to specify boundary constraints.

```
bounds ttime < 0,
       0 < x1-x10 < 1;
```

BY Statement

BY *variables* **;**

A BY statement can be used with PROC MDC to obtain separate analyses on observations in groups defined by the BY variables.

CLASS Statement

CLASS *variables* **;**

The CLASS statement names the classification variables to be used in the analysis. Classification variables can be either character or numeric.

ID Statement

ID *variable* **;**

The ID statement must be used with PROC MDC to specify the identification variable that controls multiple choice-specific cases. The MDC procedure requires only one ID statement even with multiple MODEL statements.

MODEL Statement

MODEL *dependent* = *regressors* < / *options* > **;**

The MODEL statement specifies the dependent variable and independent regressor variables for the regression model. When the nested logit model is estimated, regressors in the UTILITY statement are used for estimation.

The following options can be used in the MODEL statement after a slash (/).

CHOICE=(*variables* **)**

CHOICE=(*variable numbers* **)**

specifies the variables that contain possible choices for each individual. Choice variables must have integer values. Multiple choice variables are allowed only for nested logit models and must be specified in order from the highest level to the lowest level. For example, CHOICE=(upmode, mode) indicates that the nested logit model has two levels. The choices at the upper level are described by the upmode variable, and the choices at the lower level are described by the mode variable. If all possible alternatives are written with the variable name, the MDC procedure checks all values of the choice variable. CHOICE=(X 1 2 3) implies that the value of X should be 1, 2, or 3. On the other hand, the CHOICE=(X) considers all distinctive nonmissing values of X as elements of the choice set.

CONVERGE=*number*

specifies the convergence criterion. The CONVERGE= option is the same as the ABSGCONV= option in the NLOPTIONS statement. The ABSGCONV= option in the NLOPTIONS statement overrides the CONVERGE= option. The default value is 1E–5.

HALTONSTART=*number*

specifies the starting point of the Halton sequence. The specified number must be a positive integer. The default is HALTONSTART=11.

HEV=(*option-list* **)**

specifies options that are used to estimate the HEV model. The HEV model with a unit scale for the alternative 1 is estimated using the following SAS statement:

```
model y = x1 x2 x3 / hev=(unitscale=1);
```

The following options can be used in the HEV= option. These options are listed within parentheses and separated by commas.

INTORDER=*number*

specifies the number of summation terms for Gaussian quadrature integration. The default is INTORDER=40. The maximum order is limited to 45. This option applies only to the INTEGRATION=LAGUERRE method.

UNITSCALE=*number-list*

specifies restrictions on scale parameters of stochastic utility components.

INTEGRATE=LAGUERRE | HARDY

specifies the integration method. The INTEGRATE=HARDY option specifies an adaptive integration method, while the INTEGRATE=LAGUERRE option specifies the Gauss-Laguerre approximation method. The default is INTEGRATE=LAGUERRE.

MIXED=(*option-list* **)**

specifies options that are used for mixed logit estimation. The mixed logit model with normally distributed random parameters is specified as follows:

```
model y = x1 x2 x3 / mixed=(normalparm=x1);
```

The following options can be used in the MIXED= option. The options are listed within parentheses and separated by commas.

LOGNORMALPARM=*variables*

specifies the variables whose random coefficients are lognormally distributed. LOGNORMALPARM= *variables* must be included on the right-hand side of the MODEL statement.

NORMALEC=*variables*

specifies the error component variables whose coefficients have a normal distribution $N(0, \sigma^2)$.

NORMALPARM=*variables*

specifies the variables whose random coefficients are normally distributed. NORMALPARM= *variables* must be included on the right-hand side of the MODEL statement.

UNIFORMEC=*variables*

specifies the error component variables whose coefficients have a uniform distribution $U(-\sqrt{3}\sigma, \sqrt{3}\sigma)$.

UNIFORMPARM=*variables*

specifies the variables whose random coefficients are uniformly distributed. UNIFORMPARM= *variables* must be included on the right-hand side of the MODEL statement.

NCHOICE=*number*

specifies the number of choices for multinomial choice models when all individuals have the same choice set. When individuals have different number of choices, the NCHOICE= option is not allowed, and the CHOICE= option should be used. The NCHOICE= and CHOICE= options must not be used simultaneously, and the NCHOICE= option cannot be used for nested logit models.

NSIMUL=*number*

specifies the number of simulations when the mixed logit or multinomial probit model is estimated. The default is NSIMUL=100. In general, you need a smaller number of simulations with RANDNUM=HALTON than with RANDNUM=PSEUDO.

RANDNUM=*value*

specifies the type of the random number generator used for simulation. RANDNUM=HALTON is the default. The following option values are allowed:

PSEUDO	specifies pseudo-random number generation.
HALTON	specifies Halton sequence generation.

RANDINIT

RANDINIT=*number*

specifies that initial parameter values be perturbed by uniform pseudo-random numbers for numerical optimization of the objective function. The default is $U(-1, 1)$. When the RANDINIT=r option is specified, $U(-r, r)$ pseudo-random numbers are generated. The value r should be positive. With a RANDINIT or RANDINIT= option, there are pure random searches for a given number of trials (1,000 for conditional or nested logit, and 500 for other models) to get a maximum (or minimum) value of the objective function. For example, when there is a parameter estimate with an initial value of 1, the RANDINIT option adds a generated random number u to the initial value and computes an objective function value by using $1 + u$. This option is helpful in finding the initial value automatically if there is no guidance in setting the initial estimate.

RANK

specifies that the dependent variable contain ranks. The numbers must be positive integers starting from 1. When the dependent variable has value 1, the corresponding alternative is chosen. This option is provided only as a convenience to the user; the extra information contained in the ranks is not currently used for estimation purposes.

RESTART=(*option-list* **)**

specifies options that are used for reiteration of the optimization problem. When the ADDRANDOM option is specified, the initial value of reiteration is computed using random grid searches around the initial solution, as follows:

```
model y = x1 x2 / type=clogit
   restart=(addvalue=(.01 .01));
```

The preceding SAS statement reestimates a conditional logit model by adding ADDVALUE= values. If the ADDVALUE= option contains missing values, the RESTART= option uses the corresponding estimate from the initial stage. If no ADDVALUE= value is specified for an estimate, a default value equal to (|estimate| * 1e-3) is added to the corresponding estimate from the initial stage. If both the ADDVALUE= and ADDRANDOM(=) options are specified, ADDVALUE= is ignored.

The following options can be used in the RESTART= option. The options are listed within parentheses.

ADDMAXIT=*number*
: specifies the maximum number of iterations for the second stage of the estimation. The default is ADDMAXIT=100.

ADDRANDOM | ADDRANDOM=*value*
: specifies random added values to the estimates from the initial stage. With the ADDRANDOM option, $U(-1, 1)$ random numbers are created and added to the estimates obtained in the initial stage. When the ADDRANDOM=*r* option is specified, $U(-r, r)$ random numbers are generated. The restart initial value is determined based on the given number of random searches (1,000 for conditional or nested logit, and 500 for other models).

ADDVALUE=(*value-list*)
: specifies values added to the estimates from the initial stage. A missing value in the list is considered as a zero value for the corresponding estimate. When the ADDVALUE= option is not specified, default values equal to (|estimate| * 1e-3) are added.

SAMESCALE
: specifies that the parameters of the inclusive values be the same within a group at each level when the nested logit is estimated.

SEED=*number*
: specifies an initial seed for pseudo-random number generation. The SEED= value must be less than $2^{31} - 1$. If the SEED= value is negative or zero, the time of day from the computer's clock is used to obtain the initial seed. The default is SEED=0.

SPSCALE
: specifies that the parameters of the inclusive values be the same for any choice with only one nested choice within a group, for each level in a nested logit model. This option is useful in analyzing stated preference data.

TYPE=*value*
: specifies the type of model to be analyzed. The following model types are supported:

CONDITIONLOGIT | CLOGIT | CL	specifies a conditional logit model.
HEV	specifies a heteroscedastic extreme-value model.
MIXEDLOGIT | MXL	specifies a mixed logit model.
MULTINOMPROBIT | MPROBIT | MP	specifies a multinomial probit model.
NESTEDLOGIT | NLOGIT | NL	specifies a nested logit model.

UNITVARIANCE=(*number-list* **)**
: specifies normalization restrictions on error variances of multinomial probit for the choices whose numbers are given in the list. If the UNITVARIANCE= option is specified, it must include at least two choices. Also, for identification, additional zero restrictions are placed on the correlation coefficients for the last choice in the list.

COVEST=*value*

specifies the type of covariance matrix. The following types are supported:

OP	specifies the covariance from the outer product matrix.
HESSIAN	specifies the covariance from the Hessian matrix.
QML	specifies the covariance from the outer product and Hessian matrices.

When COVEST=OP is specified, the outer product matrix is used to compute the covariance matrix of the parameter estimates. The COVEST=HESSIAN option produces the covariance matrix by using the inverse Hessian matrix. The quasi-maximum likelihood estimates are computed with COVEST=QML. The default is COVEST=HESSIAN when the Newton-Raphson method is used. COVEST=OP is the default when the OPTMETHOD=QN option is specified.

Printing Options

ALL

requests all printing options.

COVB

displays the estimated covariances of the parameter estimates.

CORRB

displays the estimated correlation matrix of the parameter estimates.

ITPRINT

displays the initial parameter estimates, convergence criteria, and constraints of the optimization. At each iteration, the objective function value, the maximum absolute gradient element, the step size, and the slope of search direction are printed. The objective function is the full negative log-likelihood function for the maximum likelihood method. When the ITPRINT option is specified and the NLOPTIONS statement is specified, all printing options in the NLOPTIONS statement are ignored.

NOPRINT

suppresses all displayed output.

Estimation Control Options

You can also specify detailed optimization options in the NLOPTIONS statement. The OPTMETHOD= option overrides the TECHNIQUE= option in the NLOPTIONS statement. The NLOPTIONS statement is ignored if the OPTMETHOD= option is specified.

INITIAL=(*initial-values* **)**

START=(*initial-values* **)**

specifies initial values for some or all of the parameter estimates. The values specified are assigned to model parameters in the same order in which the parameter estimates are displayed in the MDC procedure output.

When you use the INITIAL= option, the initial values in the INITIAL= option must satisfy the restrictions specified for the parameter estimates. If they do not, the initial values you specify are adjusted to satisfy the restrictions.

MAXITER=*number*

sets the maximum number of iterations allowed. The MAXITER= option overrides the MAXITER= option in the NLOPTIONS statement. The default is MAXITER=100.

OPTMETHOD=*value*

specifies the optimization technique when the estimation method uses nonlinear optimization. The following techniques are supported:

QN	specifies the quasi-Newton method.
NR	specifies the Newton-Raphson method.
TR	specifies the trust region method.

The OPTMETHOD=NR option is the same as the TECHNIQUE=NEWRAP option in the NLOPTIONS statement. For the conditional and nested logit models, the default is OPTMETHOD=NR. For other models, the default is OPTMETHOD=QN.

NEST Statement

NEST LEVEL *(level-number)= (choices@choice, . . .)* **;**

The NEST statement is used when one choice variable contains all possible alternatives and the TYPE=NLOGIT option is specified. The decision tree is constructed based on the NEST statement. When the choice set is specified using multiple CHOICE= variables in the MODEL statement, the NEST statement is ignored.

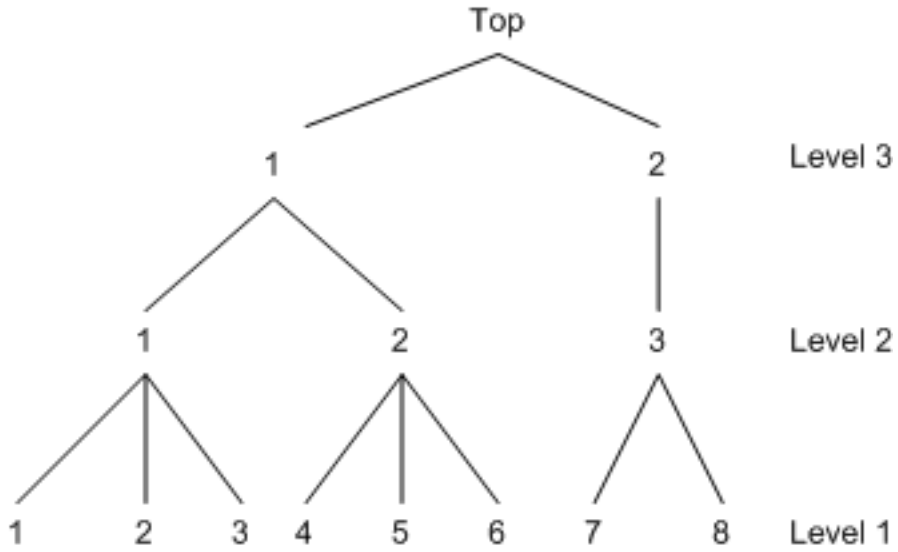
Consider the following eight choices that are nested in a three-level tree structure.

```
Level 1     Level 2     Level 3    top
   1           1           1        1
   2           1           1        1
   3           1           1        1
   4           2           1        1
   5           2           1        1
   6           2           1        1
   7           3           2        1
   8           3           2        1
```

You can use the following NEST statement to specify the tree structure displayed in Figure 25.22:

```
nest level(1) = (1 2 3 @ 1, 4 5 6 @ 2, 7 8 @ 3),
     level(2) = (1 2 @ 1, 3 @ 2),
     level(3) = (1 2 @ 1);
```

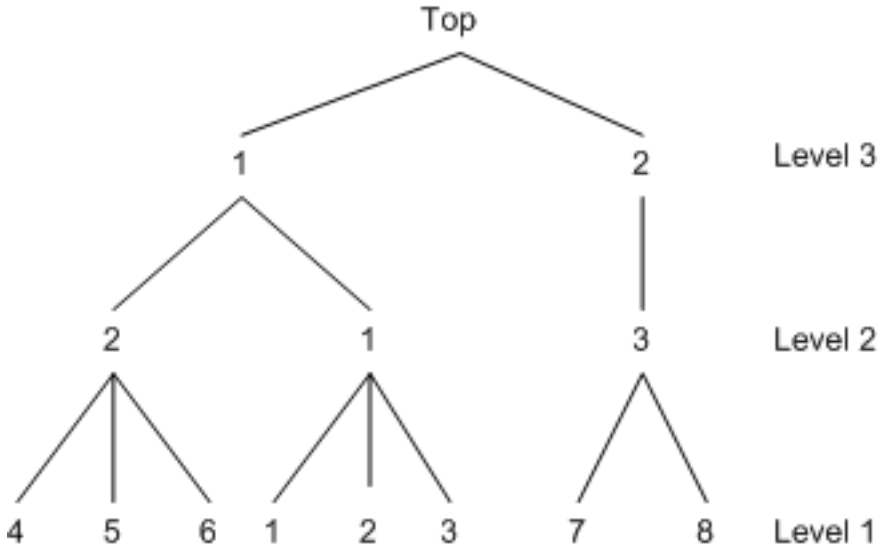
Figure 25.22 A Three-Level Tree



Note that the decision tree is constructed based on the sequence of first-level choice set specification. Therefore, specifying another order at Level 1 builds a different tree. The following NEST statement builds the tree displayed in Figure 25.23:

```
nest level(1) = (4 5 6 @ 2, 1 2 3 @ 1, 7 8 @ 3),
     level(2) = (1 2 @ 1, 3 @ 2),
     level(3) = (1 2 @ 1);
```

Figure 25.23 An Alternative Three-Level Tree



However, the NEST statement with a different sequence of choice specification at higher levels builds the same tree as displayed in Figure 25.22 if the sequence at the first level is the same:

```
nest level(1) = (1 2 3 @ 1, 4 5 6 @ 2, 7 8 @ 3),
     level(2) = (3 @ 2, 1 2 @ 1),
     level(3) = (1 2 @ 1);
```

The following specifications are equivalent:

```
nest level(2) = (3 @ 2, 1 2 @ 1)

nest level(2) = (3 @ 2, 1 @ 1, 2 @ 1)

nest level(2) = (1 @ 1, 2 @ 1, 3 @ 2)
```

Since the MDC procedure contains multiple cases for each individual, it is important to keep the data sequence in the proper order. Consider the four-choice multinomial model with one explanatory variable cost:

```
pid   choice  y   cost
 1      1     1    10
 1      2     0    25
 1      3     0    20
 1      4     0    30
 2      1     0    15
 2      2     0    22
 2      3     1    16
 2      4     0    25
```

The order of data needs to correspond to the value of choice. Therefore, the following data set is equivalent to the preceding data:

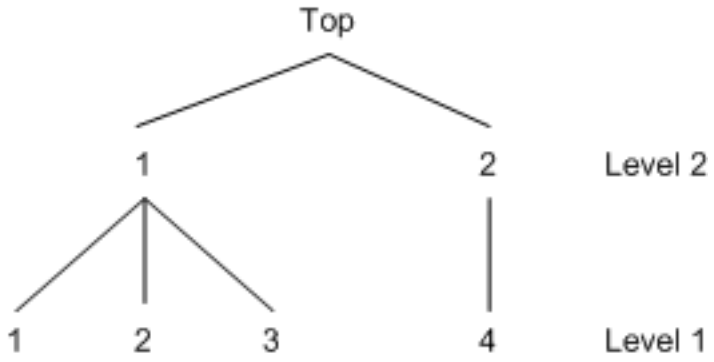
```
pid   choice  y   cost
 1      2     0    25
 1      3     0    20
 1      1     1    10
 1      4     0    30
 2      3     1    16
 2      4     0    25
 2      1     0    15
 2      2     0    22
```

The two-level nested model is estimated with a NEST statement, as follows:

```
proc mdc data=one type=nlogit;
   model y = cost / choice=(choice);
   id pid;
   utility(1,) = cost;
   nest level(1) = (1 2 3 @ 1, 4 @ 2),
        level(2) = (1 2 @ 1);
run;
```

The tree is constructed as in Figure 25.24.

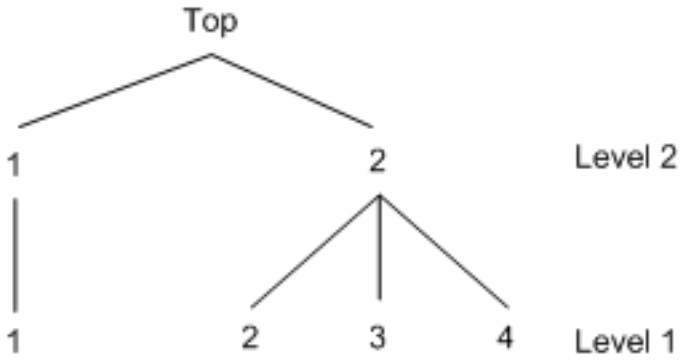
Figure 25.24 A Two-Level Tree



Another model is estimated if you specify the decision tree as in Figure 25.25. The different nested tree structure is specified in the following SAS statements:

```
proc mdc data=one type=nlogit;
   model y = cost / choice=(choice);
   id pid;
   utility u(1,) = cost;
   nest level(1) = (1 @ 1, 2 3 4 @ 2),
        level(2) = (1 2 @ 1);
run;
```

Figure 25.25 An Alternate Two-Level Tree



NLOPTIONS Statement

NLOPTIONS *options* **;**

PROC MDC uses the nonlinear optimization (NLO) subsystem to perform nonlinear optimization tasks. The NLOPTIONS statement specifies nonlinear optimization options. The NLOPTIONS statement must follow the MODEL statement. For a list of all the options of the NLOPTIONS statement, see Chapter 7, "Nonlinear Optimization Methods."

OUTPUT Statement

OUTPUT *options* **;**

The OUTPUT statement creates a new SAS data set that contains all the variables in the input data set and, optionally, the estimated linear predictors (XBETA) and predicted probabilities (P). The input data set must be sorted by the choice variables within each ID.

OUT=*SAS-data-set*
specifies the name of the output data set.

PRED=*variable name*

P=*variable name*
requests the predicted probabilities by naming the variable that contains the predicted probabilities in the output data set.

XBETA=*variable name*
names the variable that contains the linear predictor ($\mathbf{x}'\boldsymbol{\beta}$) values. However, the XBETA= option is not supported in the nested logit model.

RESTRICT Statement

RESTRICT *restriction1* < *, restriction2 . . .* > **;**

The RESTRICT statement imposes linear restrictions on the parameter estimates. You can specify any number of RESTRICT statements.

Each *restriction* is written as an expression, followed by an equality operator (=) or an inequality operator (<, >, <=, >=), followed by a second expression:

expression operator expression **;**

The *operator* can be =, <, >, <= , or >=.

Restriction expressions can be composed of parameters; multiplication ($*$), summation ($+$), and subtraction ($-$) operators; and constants. Parameters named in restriction expressions must be among the parameters estimated by the model. Parameters associated with a regressor variable are referred to by the name of the corresponding regressor variable. The restriction expressions must be a linear function of the parameters.

Lagrange multipliers are reported for all the active linear constraints. In the displayed output, the Lagrange multiplier estimates are identified with the names Restrict1, Restrict2, and so on. The probability of the Lagrange multipliers is computed using a beta distribution (LaMotte 1994).

The following are examples of using the RESTRICT statement:

```
proc mdc data=one;
model y = x1-x10 /
          type=clogit
          choice=(mode 1 2 3);
id pid;
restrict x1*2 <= x2 + x3, ;
run;
```

```
proc mdc data=newdata;
model decision = ttime /
         type=mprobit
         nchoice=3
         unitvariance=(1 2)
         covest=hess;
id pid;
restrict RHO_31 = 0, STD_3<=1;
run;
```

TEST Statement

<'label':> **TEST** *<'string':> equation <,equation. . . > </ options>* **;**

The TEST statement performs Wald, Lagrange multiplier, and likelihood ratio tests of linear hypotheses about the regression parameters in the preceding MODEL statement. Each equation specifies a linear hypothesis to be tested. All hypotheses in one TEST statement are tested jointly. Variable names in the equations must correspond to regressors in the preceding MODEL statement, and each name represents the coefficient of the corresponding regressor. The keyword INTERCEPT refers to the coefficient of the intercept.

The following options can be specified after the slash (/):

ALL
requests Wald, Lagrange multiplier, and likelihood ratio tests.

WALD
requests the Wald test.

LM
requests the Lagrange multiplier test.

LR
requests the likelihood ratio test.

The following statements illustrate the use of the TEST statement:

```
proc mdc;
   model decision = x1 x2 / type=clogit
         choice=(mode 1 2 3);
   id pid;
   test x1 = 0, 0.5 * x1 + 2 * x2 = 0;
run;
```

The test investigates the joint hypothesis that

$$\beta_1 = 0$$

and

$$0.5\beta_1 + 2\beta_2 = 0$$

Only linear equality restrictions and tests are permitted in PROC MDC. Tests expressions can be composed only of algebraic operations that use the addition symbol (+), subtraction symbol (–), and multiplication symbol (*).

The TEST statement accepts labels that are reproduced in the printed output. The TEST statement can be labeled in two ways. A TEST statement can be preceded by a label followed by a colon. Alternatively, the keyword TEST can be followed by a quoted string followed by a colon. If both are present, PROC MDC uses the label that precedes the first colon. If no label is present, PROC MDC automatically labels the tests.

UTILITY Statement

UTILITY U *(level < , choices>)= variables* **;**

The UTILITY statement specifies a utility function that can be used in estimating a nested logit model. The U()= option can have two arguments. The first argument contains level information, and the second argument is related to choice information. The second argument can be omitted for the first level when all the choices at the first level share the same variables and the same parameters. However, for any level above the first, the second argument must be provided. The UTILITY statement specifies a utility function while the NEST statement constructs the decision tree.

Consider a two-level nested logit model that has one explanatory variable at level 1. This model can be specified as follows:

```
proc mdc data=one type=nlogit;
   model y = cost / choice=(choice);
   id pid;
   utility u(1,2 3 4) = cost;
   nest level(1) = (1 @ 1, 2 3 4 @ 2),
        level(2) = (1 2 @ 1);
run;
```

You also can specify the following statement because all the variables at the first level share the same explanatory variable, cost:

```
utility u(1,) = cost;
```

The variable, cost, must be listed in the MODEL statement. When the additional explanatory variable, dummy, is included at level 2, another U()= option needs to be specified. Note that the U()= option must specify choices within any level above the first. Thus, it is specified as U(2, 1 2) in the following statements:

```
proc mdc data=one type=nlogit;
   model y = cost dummy / choice=(choice);
   id pid;
   utility u(1,) = cost,
           u(2,1 2) = dummy;
   nest level(1) = (1 @ 1, 2 3 4 @ 2),
        level(2) = (1 2 @ 1);
run;
```

Details: MDC Procedure

Multinomial Discrete Choice Modeling

When the dependent variable takes multiple discrete values, you can use multinomial discrete choice modeling to analyze the data. This section considers models for unordered multinomial data.

Let the random utility function be defined by

$$U_{ij} = V_{ij} + \epsilon_{ij}$$

where the subscript i is an index for the individual, the subscript j is an index for the alternative, V_{ij} is a nonstochastic utility function, and ϵ_{ij} is a random component (error) that captures unobserved characteristics of alternatives or individuals or both. In multinomial discrete choice models, the utility function is assumed to be linear, so that $V_{ij} = \mathbf{x}'_{ij}\boldsymbol{\beta}$.

In the conditional logit model, each ϵ_{ij} for all $j \in C_i$ is distributed independently and identically (iid) with the Type I extreme-value distribution, $\exp(-\exp(-\epsilon_{ij}))$, also known as the Gumbel distribution.

The iid assumption on the random components of the utilities of the different alternatives can be relaxed to overcome the well-known and restrictive *independence from irrelevant alternatives* (IIA) property of the conditional logit model. This allows for more flexible substitution patterns among alternatives than the one imposed by the conditional logit model. See the section "Independence from Irrelevant Alternatives (IIA)" on page 1429.

The nested logit model is derived by allowing the random components to be identical but nonindependent. Instead of independent Type I extreme-value errors, the errors are assumed to have a generalized extreme-value distribution. This model generalizes the conditional logit model to allow for particular patterns of correlation in unobserved utility (McFadden 1978).

Another generalization of the conditional logit model, the heteroscedastic extreme-value (HEV) model, is obtained by allowing independent but nonidentical errors distributed with a Type I extreme-value distribution (Bhat 1995). It permits different variances on the random components of utility across the alternatives.

Mixed logit models are also generalizations of the conditional logit model that can represent very general patterns of substitution among alternatives. See the "Mixed Logit Model" on page 1430 section for details.

The multinomial probit (MNP) model is derived when the errors, $(\epsilon_{i1}, \epsilon_{i2}, \cdots, \epsilon_{iJ})$, have a multivariate normal (MVN) distribution. Thus, this model accommodates a very general error structure.

The multinomial probit model requires burdensome computation compared to a family of multinomial choice models derived from the Gumbel distributed utility function, since it involves multi-dimensional integration (with dimension $J - 1$) in the estimation process. In addition, the multinomial probit model requires more parameters than other multinomial choice models. As a result, conditional and nested logit models are used more frequently, even though they are derived from a utility function whose random component is more restrictively defined than the multinomial probit model.

The event of a choice being made, $\{y_i = j\}$, can be expressed using a random utility function

$$U_{ij} \geq \mathbf{max}_{k \in C_i, k \neq j} U_{ik}$$

where C_i is the choice set of individual i. Individual i chooses alternative j if and only if it provides a level of utility that is greater than or equal to that of any other alternative in his choice set. Then, the probability that individual i chooses alternative j (from among the n_i choices in his choice set C_i) is

$$P_i(j) = P_{ij} = P[\mathbf{x}'_{ij}\boldsymbol{\beta} + \epsilon_{ij} \geq \mathbf{max}_{k \in C_i}(\mathbf{x}'_{ik}\boldsymbol{\beta} + \epsilon_{ik})]$$

Multinomial Logit and Conditional Logit

When explanatory variables contain only individual characteristics, the multinomial logit model is defined as

$$P(y_i = j) = P_{ij} = \frac{\exp(\mathbf{x}'_i\boldsymbol{\beta}_j)}{\sum_{k=0}^{J}\exp(\mathbf{x}'_i\boldsymbol{\beta}_k)} \quad \text{for } j = 0, \cdots, J$$

where y_i is a random variable that indicates the choice made, x_i is a vector of characteristics specific to the ith individual, and $\boldsymbol{\beta}_j$ is a vector of coefficients specific to the jth alternative. Thus, this model involves choice-specific coefficients and only individual specific regressors. For model identification, it is often assumed that $\boldsymbol{\beta}_0 = 0$. The multinomial logit model reduces to the binary logit model if $J = 1$.

The ratio of the choice probabilities for alternatives j and l (the *odds ratio* of alternatives j and l) is

$$\frac{P_{ij}}{P_{il}} = \frac{\exp(\mathbf{x}'_i\boldsymbol{\beta}_j)/\sum_{k=0}^{J}\exp(\mathbf{x}'_i\boldsymbol{\beta}_k)}{\exp(\mathbf{x}'_i\boldsymbol{\beta}_l)/\sum_{k=0}^{J}\exp(\mathbf{x}'_i\boldsymbol{\beta}_k)} = \exp[\mathbf{x}'_i(\boldsymbol{\beta}_j - \boldsymbol{\beta}_l)]$$

Note that the odds ratio of alternatives j and l does not depend on any alternatives other than j and l. For more information, see the section "Independence from Irrelevant Alternatives (IIA)" on page 1429.

The log-likelihood function of the multinomial logit model is

$$\mathcal{L} = \sum_{i=1}^{N}\sum_{j=0}^{J} d_{ij} \ln P(y_i = j)$$

where

$$d_{ij} = \begin{cases} 1 & \text{if individual } i \text{ chooses alternative } j \\ 0 & \text{otherwise} \end{cases}$$

This type of multinomial choice modeling has a couple of weaknesses: it has too many parameters (the number of individual characteristics times J), and it is difficult to interpret. The multinomial logit model can be used to predict the choice probabilities, among a given set of $J + 1$ alternatives, of an individual with known vector of characteristics $\mathbf{x}_i$.

The parameters of the multinomial logit model can be estimated with the TYPE=CLOGIT option in the MODEL statement; however, this requires modification of the conditional logit model to allow individual specific effects.

The conditional logit model, sometimes called the multinomial logit model, is similarly defined when choice-specific data are available. Using properties of Type I extreme-value (Gumbel) distribution, the probability that individual i chooses alternative j from among the choices in his choice set C_i is

$$P(y_i = j) = P_{ij} = P[\mathbf{x}'_{ij}\boldsymbol{\beta} + \epsilon_{ij} \geq \max_{k \in C_i, k \neq j}(\mathbf{x}'_{ik}\boldsymbol{\beta} + \epsilon_{ik})] = \frac{\exp(\mathbf{x}'_{ij}\boldsymbol{\beta})}{\sum_{k \in C_i}\exp(\mathbf{x}'_{ik}\boldsymbol{\beta})}$$

where $\mathbf{x}_{ij}$ is a vector of attributes specific to the jth alternative as perceived by the ith individual. It is assumed that there are n_i choices in each individual's choice set, C_i.

The log-likelihood function of the conditional logit model is

$$\mathcal{L} = \sum_{i=1}^{N} \sum_{j \in C_i} d_{ij} \ln P(y_i = j)$$

The conditional logit model can be used to predict the probability that an individual will choose a previously unavailable alternative, given knowledge of $\boldsymbol{\beta}$ and the vector $\mathbf{x}_{ij}$ of choice-specific characteristics.

Independence from Irrelevant Alternatives (IIA)

The problematic aspect of the conditional logit (and the multinomial logit) model lies in the property of independence from irrelevant alternatives (IIA). The IIA property can be derived from the probability ratio of any two choices. For the conditional logit model,

$$\frac{P_{ij}}{P_{il}} = \frac{\exp(\mathbf{x}'_{ij}\boldsymbol{\beta}) / \sum_{k \in C_i} \exp(\mathbf{x}'_{ik}\boldsymbol{\beta})}{\exp(\mathbf{x}'_{il}\boldsymbol{\beta}) / \sum_{k \in C_i} \exp(\mathbf{x}'_{ik}\boldsymbol{\beta})} = \exp[(\mathbf{x}_{ij} - \mathbf{x}_{il})'\boldsymbol{\beta}]$$

It is evident that the ratio of the probabilities for alternatives j and l does not depend on any alternatives other than j and l. This was also shown to be the case for the multinomial logit model. Thus, for the conditional and multinomial logit models, the ratio of probabilities of any two alternatives is necessarily the same regardless of what other alternatives are in the choice set or what the characteristics of the other alternatives are. This is referred to as the IIA property.

The IIA property is useful from the point of view of estimation and forecasting. For example, it allows the prediction of demand for currently unavailable alternatives. If the IIA property is appropriate for the choice situation being considered, then estimation can be based on the set of currently available alternatives, and then the estimated model can be used to calculate the probability that an individual would choose a new alternative not considered in the estimation procedure. However, the IIA property is restrictive from the point of view of choice behavior. Models that display the IIA property predict that a change in the attributes of one alternative changes the probabilities of the other alternatives proportionately such that the ratios of probabilities remain constant. Thus, cross elasticities due to a change in the attributes of an alternative j are equal for all alternatives $k \neq j$. This particular substitution pattern might be too restrictive in some choice settings.

The IIA property of the conditional logit model follows from the assumption that the random components of utility are identically and independently distributed. The other models in PROC MDC (namely, nested logit, HEV, mixed logit, and multinomial probit) relax the IIA property in different ways.

For an example of Hausman's specification test of IIA assumption, see "Example 25.6: Hausman's Specification Test" on page 1462.

Heteroscedastic Extreme-Value Model

The heteroscedastic extreme-value (HEV) model (Bhat 1995) allows the random components of the utility function to be nonidentical. Specifically, the HEV model assumes independent but nonidentical error terms distributed with the Type I extreme-value distribution. The HEV model allows the variances of the random

components of utility to differ across alternatives. Bhat (1995) argues that the HEV model does not have the IIA property. The HEV model contains the conditional logit model as a special case. The probability that an individual i will choose alternative j from the set C_i of available alternatives is

$$P_i(j) = \int_{-\infty}^{\infty} \prod_{k \in C_i, k \neq j} \Gamma\left[\frac{\mathbf{x}'_{ij}\boldsymbol{\beta} - \mathbf{x}'_{ik}\boldsymbol{\beta} + \theta_j w}{\theta_k}\right] \gamma(w) dw$$

where the choice set C_i has n_i elements and

$$\Gamma(x) = \exp(-\exp(-x))$$

$$\gamma(x) = \exp(-x)\Gamma(x)$$

are the cumulative distribution function and probability density function of the Type I extreme-value distribution. The variance of the error term for the jth alternative is $\frac{1}{6}\pi^2\theta_j^2$. If the scale parameters, θ_j, of the random components of utility of all alternatives are equal, then this choice probability is the same as that of the conditional logit model. The log-likelihood function of the HEV model can be written as

$$\mathcal{L} = \sum_{i=1}^{N} \sum_{j \in C_i} d_{ij} \ln[P_i(j)]$$

where

$$d_{ij} = \begin{cases} 1 & \text{if individual } i \text{ chooses alternative } j \\ 0 & \text{otherwise} \end{cases}$$

Since the log-likelihood function contains an improper integral function, it is computationally difficult to get a stable estimate. With the transformation $u = \exp(-w)$, the probability can be written

$$\begin{aligned} P_i(j) &= \int_0^{\infty} \Pi_{k \in C_i, k \neq j} \Gamma\left[\frac{\mathbf{x}'_{ij}\boldsymbol{\beta} - \mathbf{x}'_{ik}\boldsymbol{\beta} - \theta_j \ln(u)}{\theta_k}\right] \exp(-u) du \\ &= \int_0^{\infty} G_{ij}(u) \exp(-u) du \end{aligned}$$

Using the Gauss-Laguerre weight function, $W(x) = \exp(-u)$, the integration of the log-likelihood function can be replaced with a summation as follows:

$$\int_0^{\infty} G_{ij}(u) \exp(-u) du = \sum_{k=1}^{K} w_k G_{ij}(x_k)$$

Weights (w_k) and abscissas (x_k) are tabulated by Abramowitz and Stegun (1970).

Mixed Logit Model

In mixed logit models, an individual's utility from any alternative can be decomposed into a deterministic component, $\mathbf{x}'_{ij}\boldsymbol{\beta}$, which is a linear combination of observed variables, and a stochastic component, $\xi_{ij} + \epsilon_{ij}$,

$$U_{ij} = \mathbf{x}'_{ij}\boldsymbol{\beta} + \xi_{ij} + \epsilon_{ij}$$

where $\mathbf{x}_{ij}$ is a vector of observed variables that relate to individual i and alternative j, $\boldsymbol{\beta}$ is a vector of parameters, ξ_{ij} is an error component that can be correlated among alternatives and heteroscedastic for each individual, and ϵ_{ij} is a random term with zero mean that is independently and identically distributed over alternatives and individuals. The conditional logit model is derived if you assume ϵ_{ij} has an iid Gumbel distribution and $V(\xi_{ij}) = 0$.

The mixed logit model assumes a general distribution for ξ_{ij} and an iid Gumbel distribution for ϵ_{ij}. Denote the density function of the error component ξ_{ij} as $f(\xi_{ij}|\boldsymbol{\gamma})$, where $\boldsymbol{\gamma}$ is a parameter vector of the distribution of ξ_{ij}. The choice probability of alternative j for individual i is written as

$$P_i(j) = \int Q_i(j|\xi_{ij}) f(\xi_{ij}|\boldsymbol{\gamma}) d\xi_{ij}$$

where the conditional choice probability for a given value of ξ_{ij} is the logit

$$Q_i(j|\xi_{ij}) = \frac{\exp(\mathbf{x}'_{ij}\boldsymbol{\beta} + \xi_{ij})}{\sum_{k \in C_i} \exp(\mathbf{x}'_{ik}\boldsymbol{\beta} + \xi_{ik})}$$

Since ξ_{ij} is not given, the unconditional choice probability, $P_i(j)$, is the integral of the conditional choice probability, $Q_i(j|\xi_{ij})$, over the distribution of ξ_{ij}. This model is called "mixed logit" since the choice probability is a mixture of logits with $f(\xi_{ij}|\boldsymbol{\gamma})$ as the mixing distribution.

In general, the mixed logit model does not have an exact likelihood function because the probability $P_i(j)$ does not always have a closed form solution. Therefore, a simulation method is used for computing the approximate probability,

$$\tilde{P}_i(j) = 1/S \sum_{s=1}^{S} \tilde{Q}_i(j|\xi_{ij}^s)$$

where S is the number of simulation replications and $\tilde{P}_i(j)$ is a simulated probability. The simulated log-likelihood function is computed as

$$\tilde{\mathcal{L}} = \sum_{i=1}^{N} \sum_{j=1}^{n_i} d_{ij} \ln(\tilde{P}_i(j))$$

where

$$d_{ij} = \begin{cases} 1 & \text{if individual } i \text{ chooses alternative } j \\ 0 & \text{otherwise} \end{cases}$$

For simulation purposes, assume that the error component has a specific structure,

$$\xi_{ij} = \mathbf{z}'_{ij}\boldsymbol{\mu} + \mathbf{w}'_{ij}\boldsymbol{\beta}^*$$

where $\mathbf{z}_{ij}$ is a vector of observed data and $\boldsymbol{\mu}$ is a random vector with zero mean and density function $\psi(\boldsymbol{\mu}|\boldsymbol{\gamma})$. The observed data vector ($\mathbf{z}_{ij}$) of the error component can contain some or all elements of $\mathbf{x}_{ij}$. The component $\mathbf{z}'_{ij}\boldsymbol{\mu}$ induces heteroscedasticity and correlation across unobserved utility components of the alternatives. This allows flexible substitution patterns among the alternatives. The kth element of vector $\boldsymbol{\mu}$ is distributed as

$$\mu_k \sim (0, \sigma_k^2)$$

Therefore, μ_k can be specified as

$$\mu_k = \sigma_k \epsilon_\mu$$

where

$$\epsilon_\mu \sim N(0, 1)$$

or

$$\epsilon_\mu \sim U(-\sqrt{3}, \sqrt{3})$$

In addition, $\boldsymbol{\beta}^*$ is a vector of random parameters (random coefficients). Random coefficients allow heterogeneity across individuals in their sensitivity to observed exogenous variables. The observed data vector, $\mathbf{w}_{ij}$, is a subset of $\mathbf{x}_{ij}$. The following three types of distributions for the random coefficients are supported, where the mth element of $\boldsymbol{\beta}^*$ is denoted as β^*_m:

- Normally distributed coefficient with the mean b_m and spread s_m being estimated.

  $$\beta^*_m = b_m + s_m \epsilon_\beta \quad \text{and} \quad \epsilon_\beta \sim N(0, 1)$$

- Uniformly distributed coefficient with the mean b_m and spread s_m being estimated. A uniform distribution with mean b and spread s is $U(b - s, b + s)$.

  $$\beta^*_m = b_m + s_m \epsilon_\beta \quad \text{and} \quad \epsilon_\beta \sim U(-1, 1)$$

- Lognormally distributed coefficient. The coefficient is calculated as

  $$\beta^*_m = \exp(b_m + s_m \epsilon_\beta) \quad \text{and} \quad \epsilon_\beta \sim N(0, 1)$$

  where b_m and s_m are parameters that are estimated.

The estimate of spread for normally, uniformly, and lognormally distributed coefficients can be negative. The absolute value of the estimated spread can be interpreted as an estimate of standard deviation for normally distributed coefficients.

A detailed description of mixed logit models can be found, for example, in Brownstone and Train (1999).

Multinomial Probit

The multinomial probit model allows the random components of the utility of the different alternatives to be nonindependent and nonidentical. Thus, it does not have the IIA property. The increase in the flexibility of the error structure comes at the expense of introducing several additional parameters in the covariance matrix of the errors.

Consider the random utility function

$$U_{ij} = \mathbf{x}'_{ij} \boldsymbol{\beta} + \epsilon_{ij}$$

where the joint distribution of $(\epsilon_{i1}, \epsilon_{i2}, \cdots, \epsilon_{iJ})$ is multivariate normal:

$$\begin{bmatrix} \epsilon_{i1} \\ \epsilon_{i2} \\ \vdots \\ \epsilon_{iJ} \end{bmatrix} \sim N(\mathbf{0}, \boldsymbol{\Sigma})$$

$$\boldsymbol{\Sigma} = \left[\sigma_{jk}\right]_{j,k=1,\ldots,J}$$

The dimension of the error covariance matrix is determined by the number of alternatives J. Given $(\mathbf{x}_{i1}, \mathbf{x}_{i2}, \cdots, \mathbf{x}_{iJ})$, the jth alternative is chosen if and only if $U_{ij} \geq U_{ik}$ for all $k \neq j$. Thus, the probability that the jth alternative is chosen is

$$P(y_i = j) = P_{ij} = P[\epsilon_{i1} - \epsilon_{ij} < (\mathbf{x}_{ij} - \mathbf{x}_{i1})'\boldsymbol{\beta}, \ldots, \epsilon_{iJ} - \epsilon_{ij} < (\mathbf{x}_{ij} - \mathbf{x}_{iJ})'\boldsymbol{\beta}]$$

where y_i is a random variable that indicates the choice made. This is a cumulative probability from a $(J-1)$-variate normal distribution. Since evaluation of this probability involves multidimensional integration, it is practical to use a simulation method to estimate the model. Many studies have shown that the simulators proposed by the following authors (henceforth referred to as GHK) perform well: Geweke (1989); Hajivassiliou (1993); Keane (1994). For example, Hajivassiliou, McFadden, and Ruud (1996) compare 13 simulators using 11 different simulation methods and conclude that the GHK simulation method is the most reliable. To compute the probability of the multivariate normal distribution, the recursive simulation method is used. Refer to Hajivassiliou (1993) for more details about GHK simulators.

The log-likelihood function for the multinomial probit model can be written as

$$\mathcal{L} = \sum_{i=1}^{N} \sum_{j=1}^{J} d_{ij} \ln P(y_i = j)$$

where

$$d_{ij} = \begin{cases} 1 & \text{if individual } i \text{ chooses alternative } j \\ 0 & \text{otherwise} \end{cases}$$

For identification of the multinomial probit model, two of the diagonal elements of Σ are normalized to 1, and it is assumed that for one of the choices whose error variance is normalized to 1 (say, k), it is also true that $\sigma_{jk} = \sigma_{kj} = 0$ for $j = 1, \cdots, J$ and $j \neq k$. Thus, a model with J alternatives has at most $J(J-1)/2 - 1$ covariance parameters after normalization.

Let D and R be defined as

$$D = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \sigma_J \end{bmatrix}$$

$$R = \begin{bmatrix} 1 & \rho_{12} & \cdots & \rho_{1J} \\ \rho_{21} & 1 & \cdots & \rho_{2J} \\ \vdots & \vdots & \vdots & \vdots \\ \rho_{J1} & \rho_{J2} & \cdots & 1 \end{bmatrix}$$

where $\sigma_j^2 = \sigma_{jj}$ and $\rho_{jk} = \frac{\sigma_{jk}}{\sigma_j \sigma_k}$. Then, for identification, $\sigma_{J-1} = \sigma_J = 1$ and $\rho_{kJ} = \rho_{Jk} = 0$, for all $k \neq J$ can be imposed, and the error covariance matrix is $\mathbf{\Sigma} = DRD$.

In the standard MDC output, the parameter estimates STD_j and RHO_jk correspond to σ_j and ρ_{jk}.

In principle, the multinomial probit model is fully identified with the preceding normalizations. However, in practice, convergence in applications of the model with more than three alternatives often requires additional restrictions on the elements of $\mathbf{\Sigma}$.

It must also be noted that the unrestricted structure of the error covariance matrix makes it impossible to forecast demand for a new alternative without knowledge of the new $(J + 1)$ by $(J + 1)$ error covariance matrix.

Nested Logit

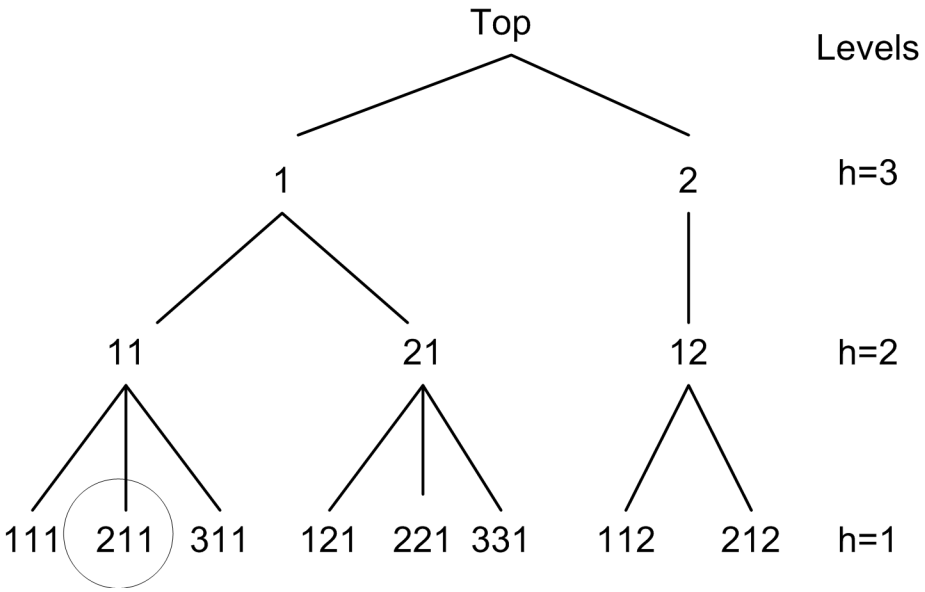
The nested logit model (McFadden 1978, 1981) allows partial relaxation of the assumption of independence of the stochastic components of utility of alternatives. In some choice situations, the IIA property holds for some pairs of alternatives but not all. In these situations, the nested logit model can be used if the set of alternatives faced by an individual can be partitioned into subsets such that the IIA property holds within subsets but not across subsets.

In the nested logit model, the joint distribution of the errors is generalized extreme value (GEV). This is a generalization of the Type I extreme-value distribution that gives rise to the conditional logit model. Note that all ϵ_{ij} within each subset are correlated with each other. Refer to McFadden (1978, 1981) for details.

Nested logit models can be described analytically following the notation of McFadden (1981). Assume that there are L levels, with 1 representing the lowest and L representing the highest level of the tree. The index of a node at level h in the tree is a pair (j_h, π_h), where $\pi_h = (j_{h+1}, \cdots, j_L)$ is the index of the adjacent node at level $h + 1$. Thus, the primitive alternatives, at level 1 in the tree, are indexed by vectors $(j_1, \cdots, j_L)$, and the alternative nodes at level L are indexed by integers j_L. The choice set C_{π_h} is the set of primitive alternatives (at level 1) that belong to branches below the node π_h. The notation C_{π_h} is also used to denote a set of indices j_h such that (j_h, π_h) is a node immediately below π_h. Note that C_{π_0} is a set with a single element, while C_{π_L} represents a choice set that contains all possible alternatives. As an example, consider the circled node at level 1 in Figure 25.26. Since it stems from node 11, $\pi_h = 11$, and since it is the second node stemming from 11, $j_h = 2$, its index is $\pi_{h-1} = \pi_0 = (j_h, \pi_h) = 211$. Similarly, $C_{11} = \{111, 211, 311\}$ contains all the possible choices below 11.

Although this notation is useful for writing closed-form solutions for probabilities, the MDC procedure allows a more flexible definition of indices. See the section "NEST Statement" on page 1420 for more details about how to describe decision trees within the MDC procedure.

Figure 25.26 Node Indices for a Three-Level Tree



Let $\mathbf{x}_{i;j_h\pi_h}^{(h)}$ denote the vector of observed variables for individual i common to the alternatives below node $j_h\pi_h$. The probability of choice at level h has a closed-form solution and is written

$$P_i(j_h|\pi_h) = \frac{\exp\left[\mathbf{x}_{i;j_h\pi_h}^{(h)\prime}\boldsymbol{\beta}^{(h)} + \sum_{k\in C_{i;j_h\pi_h}} I_{k,j_h\pi_h}\theta_{k,j_h\pi_h}\right]}{\sum_{j\in C_{i;\pi_h}} \exp\left[\mathbf{x}_{i;j\pi_h}^{(h)\prime}\boldsymbol{\beta}^{(h)} + \sum_{k\in C_{i;j\pi_h}} I_{k,j\pi_h}\theta_{k,j\pi_h}\right]}, h = 2,\cdots,L$$

where I_{π_h} is the *inclusive value* (at level $h+1$) of the branch below node π_h and is defined recursively as follows:

$$I_{\pi_h} = \ln\left\{\sum_{j\in C_{i;\pi_h}} \exp\left[\mathbf{x}_{i;j\pi_h}^{(h)\prime}\boldsymbol{\beta}^{(h)} + \sum_{k\in C_{i;j\pi_h}} I_{k,j\pi_h}\theta_{k,j\pi_h}\right]\right\}$$

$$0 \le \theta_{k,\pi_1} \le \cdots \le \theta_{k,\pi_{L-1}}$$

The inclusive value I_{π_h} denotes the average utility that the individual can expect from the branch below π_h. The *dissimilarity parameters* or *inclusive value parameters* ($\theta_{k,j\pi_h}$) are the coefficients of the inclusive values and have values between 0 and 1 if nested logit is the correct model specification. When they all take value 1, the nested logit model is equivalent to the conditional logit model.

At decision level 1, there is no inclusive value; that is, $I_{\pi_0} = 0$. Therefore, the conditional probability is

$$P_i(j_1|\pi_1) = \frac{\exp\left[\mathbf{x}_{i;j_1\pi_1}^{(1)\prime}\boldsymbol{\beta}^{(1)}\right]}{\sum_{j\in C_{i;\pi_1}} \exp\left[\mathbf{x}_{i;j\pi_1}^{(1)\prime}\boldsymbol{\beta}^{(1)}\right]}$$

The log-likelihood function at level h can then be written

$$\mathcal{L}^{(h)} = \sum_{i=1}^{N} \sum_{\pi_{h'}\in C_{i,\pi_{h+1}}} \sum_{j\in C_{i,\pi_{h'}}} y_{i,j\pi_{h'}} \ln P(C_{i,j\pi_{h'}}|C_{i,\pi_{h'}})$$

where $y_{i,j\pi_{h'}}$ is an indicator variable that has the value of 1 for the selected choice. The full log-likelihood function of the nested logit model is obtained by adding the conditional log-likelihood functions at each level:

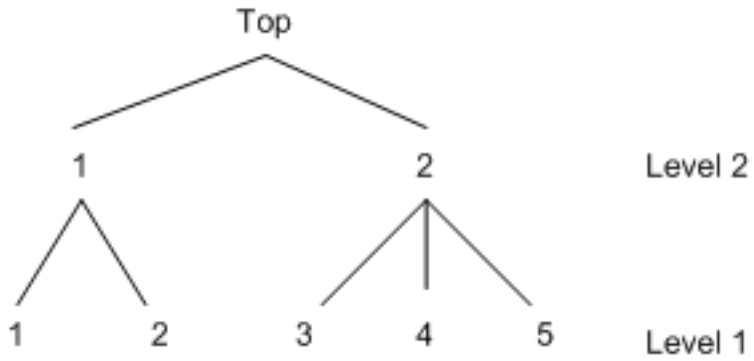
$$\mathcal{L} = \sum_{h=1}^{L} \mathcal{L}^{(h)}$$

Note that the log-likelihood functions are computed from conditional probabilities when $h < L$. The nested logit model is estimated using the full information maximum likelihood method.

Decision Tree and Nested Logit

You can view choices as a decision tree and model the decision tree by using the nested logit model. You need to use either the NEST statement or the CHOICE= option of the MODEL statement to specify the nested tree structure. Additionally, you need to identify which explanatory variables are used at each level of the decision tree. These explanatory variables are arguments for what is called a *utility function*. The utility function is specified using UTILITY statements. For example, consider a two-level decision tree. The tree structure is displayed in Figure 25.27.

Figure 25.27 Two-Level Decision Tree



A nested logit model with two levels can be specified using the following SAS statements:

```
proc mdc data=one type=nlogit;
   model decision = x1 x2 x3 x4 x5 /
         choice=(upmode 1 2, mode 1 2 3 4 5);
   id pid;
   utility u(1, 3 4 5 @ 2) = x1 x2,
           u(1, 1 2 @ 1) = x3 x4,
           u(2, 1 2) = x5;
run;
```

The DATA=one data set should be arranged as follows:

obs	pid	upmode	mode	x1	x2	x3	x4	x5	decision
1	1	1	1	#	#	#	#	#	1
2	1	1	2	#	#	#	#	#	0
3	1	2	3	#	#	#	#	#	0
4	1	2	4	#	#	#	#	#	0
5	1	2	5	#	#	#	#	#	0
6	2	1	1	#	#	#	#	#	0
7	2	1	2	#	#	#	#	#	0
8	2	2	3	#	#	#	#	#	0
9	2	2	4	#	#	#	#	#	0
10	2	2	5	#	#	#	#	#	1

All model variables, x1 through x5, are specified in the UTILITY statement. It is required that entries denoted as # have values for model estimation and prediction. The values of the level 2 utility variable x5 should be the same for all the primitive (level 1) alternatives below node 1 at level 2 and, similarly, for all the primitive alternatives below node 2 at level 2. In other words, x5 should have the same value for primitive alternatives 1 and 2 and, similarly, it should have the same value for primitive alternatives 3, 4, and 5. More generally, the values of any level 2 or higher utility function variables should be constant across primitive alternatives under each node for which the utility function applies. Since PROC MDC expects this to be the case, it uses the values of x5 only for the primitive alternatives 1 and 3, ignoring the values for the primitive alternatives 2, 4, and 5. Thus, PROC MDC uses the values of the utility function variable only for the primitive alternatives that come first under each node for which the utility function applies. This behavior applies to any utility function variables that are specified above the first level. The choice variable for level 2 (upmode) should be placed before the first-level choice variable (mode) when the CHOICE= option is specified. Alternatively, the NEST statement can be used to specify the decision tree. The following SAS statements fit the same nested logit model:

```
proc mdc data=a type=nlogit;
   model decision = x1 x2 x3 x4 x5 /
            choice=(mode 1 2 3 4 5);
   id pid;
   utility u(1, 3 4 5 @ 2) = x1 x2,
           u(1, 1 2 @ 1) = x3 x4,
           u(2, 1 2) = x5;
   nest level(1) = (1 2 @ 1, 3 4 5 @ 2),
        level(2) = (1 2 @ 1);
run;
```

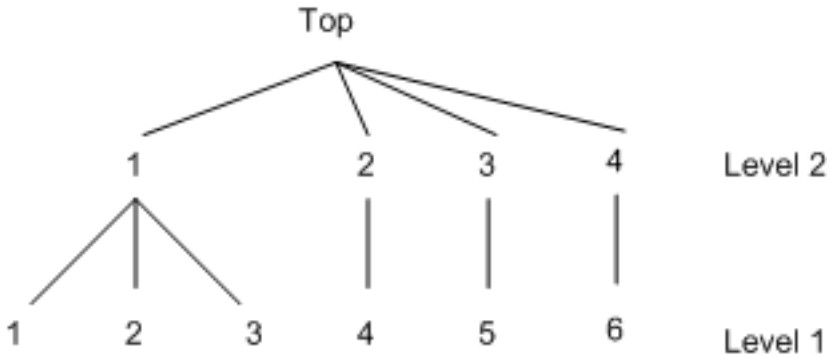
The U(1, 3 4 5 @ 2)= option specifies three choices, 3, 4, and 5, at level 1 of the decision tree. They are connected to the upper branch 2. The specified variables (x1 and x2) are used to model this utility function. The bottom level of the decision tree is level 1. All variables in the UTILITY statement must be included in the MODEL statement. When all choices at the first level share the same variables, you can omit the second argument of the U()= option for that level. However, U(1,) = x1 x2 is not equivalent to the following statements:

```
u(1, 3 4 5 @ 2) = x1 x2;
u(1, 1 2 @ 1) = x1 x2;
```

The CHOICE= variables need to be specified from the top to the bottom level. To forecast demand for new products, stated preference data are widely used. Stated preference data are attractive for market researchers

because attribute variations can be controlled. Hensher (1993) explores the advantage of combining revealed preference (market data) and stated preference data. The scale factor (V_{rp}/V_{sp}) can be estimated using the nested logit model with the decision tree structure displayed in Figure 25.28.

Figure 25.28 Decision Tree for Revealed and Stated Preference Data



Example SAS statements are as follows:

```
proc mdc data=a type=nlogit;
   model decision = x1 x2 x3 /
            spscale
            choice=(mode 1 2 3 4 5 6);
   id pid;
   utility u(1,) = x1 x2 x3;
   nest level(1) = (1 2 3 @ 1, 4 @ 2, 5 @ 3, 6 @ 4),
        level(2) = (1 2 3 4 @ 1);
run;
```

The SPSCALE option specifies that parameters of inclusive values for nodes 2, 3, and 4 at level 2 be the same. When you specify the SAMESCALE option, the MDC procedure imposes the same coefficient of inclusive values for choices 1–4.

Model Fit and Goodness-of-Fit Statistics

McFadden (1974) suggests a likelihood ratio index that is analogous to the R-square in the linear regression model:

$$R_M^2 = 1 - \frac{\ln L}{\ln L_0}$$

where L is the maximum of the log-likelihood function and L_0 is the maximum of the log-likelihood function when all coefficients, except for an intercept term, are zero. McFadden's likelihood ratio index is bounded by 0 and 1.

Estrella (1998) proposes the following requirements for a goodness-of-fit measure to be desirable in discrete choice modeling:

- The measure must take values in [0, 1], where 0 represents no fit and 1 corresponds to perfect fit.

- The measure should be directly related to the valid test statistic for the significance of all slope coefficients.
- The derivative of the measure with respect to the test statistic should comply with corresponding derivatives in a linear regression.

Estrella's measure is written as

$$R^2_{E1} = 1 - \left(\frac{\ln L}{\ln L_0}\right)^{-(2/N)\ln L_0}$$

Estrella suggests an alternative measure,

$$R^2_{E2} = 1 - [(\ln L - K)/\ln L_0]^{-(2/N)\ln L_0}$$

where $\ln L_0$ is computed with null parameter values, N is the number of observations used, and K represents the number of estimated parameters.

Other goodness-of-fit measures are summarized as follows:

$$\begin{aligned}
R^2_{CU1} &= 1 - \left(\frac{L_0}{L}\right)^{\frac{2}{N}} && \text{(Cragg-Uhler 1)}\\
R^2_{CU2} &= \frac{1-(L_0/L)^{\frac{2}{N}}}{1-L_0^{\frac{2}{N}}} && \text{(Cragg-Uhler 2)}\\
R^2_A &= \frac{2(\ln L - \ln L_0)}{2(\ln L - \ln L_0)+N} && \text{(Aldrich-Nelson)}\\
R^2_{VZ} &= R^2_A \frac{2\ln L_0 - N}{2\ln L_0} && \text{(Veall-Zimmermann)}
\end{aligned}$$

The AIC and SBC are computed as follows:

$$AIC = -2\,ln(L) + 2\,k$$
$$SBC = -2\,ln(L) + ln(n)\,k$$

where $ln(L)$ is the log-likelihood value for the model, k is the number of parameters estimated, and n is the number of observations (that is, the number of respondents).

Tests on Parameters

In general, the hypothesis to be tested can be written as

$$H_0 : \mathbf{h}(\theta) = 0$$

where $\mathbf{h}(\theta)$ is an r-by-1 vector-valued function of the parameters θ given by the r expressions specified in the TEST statement.

Let $\hat{V}$ be the estimate of the covariance matrix of $\hat{\theta}$. Let $\hat{\theta}$ be the unconstrained estimate of θ and $\tilde{\theta}$ be the constrained estimate of θ such that $h(\tilde{\theta}) = 0$. Let

$$A(\theta) = \partial h(\theta)/\partial\theta \,|_{\hat{\theta}}$$

Using this notation, the test statistics for the three kinds of tests are computed as follows:

- The Wald test statistic is defined as

$$W = h'(\hat{\theta}) \left(A(\hat{\theta}) \hat{V} A'(\hat{\theta}) \right)^{-1} h(\hat{\theta})$$

  The Wald test is not invariant to reparameterization of the model (Gregory and Veall 1985; Gallant 1987, p. 219). For more information about the theoretical properties of the Wald test, see Phillips and Park (1988).

- The Lagrange multiplier test statistic is

$$LM = \lambda' A(\tilde{\theta}) \tilde{V} A'(\tilde{\theta}) \lambda$$

  where λ is the vector of Lagrange multipliers from the computation of the restricted estimate $\tilde{\theta}$.

- The likelihood ratio test statistic is

$$LR = 2 \left(L(\hat{\theta}) - L(\tilde{\theta}) \right)$$

  where $\tilde{\theta}$ represents the constrained estimate of θ and L is the concentrated log-likelihood value.

For each kind of test, under the null hypothesis the test statistic is asymptotically distributed as a χ^2 random variable with r degrees of freedom, where r is the number of expressions in the TEST statement. The p-values reported for the tests are computed from the $\chi^2(r)$ distribution and are only asymptotically valid.

Monte Carlo simulations suggest that the asymptotic distribution of the Wald test is a poorer approximation to its small sample distribution than that of the other two tests. However, the Wald test has the lowest computational cost, since it does not require computation of the constrained estimate $\tilde{\theta}$.

The following statements are an example of using the TEST statement to perform a likelihood ratio test:

```
proc mdc;
   model decision = x1 x2 / type=clogit
         choice=(mode 1 2 3);
   id pid;
   test 0.5 * x1 + 2 * x2 = 0 / lr;
run;
```

OUTEST= Data Set

The OUTEST= data set contains all the parameters that are estimated in a MODEL statement. The OUTEST= option can be used when the PROC MDC call contains one MODEL statement. There are additional restrictions. For the HEV and multinomial probit models, you need to specify exactly all possible elements of the choice set, since additional parameters (for example, SCALE1 or STD1) are generated automatically in the MDC procedure. Therefore, the following SAS statements are not valid when the OUTEST= option is specified:

```
proc mdc data=a outest=e;
   model y = x / type=hev choice=(alter);
run;
```

You need to specify all possible choices in the CHOICE= option since the OUTEST= option is specified as follows:

```
proc mdc data=a outest=e;
   model y = x / type=hev choice=(alter 1 2 3);
run;
```

When the NCHOICE= option is specified, no additional information about possible choices is required. Therefore, the following SAS statements are correct:

```
proc mdc data=a outest=e;
   model y = x / type=mprobit nchoice=3;
run;
```

The nested logit model does not produce the OUTEST= data set unless the NEST statement is specified.

Each parameter contains the estimate for the corresponding parameter in the corresponding model. In addition, the OUTEST= data set contains the following variables:

DEPVAR	the name of the dependent variable
METHOD	the estimation method
MODEL	the label of the MODEL statement if one is specified, or blank otherwise
STATUS	a character variable that indicates whether the optimization process reached convergence or failed to converge: 0 indicates that the convergence was reached, 1 indicates that the maximum number of iterations allowed was exceeded, 2 indicates a failure to improve the function value, and 3 indicates a failure to converge because the objective function or its derivatives could not be evaluated or improved, or linear constraints were dependent, or the algorithm failed to return to feasible region, or the number of iterations was greater than prespecified.
NAME	the name of the row of the covariance matrix for the parameter estimate, if the COVOUT option is specified, or blank otherwise
LIKLHD	the log-likelihood value
STDERR	standard error of the parameter estimate, if the COVOUT option is specified
TYPE	PARMS for observations that contain parameter estimates, or COV for observations that contain covariance matrix elements

The OUTEST= data set contains one observation for the MODEL statement giving the parameter estimates for that model. If the COVOUT option is specified, the OUTEST= data set includes additional observations for the MODEL statement giving the rows of the covariance matrix of parameter estimates. For covariance observations, the value of the _TYPE_ variable is COV, and the _NAME_ variable identifies the parameter associated with that row of the covariance matrix.

ODS Table Names

PROC MDC assigns a name to each table it creates. You can use these names to denote the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in the Table 25.3.

Table 25.3 ODS Tables Produced in PROC MDC

ODS Table Name	Description	Option
ODS Tables Created by the MODEL Statement		
ResponseProfile	Response profile	Default
ClassLevels	Class levels	Default
FitSummary	Summary of nonlinear estimation	Default
GoodnessOfFit	Pseudo-R-square measures	Default
ConvergenceStatus	Convergence status	Default
ParameterEstimates	Parameter estimates	Default
CovB	Covariance of parameter estimates	COVB
CorrB	Correlation of parameter estimates	CORRB
LinCon	Linear constraints	ITPRINT
InputOptions	Input options	ITPRINT
ProblemDescription	Problem description	ITPRINT
IterStart	Optimization start	ITPRINT
IterHist	Iteration history	ITPRINT
IterStop	Optimization results	ITPRINT
ConvergenceStatus	Convergence status	ITPRINT
ParameterEstimatesResults	Resulting parameters	ITPRINT
LinConSol	Linear constraints evaluated at solution	ITPRINT
ODS Tables Created by the TEST Statement		
TestResults	Test results	Default

Examples: MDC Procedure

Example 25.1: Binary Data Modeling

The MDC procedure supports various multinomial choice models. However, you can also use PROC MDC to estimate binary choice models such as binary logit and probit because these models are special cases of multinomial models.

Spector and Mazzeo (1980) studied the effectiveness of a new teaching method on students' performance in an economics course. They reported grade point average (gpa), previous knowledge of the material (tuce), a dummy variable for the new teaching method (psi), and the final course grade (grade). A value of 1 is recorded for grade if a student earned the letter grade "A," and 0 otherwise.

The binary logit can be estimated using the conditional logit model. In order to use the MDC procedure, the data are converted as follows so that each possible choice corresponds to one observation:

```
data smdata;
   input gpa tuce psi grade;
datalines;
2.66      20      0        0
2.89      22      0        0
3.28      24      0        0
2.92      12      0        0

   ... more lines ...


data smdata1;
   set smdata;
   retain id 0;
   id + 1;

   /*-- first choice --*/
   choice1 = 1;
   choice2 = 0;
   decision = (grade = 0);
   gpa_2 = 0;
   tuce_2 = 0;
   psi_2 = 0;
   output;

   /*-- second choice --*/
   choice1 = 0;
   choice2 = 1;
   decision = (grade = 1);
   gpa_2 = gpa;
   tuce_2 = tuce;
   psi_2 = psi;
   output;
run;
```

The first 10 observations are displayed in Output 25.1.1. The variables related to grade=0 are omitted since these are not used for binary choice model estimation.

Output 25.1.1 Converted Binary Data

id	decision	choice2	gpa_2	tuce_2	psi_2
1	1	0	0.00	0	0
1	0	1	2.66	20	0
2	1	0	0.00	0	0
2	0	1	2.89	22	0
3	1	0	0.00	0	0
3	0	1	3.28	24	0
4	1	0	0.00	0	0
4	0	1	2.92	12	0
5	0	0	0.00	0	0
5	1	1	4.00	21	0

Consider the choice probability of the conditional logit model for binary choice:

$$P_i(j) = \frac{\exp(\mathbf{x}_{ij}'\boldsymbol{\beta})}{\sum_{k=1}^{2}\exp(\mathbf{x}_{ik}'\boldsymbol{\beta})}, \quad j = 1, 2$$

The choice probability of the binary logit model is computed based on normalization. The preceding conditional logit model can be converted as

$$P_i(1) = \frac{1}{1 + \exp((\mathbf{x}_{i2} - \mathbf{x}_{i1})'\boldsymbol{\beta})}$$

$$P_i(2) = \frac{\exp((\mathbf{x}_{i2} - \mathbf{x}_{i1})'\boldsymbol{\beta})}{1 + \exp((\mathbf{x}_{i2} - \mathbf{x}_{i1})'\boldsymbol{\beta})}$$

Therefore, you can interpret the binary choice data as the difference between the first and second choice characteristics. In the following statements, it is assumed that $\mathbf{x}_{i1} = \mathbf{0}$. The binary logit model is estimated and displayed in Output 25.1.2.

```
/*-- Conditional Logit --*/
proc mdc data=smdata1;
   model decision = choice2 gpa_2 tuce_2 psi_2 /
            type=clogit
            nchoice=2
            covest=hess;
   id id;
run;
```

Output 25.1.2 Binary Logit Estimates

The MDC Procedure

Conditional Logit Estimates

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
choice2	1	-13.0213	4.9313	-2.64	0.0083
gpa_2	1	2.8261	1.2629	2.24	0.0252
tuce_2	1	0.0952	0.1416	0.67	0.5014
psi_2	1	2.3787	1.0646	2.23	0.0255

Consider the choice probability of the multinomial probit model:

$$P_i(j) = P[\epsilon_{i1} - \epsilon_{ij} < (\mathbf{x}_{ij} - \mathbf{x}_{i1})'\boldsymbol{\beta}, \ldots, \epsilon_{iJ} - \epsilon_{ij} < (\mathbf{x}_{ij} - \mathbf{x}_{iJ})'\boldsymbol{\beta}]$$

The probabilities of choice of the two alternatives can be written as

$$P_i(1) = P[\epsilon_{i2} - \epsilon_{i1} < (\mathbf{x}_{i1} - \mathbf{x}_{i2})'\boldsymbol{\beta}]$$

$$P_i(2) = P[\epsilon_{i1} - \epsilon_{i2} < (\mathbf{x}_{i2} - \mathbf{x}_{i1})'\boldsymbol{\beta}]$$

where $\begin{bmatrix} \epsilon_{i1} \\ \epsilon_{i2} \end{bmatrix} \sim N\left(\mathbf{0}, \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{bmatrix}\right)$. Assume that $\mathbf{x}_{i1} = 0$ and $\sigma_{12} = 0$. The binary probit model is estimated and displayed in Output 25.1.3. You do not get the same estimates as that of the usual binary probit model. The probabilities of choice in the binary probit model are

$$P_i(2) = P[\epsilon_i < \mathbf{x}_i'\boldsymbol{\beta}]$$

$$P_i(1) = 1 - P[\epsilon_i < \mathbf{x}_i'\boldsymbol{\beta}]$$

where $\epsilon_i \sim N(0, 1)$. However, the multinomial probit model has the error variance $\text{Var}(\epsilon_{i2} - \epsilon_{i1}) = \sigma_1^2 + \sigma_2^2$ if ϵ_{i1} and ϵ_{i2} are independent ($\sigma_{12} = 0$). In the following statements, unit variance restrictions are imposed on choices 1 and 2 ($\sigma_1^2 = \sigma_2^2 = 1$). Therefore, the usual binary probit estimates (and standard errors) can be obtained by multiplying the multinomial probit estimates (and standard errors) in Output 25.1.3 by $1/\sqrt{2}$.

```
/*-- Multinomial Probit --*/
proc mdc data=smdata1;
   model decision = choice2 gpa_2 tuce_2 psi_2 /
            type=mprobit
            nchoice=2
            covest=hess
            unitvariance=(1 2);
   id id;
run;
```

Output 25.1.3 Binary Probit Estimates

The MDC Procedure

Multinomial Probit Estimates

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
choice2	1	-10.5392	3.5956	-2.93	0.0034
gpa_2	1	2.2992	0.9813	2.34	0.0191
tuce_2	1	0.0732	0.1186	0.62	0.5375
psi_2	1	2.0171	0.8415	2.40	0.0165

Example 25.2: Conditional Logit and Data Conversion

In this example, data are prepared for use by the MDCDATA statement. Sometimes, choice-specific information is stored in multiple variables. Since the MDC procedure requires multiple observations for each decision maker, you need to arrange the data so that there is an observation for each subject-alternative (individual-choice) combination. Simple binary choice data are obtained from Ben-Akiva and Lerman (1985). The following statements create the SAS data set:

```
data travel;
   length mode $ 8;
   input auto transit mode $;
datalines;
52.9   4.4 Transit
4.1   28.5 Transit
4.1   86.9 Auto
56.2  31.6 Transit
51.8  20.2 Transit
0.2   91.2 Auto
27.6  79.7 Auto
89.9  2.2  Transit
41.5  24.5 Transit
95.0  43.5 Transit
99.1  8.4  Transit

   ... more lines ...
```

The travel time is stored in two variables, auto and transit. In addition, the chosen alternatives are stored in a character variable, mode. The choice variable, mode, is converted to a numeric variable, decision, since the MDC procedure supports only numeric variables. The following statements convert the original data set, travel, and estimate the binary logit model. The first 10 observations of a relevant subset of the new data set and the parameter estimates are displayed in Output 25.2.1 and Output 25.2.2, respectively.

```
data new;
   set travel;
   retain id 0;
   id+1;
   /*-- create auto variable --*/
```

```
   decision = (upcase(mode) = 'AUTO');
   ttime = auto;
   autodum = 1;
   trandum = 0;
   output;
   /*-- create transit variable --*/
   decision = (upcase(mode) = 'TRANSIT');
   ttime = transit;
   autodum = 0;
   trandum = 1;
   output;
run;

proc print data=new(obs=10);
   var decision autodum trandum ttime;
   id id;
run;
```

Output 25.2.1 Converted Data

id	decision	autodum	trandum	ttime
1	0	1	0	52.9
1	1	0	1	4.4
2	0	1	0	4.1
2	1	0	1	28.5
3	1	1	0	4.1
3	0	0	1	86.9
4	0	1	0	56.2
4	1	0	1	31.6
5	0	1	0	51.8
5	1	0	1	20.2

The following statements perform the binary logit estimation:

```
proc mdc data=new;
   model decision = autodum ttime /
            type=clogit
            nchoice=2;
   id id;
run;
```

Output 25.2.2 Binary Logit Estimation of Modal Choice Data

The MDC Procedure

Conditional Logit Estimates

Parameter Estimates					
Parameter	**DF**	**Estimate**	**Standard Error**	**t Value**	**Approx Pr > |t|**
autodum	1	-0.2376	0.7505	-0.32	0.7516
ttime	1	-0.0531	0.0206	-2.57	0.0101

In order to handle more general cases, you can use the MDCDATA statement. Choice-specific dummy variables are generated and multiple observations for each individual are created. The following example converts the original data set travel by using the MDCDATA statement and performs conditional logit analysis. Interleaved data are output into the new data set new3. This data set has twice as many observations as the original travel data set.

```
proc mdc data=travel;
   mdcdata varlist( x1 = (auto transit) )
           select=mode
           id=id
           alt=alternative
           decvar=Decision / out=new3;
   model decision = auto x1 /
            nchoice=2
            type=clogit;
   id id;
run;
```

The first nine observations of the modified data set are shown in Output 25.2.3. The result of the preceding program is listed in Output 25.2.4.

Output 25.2.3 Transformed Model Choice Data

Obs	MODE	AUTO	TRANSIT	X1	ID	ALTERNATIVE	DECISION
1	TRANSIT	1	0	52.9	1	1	0
2	TRANSIT	0	1	4.4	1	2	1
3	TRANSIT	1	0	4.1	2	1	0
4	TRANSIT	0	1	28.5	2	2	1
5	AUTO	1	0	4.1	3	1	1
6	AUTO	0	1	86.9	3	2	0
7	TRANSIT	1	0	56.2	4	1	0
8	TRANSIT	0	1	31.6	4	2	1
9	TRANSIT	1	0	51.8	5	1	0

Output 25.2.4 Results Using MDCDATA Statement

The MDC Procedure

Conditional Logit Estimates

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
AUTO	1	-0.2376	0.7505	-0.32	0.7516
X1	1	-0.0531	0.0206	-2.57	0.0101

Example 25.3: Correlated Choice Modeling

Often, it is not realistic to assume that the random components of utility for all choices are independent. This example shows the solution to the problem of correlated random components by using multinomial probit and nested logit.

To analyze correlated data, trinomial choice data (1,000 observations) are created using a pseudo-random number generator by using the following statements. The random utility function is

$$U_{ij} = V_{ij} + \epsilon_{ij}, \quad j = 1, 2, 3$$

where

$$\epsilon_{ij} \sim N\left(0, \begin{bmatrix} 2 & .6 & 0 \\ .6 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right)$$

```
/*-- generate simulated series --*/
%let ndim = 3;
%let nobs = 1000;

data trichoice;
   array error{&ndim} e1-e3;
   array vtemp{&ndim} _temporary_;
   array lm{6} _temporary_ (1.4142136 0.4242641 1 0 0 1);
   retain nseed 345678;

   do id = 1 to &nobs;
      index = 0;
      /* generate independent normal variate */
      do i = 1 to &ndim;
         /* index of diagonal element */
         vtemp{i} = rannor(nseed);
      end;
      /* get multivariate normal variate */
      index = 0;
      do i = 1 to &ndim;
         error{i} = 0;
         do j = 1 to i;
            error{i} = error{i} + lm{index+j}*vtemp{j};
         end;
         index = index + i;
      end;
      x1 = 1.0 + 2.0 * ranuni(nseed);
      x2 = 1.2 + 2.0 * ranuni(nseed);
      x3 = 1.5 + 1.2 * ranuni(nseed);
      util1 = 2.0 * x1 + e1;
      util2 = 2.0 * x2 + e2;
      util3 = 2.0 * x3 + e3;
      do i = 1 to &ndim;
         vtemp{i} = 0;
      end;
      if ( util1 > util2 & util1 > util3 ) then
         vtemp{1} = 1;
      else if ( util2 > util1 & util2 > util3 ) then
         vtemp{2} = 1;
      else if ( util3 > util1 & util3 > util2 ) then
         vtemp{3} = 1;
      else continue;
      /*-- first choice --*/
```

```
      x = x1;
      mode = 1;
      decision = vtemp{1};
      output;
      /*-- second choice --*/
      x = x2;
      mode = 2;
      decision = vtemp{2};
      output;
      /*-- third choice --*/
      x = x3;
      mode = 3;
      decision = vtemp{3};
      output;
   end;
run;
```

First, the multinomial probit model is estimated (see the following statements). Results show that the standard deviation, correlation, and slope estimates are close to the parameter values. Note that $\rho_{12} = \frac{\sigma_{12}}{\sqrt{(\sigma_1^2)(\sigma_2^2)}} = \frac{0.6}{\sqrt{(2)(1)}} = 0.42$, $\sigma_1 = \sqrt{2} = 1.41$, $\sigma_2 = \sqrt{1} = 1$, and the parameter value for the variable x is 2.0. (See Output 25.3.1.)

```
/*-- Trinomial Probit --*/
proc mdc data=trichoice randnum=halton nsimul=100;
   model decision = x /
            type=mprobit
            choice=(mode 1 2 3)
            covest=op
            optmethod=qn;
   id id;
run;
```

Output 25.3.1 Trinomial Probit Model Estimation

The MDC Procedure

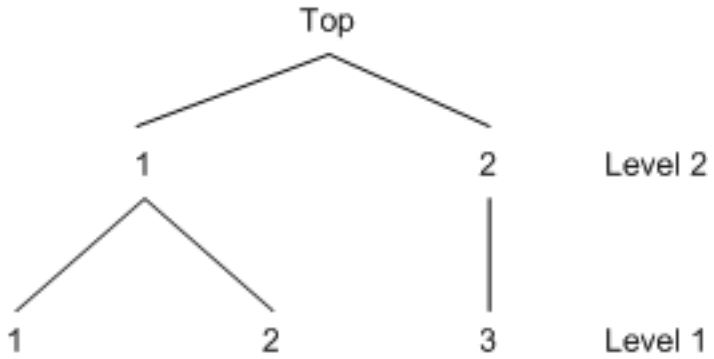
Multinomial Probit Estimates

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
x	1	1.7685	0.1191	14.85	<.0001
STD_1	1	1.2514	0.1494	8.38	<.0001
RHO_21	1	0.3971	0.1087	3.65	0.0003

Figure 25.29 shows a two-level decision tree.

Figure 25.29 Nested Tree Structure



The following statements estimate the nested model shown in Figure 25.29:

```
/*-- Two-Level Nested Logit --*/
proc mdc data=trichoice;
   model decision = x /
            type=nlogit
            choice=(mode 1 2 3)
            covest=op
            optmethod=qn;
   id id;
   utility u(1,) = x;
   nest level(1) = (1 2 @ 1, 3 @ 2),
        level(2) = (1 2 @ 1);
run;
```

The estimated result (see Output 25.3.2) shows that the data support the nested tree model since the estimates of the inclusive value parameters are significant and are less than 1.

Output 25.3.2 Two-Level Nested Logit

The MDC Procedure

Nested Logit Estimates

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
x_L1	1	2.5907	0.1958	13.23	<.0001
INC_L2G1C1	1	0.8103	0.0859	9.43	<.0001
INC_L2G1C2	1	0.8189	0.0955	8.57	<.0001

Example 25.4: Testing for Homoscedasticity of the Utility Function

The conditional logit model imposes equal variances on random components of utility of all alternatives. This assumption can often be too restrictive and the calculated results misleading. This example shows several approaches to testing the homoscedasticity assumption.

The section "Getting Started: MDC Procedure" on page 1395 analyzes an HEV model by using Daganzo's trinomial choice data and displays the HEV parameter estimates in Figure 25.15. The inverted scale estimates for mode "2" and mode "3" suggest that the conditional logit model (which imposes equal variances on random components of utility of all alternatives) might be misleading. The HEV estimation summary from that analysis is repeated in Output 25.4.1.

Output 25.4.1 HEV Estimation Summary ($\theta_1 = 1$)

Model Fit Summary	
Dependent Variable	decision
Number of Observations	50
Number of Cases	150
Log Likelihood	-33.41383
Maximum Absolute Gradient	0.0000218
Number of Iterations	11
Optimization Method	Dual Quasi-Newton
AIC	72.82765
Schwarz Criterion	78.56372

You can estimate the HEV model with unit scale restrictions on all three alternatives ($\theta_1 = \theta_2 = \theta_3 = 1$) with the following statements.

```
/*-- HEV Estimation --*/
proc mdc data=newdata;
   model decision = ttime /
            type=hev
            nchoice=3
            hev=(unitscale=1 2 3, integrate=laguerre)
            covest=hess;
   id pid;
run;
```

Output 25.4.2 displays the estimation summary.

Output 25.4.2 HEV Estimation Summary ($\theta_1 = \theta_2 = \theta_3 = 1$)

The MDC Procedure

Heteroscedastic Extreme Value Model Estimates

Model Fit Summary	
Dependent Variable	decision
Number of Observations	50
Number of Cases	150
Log Likelihood	-34.12756
Maximum Absolute Gradient	6.7951E-9
Number of Iterations	5
Optimization Method	Dual Quasi-Newton
AIC	70.25512
Schwarz Criterion	72.16714

The test for scale equivalence (SCALE2=SCALE3=1) is performed using a likelihood ratio test statistic. The following SAS statements compute the test statistic (1.4276) and its p-value (0.4898) from the log-likelihood values in Output 25.4.1 and Output 25.4.2:

```
data _null_;
   /*-- test for H0: scale2 = scale3 = 1 --*/
   /*  ln L(max) = -34.1276                */
   /*  ln L(0)   = -33.4138                */
   stat = -2 * ( - 34.1276 + 33.4138 );
   df = 2;
   p_value = 1 - probchi(stat, df);
   put stat= p_value=;
run;
```

The test statistic fails to reject the null hypothesis of equal scale parameters, which implies that the random utility function is homoscedastic.

A multinomial probit model also allows heteroscedasticity of the random components of utility for different alternatives. Consider the utility function

$$U_{ij} = V_{ij} + \epsilon_{ij}$$

where

$$\boldsymbol{\epsilon}_i \sim N\left(\mathbf{0}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \sigma_3^2 \end{bmatrix}\right)$$

This multinomial probit model is estimated by using the following statements:

```
/*-- Heteroscedastic Multinomial Probit --*/
proc mdc data=newdata;
   model decision = ttime /
            type=mprobit
            nchoice=3
            unitvariance=(1 2)
            covest=hess;
   id pid;
   restrict RHO_31 = 0;
run;
```

The estimation summary is displayed in Output 25.4.3.

Output 25.4.3 Heteroscedastic Multinomial Probit Estimation Summary

The MDC Procedure

Multinomial Probit Estimates

Model Fit Summary	
Dependent Variable	decision
Number of Observations	50
Number of Cases	150
Log Likelihood	-33.88604
Log Likelihood Null (LogL(0))	-54.93061
Maximum Absolute Gradient	5.60277E-6
Number of Iterations	8
Optimization Method	Dual Quasi-Newton
AIC	71.77209
Schwarz Criterion	75.59613
Number of Simulations	100
Starting Point of Halton Sequence	11

Next, the multinomial probit model with unit variances ($\sigma_1 = \sigma_2 = \sigma_3 = 1$) is estimated in the following statements:

```
/*-- Homoscedastic Multinomial Probit --*/
proc mdc data=newdata;
   model decision = ttime /
             type=mprobit
             nchoice=3
             unitvariance=(1 2 3)
             covest=hess;
   id pid;
   restrict RHO_21 = 0;
run;
```

The estimation summary is displayed in Output 25.4.4.

Output 25.4.4 Homoscedastic Multinomial Probit Estimation Summary

The MDC Procedure

Multinomial Probit Estimates

Model Fit Summary	
Dependent Variable	decision
Number of Observations	50
Number of Cases	150
Log Likelihood	-34.54252
Log Likelihood Null (LogL(0))	-54.93061
Maximum Absolute Gradient	1.37303E-7
Number of Iterations	5
Optimization Method	Dual Quasi-Newton
AIC	71.08505
Schwarz Criterion	72.99707
Number of Simulations	100
Starting Point of Halton Sequence	11

The test for homoscedasticity ($\sigma_3 = 1$) under $\sigma_1 = \sigma_2 = 1$ shows that the error variance is not heteroscedastic since the test statistic (1.313) is less than $\chi^2_{0.05,1} = 3.84$. The marginal probability or p-value computed in the following statements from the PROBCHI function is 0.2519:

```
data _null_;
   /*-- test for H0: sigma3 = 1 --*/
   /*   ln L(max) = -33.8860      */
   /*   ln L(0)   = -34.5425      */
   stat = -2 * ( -34.5425 + 33.8860 );
   df = 1;
   p_value = 1 - probchi(stat, df);
   put stat= p_value=;
run;
```

Example 25.5: Choice of Time for Work Trips: Nested Logit Analysis

This example uses sample data of 527 automobile commuters in the San Francisco Bay Area to demonstrate the use of nested logit model.

Brownstone and Small (1989) analyzed a two-level nested logit model that is displayed in Figure 25.30. The probability of choosing j at level 2 is written as

$$P_i(j) = \frac{\exp(\tau_j I_j)}{\sum_{j'=1}^{3} \exp(\tau_{j'} I_{j'})}$$

where $I_{j'}$ is an inclusive value and is computed as

$$I_{j'} = \ln \left[\sum_{k' \in C_{j'}} \exp(\mathbf{x}'_{ik'} \boldsymbol{\beta}) \right]$$

The probability of choosing an alternative k is denoted as

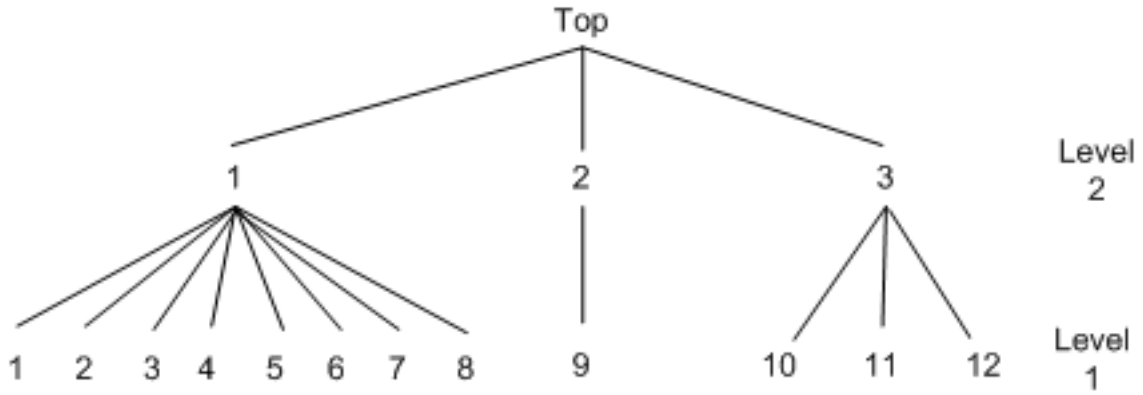
$$P_i(k|j) = \frac{\exp(\mathbf{x}'_{ik} \boldsymbol{\beta})}{\sum_{k' \in C_j} \exp(\mathbf{x}'_{ik'} \boldsymbol{\beta})}$$

The full information maximum likelihood (FIML) method maximizes the following log-likelihood function:

$$\mathcal{L} = \sum_{i=1}^{N} \sum_{j=1}^{J} d_{ij} \left[\ln(P_i(k|j)) + \ln(P_i(j)) \right]$$

where $d_{ij} = 1$ if a decision maker i chooses j, and 0 otherwise.

Figure 25.30 Decision Tree for Two-Level Nested Logit



Sample data of 527 automobile commuters in the San Francisco Bay Area have been analyzed by Small (1982); Brownstone and Small (1989). The regular time of arrival is recorded as between 42.5 minutes early and 17.5 minutes late, and indexed by 12 alternatives, using five-minute interval groups. Refer to Small (1982) for more details on these data. The following statements estimate the two-level nested logit model:

```
/*-- Two-level Nested Logit --*/
proc mdc data=small maxit=200 outest=a;
   model decision = r15 r10 ttime ttime_cp sde sde_cp
                    sdl sdlx d2l /
            type=nlogit
            choice=(alt);
   id id;
   utility u(1, ) = r15 r10 ttime ttime_cp sde sde_cp
                    sdl sdlx d2l;
   nest level(1) = (1 2 3 4 5 6 7 8 @ 1, 9 @ 2, 10 11 12 @ 3),
        level(2) = (1 2 3 @ 1);
run;
```

The following statements add the upalt variable, which describes the choice at the upper level of the nested tree to the data set.

```
data small;
   set small;
   upalt=1;
   if alt=9 then upalt=2;
   if alt>9 then upalt=3;
run;
```

The following statements show and alternative specification, which uses the CHOICE= option with two nested levels that are represented by upalt and alt:

```
proc mdc data=upalt maxit=200;
   model decision = r15 r10 ttime ttime_cp sde sde_cp
                    sdl sdlx d2l /
            type=nlogit
            choice=(upalt,alt);
```

```
   id id;
   utility u(1, ) = r15 r10 ttime ttime_cp sde sde_cp
                    sdl sdlx d2l;
run;
```

The estimation summary, discrete response profile, and the FIML estimates are displayed in Output 25.5.1 through Output 25.5.3.

Output 25.5.1 Nested Logit Estimation Summary

The MDC Procedure

Nested Logit Estimates

Model Fit Summary	
Dependent Variable	decision
Number of Observations	527
Number of Cases	6324
Log Likelihood	-990.81912
Log Likelihood Null (LogL(0))	-1310
Maximum Absolute Gradient	4.93868E-6
Number of Iterations	18
Optimization Method	Newton-Raphson
AIC	2006
Schwarz Criterion	2057

Output 25.5.2 Discrete Choice Characteristics

Discrete Response Profile			
Index	**alt**	**Frequency**	**Percent**
0	1	6	1.14
1	2	10	1.90
2	3	61	11.57
3	4	15	2.85
4	5	27	5.12
5	6	80	15.18
6	7	55	10.44
7	8	64	12.14
8	9	187	35.48
9	10	13	2.47
10	11	8	1.52
11	12	1	0.19

Output 25.5.3 Nested Logit Estimates

The MDC Procedure

Nested Logit Estimates

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
r15_L1	1	1.1034	0.1221	9.04	<.0001
r10_L1	1	0.3931	0.1194	3.29	0.0010
ttime_L1	1	-0.0465	0.0235	-1.98	0.0474
ttime_cp_L1	1	-0.0498	0.0305	-1.63	0.1028
sde_L1	1	-0.6618	0.0833	-7.95	<.0001
sde_cp_L1	1	0.0519	0.1278	0.41	0.6850
sdl_L1	1	-2.1006	0.5062	-4.15	<.0001
sdlx_L1	1	-3.5240	1.5346	-2.30	0.0217
d2l_L1	1	-1.0941	0.3273	-3.34	0.0008
INC_L2G1C1	1	0.6762	0.2754	2.46	0.0141
INC_L2G1C2	1	1.0906	0.3090	3.53	0.0004
INC_L2G1C3	1	0.7622	0.1649	4.62	<.0001

Now policy makers are particularly interested in predicting shares of each alternative to be chosen by population. One application of such predictions are market shares. Going even further, it is extremely useful to predict choice probabilities out of sample; that is, under alternative policies.

Suppose that in this particular transportation example you are interested in projecting the effect of a new program that indirectly shifts individual preferences with respect to late arrival to work. This means that you manage to decrease the coefficient for the "late dummy" D2L, which is a penalty for violating some margin of arriving on time. Suppose that you alter it from an estimated -1.0941 to almost twice that level, -2.0941.

But first, in order to have a benchmark share, you predict probabilities to choose each particular option and output them to the new data set with the following additional statement:

```
/*-- Create new data set with predicted probabilities --*/
output out=predicted1 p=probs;
```

Having these in sample predictions, you sort the data by alternative and aggregate across each of them as shown in the following statements:

```
/*-- Sort the data by alternative --*/
proc sort data=predicted1;
   by alt;
run;

/*-- Calculate average probabilities of each alternative --*/
proc means data=predicted1 nonobs mean;
   var probs;
   class alt;
run;
```

Output 25.5.4 shows the summary table that is produced by the preceding statements.

Output 25.5.4 Average Probabilities of Choosing Each Particular Alternative

The MEANS Procedure

Analysis Variable : probs	
alt	Mean
1	0.0178197
2	0.0161712
3	0.0972584
4	0.0294659
5	0.0594076
6	0.1653871
7	0.1118181
8	0.1043445
9	0.3564940
10	0.0272324
11	0.0096334
12	0.0049677

Now you change the preference parameter for variable D2L. In order to fix all the parameters, you use the MAXIT=0 option to prevent optimization and the START= option in MODEL statement to specify initial parameters.

```
/*-- Two-level Nested Logit --*/
proc mdc data=small maxit=0 outest=a;
   model decision = r15 r10 ttime ttime_cp sde sde_cp
                    sdl sdlx d2l /
            type=nlogit
            choice=(alt)
            start=( 1.1034 0.3931 -0.0465 -0.0498
                   -0.6618 0.0519 -2.1006 -3.5240
                   -2.0941 0.6762  1.0906  0.7622);
   id id;
   utility u(1, ) = r15 r10 ttime ttime_cp sde sde_cp
                    sdl sdlx d2l;
   nest level(1) = (1 2 3 4 5 6 7 8 @ 1, 9 @ 2, 10 11 12 @ 3),
        level(2) = (1 2 3 @ 1);
   output out=predicted2 p=probs;
run;
```

You apply the same SORT and MEANS procedures as applied earlier to obtain the following summary table in Output 25.5.5.

Output 25.5.5 Average Probabilities of Choosing Each Particular Alternative after Changing the Preference Parameter

The MEANS Procedure

Analysis Variable : probs	
alt	Mean
1	0.0207766
2	0.0188966
3	0.1138816
4	0.0345654
5	0.0697830
6	0.1944572
7	0.1315588
8	0.1228049
9	0.2560674
10	0.0236178
11	0.0090781
12	0.0045128

Comparing the two tables shown in Output 25.5.4 and Output 25.5.5, you clearly see the effect of increased dislike of late arrival. People shifted their choices towards earlier times (alternatives 1–8) from the on-time option (alternative 9).

Brownstone and Small (1989) also estimate the two-level nested logit model with equal scale parameter constraints, $\tau_1 = \tau_2 = \tau_3$. Replication of their model estimation is shown in the following statements:

```
/*-- Nested Logit with Equal Dissimilarity Parameters --*/
proc mdc data=small maxit=200 outest=a;
   model decision = r15 r10 ttime ttime_cp sde sde_cp
                    sdl sdlx d2l /
            samescale
            type=nlogit
            choice=(alt);
   id id;
   utility u(1, ) = r15 r10 ttime ttime_cp sde sde_cp
                    sdl sdlx d2l;
   nest level(1) = (1 2 3 4 5 6 7 8 @ 1, 9 @ 2, 10 11 12 @ 3),
        level(2) = (1 2 3 @ 1);
run;
```

The parameter estimates and standard errors are almost identical to those in Brownstone and Small (1989, p. 69). Output 25.5.6 and Output 25.5.7 display the results.

Output 25.5.6 Nested Logit Estimation Summary with Equal Dissimilarity Parameters

The MDC Procedure

Nested Logit Estimates

Model Fit Summary	
Dependent Variable	decision
Number of Observations	527
Number of Cases	6324
Log Likelihood	-994.39402
Log Likelihood Null (LogL(0))	-1310
Maximum Absolute Gradient	2.97172E-6
Number of Iterations	16
Optimization Method	Newton-Raphson
AIC	2009
Schwarz Criterion	2051

Output 25.5.7 Nested Logit Estimates with Equal Dissimilarity Parameters

The MDC Procedure

Nested Logit Estimates

Parameter Estimates					
Parameter	**DF**	**Estimate**	**Standard Error**	**t Value**	**Approx Pr > |t|**
r15_L1	1	1.1345	0.1092	10.39	<.0001
r10_L1	1	0.4194	0.1081	3.88	0.0001
ttime_L1	1	-0.1626	0.0609	-2.67	0.0076
ttime_cp_L1	1	0.1285	0.0853	1.51	0.1319
sde_L1	1	-0.7548	0.0669	-11.28	<.0001
sde_cp_L1	1	0.2292	0.0981	2.34	0.0195
sdl_L1	1	-2.0719	0.4860	-4.26	<.0001
sdlx_L1	1	-2.8216	1.2560	-2.25	0.0247
d2l_L1	1	-1.3164	0.3474	-3.79	0.0002
INC_L2G1	1	0.8059	0.1705	4.73	<.0001

However, the test statistic for $H_0 : \tau_1 = \tau_2 = \tau_3$ rejects the null hypothesis at the 5% significance level since $-2 * (\ln L(0) - \ln L) = 7.15 > \chi^2_{0.05,2} = 5.99$. The p-value is computed in the following statements and is equal to 0.0280:

```
data _null_;
   /*-- test for H0: tau1 = tau2 = tau3 --*/
   /*   ln L(max) = -990.8191             */
   /*   ln L(0)   = -994.3940             */
   stat = -2 * ( -994.3940 + 990.8191 );
   df = 2;
   p_value = 1 - probchi(stat, df);
   put stat= p_value=;
run;
```

Example 25.6: Hausman's Specification Test

As discussed under multinomial and conditional logits, the odds ratios in the multinomial or conditional logits are independent of the other alternatives. See the section "Multinomial Logit and Conditional Logit" on page 1428. This property of the logit models is often viewed as rather restrictive and provides substitution patterns that do not represent the actual relationship among choice alternatives.

This independence assumption, called independence of irrelevant alternatives (IIA), can be tested with Hausman's specification test. According to Hausman and McFadden (1984), if a subset of choice alternatives is irrelevant, it can be omitted from the sample without changing the remaining parameters systematically.

Under the null hypothesis (IIA holds), omitting the irrelevant alternatives leads to consistent and efficient parameter estimates β_R, while parameter estimates β_U from the unrestricted model are consistent but inefficient. Under the alternative, only the parameter estimates β_U obtained from the unrestricted model are consistent.

This example demonstrates the use of Hausman's specification test to analyze the IIA assumption and decide on an appropriate model that provides less restrictive substitution patterns (nested logit or multinomial probit). A sample data set of 527 automobile commuters in the San Francisco Bay Area is used (Small 1982). The regular time of arrival is recorded as between 42.5 minutes early and 17.5 minutes late, and is indexed by 12 alternatives, using five-minute interval groups. See Small (1982) for more details about these data.

The data can be divided into three groups: commuters who arrive early (alternatives 1 – 8), commuters who arrive on time (alternative 9), and commuters who arrive late (alternatives 10 – 12). Suppose that you want to test whether the IIA assumption holds for commuters who arrived on time (alternative 9).

Hausman's specification test is distributed as χ^2 with k degrees of freedom (equal to the number of independent variables) and can be written as

$$\chi^2 = (\hat{\beta_U} - \hat{\beta_R})'[\hat{V_U} - \hat{V_R}]^{-1}(\hat{\beta_U} - \hat{\beta_R})$$

where $\hat{\beta_R}$ and $\hat{V_R}$ represent parameter estimates and the variance-covariance matrix, respectively, from the model where the ninth alternative was omitted, and $\hat{\beta_U}$ and $\hat{V_U}$ represent parameter estimates and the variance-covariance matrix, respectively, from the full model. The following macro can be used to perform the IIA test for the ninth alternative:

```
/*-------------------------------------------------------------
 * name: %IIA
 * note: This macro test the IIA hypothesis using the Hausman's
 *       specification test. Inputs into the macro are as follows:
 *       indata:    input data set
 *       varlist:   list of RHS variables
 *       nchoice:   number of choices for each individual
 *       choice:    list of choices
 *       nvar:      number of independent variables
 *       nIIA:      number of choice alternatives used to test IIA
 *       IIA:       choice alternatives used to test IIA
 *       id:        ID variable
 *       decision:  0-1 LHS variable representing nchoice choices
 * purpose: Hausman's specification test
```

```
 *-------------------------------------------------------------*/

%macro IIA(indata=, varlist=, nchoice=, choice= , nvar= , IIA= ,
              nIIA=, id= , decision=);

   %let n=%eval(&nchoice-&nIIA);

   proc mdc data=&indata outest=cov covout ;
      model &decision = &varlist /
               type=clogit
               nchoice=&nchoice;
      id &id;
   run;

   data two;
      set &indata;
      if &choice in &IIA and &decision=1 then output;
   run;

   data two;
      set two;
      keep &id ind;
      ind=1;
   run;

   data merged;
      merge &indata two;
      by &id;
      if ind=1 or &choice in &IIA then delete;
   run;

   proc mdc data=merged outest=cov2 covout ;
      model &decision = &varlist /
               type=clogit
               nchoice=&n;
      id &id;
   run;

   proc IML;
      use cov var{_TYPE_ &varlist};
         read first into BetaU;
         read all into CovVarU where(_TYPE_='COV');
      close cov;

      use cov2 var{_TYPE_ &varlist};
         read first into BetaR;
         read all into CovVarR where(_TYPE_='COV');
      close cov;

      tmp = BetaU-BetaR;
      ChiSq=tmp*ginv(CovVarR-CovVarU)*tmp`;
      if ChiSq<0 then ChiSq=0;
      Prob=1-Probchi(ChiSq, &nvar);
      Print "Hausman Test for IIA for Variable &IIA";
```

```
      Print ChiSq Prob;
   run; quit;

%mend IIA;
```

The following statement invokes the %IIA macro to test IIA for commuters who arrive on time:

```
%IIA( indata=small,
      varlist=r15 r10 ttime ttime_cp sde sde_cp sdl sdlx d2l,
      nchoice=12,
      choice=alt,
      nvar=9,
      nIIA=1,
      IIA=(9),
      id=id,
      decision=decision );
```

The obtained χ^2 of 7.9 and the p-value of 0.54 indicate that IIA holds for commuters who arrive on time (alternative 9). If the IIA assumption did not hold, the following model (nested logit), which reserves a subcategory for alternative 9, might be more appropriate. See Output 25.30.

```
proc mdc data=small maxit=200 outest=a;
   model decision = r15 r10 ttime ttime_cp sde sde_cp
                    sdl sdlx d2l /
            type=nlogit
            choice=(alt);
   id id;
   utility u(1, ) = r15 r10 ttime ttime_cp sde sde_cp
                    sdl sdlx d2l;
   nest level(1) = (1 2 3 4 5 6 7 8 @ 1, 9 @ 2, 10 11 12 @ 3),
        level(2) = (1 2 3 @ 1);
run;
```

Similarly, IIA could be tested for commuters who arrive approximately on time (alternative 8, 9, 10), as follows:

```
%IIA( indata=small,
      varlist=r15 r10 ttime ttime_cp sde sde_cp sdl sdlx d2l,
      nchoice=12,
      choice=alt,
      nvar=9,
      nIIA=3,
      IIA=(8 9 10),
      id=id,
      decision=decision );
```

Based on this test, independence of irrelevant alternatives is not rejected for this subgroup ($\chi^2 = 10.3$ and p-value=0.326), and it is concluded that a more complex nested logit model with commuters who arrive approximately on time in one subcategory is not needed. Since the two Hausman's specification tests just performed did not reject IIA, it might be a good idea to test whether the nested logit model is even needed. This is done using the likelihood ratio test in the next example.

Example 25.7: Likelihood Ratio Test

This example is an extension of Example 25.6; it performs another specification test, the likelihood ratio test (LR). Suppose you are interested in testing whether the nested logit model (Output 25.30) with three subgroups that represent commuters who arrive early, on time, and late is more appropriate than the standard multinomial logit. This can be done by adding the TEST statement to the model as follows:

```
/*-- Restricted Model with Inclusive Value Parameters
     Constrained to One --*/
proc mdc data=small maxit=200 outest=a;
   model decision = r15 r10 ttime ttime_cp sde sde_cp
                    sdl sdlx d2l /
            type=nlogit
            choice=(alt);
   id id;
   utility u(1, ) = r15 r10 ttime ttime_cp sde sde_cp
                    sdl sdlx d2l;
   nest level(1) = (1 2 3 4 5 6 7 8 @ 1, 9 @ 2, 10 11 12 @ 3),
        level(2) = (1 2 3 @ 1);
   test INC_L2G1C1=1, INC_L2G1C2=1, INC_L2G1C3=1 /LR;
run;
```

Output 25.7.1 Likelihood Ratio Test

The MDC Procedure

Nested Logit Estimates

Test Results

Test	Type	Statistic	Pr > ChiSq	Label
Test0	L.R.	8.11	0.0438	INC_L2G1C1 = 1 , INC_L2G1C2 = 1 , INC_L2G1C3 = 1

Based on this test, you can conclude that the inclusive values, INC_L2G1C1, INC_L2G1C2, and INC_L2G1C3 are jointly statistically different from the value 1 at the 5% level and therefore the nested logit is a more appropriate model. The LR test can be used to test other types of restrictions in the nested logit setting as long as one model can be nested within another.

Acknowledgments: MDC Procedure

Professor Kenneth Small provided the work trip data that are used in the "Examples" section. These data were collected for the urban travel demand forecasting project, which was carried out by McFadden, Talvitie, and Associates (1977). The project was supported by the National Science Foundation, Research Applied to National Needs Program through grants GI-43740 and APR74-20392, and the Alfred P. Sloan Foundation, through grant 74-21-8.

References

Abramowitz, M., and Stegun, I. A., eds. (1970). *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.* 9th printing. New York: Dover.

Amemiya, T. (1981). "Qualitative Response Models: A Survey." *Journal of Economic Literature* 19:483–536.

Amemiya, T. (1985). *Advanced Econometrics*. Cambridge, MA: Harvard University Press.

Ben-Akiva, M., and Lerman, S. R. (1985). *Discrete Choice Analysis: Theory and Application to Travel Demand*. Cambridge, MA: MIT Press.

Bhat, C. R. (1995). "A Heteroscedastic Extreme Value Model of Intercity Travel Mode Choice." *Transportation Research* 29:471–483.

Brownstone, D., and Small, K. A. (1989). "Efficient Estimation of Nested Logit Models." *Journal of Business and Economic Statistics* 7:67–74.

Brownstone, D., and Train, K. (1999). "Forecasting New Product Penetration with Flexible Substitution Patterns." *Journal of Econometrics* 89:109–129.

Daganzo, C. (1979). *Multinomial Probit: The Theory and Its Application to Demand Forecasting*. New York: Academic Press.

Daganzo, C., and Kusnic, M. (1993). "Two Properties of the Nested Logit Model." *Transportation Science* 27:395–400.

Estrella, A. (1998). "A New Measure of Fit for Equations with Dichotomous Dependent Variables." *Journal of Business and Economic Statistics* 16:198–205.

Gallant, A. R. (1987). *Nonlinear Statistical Models*. New York: John Wiley & Sons.

Geweke, J. (1989). "Bayesian Inference in Econometric Models Using Monte Carlo Integration." *Econometrica* 57:1317–1339.

Geweke, J., Keane, M. P., and Runkle, D. (1994). "Alternative Computational Approaches to Inference in the Multinomial Probit Model." *Review of Economics and Statistics* 76:609–632.

Godfrey, L. G. (1988). *Misspecification Tests in Econometrics*. Cambridge: Cambridge University Press.

Greene, W. H. (1997). *Econometric Analysis*. 3rd ed. Upper Saddle River, NJ: Prentice-Hall.

Gregory, A. W., and Veall, M. R. (1985). "On Formulating Wald Tests for Nonlinear Restrictions." *Econometrica* 53:1465–1468.

Hajivassiliou, V. A. (1993). "Simulation Estimation Methods for Limited Dependent Variable Models." In *Econometrics*, edited by G. S. Maddala, C. R. Rao, and H. D. Vinod. Vol. 11 of Handbook of Statistics, 519–543. New York: Elsevier Science.

Hajivassiliou, V. A., McFadden, D. L., and Ruud, P. A. (1996). "Simulation of Multivariate Normal Rectangle Probabilities and Their Derivatives: Theoretical and Computational Results." *Journal of Econometrics* 72:85–134.

Hausman, J. A., and McFadden, D. (1984). "Specification Tests for the Multinomial Logit Model." *Econometrica* 52:1219–1240.

Hensher, D. A. (1986). "Sequential and Full Information Maximum Likelihood Estimation of a Nested Logit Model." *Review of Economics and Statistics* 68:657–667.

Hensher, D. A. (1993). "Using Stated Response Choice Data to Enrich Revealed Preference Discrete Choice Models." *Marketing Letters* 4:139–151.

Keane, M. P. (1994). "A Computationally Practical Simulation Estimator for Panel Data." *Econometrica* 62:95–116.

Keane, M. P. (1997). "Current Issues in Discrete Choice Modeling." *Marketing Letters* 8:307–322.

LaMotte, L. R. (1994). "A Note on the Role of Independence in *t* Statistics Constructed from Linear Statistics in Regression Models." *American Statistician* 48:238–240.

Luce, R. D. (1959). *Individual Choice Behavior: A Theoretical Analysis*. New York: John Wiley & Sons.

Maddala, G. S. (1983). *Limited-Dependent and Qualitative Variables in Econometrics*. New York: Cambridge University Press.

McFadden, D. (1974). "Conditional Logit Analysis of Qualitative Choice Behavior." In *Frontiers in Econometrics*, edited by P. Zarembka, 105–142. New York: Academic Press.

McFadden, D. (1978). "Modelling the Choice of Residential Location." In *Spatial Interaction Theory and Planning Models*, edited by A. Karlqvist, L. Lundqvist, F. Snickars, and J. Weibull, 75–96. Amsterdam: North-Holland.

McFadden, D. (1981). "Econometric Models of Probabilistic Choice." In *Structural Analysis of Discrete Data with Econometric Applications*, edited by C. F. Manski, and D. McFadden, 2–50. Cambridge, MA: MIT Press.

McFadden, D., Talvitie, A. P., and Associates (1977). *The Urban Travel Demand Forecasting Project: Phase I Final Report Series.* Vol. 5. Berkeley: Institute of Transportation Studies, University of California, Berkeley.

McFadden, D. L., and Ruud, P. A. (1994). "Estimation by Simulation." *Review of Economics and Statistics* 76:591–608.

Phillips, C. B., and Park, J. Y. (1988). "On Formulating Wald Tests of Nonlinear Restrictions." *Econometrica* 56:1065–1083.

Powers, D. A., and Xie, Y. (2000). *Statistical Methods for Categorical Data Analysis*. San Diego: Academic Press.

Schmidt, P., and Strauss, R. (1975). "The Prediction of Occupation Using Multiple Logit Models." *International Economic Review* 16:471–486.

Small, K. (1982). "The Scheduling of Consumer Activities: Work Trips." *American Economic Review* 72:467–479.

Spector, L., and Mazzeo, M. (1980). "Probit Analysis and Economic Education." *Journal of Economic Education* 11:37–44.

Swait, J., and Bernardino, A. (2000). "Distinguishing Taste Variation from Error Structure in Discrete Choice Data." *Transportation Research Part B* 34:1–15.

Theil, H. (1969). "A Multinomial Extension of the Linear Logit Model." *International Economic Review* 10:251–259.

Train, K. E., Ben-Akiva, M., and Atherton, T. (1989). "Consumption Patterns and Self-Selecting Tariffs." *Review of Economics and Statistics* 71:62–73.

Chapter 26
The MODEL Procedure

Contents

Overview: MODEL Procedure **1471**
Getting Started: MODEL Procedure **1475**
Nonlinear Regression Analysis 1475
Nonlinear Systems Regression 1479
General Form Models 1479
Solving Simultaneous Nonlinear Equation Systems 1482
Working with Model Files 1485
Monte Carlo Simulation 1486
Syntax: MODEL Procedure **1489**
Functional Summary 1490
PROC MODEL Statement 1496
BOUNDS Statement 1502
BY Statement 1504
CONTROL Statement 1507
DELETEMODEL Statement 1507
ENDOGENOUS Statement 1507
EQGROUP Statement 1507
ERRORMODEL Statement 1508
ESTIMATE Statement 1509
EXOGENOUS Statement 1511
FIT Statement 1511
ID Statement 1520
INCLUDE Statement 1520
INSTRUMENTS Statement 1520
LABEL Statement 1522
MOMENT Statement 1522
OUTVARS Statement 1524
PARAMETERS Statement 1524
Programming Statements 1524
RANGE Statement 1524
RESET Statement 1525
RESTRICT Statement 1525
SOLVE Statement 1527
TEST Statement 1533
VAR Statement 1534
VARGROUP Statement 1534

WEIGHT Statement . . . 1535
Details: Estimation by the MODEL Procedure . . . **1535**
Estimation Methods . . . 1535
Properties of the Estimates . . . 1552
Minimization Methods . . . 1555
Convergence Criteria . . . 1556
Troubleshooting Convergence Problems . . . 1558
Iteration History . . . 1567
Computer Resource Requirements . . . 1570
Testing for Normality . . . 1574
Heteroscedasticity . . . 1576
Testing for Autocorrelation . . . 1583
Transformation of Error Terms . . . 1584
Error Covariance Structure Specification . . . 1586
Ordinary Differential Equations . . . 1590
Restrictions and Bounds on Parameters . . . 1600
Tests on Parameters . . . 1601
Hausman Specification Test . . . 1603
Chow Tests . . . 1604
Profile Likelihood Confidence Intervals . . . 1605
Choice of Instruments . . . 1607
Autoregressive Moving-Average Error Processes . . . 1610
Distributed Lag Models and the %PDL Macro . . . 1625
Input Data Sets . . . 1627
Output Data Sets . . . 1632
ODS Table Names . . . 1635
ODS Graphics . . . 1637
Details: Simulation by the MODEL Procedure . . . **1638**
Solution Modes . . . 1639
Multivariate *t* Distribution Simulation . . . 1643
Alternate Distribution Simulation . . . 1645
Mixtures of Distributions—Copulas . . . 1647
Solution Mode Output . . . 1654
Goal Seeking: Solving for Right-Hand-Side Variables . . . 1660
Numerical Solution Methods . . . 1662
Numerical Integration . . . 1669
Limitations . . . 1670
SOLVE Data Sets . . . 1671
Programming Language Overview: MODEL Procedure . . . **1673**
Variables in the Model Program . . . 1673
Equation Translations . . . 1677
Derivatives . . . 1679
Mathematical Functions . . . 1680
Functions across Time . . . 1681

Language Differences 1685
Storing Programs in Model Files 1688
Macro Return Codes (SYSINFO) 1689
Diagnostics and Debugging 1689
Analyzing the Structure of Large Models 1694
Examples: MODEL Procedure **1704**
Example 26.1: OLS Single Nonlinear Equation 1704
Example 26.2: A Consumer Demand Model 1707
Example 26.3: Vector AR(1) Estimation 1711
Example 26.4: MA(1) Estimation 1714
Example 26.5: Polynomial Distributed Lags by Using %PDL 1718
Example 26.6: General Form Equations 1721
Example 26.7: Spring and Damper Continuous System 1725
Example 26.8: Nonlinear FIML Estimation 1730
Example 26.9: Circuit Estimation 1732
Example 26.10: Systems of Differential Equations 1733
Example 26.11: Monte Carlo Simulation 1737
Example 26.12: Cauchy Distribution Estimation 1738
Example 26.13: Switching Regression Example 1740
Example 26.14: Simulating from a Mixture of Distributions 1743
Example 26.15: Simulated Method of Moments—Simple Linear Regression 1751
Example 26.16: Simulated Method of Moments—AR(1) Process 1753
Example 26.17: Simulated Method of Moments—Stochastic Volatility Model 1754
Example 26.18: Duration Data Model with Unobserved Heterogeneity 1756
Example 26.19: EMM Estimation of a Stochastic Volatility Model 1757
Example 26.20: Illustration of ODS Graphics 1762
Example 26.21: A Translog Cost Function and Derived Demands 1772
Example 26.22: Reducing Parameter Variance in a Tree Biomass Model 1782
References **1786**

Overview: MODEL Procedure

The MODEL procedure analyzes models in which the relationships among the variables form a system of one or more nonlinear equations. Primary uses of the MODEL procedure are estimation, simulation, and forecasting of nonlinear simultaneous equation models.

PROC MODEL features include the following:

- SAS programming statements to define simultaneous systems of nonlinear equations
- tools to analyze the structure of the simultaneous equation system
- ARIMA, PDL, and other dynamic modeling capabilities

- tools to specify and estimate the error covariance structure
- tools to estimate and solve ordinary differential equations
- the following methods of parameter estimation:
  - ordinary least squares (OLS)
  - two-stage least squares (2SLS)
  - seemingly unrelated regression (SUR) and iterative SUR (ITSUR)
  - three-stage least squares (3SLS) and iterative 3SLS (IT3SLS)
  - generalized method of moments (GMM)
  - simulated method of moments (SMM)
  - full information maximum likelihood (FIML)
  - general log-likelihood maximization
- simulation and forecasting capabilities
- Monte Carlo simulation
- goal-seeking solutions

A system of equations can be nonlinear in the parameters, nonlinear in the observed variables, or nonlinear in both the parameters and the variables. *Nonlinear* in the parameters means that the mathematical relationship between the variables and parameters is not required to have a linear form. (A linear model is a special case of a nonlinear model.) A general nonlinear system of equations can be written as

$$\begin{aligned} q_1(y_{1,t}, y_{2,t}, \ldots, y_{g,t}, x_{1,t}, x_{2,t}, \ldots, x_{m,t}, \theta_1, \theta_2, \ldots, \theta_p) &= \epsilon_{1,t} \\ q_2(y_{1,t}, y_{2,t}, \ldots, y_{g,t}, x_{1,t}, x_{2,t}, \ldots, x_{m,t}, \theta_1, \theta_2, \ldots, \theta_p) &= \epsilon_{2,t} \\ &\vdots \\ q_g(y_{1,t}, y_{2,t}, \ldots, y_{g,t}, x_{1,t}, x_{2,t}, \ldots, x_{m,t}, \theta_1, \theta_2, \ldots, \theta_p) &= \epsilon_{g,t} \end{aligned}$$

where $y_{i,t}$ is an endogenous variable, $x_{i,t}$ is an exogenous variable, θ_i is a parameter, and ϵ_i is the unknown error. The subscript t represents time or some index to the data.

In econometrics literature, the observed variables are either *endogenous* (dependent) variables or *exogenous* (independent) variables. This system can be written more succinctly in vector form as

$$\mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \boldsymbol{\theta}) = \boldsymbol{\epsilon}_t$$

This system of equations is in *general form* because the error term is by itself on one side of the equality. Systems can also be written in *normalized form* by placing the endogenous variable on one side of the equality, with each equation defining a predicted value for a unique endogenous variable. A normalized form equation system can be written in vector notation as

$$\mathbf{y}_t = \mathbf{f}(\mathbf{y}_t, \mathbf{x}_t, \boldsymbol{\theta}) + \boldsymbol{\epsilon}_t.$$

PROC MODEL handles equations written in both forms.

Econometric models often explain the current values of the endogenous variables as functions of past values of exogenous and endogenous variables. These past values are referred to as *lagged* values, and the variable x_{t-i} is called lag i of the variable x_t. Using lagged variables, you can create a *dynamic*, or time-dependent, model. In the preceding model systems, the lagged exogenous and endogenous variables are included as part of the exogenous variables.

If the data are time series, so that t indexes time (see Chapter 4, "Working with Time Series Data," for more information on time series), it is possible that $\boldsymbol{\epsilon}_t$ depends on $\boldsymbol{\epsilon}_{t-i}$ or, more generally, the $\boldsymbol{\epsilon}_t$'s are not identically and independently distributed. If the errors of a model system are autocorrelated, the standard error of the estimates of the parameters of the system will be inflated.

Sometimes the $\boldsymbol{\epsilon}_i$'s are not identically distributed because the variance of $\boldsymbol{\epsilon}$ is not constant. This is known as *heteroscedasticity*. Heteroscedasticity in an estimated model can also inflate the standard error of the estimates of the parameters. Using a weighted estimation can sometimes eliminate this problem. Alternately, a variance model such as GARCH or EGARCH can be estimated to correct for heteroscedasticity. If the proper weighting scheme and the form of the error model is difficult to determine, generalized methods of moments (GMM) estimation can be used to determine parameter estimates with very few assumptions about the form of the error process.

Other problems can also arise when estimating systems of equations. Consider the following system of equations, which is nonlinear in its parameters and cannot be estimated with linear regression:

$$\begin{aligned} y_{1,t} &= \theta_1 + (\theta_2 + \theta_3\theta_4^t)^{-1} + \theta_5 y_{2,t} + \epsilon_{1,t} \\ y_{2,t} &= \theta_6 + (\theta_7 + \theta_8\theta_9^t)^{-1} + \theta_{10} y_{1,t} + \epsilon_{2,t} \end{aligned}$$

This system of equations represents a rudimentary predator-prey process with y_1 as the prey and y_2 as the predator (the second term in both equations is a logistics curve). The two equations must be estimated simultaneously because of the cross-dependency of y's. This cross-dependency makes ϵ_1 and ϵ_2 violate the assumption of independence. Nonlinear ordinary least squares estimation of these equations produce biased and inconsistent parameter estimates. This is called *simultaneous equation bias*.

One method to remove simultaneous equation bias, in the linear case, is to replace the endogenous variables on the right-hand side of the equations with predicted values that are uncorrelated with the error terms. These predicted values can be obtained through a preliminary, or "first-stage," *instrumental variable regression*. *Instrumental variables*, which are uncorrelated with the error term, are used as regressors to model the predicted values. The parameter estimates are obtained by a second regression by using the predicted values of the regressors. This process is called *two-stage least squares*.

In the nonlinear case, nonlinear ordinary least squares estimation is performed iteratively by using a linearization of the model with respect to the parameters. The instrumental solution to simultaneous equation bias in the nonlinear case is the same as the linear case, except the linearization of the model with respect to the parameters is predicted by the instrumental regression. Nonlinear two-stage least squares is one of several instrumental variables methods available in the MODEL procedure to handle simultaneous equation bias.

When you have a system of several regression equations, the random errors of the equations can be correlated. In this case, the large-sample efficiency of the estimation can be improved by using a joint generalized least squares method that takes the cross-equation correlations into account. If the equations are not simultaneous (no dependent regressors), then *seemingly unrelated regression* (SUR) can be used. The SUR method requires an estimate of the cross-equation error covariance matrix, Σ. The usual approach is to first fit the equations by using OLS, compute an estimate $\hat{\Sigma}$ from the OLS residuals, and then perform the SUR estimation based on $\hat{\Sigma}$. The MODEL procedure estimates Σ by default, or you can supply your own estimate of Σ.

If the equation system is simultaneous, you can combine the 2SLS and SUR methods to take into account both simultaneous equation bias and cross-equation correlation of the errors. This is called *three-stage least squares* or 3SLS.

A different approach to the simultaneous equation bias problem is the full information maximum likelihood (FIML) estimation method. FIML does not require instrumental variables, but it assumes that the equation errors have a multivariate normal distribution. 2SLS and 3SLS estimation do not assume a particular distribution for the errors.

Other nonnormal error distribution models can be estimated as well. The centered *t* distribution with estimated degrees of freedom and nonconstant variance is an additional built-in likelihood function. If the distribution of the equation errors is not normal or *t* but known, then the log likelihood can be specified by using the ERRORMODEL statement.

Once a nonlinear model has been estimated, it can be used to obtain forecasts. If the model is linear in the variables you want to forecast, a simple linear solve can generate the forecasts. If the system is nonlinear, an iterative procedure must be used. The preceding example system is linear in its endogenous variables. The MODEL procedure's SOLVE statement is used to forecast nonlinear models.

One of the main purposes of creating models is to obtain an understanding of the relationship among the variables. There are usually only a few variables in a model you can control (for example, the amount of money spent on advertising). Often you want to determine how to change the variables under your control to obtain some target goal. This process is called *goal seeking*. PROC MODEL allows you to solve for any subset of the variables in a system of equations given values for the remaining variables.

The nonlinearity of a model creates two problems with the forecasts: the forecast errors are not normally distributed with zero mean, and no formula exists to calculate the forecast confidence intervals. PROC MODEL provides Monte Carlo techniques, which, when used with the covariance of the parameters and error covariance matrix, can produce approximate error bounds on the forecasts. The following distributions on the errors are supported for multivariate Monte Carlo simulation:

- Cauchy
- chi-squared
- empirical
- *F*
- Poisson
- *t*
- uniform

A transformation technique is used to create a covariance matrix for generating the correct innovations in a Monte Carlo simulation.

Getting Started: MODEL Procedure

This section introduces the MODEL procedure and shows how to use PROC MODEL for several kinds of nonlinear regression analysis and nonlinear systems simulation problems.

Nonlinear Regression Analysis

One of the most important uses of PROC MODEL is to estimate unknown parameters in a nonlinear model. A simple nonlinear model has the form:

$$y = f(\mathbf{x}, \boldsymbol{\theta}) + \epsilon$$

where $\mathbf{x}$ is a vector of exogenous variables. To estimate unknown parameters by using PROC MODEL, do the following:

1. Use the DATA= option in a PROC MODEL statement to specify the input SAS data set that contains $\mathbf{y}$ and $\mathbf{x}$, the observed values of the variables.
2. Write the equation for the model by using SAS programming statements, including all parameters and arithmetic operators but leaving off the unobserved error component, ϵ.
3. Use a FIT statement to fit the model equation to the input data to determine the unknown parameters, $\boldsymbol{\theta}$.

An Example

The SASHELP library contains the data set CITIMON, which contains the variable LHUR, the monthly unemployment figures, and the variable IP, the monthly industrial production index. You suspect that the unemployment rates are inversely proportional to the industrial production index. Assume that these variables are related by the following nonlinear equation:

$$lhur = \frac{1}{a \cdot \mathrm{ip} + b} + c + \epsilon$$

In this equation a, b, and c are unknown coefficients and ϵ is an unobserved random error.

The following statements illustrate how to use PROC MODEL to estimate values for a, b, and c from the data in SASHELP.CITIMON.

```
proc model data=sashelp.citimon;
   lhur = 1/(a * ip + b) + c;
   fit lhur;
run;
```

Notice that the model equation is written as a SAS assignment statement. The variable LHUR is assumed to be the dependent variable because it is named in the FIT statement and is on the left-hand side of the assignment.

PROC MODEL determines that LHUR and IP are observed variables because they are in the input data set. A, B, and C are treated as unknown parameters to be estimated from the data because they are not in the input data set. If the data set contained a variable named A, B, or C, you would need to explicitly declare the parameters with a PARMS statement.

In response to the FIT statement, PROC MODEL estimates values for A, B, and C by using nonlinear least squares and prints the results. The first part of the output is a "Model Summary" table, shown in Figure 26.1.

Figure 26.1 Model Summary Report

The MODEL Procedure

Model Summary	
Model Variables	1
Parameters	3
Equations	1
Number of Statements	1

Model Variables	LHUR
Parameters	a b c
Equations	LHUR

This table details the size of the model, including the number of programming statements that define the model, and lists the dependent variables (LHUR in this case), the unknown parameters (A, B, and C), and the model equations. In this case the equation is named for the dependent variable, LHUR.

PROC MODEL then prints a summary of the estimation problem, as shown in Figure 26.2.

Figure 26.2 Estimation Problem Report

The Equation to Estimate is
LHUR = F(a, b, c(1))

The notation used in the summary of the estimation problem indicates that LHUR is a function of A, B, and C, which are to be estimated by fitting the function to the data. If the partial derivative of the equation with respect to a parameter is a simple variable or constant, the derivative is shown in parentheses after the parameter name. In this case, the derivative with respect to the intercept C is 1. The derivatives with respect to A and B are complex expressions and so are not shown.

Next, PROC MODEL prints an estimation summary as shown in Figure 26.3.

Figure 26.3 Estimation Summary Report

The MODEL Procedure
OLS Estimation Summary

Data Set Options	
DATA=	SASHELP.CITIMON

Minimization Summary	
Parameters Estimated	3
Method	Gauss
Iterations	10

Final Convergence Criteria	
R	0.000737
PPC(b)	0.003943
RPC(b)	0.00968
Object	4.784E-6
Trace(S)	0.533325
Objective Value	0.522214

Observations Processed	
Read	145
Solved	145
Used	144
Missing	1

The estimation summary provides information on the iterative process used to compute the estimates. The heading "OLS Estimation Summary" indicates that the nonlinear ordinary least squares (OLS) estimation method is used. This table indicates that all three parameters were estimated successfully by using 144 nonmissing observations from the data set SASHELP.CITIMON. Calculating the estimates required 10 iterations of the GAUSS method. Various measures of how well the iterative process converged are also shown. For example, the "RPC(B)" value 0.00968 means that on the final iteration the largest relative change in any estimate was for parameter B, which changed by 0.968 percent. See the section "Convergence Criteria" on page 1556 for details.

PROC MODEL then prints the estimation results. The first part of this table is the summary of residual errors, shown in Figure 26.4.

Figure 26.4 Summary of Residual Errors Report

The MODEL Procedure

Nonlinear OLS Summary of Residual Errors

Equation	DF Model	DF Error	SSE	MSE	Root MSE	R-Square	Adj R-Sq	Label
LHUR	3	141	75.1989	0.5333	0.7303	0.7472	0.7436	UNEMPLOYMENT RATE: ALL WORKERS, 16 YEARS

This table lists the sum of squared errors (SSE), the mean squared error (MSE), the root mean squared error (root MSE), and the R^2 and adjusted R^2 statistics. The R^2 value of 0.7472 means that the estimated model explains approximately 75 percent more of the variability in LHUR than a mean model explains.

Following the summary of residual errors is the parameter estimates table, shown in Figure 26.5.

Figure 26.5 Parameter Estimates

Nonlinear OLS Parameter Estimates

Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|
a	0.009046	0.00343	2.63	0.0094
b	-0.57059	0.2617	-2.18	0.0309
c	3.337151	0.7297	4.57	<.0001

Because the model is nonlinear, the standard error of the estimate, the *t* value, and its significance level are only approximate. These values are computed using asymptotic formulas that are correct for large sample sizes but only approximately correct for smaller samples. Thus, you should use caution in interpreting these statistics for nonlinear models, especially for small sample sizes. For linear models, these results are exact and are the same as standard linear regression.

The last part of the output produced by the FIT statement is shown in Figure 26.6.

Figure 26.6 System Summary Statistics

Number of Observations		Statistics for System	
Used	144	Objective	0.5222
Missing	1	Objective*N	75.1989

This table lists the objective value for the estimation of the nonlinear system. Since there is only a single equation in this case, the objective value is the same as the residual MSE for LHUR except that the objective value does not include a degrees-of-freedom correction. This can be seen in the fact that "Objective*N" equals the residual SSE, 75.1989. N is 144, the number of observations used.

Convergence and Starting Values

Computing parameter estimates for nonlinear equations requires an iterative process. Starting with an initial guess for the parameter values, PROC MODEL tries different parameter values until the objective function of the estimation method is minimized. (The objective function of the estimation method is sometimes called the *fitting function*.) This process does not always succeed, and whether it does succeed depends greatly on the starting values used. By default, PROC MODEL uses the starting value 0.0001 for all parameters.

Consequently, in order to use PROC MODEL to achieve convergence of parameter estimates, you need to know two things: how to recognize convergence failure by interpreting diagnostic output, and how to specify reasonable starting values. The MODEL procedure includes alternate iterative techniques and grid search capabilities to aid in finding estimates. See the section "Troubleshooting Convergence Problems" on page 1558 for more details.

Nonlinear Systems Regression

If a model has more than one endogenous variable, several facts need to be considered in the choice of an estimation method. If the model has endogenous regressors, then an instrumental variables method such as 2SLS or 3SLS can be used to avoid simultaneous equation bias. Instrumental variables must be provided to use these methods. A discussion of possible choices for instrumental variables is provided in the section "Choice of Instruments" on page 1607 in this chapter.

The following is an example of the use of 2SLS and the INSTRUMENTS statement:

```
proc model data=test2;
   exogenous x1 x2;
   parms a1 a2 b2 2.5 c2 55 d1;

   y1 = a1 * y2 + b2 * x1 * x1 + d1;
   y2 = a2 * y1 + b2 * x2 * x2 + c2 / x2 + d1;

   fit y1 y2 / 2sls;
   instruments b2 c2 _exog_;
run;
```

The estimation method selected is added after the slash (/) on the FIT statement. The INSTRUMENTS statement follows the FIT statement and in this case selects all the exogenous variables as instruments with the _EXOG_ keyword. The parameters B2 and C2 in the instruments list request that the derivatives with respect to B2 and C2 be additional instruments.

Full information maximum likelihood (FIML) can also be used to avoid simultaneous equation bias. FIML is computationally more expensive than an instrumental variables method and assumes that the errors are normally distributed. On the other hand, FIML does not require the specification of instruments. FIML is selected with the FIML option on the FIT statement.

The preceding example is estimated with FIML by using the following statements:

```
proc model data=test2;
   exogenous x1 x2;
   parms a1 a2 b2 2.5 c2 55 d1;

   y1 = a1 * y2 + b2 * x1 * x1 + d1;
   y2 = a2 * y1 + b2 * x2 * x2 + c2 / x2 + d1;

   fit y1 y2 / fiml;
run;
```

General Form Models

The single equation example shown in the preceding section was written in normalized form and specified as an assignment of the regression function to the dependent variable LHUR. However, sometimes it is impossible or inconvenient to write a nonlinear model in normalized form.

To write a general form equation, give the equation a name with the prefix "EQ.". This EQ.-prefixed variable represents the equation error. Write the equation as an assignment to this variable.

For example, suppose you have the following nonlinear model that relates the variables x and y :

$$\epsilon = a + b \ln(cy + dx)$$

Naming this equation 'one', you can fit this model with the following statements:

```
proc model data=xydata;
   eq.one = a + b * log( c * y + d * x );
   fit one;
run;
```

The use of the EQ. prefix tells PROC MODEL that the variable is an error term and that it should not expect actual values for the variable ONE in the input data set.

Supply and Demand Models

General form specifications are often useful when you have several equations for the same dependent variable. This is common in supply and demand models, where both the supply equation and the demand equation are written as predictions for quantity as functions of price.

For example, consider the following supply and demand system:

$$\begin{aligned} \text{(supply)} \quad & \text{quantity} = \alpha_1 + \alpha_2 \text{ price} + \epsilon_1 \\ \text{(demand)} \quad & \text{quantity} = \beta_1 + \beta_2 \text{ price} + \beta_3 \text{ income} + \epsilon_2 \end{aligned}$$

Assume the *quantity* of interest is the amount of energy consumed in the U.S., the *price* is the price of gasoline, and the *income* variable is the consumer debt. When the market is at equilibrium, these equations determine the market price and the equilibrium quantity. These equations are written in general form as

$$\epsilon_1 = quantity - (\alpha_1 + \alpha_2\ price)$$

$$\epsilon_2 = quantity - (\beta_1 + \beta_2\ price + \beta_3\ income)$$

Note that the endogenous variables *quantity* and *price* depend on two error terms so that OLS should not be used. The following example uses three-stage least squares estimation.

Data for this model is obtained from the SASHELP.CITIMON data set.

```
title1 'Supply-Demand Model using General-form Equations';
proc model data=sashelp.citimon;
   endogenous eegp eec;
   exogenous exvus cciutc;
   parameters a1 a2 b1 b2 b3 ;
   label eegp   = 'Gasoline Retail Price'
         eec    = 'Energy Consumption'
         cciutc = 'Consumer Debt';

   /* -------- Supply equation ------------- */
```

```
    eq.supply = eec - (a1 + a2 * eegp);

    /* -------- Demand equation ------------- */
    eq.demand = eec - (b1 + b2 * eegp + b3 * cciutc);

    /* -------- Instrumental variables -------*/
    lageegp = lag(eegp); lag2eegp=lag2(eegp);

    /* -------- Estimate parameters --------- */
    fit supply demand / n3sls fsrsq;
    instruments _EXOG_ lageegp lag2eegp;
run;
```

The FIT statement specifies the two equations to estimate and the method of estimation, N3SLS. Note that '3SLS' is an alias for N3SLS. The option FSRSQ is selected to get a report of the first stage R^2 to determine the acceptability of the selected instruments.

Since three-stage least squares is an instrumental variables method, instruments are specified with the INSTRUMENTS statement. The instruments selected are all the exogenous variables, selected with the _EXOG_ option, and two lags of the variable EEGP: LAGEEGP and LAG2EEGP.

The data set CITIMON has four observations that generate missing values because values for EEGP, EEC, or CCIUTC are missing. This is revealed in the "Observations Processed" output shown in Figure 26.7. Missing values are also generated when the equations cannot be computed for a given observation. Missing observations are not used in the estimation.

Figure 26.7 Supply-Demand Observations Processed

Supply-Demand Model using General-form Equations

The MODEL Procedure
3SLS Estimation Summary

Observations Processed	
Read	145
Solved	143
First	3
Last	145
Used	139
Missing	4
Lagged	2

The lags used to create the instruments also reduce the number of observations used. In this case, the first two observations were used to fill the lags of EEGP.

The data set has a total of 145 observations, of which four generated missing values and two were used to fill lags, which left 139 observations for the estimation. In the estimation summary, in Figure 26.8, the total degrees of freedom for the model and error is 139.

Figure 26.8 Supply-Demand Parameter Estimates

Supply-Demand Model using General-form Equations

The MODEL Procedure

Nonlinear 3SLS Summary of Residual Errors

Equation	DF Model	DF Error	SSE	MSE	Root MSE	R-Square	Adj R-Sq
supply	2	137	43.2677	0.3158	0.5620		
demand	3	136	39.5791	0.2910	0.5395		

Nonlinear 3SLS Parameter Estimates

Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|	1st Stage R-Square
a1	7.30952	0.3799	19.24	<.0001	1.0000
a2	-0.00853	0.00328	-2.60	0.0103	0.9617
b1	6.82196	0.3788	18.01	<.0001	1.0000
b2	-0.00614	0.00303	-2.02	0.0450	0.9617
b3	9E-7	3.165E-7	2.84	0.0051	1.0000

One disadvantage of specifying equations in general form is that there are no actual values associated with the equation, so the R^2 statistic cannot be computed.

Solving Simultaneous Nonlinear Equation Systems

You can use a SOLVE statement to solve the nonlinear equation system for some variables when the values of other variables are given.

Consider the supply and demand model shown in the preceding example. The following statement computes equilibrium price (EEGP) and quantity (EEC) values for given observed cost (CCIUTC) values and stores them in the output data set EQUILIB.

```
title1 'Supply-Demand Model using General-form Equations';
proc model data=sashelp.citimon(where=(eec ne .));
   endogenous eegp eec;
   exogenous exvus cciutc;
   parameters a1 a2 a3 b1 b2 ;
   label eegp   = 'Gasoline Retail Price'
         eec    = 'Energy Consumption'
         cciutc = 'Consumer Debt';

   /* -------- Supply equation ------------- */
   eq.supply = eec - (a1 + a2 * eegp);

   /* -------- Demand equation ------------- */
   eq.demand = eec - (b1 + b2 * eegp  + b3 * cciutc);

   /* -------- Instrumental variables -------*/
   lageegp = lag(eegp); lag2eegp=lag2(eegp);
```

```
   /* -------- Estimate parameters --------- */
   instruments _EXOG_ lageegp lag2eegp;
   fit supply demand / n3sls ;
   solve eegp eec / out=equilib;
run;
```

As a second example, suppose you want to compute points of intersection between the square root function and hyperbolas of the form $a + b/x$. That is, you want to solve the system:

$$\text{(square root)} \quad y = \sqrt{x}$$

$$\text{(hyperbola)} \quad y = a + \frac{b}{x}$$

The following statements read parameters for several hyperbolas in the input data set TEST and solve the nonlinear equations. The SOLVEPRINT option in the SOLVE statement prints the solution values. The ID statement is used to include the values of A and B in the output of the SOLVEPRINT option.

```
title1 'Solving a Simultaneous System';
data test;
   input a b @@;
datalines;
  0 1   1 1   1 2
;

proc model data=test;
   eq.sqrt      = sqrt(x) - y;
   eq.hyperbola = a + b / x - y;
   solve x y / solveprint;
   id a b;
run;
```

The printed output produced by this example consists of a model summary report, a listing of the solution values for each observation, and a solution summary report. The model summary for this example is shown in Figure 26.9.

Figure 26.9 Model Summary Report

Solving a Simultaneous System

The MODEL Procedure

Model Summary	
Model Variables	2
ID Variables	2
Equations	2
Number of Statements	2

Model Variables	x y
Equations	sqrt hyperbola

The output produced by the SOLVEPRINT option is shown in Figure 26.10.

Figure 26.10 Solution Values for Each Observation

Solving a Simultaneous System

The MODEL Procedure
Simultaneous Simulation

Observation	1	**a**	0	**b**	1.0000	**eq.hyperbola**	0.000000
		Iterations	17	**CC**	0.000000		

Solution Values	
x	**y**
1.000000	1.000000

Observation	2	**a**	1.0000	**b**	1.0000	**eq.hyperbola**	0.000000
		Iterations	5	**CC**	0.000000		

Solution Values	
x	**y**
2.147899	1.465571

Observation	3	**a**	1.0000	**b**	2.0000	**eq.hyperbola**	0.000000
		Iterations	4	**CC**	0.000000		

Solution Values	
x	**y**
2.875130	1.695621

For each observation, a heading line is printed that lists the values of the ID variables for the observation and information about the iterative process used to compute the solution. The number of iterations required, and the convergence measure (labeled CC) are printed. This convergence measure indicates the maximum error by which solution values fail to satisfy the equations. When this error is small enough (as determined by the CONVERGE= option), the iterations terminate. The equation with the largest error is indicated. For example, for observation 3 the HYPERBOLA equation has an error of 4.42×10^{-13} while the error of the SQRT equation is even smaller. Following the heading line for the observation, the solution values are printed.

The last part of the SOLVE statement output is the solution summary report shown in Figure 26.11. This report summarizes the solution method used (Newton's method by default), the iteration history, and the observations processed.

Figure 26.11 Solution Summary Report

Solving a Simultaneous System

The MODEL Procedure
Simultaneous Simulation

Data Set Options
DATA= TEST

Figure 26.11 *continued*

Solution Summary	
Variables Solved	2
Implicit Equations	2
Solution Method	NEWTON
CONVERGE=	1E-8
Maximum CC	9.176E-9
Maximum Iterations	17
Total Iterations	26
Average Iterations	8.666667

Observations Processed	
Read	3
Solved	3

Variables Solved For	x y
Equations Solved	sqrt hyperbola

Working with Model Files

Model files enable you to save models that are specified in a PROC MODEL step to a SAS library file. Model files store the SAS programming statements and variable declarations that constitute a model, and they make those statements and declarations available for use in subsequent PROC MODEL steps. Typically you specify the OUTMODEL= option in one PROC MODEL step to save a model specification to a model file, and later you specify the MODEL= option in one or more other PROC MODEL steps to read one or more model files. For more information, see the section "Storing Programs in Model Files" on page 1688.

Model files can help organize modeling efforts when many statements are required to specify, estimate, and simulate models. For example, in the supply and demand model analyzed previously, the following statements specify the system of equations once and save them to the model file SUPDEM:

```
proc model outmodel=supdem;
   endogenous eegp eec;
   exogenous exvus cciutc;
   parameters a1 a2 b1 b2 b3 ;
   label eegp   = 'Gasoline Retail Price'
         eec    = 'Energy Consumption'
         cciutc = 'Consumer Debt';

   /* -------- Supply equation ------------- */
   eq.supply = eec - (a1 + a2 * eegp );

   /* -------- Demand equation ------------- */
   eq.demand = eec - (b1 + b2 * eegp + b3 * cciutc);
quit;
```

When the model has been defined and saved, its parameters can be estimated in a separate PROC MODEL step. The following estimation step defines the instruments LAGEEGP and LAG2EEGP (which do not appear in the supply and demand model equations) and performs the three-stage least squares estimation:

```
proc model data=sashelp.citimon model=supdem outmodel=supdem;
   /* -------- Instrumental variables ------ */
   lageegp = lag(eegp); lag2eegp=lag2(eegp);
   /* -------- Estimate parameters --------- */
   instruments _EXOG_ lageegp lag2eegp;
   fit supply demand / n3sls;
quit;
```

Finally, the following statements use the supply and demand model together with its parameter estimates to solve for equilibrium prices and quantities:

```
proc model data=sashelp.citimon(where=(eec ne .)) model=supdem;
   solve eegp eec / out=equilib;
quit;
```

Monte Carlo Simulation

The RANDOM= option is used to request Monte Carlo (or stochastic) simulation to generate confidence intervals for a forecast. The confidence intervals are implied by the model's relationship to implicit random error term ϵ and the parameters.

The Monte Carlo simulation generates a random set of additive error values, one for each observation and each equation, and computes one set of perturbations of the parameters. These new parameters, along with the additive error terms, are then used to compute a new forecast that satisfies this new simultaneous system. Then a new set of additive error values and parameter perturbations is computed, and the process is repeated the requested number of times.

Consider the following exchange rate model for the U.S. dollar with the German mark and the Japanese yen:

$$rate\_jp = a_1 + b_1 im\_jp + c_1 di\_jp;$$

$$rate\_wg = a_2 + b_2 im\_wg + c_1 di\_wg;$$

where $rate\_jp$ and $rate\_wg$ are the exchange rate of the Japanese yen and the German mark versus the U.S. dollar, respectively; *im_jp* and *im_wg* are the imports from Japan and Germany in 1984 dollars, respectively; and *di_jp* and *di_wg* are the differences in inflation rate of Japan and the U.S., and Germany and the U.S., respectively. The Monte Carlo capabilities of the MODEL procedure are used to generate error bounds on a forecast by using this model.

```
proc model data=exchange;
   endo im_jp im_wg;
   exo di_jp di_wg;
   parms a1 a2 b1 b2 c1 c2;
   label rate_jp = 'Exchange Rate of Yen/$'
         rate_wg = 'Exchange Rate of Gm/$'
         im_jp = 'Imports to US from Japan in 1984 $'
         im_wg = 'Imports to US from WG in 1984 $'
         di_jp = 'Difference in Inflation Rates US-JP'
         di_wg = 'Difference in Inflation Rates US-WG';

   rate_jp = a1 + b1*im_jp + c1*di_jp;
   rate_wg = a2 + b2*im_wg + c2*di_wg;

            /* Fit the EXCHANGE data */
   fit rate_jp rate_wg / sur outest=xch_est outcov outs=s;

            /* Solve using the WHATIF data set */
   solve rate_jp rate_wg / data=whatif estdata=xch_est sdata=s
          random=100 seed=123 out=monte forecast;
   id yr;
   range yr=1986;
run;
```

Data for the EXCHANGE data set was obtained from the Department of Commerce and the yearly "Economic Report of the President."

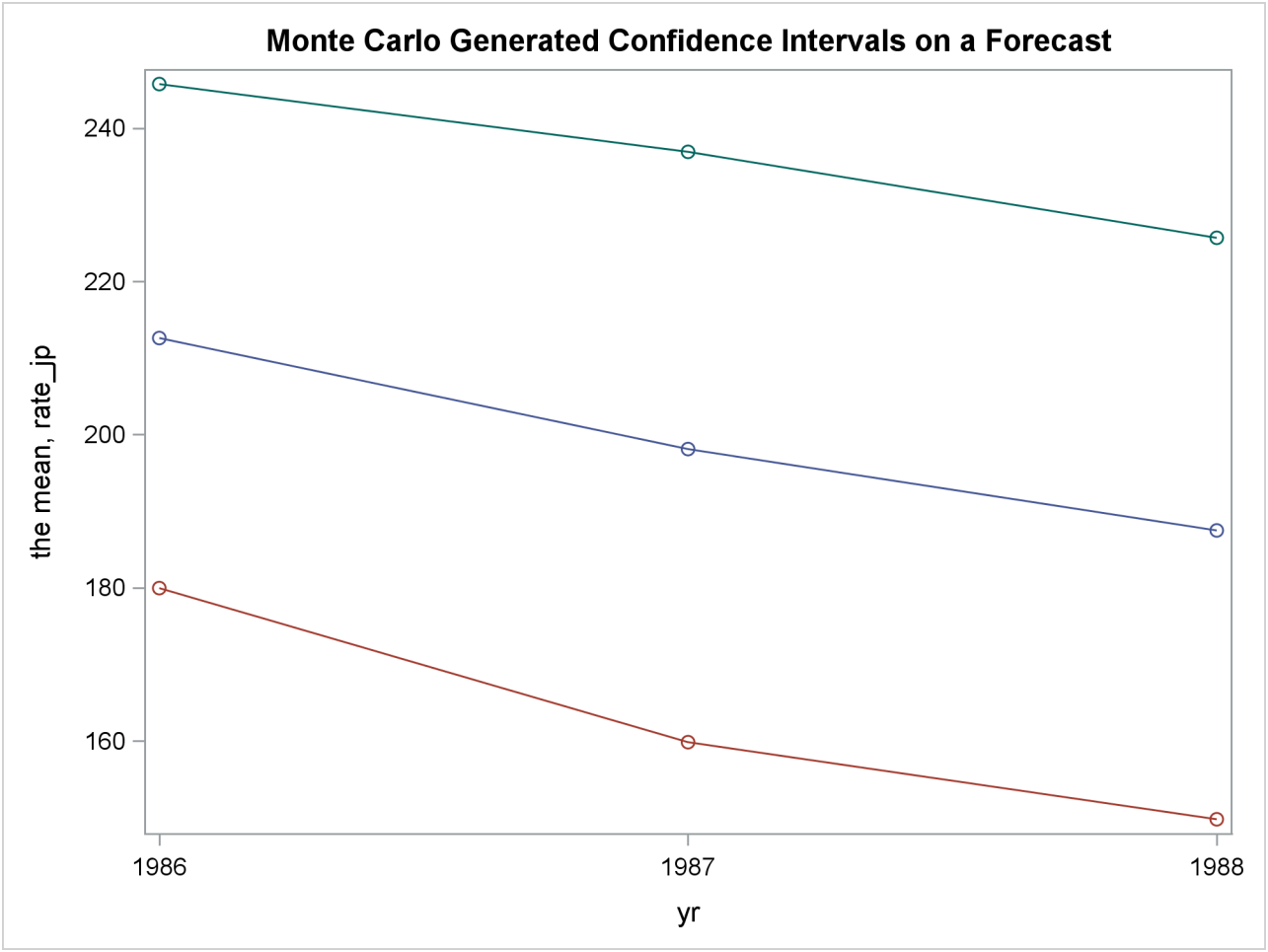
First, the parameters are estimated using SUR selected by the SUR option in the FIT statement. The OUTEST= option is used to create the XCH_EST data set which contains the estimates of the parameters. The OUTCOV option adds the covariance matrix of the parameters to the XCH_EST data set. The OUTS= option is used to save the covariance of the equation error in the data set S.

Next, Monte Carlo simulation is requested by using the RANDOM= option in the SOLVE statement. The data set WHATIF is used to drive the forecasts. The ESTDATA= option reads in the XCH_EST data set which contains the parameter estimates and covariance matrix. Because the parameter covariance matrix is included, perturbations of the parameters are performed. The SDATA= option causes the Monte Carlo simulation to use the equation error covariance in the S data set to perturb the equation errors. The SEED= option selects the number 123 as a seed value for the random number generator. The output of the Monte Carlo simulation is written to the data set MONTE selected by the OUT= option.

To generate a confidence interval plot for the forecast, use PROC UNIVARIATE to generate percentile bounds and use PROC SGPLOT to plot the graph. The following SAS statements produce the graph in Figure 26.12.

```
proc sort data=monte;
   by yr;
run;

proc univariate data=monte noprint;
   by yr;
   var rate_jp rate_wg;
   output out=bounds mean=mean p5=p5 p95=p95;
run;

title "Monte Carlo Generated Confidence Intervals on a Forecast";
proc sgplot data=bounds noautolegend;
  series x=yr y=mean / markers;
  series x=yr y=p5 / markers;
  series x=yr y=p95 / markers;
run;
```

Figure 26.12 Monte Carlo Confidence Interval Plot



Syntax: MODEL Procedure

The following statements can be used with the MODEL procedure:

PROC MODEL *options* ;
ABORT ;
ARRAY *arrayname variable-list . . .* ;
ATTRIB *variable-list1 attribute-list1 < variable-list2 attribute-list2 . . . >* ;
BOUNDS *bound1 < , bound2 . . . >* ;
BY *variable-list* ;
CALL *name* ;
CALL *name(expression1 < , expression2 . . . >)* ;
CONTROL *variable < value > . . .* ;
DELETE ;
DO ;
DO *variable = expression < TO expression > < BY expression > < , expression TO expression < BY expression > . . . > < WHILE expression > < UNTIL expression >* ;
END ;
DROP *variable . . .* ;
ENDOGENOUS *variable < initial-values > . . .* ;
ERRORMODEL *equation-name ~ distribution < CDF=(CDF(options)) >* ;
ESTIMATE *item1 < , item2 . . . > < ,/ options >* ;
EXOGENOUS *variable < initial values > . . .* ;
FIT *equations < PARMS=(parameter values . . .) > < START=(parameter values . . .) > < DROP=(parameters) > < / options >* ;
FORMAT *variable-list < format > < DEFAULT= default-format >* ;
GOTO *statement-label* ;
ID *variable-list* ;
IF *expression* ;
IF *expression THEN programming-statement1 ; < ELSE programming-statement2 >* ;
variable *= expression* ;
variable *+ expression* ;
INCLUDE *model-file . . .* ;
INSTRUMENTS *< instruments > < _EXOG_ > < EXCLUDE=(parameters) > < / options >* ;
KEEP *variable . . .* ;
LABEL *variable ='label' . . .* ;
LENGTH *variable-list < $ > length . . . < DEFAULT=length >* ;
LINK *statement-label* ;
MOMENT *variable-list = moment-specification . . .* ;
OUTVARS *variable . . .* ;
PARAMETERS *variable1 < value1 > < variable2 < value2 . . . > >* ;
PUT *print-item . . . < @ > < @@ >* ;
RANGE *variable < = first > < TO last >* ;
RENAME *old-name1 = new-name1 < . . . old-name2 = new-name2 >* ;

RESET *options* ;
RESTRICT *restriction1 < , restriction2 . . . >* ;
RETAIN *variable-list1 value1 < variable-list2 value2 . . . >* ;
RETURN ;
SOLVE *variable-list < SATISFY=(equations) > < / options >* ;
SUBSTR *(variable, index, length) = expression* ;
SELECT *< (expression) >* ;
OTHERWISE *programming-statement* ;
STOP ;
TEST *< "name" > test1 < , test2 . . . > < ,/ options >* ;
VAR *variable < initial-values > . . .* ;
WEIGHT *variable* ;
WHEN *(expression) programming-statement* ;

Functional Summary

The statements and options in the MODEL procedure are summarized in the following table.

Description	Statement	Option
Data Set Options		
Specifies the input data set for the variables	FIT, SOLVE	DATA=
Specifies the input data set for parameters	FIT, SOLVE	ESTDATA=
Specifies the method for handling missing values	FIT	MISSING=
Specifies the input data set for parameters	MODEL	PARMSDATA=
Requests that the procedure produce graphics via the Output Delivery System	MODEL	PLOTS=
Specifies the output data set for residual, predicted, or actual values	FIT	OUT=
Specifies the output data set for solution mode results	SOLVE	OUT=
Writes the actual values to OUT= data set	FIT	OUTACTUAL
Selects all output options	FIT	OUTALL
Writes the covariance matrix of the estimates	FIT	OUTCOV
Writes the parameter estimates to a data set	FIT	OUTEST=
Writes the parameter estimates to a data set	MODEL	OUTPARMS=
Writes the observations used to start the lags	SOLVE	OUTLAGS
Writes the predicted values to the OUT= data set	FIT	OUTPREDICT
Writes the residual values to the OUT= data set	FIT	OUTRESID
Writes the covariance matrix of the equation errors to a data set	FIT	OUTS=
Writes the **S** matrix used in the objective function definition to a data set	FIT	OUTSUSED=

Description	Statement	Option
Writes the estimate of the variance matrix of the moment generating function	FIT	OUTV=
Reads the covariance matrix of the equation errors	FIT, SOLVE	SDATA=
Reads the covariance matrix for GMM and IT-GMM	FIT	VDATA=
Specifies the name of the time variable	FIT, SOLVE, MODEL	TIME=
Selects the estimation type to read	FIT, SOLVE	TYPE=
General ESTIMATE Statement Options		
Specifies the name of the data set in which the estimate of the functions of the parameters are to be written	ESTIMATE	OUTEST=
Writes the covariance matrix of the functions of the parameters to the OUTEST= data set	ESTIMATE	OUTCOV
Prints the covariance matrix of the functions of the parameters	ESTIMATE	COVB
Prints the correlation matrix of the functions of the parameters	ESTIMATE	CORRB
Printing Options for FIT Tasks		
Prints the modified Breusch-Pagan test for heteroscedasticity	FIT	BREUSCH
Prints the Chow test for structural breaks	FIT	CHOW=
Prints collinearity diagnostics	FIT	COLLIN
Prints the correlation matrices	FIT	CORR
Prints the correlation matrix of the parameters	FIT	CORRB
Prints the correlation matrix of the residuals	FIT	CORRS
Prints the covariance matrices	FIT	COV
Prints the covariance matrix of the parameters	FIT	COVB
Prints the covariance matrix of the residuals	FIT	COVS
Prints Durbin-Watson d statistics	FIT	DW
Prints first-stage R^2 statistics	FIT	FSRSQ
Prints Godfrey's tests for autocorrelated residuals for each equation	FIT	GODFREY
Prints Hausman's specification test	FIT	HAUSMAN
Prints tests of normality of the model residuals	FIT	NORMAL
Prints the predictive Chow test for structural breaks	FIT	PCHOW=
Specifies all the printing options	FIT	PRINTALL
Prints White's test for heteroscedasticity	FIT	WHITE

Description	Statement	Option
Options to Control FIT Iteration Output		
Prints the inverse of the crossproducts Jacobian matrix	FIT	I
Prints a summary iteration listing	FIT	ITPRINT
Prints a detailed iteration listing	FIT	ITDETAILS
Prints the crossproduct Jacobian matrix	FIT	XPX
Specifies all the iteration printing-control options	FIT	ITALL
Options to Control the Minimization Process		
Specifies the convergence criteria	FIT	CONVERGE=
Selects the Hessian approximation used for FIML	FIT	HESSIAN=
Specifies the local truncation error bound for the integration	FIT, SOLVE, MODEL	LTEBOUND=
Specifies the maximum number of iterations allowed	FIT	MAXITER=
Specifies the maximum number of subiterations allowed	FIT	MAXSUBITER=
Selects the iterative minimization method to use	FIT	METHOD=
Specifies the smallest allowed time step to be used in the integration	FIT, SOLVE, MODEL	MINTIMESTEP=
Modify the iterations for estimation methods that iterate the **S** matrix or the **V** matrix	FIT	NESTIT
Specifies the smallest pivot value	MODEL, FIT, SOLVE	SINGULAR
Specifies the number of minimization iterations to perform at each grid point	FIT	STARTITER=
Specifies a weight variable	WEIGHT	
Options to Read and Write Model Files		
Deletes a model from a model file	DELETEMODEL	MODNAME=
Reads a model from one or more input model files	INCLUDE	MODEL=
Suppresses the default output of the model file	MODEL, RESET	NOSTORE
Specifies the name of an output model file	MODEL, RESET	OUTMODEL=
Deletes the current model	RESET	PURGE
Options to List or Analyze the Structure of the Model		
Identifies equations in a dependency analysis	EQGROUP	
Identifies variables in a dependency analysis	VARGROUP	

Description	Statement	Option
Prints a dependency analysis of a simulation model	SOLVE	ANALYZEDEP=
Prints a dependency structure of a normal form model	MODEL	BLOCK
Prints a graph of the dependency structure of a normal form model	MODEL	GRAPH
Prints the model program and variable lists	MODEL	LIST
Prints the derivative tables and compiled model program code	MODEL	LISTCODE
Prints a dependency list	MODEL	LISTDEP
Prints a table of derivatives	MODEL	LISTDER
Prints a cross-reference of the variables	MODEL	XREF
General Printing Control Options		
Expands parts of the printed output	FIT, SOLVE	DETAILS
Prints a message for each statement as it is executed	FIT, SOLVE	FLOW
Selects the maximum number of execution errors that can be printed	FIT, SOLVE	MAXERRORS=
Requests a comprehensive memory usage summary	FIT, SOLVE, MODEL, RESET	MEMORYUSE
Selects the number of decimal places shown in the printed output	FIT, SOLVE	NDEC=
Suppresses the normal printed output	FIT, SOLVE	NOPRINT
Turns off the NOPRINT option	RESET	PRINT
Specifies all the noniteration printing options	FIT, SOLVE	PRINTALL
Prints tables which summarize missing value calculations	FIT, SOLVE, MODEL	REPORTMISSINGS
Prints the result of each operation as it is executed	FIT, SOLVE	TRACE
Statements that Declare Variables		
Associates a name with a list of variables and constants	ARRAY	
Declares a variable to have a fixed value	CONTROL	
Declares a variable to be a dependent or endogenous variable	ENDOGENOUS	
Declares a variable to be an independent or exogenous variable	EXOGENOUS	
Specifies identifying variables	ID	
Assigns a label to a variable	LABEL	
Selects additional variables to be output	OUTVARS	
Declares a variable to be a parameter	PARAMETERS	
Forces a variable to hold its value from a previous observation	RETAIN	

Description	Statement	Option
Declares a model variable	VAR	
Declares an instrumental variable	INSTRUMENTS	
Omits the default intercept term in the instruments list	INSTRUMENTS	NOINT
General FIT Statement Options		
Omits parameters from the estimation	FIT	DROP=
Associates a variable with an initial value as a parameter or a constant	FIT	INITIAL=
Bypasses OLS to get initial parameter estimates for GMM, ITGMM, or FIML	FIT	NOOLS
Bypasses 2SLS to get initial parameter estimates for GMM, ITGMM, or FIML	FIT	NO2SLS
Specifies the parameters to estimate	FIT	PARMS=
Requests confidence intervals on estimated parameters	FIT	PRL=
Selects a grid search	FIT	START=
Options to Control the Estimation Method Used		
Specifies nonlinear ordinary least squares	FIT	OLS
Specifies iterated nonlinear ordinary least squares	FIT	ITOLS
Specifies seemingly unrelated regression	FIT	SUR
Specifies iterated seemingly unrelated regression	FIT	ITSUR
Specifies two-stage least squares	FIT	2SLS
Specifies iterated two-stage least squares	FIT	IT2SLS
Specifies three-stage least squares	FIT	3SLS
Specifies iterated three-stage least squares	FIT	IT3SLS
Specifies full information maximum likelihood	FIT	FIML
Specifies simulated method of moments	FIT	NDRAW
Specifies number of draws for the V matrix	FIT	NDRAWV
Specifies number of initial observations for SMM	FIT	NPREOBS
Selects the variance-covariance estimator used for FIML	FIT	COVBEST=
Specifies generalized method of moments	FIT	GMM
Specifies the kernel for GMM and ITGMM	FIT	KERNEL=
Specifies iterated generalized method of moments	FIT	ITGMM
Specifies the type of generalized inverse used for the covariance matrix	FIT	GINV=
Specifies the denominator for computing variances and covariances	FIT	VARDEF=

Description	Statement	Option
Specifies adding the variance adjustment for SMM	FIT	ADJSMMV
Specifies variance correction for heteroscedasticity	FIT	HCCME=
Specifies GMM variance under arbitrary weighting matrix	FIT	GENGMMV
Specifies GMM variance under optimal weighting matrix	FIT	NOGENGMMV
Solution Mode Options		
Selects a subset of the model equations	SOLVE	SATISFY=
Solves only for missing variables	SOLVE	FORECAST
Solves for all solution variables	SOLVE	SIMULATE
Solution Mode Options: Lag Processing		
Uses solved values in the lag functions	SOLVE	DYNAMIC
Uses actual values in the lag functions	SOLVE	STATIC
Produces successive forecasts to a fixed forecast horizon	SOLVE	NAHEAD=
Selects the observation to start dynamic solutions	SOLVE	START=
Solution Mode Options: Numerical Methods		
Specifies the maximum number of iterations allowed	SOLVE	MAXITER=
Specifies the maximum number of subiterations allowed	SOLVE	MAXSUBITER=
Specifies the convergence criteria	SOLVE	CONVERGE=
Computes a simultaneous solution using a Jacobi-like iteration	SOLVE	JACOBI
Computes a simultaneous solution using a Gauss-Seidel-like iteration	SOLVE	SEIDEL
Computes a simultaneous solution using Newton's method	SOLVE	NEWTON
Computes a nonsimultaneous solution	SOLVE	SINGLE
Monte Carlo Simulation Options		
Specifies quasi-random number generator	SOLVE	QUASI=
Specifies pseudo-random number generator	SOLVE	PSUEDO=
Repeats the solution multiple times	SOLVE	RANDOM=
Initializes the pseudo-random number generator	SOLVE	SEED=
Specifies copula options	SOLVE	COPULA=

Description	Statement	Option
Solution Mode Printing Options		
Prints between data points integration values for the DERT. variables and the auxiliary variables	FIT, SOLVE, MODEL	INTGPRINT
Prints the solution approximation and equation errors	SOLVE	ITPRINT
Prints the solution values and residuals at each observation	SOLVE	SOLVEPRINT
Prints various summary statistics	SOLVE	STATS
Prints tables of Theil inequality coefficients	SOLVE	THEIL
Specifies all printing control options	SOLVE	PRINTALL
General TEST Statement Options		
Specifies that a Wald test be computed	TEST	WALD
Specifies that a Lagrange multiplier test be computed	TEST	LM
Specifies that a likelihood ratio test be computed	TEST	LR
Request all three types of tests	TEST	ALL
Specifies the name of an output SAS data set that contains the test results	TEST	OUT=
Miscellaneous Statements		
Specifies the range of observations to be used	RANGE	
Subsets the data set with BY variables	BY	

PROC MODEL Statement

PROC MODEL *options* **;**

The following options can be specified in the PROC MODEL statement. All of the nonassignment options (the options that do not accept a value after an equal sign) can have NO prefixed to the option name in the RESET statement to turn the option off. The default case is not explicitly indicated in the discussion that follows. Thus, for example, the option DETAILS is documented in the following, but NODETAILS is not documented since it is the default. Also, the NOSTORE option is documented because STORE is the default.

Data Set Options

DATA=*SAS-data-set*

names the input data set. Variables in the model program are looked up in the DATA= data set and, if found, their attributes (type, length, label, format) are set to be the same as those in the input data set (if not previously defined otherwise). The values for the variables in the program are read from the input data set when the model is estimated or simulated by FIT and SOLVE statements.

OUTPARMS=*SAS-data-set*
writes the parameter estimates to a SAS data set. See the section "Output Data Sets" on page 1632 for details.

PARMSDATA=*SAS-data-set*
names the SAS data set that contains the parameter estimates. In PROC MODEL, you have several options to specify starting values for the parameters to be estimated. When more than one option is specified, the options are implemented in the following order of precedence (from highest to lowest): the START= option, the PARMS statement initialization value, the ESTDATA= option, and the PARMSDATA= option. If no options are specified for the starting value, the default value of 0.0001 is used. See the section "Input Data Sets" on page 1627 for details.

PLOTS< *(global-plot-options)* > < *=(plot-request . . .)* >
selects plots that the MODEL procedure produces via the Output Delivery System. For general information about ODS Graphics, see Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*). The *global-plot-options* apply to all relevant plots generated by the MODEL procedure. The *global-plot-options* and specific *plot-request* options supported by the MODEL procedure follow.

Global Plot Options

ONLY suppresses the default plots. Only the plots specifically requested are produced.

UNPACKPANEL displays each graph separately. (By default, some graphs can appear together in a single panel.)

Specific Plot Options

ALL requests that all plots appropriate for the particular analysis be produced.

ACF produces the autocorrelation function plot.

DEPENDENCY< (OUTLINE=ON | OFF) > produces the dependency analysis plots. Specifying the OUTLINE= option displays, or suppresses outlines around the dependency cells.

IACF produces the inverse autocorrelation function plot of residuals.

PACF produces the partial autocorrelation function plot of residuals.

FITPLOT plots the predicted and actual values.

COOKSD produces the Cook's D plot.

QQ produces a QQ plot of residuals.

RESIDUAL | RES plots the residuals.

STUDENTRESIDUAL plots the studentized residuals.

RESIDUALHISTOGRAM | RESIDHISTOGRAM plots the histogram of residuals.

NONE suppresses all plots.

Options to Read and Write Model Files

MODEL=*model-name*

MODEL=*(model-list)*

reads the model from one or more input model files that were created by specifying the OUTMODEL= option in previous PROC MODEL executions.

NOSTORE

suppresses the default output of the model file. This option is applicable only when FIT or SOLVE statements are not used, the MODEL= option is not used, and when a model is specified.

OUTCAT=(*outcat-name* **MODNAME=***model-key* **<** *outcat-options* **>)**

SLIST=(*outcat-name* **MODNAME=***model-key* **<** *outcat-options* **>)**

specifies the name and *model-key* for writing fitted model files. The *model-key* is a SAS name. Files written using the OUTCAT= option are used by SAS Risk Dimensions. The OUTCAT= option only applies to FIT statements. You can specify the following *outcat-options*:

DIM=*n* specifies the dimensionality of the model.

GROUP|MODGROUP=*group* specifies a SAS name which is the group for the model.

INTERVAL=*interval* specifies the time interval between observations.

MODLABEL=*label* specifies a label for the model.

STARTDATE=*date* specifies the starting date of the model.

OUTMODEL=*model-name*

specifies the name of an output model file to which the model is to be written. Starting with SAS 9.2, model files are being stored as XML-based SAS data sets instead of being stored as members of a SAS catalog as in earlier releases. This makes MODEL files more readily extendable in the future and enables Java-based applications to read the MODEL files directly. To change this behavior, use the SAS *global-CMPMODEL-options*. You can choose the format in which the output model file is stored and read by using the CMPMODEL=*global-CMPMODEL-options* in an OPTIONS statement as follows.

OPTIONS CMPMODEL=*global-CMPMODEL-options***;**

You can specify the following *global-CMPMODEL-options*:

CATALOG specifies that model files be written and read from SAS catalogs only.

XML specifies that model files be written and read from XML data sets only.

BOTH specifies that model files be written to both XML and CATALOG formats. When BOTH is specified, model files are read from the data set first and read from the SAS catalog only if the data set is not found. This is the default.

Options to List or Analyze the Structure of the Model

These options produce reports on the structure of the model or list the programming statements that define the models. These options are automatically reset (turned off) after the reports are printed. To turn these options back on after a RUN statement has been entered, use the RESET statement or specify the options in a FIT or SOLVE statement.

ANALYZEDEP=(*dependency-plot1* < *dependency-plot2 . . .* >**)**
plots analyses of the dependencies among equations and solve variables. Each *dependency-plot* is one of the following:

BLOCK	specifies a block dependency matrix of the entire system.
BLOCK(*eq-list*,*var-list*)	specifies a block dependency matrix for a subset of equations and solve variables.
DETAILS	specifies a dependency matrix of all equations and solve variables.
DETAILS(*eq-list*,*var-list*)	specifies a dependency matrix for a subset of equations and solve variables.
NOLISTBLOCK	suppresses the listing of dependency blocks.

You can specify which equations and solve variables are included in the dependency analysis by qualifying both the BLOCK and DETAILS *dependency-plot* options with a pair of lists. The first list in the pair is the *eq-list*. It specifies which equations to include in the dependency analysis. You can specify a mix of equation names and equation group labels in the *eq-list*. The MODEL procedure replaces each equation group label in the *eq-list* with the list of equations that are specified in the corresponding EQGROUP statement. The second list in the pair is the *var-list*. It specifies which solve variables to include in the dependency analysis. You can specify a mix of variable names and variable group labels in the *var-list*. The MODEL procedure replaces each variable group label in the *var-list* with the list of variables that are specified in the corresponding VARGROUP statement. By default, when you specify a BLOCK option, a listing of the equations and solve variables that form each dependency block is generated. The NOLISTBLOCK option suppresses this listing. The ANALYZEDEP= option applies only to SOLVE steps. For more information about the analyses that are performed by the ANALYZEDEP= option, see the section "Diagnostics and Debugging" on page 1689.

BLOCK
prints an analysis of the structure of the model given by the assignments to model variables that appear in the model program. This analysis includes a classification of model variables into endogenous (dependent) and exogenous (independent) groups based on the presence of the variable on the left side of an assignment statement. The endogenous variables are grouped into simultaneously determined blocks. The dependency structure of the simultaneous blocks and exogenous variables is also printed. The BLOCK option cannot analyze dependencies implied by general form equations.

GRAPH
prints the graph of the dependency structure of the model. The GRAPH option also invokes the BLOCK option and produces a graphical display of the information listed by the BLOCK option.

LIST
prints the model program and variable lists, including the statements added by PROC MODEL and macros.

LISTALL
selects the LIST, LISTDEP, LISTDER, and LISTCODE options.

LISTCODE
prints the derivative tables and compiled model program code. LISTCODE is a debugging feature and is not normally needed.

LISTDEP
prints a report that lists for each variable in the model program the variables that depend on it and that it depends on. These lists are given separately for current-period values and for lagged values of the variables.

The information displayed is the same as that used to construct the BLOCK report but differs in that the information is listed for all variables (including parameters, control variables, and program variables), not just for the model variables. Classification into endogenous and exogenous groups and analysis of simultaneous structure is not done by the LISTDEP report.

LISTDER
prints a table of derivatives for FIT and SOLVE tasks. (The LISTDER option is applicable only for the default NEWTON method for SOLVE tasks.) The derivatives table shows each nonzero derivative computed for the problem. The derivative listed can be a constant, a variable in the model program, or a special derivative variable created to hold the result of the derivative expression. This option is turned on by the LISTCODE and PRINTALL options.

XREF
prints a cross-reference of the variables in the model program that shows where each variable was referenced or given a value. The XREF option is normally used in conjunction with the LIST option. A more detailed description is given in the section "Diagnostics and Debugging" on page 1689.

General Printing Control Options

DETAILS
specifies the detailed printout. Parts of the printed output are expanded when the DETAILS option is specified. The following additional graphs of the residuals are produced when graphics output is enabled: ACF, PACF, IACF, white noise, and QQ plot versus the normal.

FLOW
prints a message for each statement in the model program as it is executed. This debugging option is needed very rarely and produces voluminous output.

MAXERRORS=*n*
specifies the maximum number of execution errors that can be printed. The default is MAXERRORS=50.

MEMORYUSE
prints a report of the memory required for the various parts of the analysis.

NDEC=*n*
specifies the precision of the format that PROC MODEL uses when printing various numbers. The default is NDEC=3, which means that PROC MODEL attempts to print values by using the D format

but ensures that at least three significant digits are shown. If the NDEC= value is greater than nine, the BEST. format is used. The smallest value allowed is NDEC=2.

The NDEC= option affects the format of most, but not all, of the floating point numbers that PROC MODEL can print. For some values (such as parameter estimates), a precision limit one or two digits greater than the NDEC= value is used. This option does not apply to the precision of the variables in the output data set.

NOPRINT
suppresses the normal printed output but does not suppress error listings. Using any other print option turns the NOPRINT option off. The PRINT option can be used with the RESET statement to turn off NOPRINT.

PRINTALL
turns on all the printing-control options. The options set by PRINTALL are DETAILS; the model information options LIST, LISTDEP, LISTDER, XREF, BLOCK, and GRAPH; the FIT task printing options FSRSQ, COVB, CORRB, COVS, CORRS, DW, and COLLIN; and the SOLVE task printing options STATS, THEIL, SOLVEPRINT, and ITPRINT.

REPORTMISSINGS
prints tables that summarize missing values that are encountered during a SOLVE or FIT task. The missing values that are summarized in these tabular reports can be produced by missing values in the DATA= data set or by calculations in the model program that generate missing values. The number of missing values that are reported can be limited by using the MAXERRORS= option.

TRACE
prints the result of each operation in each statement in the model program as it is executed, in addition to the information printed by the FLOW option. This debugging option is needed very rarely and produces voluminous output.

FIT Task Options

The following options are used in the FIT statement (parameter estimation) and can also be used in the PROC MODEL statement: COLLIN, CONVERGE=, CORR, CORRB, CORRS, COVB, COVBEST=, COVS, DW, FIML, FSRSQ, GMM, HESSIAN=, I, INTGPRINT, ITALL, ITDETAILS, ITGMM, ITPRINT, ITOLS, ITSUR, IT2SLS, IT3SLS, KERNEL=, LTEBOUND=, MAXITER=, MAXSUBITER=, METHOD=, MINTIMESTEP=, NESTIT, N2SLS, N3SLS, OLS, OUTPREDICT, OUTRESID, OUTACTUAL, OUTLAGS, OUTALL, OUTCOV, SINGULAR=, STARTITER=, SUR, TIME=, VARDEF, and XPX. See the section "FIT Statement" on page 1511 for a description of these options.

When used in the PROC MODEL or RESET statement, these are default options for subsequent FIT statements. For example, the statement

```
proc model n2sls ... ;
```

makes two-stage least squares the default parameter estimation method for FIT statements that do not specify an estimation method.

SOLVE Task Options

The following options for the SOLVE statement can also be used in the PROC MODEL statement: CONVERGE=, DYNAMIC, FORECAST, INTGPRINT, ITPRINT, JACOBI, LTEBOUND=, MAXITER=, MAXSUBITER=, MINTIMESTEP=, NAHEAD=, NEWTON, OUTPREDICT, OUTRESID, OUTACTUAL, OUTLAGS, OUTERRORS, OUTALL, SEED=, SEIDEL, SIMULATE, SINGLE, SINGULAR=, SOLVEPRINT, START=, STATIC, STATS, THEIL, TIME=, and TYPE=. For more information about these options, see section "SOLVE Statement" on page 1527.

When used in the PROC MODEL or RESET statement, these options provide default values for subsequent SOLVE statements.

BOUNDS Statement

BOUNDS *bound1* < , *bound2* . . . > **;**

The BOUNDS statement imposes simple boundary constraints either on the parameters in an estimation or on the solution variables specified in a solve operation. A BOUNDS statement that applies to parameters constrains the parameters estimated in the preceding FIT statement or, in the absence of a preceding FIT statement, in the following FIT statement. A BOUNDS statement that is applied to solution variables constrains the solution of the preceding SOLVE statement or, in the absence of a preceding SOLVE statement, of the following SOLVE statement. You can specify any number of BOUNDS statements.

Each *bound* is composed of either parameters or solution variables, constants, and inequality operators:

item operator item **<** *operator item* **<** *operator item* **. . . > >**

For BOUNDS statements that apply to FIT statements, each *item* is a constant, the name of an estimated parameter, or a list of parameter names. For BOUNDS statements that apply to SOLVE statements, each *item* is a constant, the name of a solution variable, or a list of solution variables. Each *operator* is <, >, <=, or >=.

You can use either the BOUNDS statement or the RESTRICT statement to impose boundary constraints when estimating parameters or solving for solution variables.

The BOUNDS statement provides a simpler syntax for specifying boundary constraints than the RESTRICT statement. For more information about the computational details of estimation and solutions with inequality restrictions, see the section "RESTRICT Statement" on page 1525.

Parameter Estimates

Each active boundary constraint on estimated parameters is associated with a Lagrange multiplier. In the printed output and in the OUTEST= data set, the Lagrange multiplier estimates are identified with the names BOUND0, BOUND1, and so forth. The probabilities of the Lagrange multipliers are computed by using a beta distribution (LaMotte 1994). To give the constraints more descriptive names, use the RESTRICT statement instead of the BOUNDS statement.

The following BOUNDS statement constrains the estimates of the parameters A and B and the ten parameters P1 through P10 to be between 0 and 1. This example illustrates the use of parameter lists to specify boundary constraints.

```
bounds 0 < a b p1-p10 < 1;
```

The following statements show how to use the BOUNDS statement, and they produce the output shown in Figure 26.13:

```
title 'Holzman Function (1969), Himmelblau No. 21, N=3';
data zero;
   do i = 1 to 99;
      output;
   end;
run;

proc model data=zero;
   parms x1= 100 x2= 12.5 x3=  3;
   bounds .1 <= x1 <= 100,
           0 <= x2 <=  25.6,
           0 <= x3 <=   5;

   t = 2 / 3;
   u = 25 + (-50 * log(0.01 * i )) ** t;
   v = (u - x2) ** x3;
   w = exp(-v / x1);
   eq.foo = -.01 * i + w;

   fit foo / method=marquardt;
run;
```

Figure 26.13 Output from Bounded Estimation

Holzman Function (1969), Himmelblau No. 21, N=3

The MODEL Procedure

Nonlinear OLS Parameter Estimates

Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|
x1	49.99999	0	.	.
x2	25	0	.	.
x3	1.5	0	.	.

Solution Variables

Boundary constraints on solution variables can be used to specify which solution is reported when an equation has multiple solutions. The BOUNDS statement in the following example causes its associated SOLVE statement to compute only the negative value of the solution variable shown in Figure 26.14:

```
data d;
   date = 0;
run;

proc model data=d;
   endo x;
```

```
   bounds x < 0;

   eq.sqrt = x**2 - 4;

   solve / optimize out=o;
run;

proc print data = o; run;
```

Figure 26.14 Listing of OUT= Data Set Created by a Bounded SOLVE Statement

Obs	_TYPE_	_MODE_	_ERRORS_	x
1	PREDICT	SIMULATE	0	-2

BY Statement

BY *variables* **;**

A BY statement is used with the FIT statement to obtain separate estimates for observations in groups defined by the BY variables. If an output model file is written using the OUTMODEL= option, the parameter values that are stored are those from the last BY group processed. To save parameter estimates for each BY group, use the OUTEST= option in the FIT statement.

A BY statement is used with the SOLVE statement to obtain solutions for observations in groups defined by the BY variables. If the BY variables in the DATA= data set and the ESTDATA= data set are identical, then the two data sets are synchronized and the calculations are performed by using the data and parameters for each BY group. This holds for BY variables in the SDATA= data set as well. If the BY variables do not match, BY-group processing is abandoned in either the ESTDATA= data set or the SDATA= data set, whichever has the missing BY value. If the DATA= data set does not contain BY variables and the ESTDATA= data set or the SDATA= data set does, then BY-group processing is performed for the ESTDATA= data set and the SDATA= data set by reusing the data in the DATA= data set for each BY group.

If both FIT and SOLVE tasks require BY-group processing, then two separate BY statements are needed. If parameters for each BY group in the OUTEST = data set that is obtained from the FIT task are to be used for the corresponding BY group for the SOLVE task, then one of the two BY statements must appear after the SOLVE statement.

The following linear regression example illustrates the use of BY-group processing. Both the data sets A and D to be used for fitting and solving, respectively, have three groups.

```
/*------ data set for fit task------ */
data a ;
   do group = 1 to 3 ;
      do i = 1 to 100 ;
         x = normal(1);
         y = 2 + 3*x + rannor(1) ;
         output ;
      end ;
   end ;
run ;
```

```
/*------ data set for solve task------ */
data d ;
   do group = 1 to 3 ;
      x = normal(1) ;
      output ;
   end ;
run ;

/* ------  2 BY statements, one of them appear after SOLVE statement ------ */
proc model data = a ;
   by group ;
   y = a0 + a1*x ;
   fit y / outest = b1 ;
   solve y / data = d estdata = b1 out = c1 ;
   by group ;
run;

proc print data = b1 ;run;
proc print data = c1 ; run;
```

Each of the parameter estimates obtained from the BY group processing in the FIT statement shown in Figure 26.15 is used in the corresponding BY group variables in the SOLVE statement. The output dataset is shown in Figure 26.16.

Figure 26.15 Listing of OUTEST= Data Set Created in the FIT Statement with Two BY Statements

Obs	group	_NAME_	_TYPE_	_STATUS_	_NUSED_	a0	a1
1	1		OLS	0 Converged	100	2.00338	3.00298
2	2		OLS	0 Converged	100	2.05091	3.08808
3	3		OLS	0 Converged	100	2.15528	3.04290

Figure 26.16 Listing of OUT= Data Set Created in the SOLVE Statement with Two BY Statements

Obs	group	_TYPE_	_MODE_	_ERRORS_	y	x
1	1	PREDICT	SIMULATE	0	7.42322	1.80482
2	2	PREDICT	SIMULATE	0	1.80413	-0.07992
3	3	PREDICT	SIMULATE	0	3.36202	0.39658

If only one BY statement is used and it appears before the SOLVE statement, then parameters for the last BY group in the OUTEST = data set are used for all BY groups for the SOLVE task.

```
/*------ 1 BY statement that appears before SOLVE statement------ */
proc model data = a ;
   by group ;
   y = a0 + a1*x ;
   fit y / outest = b2 ;
   solve y / data = d estdata = b2 out = c2 ;
run;

proc print data = b2 ; run;
proc print data = c2 ; run;
```

The estimates of the parameters are shown in Figure 26.17, and the output data set of the SOLVE statement is shown in Figure 26.18. Hence, the estimates and the predicted values obtained in the last BY group variable of both DATA C1 and C2 are the same while the others do not match.

Figure 26.17 Listing of OUTEST= Data Set Created in the FIT Statement with One BY Statement That Appears before the SOLVE Statement

Obs	group	_NAME_	_TYPE_	_STATUS_	_NUSED_	a0	a1
1	1		OLS	0 Converged	100	2.00338	3.00298
2	2		OLS	0 Converged	100	2.05091	3.08808
3	3		OLS	0 Converged	100	2.15528	3.04290

Figure 26.18 Listing of OUT= Data Set Created in the SOLVE Statement with One BY Statement That Appears before the SOLVE Statement

Obs	_TYPE_	_MODE_	_ERRORS_	y	x
1	PREDICT	SIMULATE	0	7.64717	1.80482
2	PREDICT	SIMULATE	0	1.91211	-0.07992
3	PREDICT	SIMULATE	0	3.36202	0.39658

If only one BY statement is used and it appears after the SOLVE statement, then BY group processing does not apply to the FIT task. In this case, the OUTEST=data set does not contain the BY variable, and the single set of parameter estimates obtained from the FIT task are used for all BY groups during the SOLVE task.

```
/*------ 1 BY statement that appears after SOLVE statement------*/
proc model data = a ;
   y = a0 + a1*x ;
   fit y / outest = b3 ;
   solve y / data = d estdata = b3 out = c3 ;
   by group ;
run;

proc print data = b3 ; run;
proc print data = c3 ; run;
```

The output data B3 and C3 are listed in Figure 26.19 and Figure 26.20, respectively.

Figure 26.19 Listing of OUTEST= Data Set Created in the FIT Statement with One BY Statement That Appears after the SOLVE Statement

Obs	_NAME_	_TYPE_	_STATUS_	_NUSED_	a0	a1
1		OLS	0 Converged	300	2.06624	3.04219

Figure 26.20 Listing of OUT= Data Set Created in the First SOLVE Statement with One BY Statement That Appears after the SOLVE Statement

Obs	group	_TYPE_	_MODE_	_ERRORS_	y	x
1	1	PREDICT	SIMULATE	0	7.55686	1.80482
2	2	PREDICT	SIMULATE	0	1.82312	-0.07992
3	3	PREDICT	SIMULATE	0	3.27270	0.39658

CONTROL Statement

CONTROL *variable* < *value* > . . . **;**

The CONTROL statement declares control variables and specifies their values. A control variable is like a parameter except that it has a fixed value and is not estimated from the data. You can use control variables for constants in model equations that you might want to change in different solution cases. You can use control variables to vary the program logic. Unlike the retained variables, these values are fixed across iterations.

DELETEMODEL Statement

DELETEMODEL *model* < *MODNAME=model-name* > **;**

The DELETEMODEL statement deletes a model created using the OUTMODEL= option in a previous PROC MODEL execution. The *model* argument specifies the catalog or XML-based data set containing the model to be deleted, and the *model-name* argument specifies which model is to be deleted.

ENDOGENOUS Statement

ENDOGENOUS *variable* < *initial-values* > . . . **;**

The ENDOGENOUS statement declares model variables and identifies them as endogenous. You can declare model variables with an ENDOGENOUS statement instead of with a VAR statement to help document the model or to indicate the default solution variables. The variables declared endogenous are solved when a SOLVE statement does not indicate which variables to solve. Valid abbreviations for the ENDOGENOUS statement are ENDOG and ENDO.

The DEPENDENT statement is equivalent to the ENDOGENOUS statement and is provided for the convenience of noneconometric practitioners.

The ENDOGENOUS statement optionally provides initial values for lagged dependent variables. See the section "Lag Logic" on page 1682 for more information.

EQGROUP Statement

EQGROUP *label=equation. . .* **;**

The EQGROUP statement applies a group label to the specified list of equations in the model program. Equation groups identify sets of related equations. The equation groups can be used by the ANALYZEDEP= option in a subsequent SOLVE statement to help specify and understand the role of groups of equations in a SOLVE step. If an equation appears in more than one EQGROUP statement, the label that is specified in the last EQGROUP statement is applied to that equation.

ERRORMODEL Statement

ERRORMODEL *equation-name* ∼ *distribution* < **CDF=** *CDF(options)* > **;**

The ERRORMODEL statement is the mechanism for specifying the distribution of the residuals. You must specify the dependent/endogenous variables or general form model name, a tilde (∼), and then a *distribution* with its parameters. You can specify the following options:

Options to Specify the Distribution

CAUCHY(< *location, scale* **>)**
specifies the Cauchy distribution. This option is supported only for simulation. The arguments correspond to the arguments of the SAS CDF function that computes the cumulative distribution function (ignoring the random variable argument).

CHISQUARED (*df* **<** *, nc* **>)**
specifies the χ^2 distribution. This option is supported only for simulation. The arguments correspond to the arguments of the SAS CDF function (ignoring the random variable argument).

GENERAL(*Likelihood* **<** *, parm1, parm2, . . . parmn* **>)**
specifies the negative of a general log-likelihood function that you construct by using SAS programming statements. The procedure minimizes the negative log-likelihood function specified. *parm1*, *parm2*, . . . *parmn* are optional parameters for this distribution and are used for documentation purposes only.

F(*ndf, ddf* **<** *, nc* **>)**
specifies the F distribution. This option is supported only for simulation. The arguments correspond to the arguments of the SAS CDF function (ignoring the random variable argument).

NORMAL(v_1 v_2 $\ldots$ v_n **)**
specifies a multivariate normal (Gaussian) distribution with mean *0* and variances v_1 through v_n.

POISSON(*mean* **)**
specifies the Poisson distribution. This option is supported only for simulation. The arguments correspond to the arguments of the SAS CDF function (ignoring the random variable argument).

T($v_1 v_2 \cdots v_n$, *df* **)**
specifies a multivariate *t* distribution with noncentrality *0*, variance v_1 through v_n, and common degrees of freedom df.

UNIFORM(< *left, right* **>)**
specifies the uniform distribution. This option is supported only for simulation. The arguments correspond to the arguments of the SAS CDF function (ignoring the random variable argument).

Options to Specify the CDF for Simulation

CDF=(*CDF(options)* **)**
specifies the univariate distribution that is used for simulation so that the estimation can be done for one set of distributional assumptions and the simulation for another. The *CDF* can be any of the distributions from the previous section with the exception of the general likelihood. In addition, you can specify the empirical distribution of the residuals.

EMPIRICAL= (< TAILS=*(options)* >)
uses the sorted residual data to create an empirical CDF.

TAILS=(*tail-options* **)**
specifies how to handle the tails in computing the inverse CDF from an empirical distribution, where *tail-options* are:

NORMAL	specifies the normal distribution to extrapolate the tails.
T(*df*)	specifies the *t* distribution to extrapolate the tails.
PERCENT= *p*	specifies the percentage of the observations to use in constructing each tail. The default for the PERCENT= option is 10. A normal distribution or a *t* distribution is used to extrapolate the tails to infinity. The variance for the tail distribution is obtained from the data so that the empirical CDF is continuous.

ESTIMATE Statement

ESTIMATE *item* < *, item . . .* > < *,/ options* > **;**

The ESTIMATE statement computes estimates of functions of the parameters.

The ESTIMATE statement refers to the parameters estimated by the associated FIT statement (that is, to either the preceding FIT statement or, in the absence of a preceding FIT statement, to the following FIT statement). You can use any number of ESTIMATE statements.

Let $\mathbf{h}(\theta)$ denote the function of parameters that needs to be estimated. Let $\hat{\theta}$ denote the unconstrained estimate of the parameter of interest, θ. Let $\hat{\mathbf{V}}$ be the estimate of the covariance matrix of θ. Denote

$$\mathbf{A}(\theta) = \partial h(\theta)/\partial\theta \mid_{\hat{\theta}}$$

Then the standard error of the parameter function estimate is computed by obtaining the square root of $\mathbf{A}(\hat{\theta})\hat{\mathbf{V}}\mathbf{A}'(\hat{\theta})$. This is the same as the variance needed for a Wald type test statistic with null hypothesis $h(\theta) = 0$.

If the expression of the function in the ESTIMATE statement includes a variable, then the value used in computing the function estimate is the last observation of the variable in the DATA= data set.

If you specify options on the ESTIMATE statement, a comma is required before the "/" character that separates the test expressions from the options, since the "/" character can also be used within test expressions to indicate division. Each *item* is written as an optional name followed by an expression,

< "name" > expression

where *"name"* is a string used to identify the estimate in the printed output and in the OUTEST= data set.

Expressions can be composed of parameter names, arithmetic operators, functions, and constants. Comparison operators (such as = or <) and logical operators (such as &) cannot be used in ESTIMATE statement expressions. Parameters named in ESTIMATE expressions must be among the parameters estimated by the associated FIT statement.

You can use the following options in the ESTIMATE statement:

OUTEST=
specifies the name of the data set in which the estimate of the functions of the parameters are to be written. The format for this data set is identical to the OUTEST= data set for the FIT statement.

If you specify a *name* in the ESTIMATE statement, that name is used as the parameter name for the estimate in the OUTEST= data set. If no *name* is provided and the expression is just a symbol, the symbol name is used; otherwise, the string "_Estimate #" is used, where "#" is the variable number in the OUTEST= data set.

OUTCOV
writes the covariance matrix of the functions of the parameters to the OUTEST= data set in addition to the parameter estimates.

COVB
prints the covariance matrix of the functions of the parameters.

CORRB
prints the correlation matrix of the functions of the parameters.

The following statements are an example of the use of the ESTIMATE statement in a segmented model and produce the output shown in Figure 26.21:

```
data a;
   input y x @@;
datalines;
   .46 1  .47  2 .57  3 .61  4 .62  5 .68  6 .69  7
   .78 8  .70  9 .74 10 .77 11 .78 12 .74 13 .80 13
   .80 15 .78 16
;

title 'Segmented Model -- Quadratic with Plateau';
proc model data=a;

   x0 = -.5 * b / c;

   if x < x0 then y = a + b*x + c*x*x;
   else           y = a + b*x0 + c*x0*x0;

   fit y start=( a .45 b .5 c -.0025 );

   estimate 'Join point' x0 ,
            'plateau' a + b*x0 + c*x0**2 ;
run;
```

Figure 26.21 ESTIMATE Statement Output

Segmented Model -- Quadratic with Plateau

The MODEL Procedure

Nonlinear OLS Estimates

Term	Estimate	Approx Std Err	t Value	Approx Pr > |t|	Label
Join point	12.7504	1.2785	9.97	<.0001	x0
plateau	0.777516	0.0123	63.10	<.0001	a + b*x0 + c*x0**2

EXOGENOUS Statement

EXOGENOUS *variable* < *initial-values* > . . . **;**

The EXOGENOUS statement declares model variables and identifies them as exogenous. You can declare model variables with an EXOGENOUS statement instead of with a VAR statement to help document the model or to indicate the default instrumental variables. The variables declared exogenous are used as instruments when an instrumental variables estimation method is requested (such as N2SLS or N3SLS) and an INSTRUMENTS statement is not used. Valid abbreviations for the EXOGENOUS statement are EXOG and EXO.

The INDEPENDENT statement is equivalent to the EXOGENOUS statement and is provided for the convenience of non-econometric practitioners.

The EXOGENOUS statement optionally provides initial values for lagged exogenous variables. See the section "Lag Logic" on page 1682 for more information.

FIT Statement

FIT < *equations* > < *PARMS=(parameter* < *values* > . . .) > < *START=(parameter values . . .)* > < *DROP=(parameter . . .)* > < *INITIAL=(variable* < = *parameter* | *constant* > . . .) > < */ options* > **;**

The FIT statement estimates model parameters by fitting the model equations to input data and optionally selects the equations to be fit. If the list of equations is omitted, all model equations that contain parameters are fitted.

The following options can be used in the FIT statement.

DROP= *(parameters . . .)*

specifies that the named parameters not be estimated. All the parameters in the equations fit are estimated except those listed in the DROP= option. The dropped parameters retain their previous values and are not changed by the estimation.

INITIAL= *(variable =* < *parameter* | *constant* > . . .)

associates a *variable* with an initial value as a *parameter* or a *constant*. This option applies only to ordinary differential equations. See the section "Ordinary Differential Equations" on page 1590 for more information.

PARMS= *(parameters [values] . . .)*
selects a subset of the parameters for estimation. When the PARMS= option is used, only the named parameters are estimated. Any parameters not specified in the PARMS= list retain their previous values and are not changed by the estimation.

In PROC MODEL, you have several options to specify starting values for the parameters to be estimated. When more than one option is specified, the options are implemented in the following order of precedence (from highest to lowest): the START= option, the PARMS statement initialization value, the ESTDATA= option, and the PARMSDATA= option. If no options are specified for the starting value, the default value of 0.0001 is used.

PRL= WALD | LR | BOTH
requests confidence intervals on estimated parameters. By default, the PRL option produces 95% likelihood ratio confidence limits. The coverage of the confidence interval is controlled by the ALPHA= option in the FIT statement.

START= *(parameter values . . .)*
supplies starting values for the parameter estimates. In PROC MODEL, you have several options to specify starting values for the parameters to be estimated. When more than one option is specified, the options are implemented in the following order of precedence (from highest to lowest): the START= option, the PARMS statement initialization value, the ESTDATA= option, and the PARMSDATA= option. If no options are specified for the starting value, the default value of 0.0001 is used. If the START= option specifies more than one starting value for one or more parameters, a grid search is performed over all combinations of the values, and the best combination is used to start the iterations. For more information, see the STARTITER= option.

Options to Control the Estimation Method Used

ADJSMMV
specifies adding the variance adjustment from simulating the moments to the variance-covariance matrix of the parameter estimators. By default, no adjustment is made.

COVBEST=GLS | CROSS | FDA
specifies the variance-covariance estimator used for FIML. COVBEST=GLS selects the generalized least squares estimator. COVBEST=CROSS selects the crossproducts estimator. COVBEST=FDA selects the inverse of the finite difference approximation to the Hessian. The default is COVBEST=CROSS.

DYNAMIC
specifies dynamic estimation of ordinary differential equations. See the section "Ordinary Differential Equations" on page 1590 for more details.

FIML
specifies full information maximum likelihood estimation.

GINV=G2 | G4
specifies the type of generalized inverse to be used when computing the covariance matrix. G4 selects the Moore-Penrose generalized inverse. The default is GINV=G2.

Rather than deleting linearly related rows and columns of the covariance matrix, the Moore-Penrose generalized inverse averages the variance effects between collinear rows. When the option GINV=G4

is used, the Moore-Penrose generalized inverse is used to calculate standard errors and the covariance matrix of the parameters as well as the change vector for the optimization problem. For singular systems, a normal G2 inverse is used to determine the singular rows so that the parameters can be marked in the parameter estimates table. A G2 inverse is calculated by satisfying the first two properties of the Moore-Penrose generalized inverse; that is, $\mathbf{A}\mathbf{A}^{+}\mathbf{A} = \mathbf{A}$ and $\mathbf{A}^{+}\mathbf{A}\mathbf{A}^{+} = \mathbf{A}^{+}$. Whether or not you use a G4 inverse, if the covariance matrix is singular, the parameter estimates are not unique. Refer to Noble and Daniel (1977, pp. 337–340) for more details about generalized inverses.

GENGMMV

specify GMM variance under arbitrary weighting matrix. See the section "Estimation Methods" on page 1535 for more details.

This is the default method for GMM estimation.

GMM

specifies generalized method of moments estimation.

HCCME= 0 | 1 | 2 | 3 | NO

specifies the type of heteroscedasticity-consistent covariance matrix estimator to use for OLS, 2SLS, 3SLS, SUR, and the iterated versions of these estimation methods. The number corresponds to the type of covariance matrix estimator to use as

$$
\begin{aligned}
HC_0: &\quad \hat{\epsilon}_t^2 \\
HC_1: &\quad \frac{n}{n-df}\hat{\epsilon}_t^2 \\
HC_2: &\quad \hat{\epsilon}_t^2/(1-\hat{h}_t) \\
HC_3: &\quad \hat{\epsilon}_t^2/(1-\hat{h}_t)^2
\end{aligned}
$$

The default is NO.

ITGMM

specifies iterated generalized method of moments estimation.

ITOLS

specifies iterated ordinary least squares estimation. This is the same as OLS unless there are cross-equation parameter restrictions.

ITSUR

specifies iterated seemingly unrelated regression estimation

IT2SLS

specifies iterated two-stage least squares estimation. This is the same as 2SLS unless there are cross-equation parameter restrictions.

IT3SLS

specifies iterated three-stage least squares estimation.

KERNEL=(PARZEN | BART | QS, < c > , < e >)

KERNEL=PARZEN | BART | QS

specifies the kernel to be used for GMM and ITGMM. PARZEN selects the Parzen kernel, BART selects the Bartlett kernel, and QS selects the quadratic spectral kernel. $e \geq 0$ and $c \geq 0$ are used to compute the bandwidth parameter. The default is KERNEL=(PARZEN, 1, 0.2). See the section "Estimation Methods" on page 1535 for more details.

N2SLS | 2SLS
specifies nonlinear two-stage least squares estimation. This is the default when an INSTRUMENTS statement is used.

N3SLS | 3SLS
specifies nonlinear three-stage least squares estimation.

NDRAW <=*number of draws***>**
requests the simulation method for parameter estimation where the contribution of each observation to the estimation is approximated by using *number of draws* evaluations of the model program. If *number of draws* is not specified, the default value of 10 is used.

NOOLS

NO2SLS
specifies bypassing OLS or 2SLS to get initial parameter estimates for GMM, ITGMM, or FIML. This is important for certain models that are poorly defined in OLS or 2SLS, or if good initial parameter values are already provided. Note that for GMM, the **V** matrix is created by using the initial values specified and this might not be consistently estimated.

NO3SLS
specifies not to use 3SLS automatically for FIML initial parameter starting values.

NOGENGMMV
specifies not to use GMM variance under arbitrary weighting matrix. Use GMM variance under optimal weighting matrix instead. See the section "Estimation Methods" on page 1535 for more details.

NPREOBS =*number of obs to initialize*
specifies the initial number of observations to run the simulation before the simulated values are compared to observed variables. This option is most useful in cases where the program statements involve lag operations. Use this option to avoid the effect of the starting point on the simulation.

NVDRAW =*number of draws for V matrix*
specifies H', the number of draws for V matrix. If this option is not specified, the default H' is set to 20.

OLS
specifies ordinary least squares estimation. This is the default.

SUR
specifies seemingly unrelated regression estimation.

VARDEF=N | WGT | DF | WDF
specifies the denominator to be used in computing variances and covariances, MSE, root MSE measures, and so on. VARDEF=N specifies that the number of nonmissing observations be used. VARDEF=WGT specifies that the sum of the weights be used. VARDEF=DF specifies that the number of nonmissing observations minus the model degrees of freedom (number of parameters) be used. VARDEF=WDF specifies that the sum of the weights minus the model degrees of freedom be used. The default is VARDEF=DF. For FIML estimation the VARDEF= option does not affect the calculation of the parameter covariance matrix, which is determined by the COVBEST= option.

Data Set Options

DATA=*SAS-data-set*
specifies the input data set. Values for the variables in the program are read from this data set. If the DATA= option is not specified on the FIT statement, the data set specified by the DATA= option on the PROC MODEL statement is used.

ESTDATA=*SAS-data-set*
specifies a data set whose first observation provides initial values for some or all of the parameters.

MISSING=PAIRWISE | DELETE
specifies how missing values are handled. MISSING=PAIRWISE specifies that missing values are tracked on an equation-by-equation basis. MISSING=DELETE specifies that the entire observation is omitted from the analysis when any equation has a missing predicted or actual value for the equation. The default is MISSING=DELETE.

OUT=*SAS-data-set*
names the SAS data set to contain the residuals, predicted values, or actual values from each estimation. The residual values written to the OUT= data set are defined as the $actual - predicted$, which is the negative of RESID.*variable* as defined in the section "Equation Translations" on page 1677. Only the residuals are output by default.

OUTACTUAL
writes the actual values of the endogenous variables of the estimation to the OUT= data set. This option is applicable only if the OUT= option is specified.

OUTALL
selects the OUTACTUAL, OUTERRORS, OUTLAGS, OUTPREDICT, and OUTRESID options.

OUTCOV
COVOUT
writes the covariance matrix of the estimates to the OUTEST= data set in addition to the parameter estimates. The OUTCOV option is applicable only if the OUTEST= option is also specified.

OUTEST=*SAS-data-set*
names the SAS data set to contain the parameter estimates and optionally the covariance of the estimates.

OUTLAGS
writes the observations used to start the lags to the OUT= data set. This option is applicable only if the OUT= option is specified.

OUTPREDICT
writes the predicted values to the OUT= data set. This option is applicable only if OUT= is specified.

OUTRESID
writes the residual values computed from the parameter estimates to the OUT= data set. The OUTRESID option is the default if neither OUTPREDICT nor OUTACTUAL is specified. This option is applicable only if the OUT= option is specified. If the h.var equation is specified, the residual values written to the OUT= data set are the normalized residuals, defined as $actual - predicted$, divided by the square root of the h.var value. If the WEIGHT statement is used, the residual values are calculated as $actual - predicted$ multiplied by the square root of the WEIGHT variable.

OUTS=*SAS-data-set*
names the SAS data set to contain the estimated covariance matrix of the equation errors. This is the covariance of the residuals computed from the parameter estimates.

OUTSN=*SAS-data-set*
names the SAS data set to contain the estimated normalized covariance matrix of the equation errors. This is valid for multivariate *t* distribution estimation.

OUTSUSED=*SAS-data-set*
names the SAS data set to contain the **S** matrix used in the objective function definition. The OUTSUSED= data set is the same as the OUTS= data set for the methods that iterate the **S** matrix.

OUTUNWGTRESID
writes the unweighted residual values computed from the parameter estimates to the OUT= data set. These are residuals computed as $actual - predicted$ with no accounting for the WEIGHT statement, the _WEIGHT_ variable, or any variance expressions. This option is applicable only if the OUT= option is specified.

OUTV=*SAS-data-set*
names the SAS data set to contain the estimate of the variance matrix for GMM and ITGMM.

SDATA=*SAS-data-set*
specifies a data set that provides the covariance matrix of the equation errors. The matrix read from the SDATA= data set is used for the equation covariance matrix (**S** matrix) in the estimation. (The SDATA= S matrix is used to provide only the initial estimate of **S** for the methods that iterate the **S** matrix.)

TIME=*name*
specifies the name of the time variable. This variable must be in the data set.

TYPE=*name*
specifies the estimation type to read from the SDATA= and ESTDATA= data sets. The name specified in the TYPE= option is compared to the _TYPE_ variable in the ESTDATA= and SDATA= data sets to select observations to use in constructing the covariance matrices. When the TYPE= option is omitted, the last estimation type in the data set is used. Valid values are the estimation methods used in PROC MODEL.

VDATA=*SAS-data-set*
specifies a data set that contains a variance matrix for GMM and ITGMM estimation. See the section "Output Data Sets" on page 1632 for details.

Printing Options for FIT Tasks

BREUSCH=*(variable-list)*
specifies the modified Breusch-Pagan test, where *variable-list* is a list of variables used to model the error variance.

CHOW=*obs*
CHOW=*(obs1 obs2 . . . obsn)*
prints the Chow test for break points or structural changes in a model. The argument is the number of observations in the first sample or a parenthesized list of first sample sizes. If the size of the one of the two groups in which the sample is partitioned is less than the number of parameters, then a predictive Chow test is automatically used. See the section "Chow Tests" on page 1604 for details.

COLLIN
prints collinearity diagnostics for the Jacobian crossproducts matrix (**XPX**) after the parameters have converged. Collinearity diagnostics are also automatically printed if the estimation fails to converge.

CORR
prints the correlation matrices of the residuals and parameters. Using CORR is the same as using both CORRB and CORRS.

CORRB
prints the correlation matrix of the parameter estimates.

CORRS
prints the correlation matrix of the residuals.

COV
prints the covariance matrices of the residuals and parameters. Specifying COV is the same as specifying both COVB and COVS.

COVB
prints the covariance matrix of the parameter estimates.

COVS
prints the covariance matrix of the residuals.

DW < = >
prints Durbin-Watson *d* statistics, which measure autocorrelation of the residuals. When the residual series is interrupted by missing observations, the Durbin-Watson statistic calculated is d' as suggested by Savin and White (1978). This is the usual Durbin-Watson computed by ignoring the gaps. Savin and White show that it has the same null distribution as the DW with no gaps in the series and can be used to test for autocorrelation using the standard tables. The Durbin-Watson statistic is not valid for models that contain lagged endogenous variables.

You can use the DW= option to request higher-order Durbin-Watson statistics. Since the ordinary Durbin-Watson statistic tests only for first-order autocorrelation, the Durbin-Watson statistics for higher-order autocorrelation are called *generalized Durbin-Watson* statistics.

DWPROB
prints the significance level (*p*-values) for the Durbin-Watson tests. Since the Durbin-Watson *p*-values are computationally expensive, they are not reported by default. In the Durbin-Watson test, the null hypothesis is that there is autocorrelation at a specific lag.

See the section "Generalized Durbin-Watson Tests" for limitations of the statistic in the Chapter 9, "The AUTOREG Procedure."

FSRSQ
prints the first-stage R^2 statistics for instrumental estimation methods. These R^2 statistics measure the proportion of the variance retained when the Jacobian columns associated with the parameters are projected through the instruments space.

GODFREY

GODFREY=*n*

performs Godfrey's tests for autocorrelated residuals for each equation, where *n* is the maximum autoregressive order, and specifies that Godfrey's tests be computed for lags 1 through *n*. The default number of lags is one.

HAUSMAN

performs Hausman's specification test, or *m*-statistics.

NORMAL

performs tests of normality of the model residuals.

PCHOW=*obs*

PCHOW=*(obs1 obs2 . . . obsn)*

prints the predictive Chow test for break points or structural changes in a model. The argument is the number of observations in the first sample or a parenthesized list of first sample sizes. See the section "Chow Tests" on page 1604 for details.

PRINTALL

specifies the printing options COLLIN, CORRB, CORRS, COVB, COVS, DETAILS, DW, and FSRSQ.

WHITE

specifies White's test.

Options to Control Iteration Output

Details of the output produced are discussed in the section "Iteration History" on page 1567.

I

prints the inverse of the crossproducts Jacobian matrix at each iteration.

ITALL

specifies all iteration printing-control options (I, ITDETAILS, ITPRINT, and XPX). ITALL also prints the crossproducts matrix (labeled CROSS), the parameter change vector, and the estimate of the cross-equation covariance of residuals matrix at each iteration.

ITDETAILS

prints a detailed iteration listing. This includes the ITPRINT information and additional statistics.

ITPRINT

prints the parameter estimates, objective function value, and convergence criteria at each iteration.

XPX

prints the crossproducts Jacobian matrix at each iteration.

Options to Control the Minimization Process

The following options can be helpful when you experience a convergence problem:

CONVERGE=*value1*

CONVERGE=*(value1, value2)*

specifies the convergence criteria. The convergence measure must be less than *value1* before convergence is assumed. *value2* is the convergence criterion for the **S** and **V** matrices for **S** and **V** iterated methods. *value2* defaults to *value1*. See the section "Convergence Criteria" on page 1556 for details. The default value is CONVERGE=0.001.

HESSIAN=CROSS | GLS | FDA

specifies the Hessian approximation used for FIML. HESSIAN=CROSS selects the crossproducts approximation to the Hessian, HESSIAN=GLS selects the generalized least squares approximation to the Hessian, and HESSIAN=FDA selects the finite difference approximation to the Hessian. HESSIAN=GLS is the default.

LTEBOUND=*n*

specifies the local truncation error bound for the integration. This option is ignored if no ordinary differential equations (ODEs) are specified.

EPSILON =*value*

specifies the tolerance value used to transform strict inequalities into inequalities when restrictions on parameters are imposed. By default, EPSILON=1E–8. See the section "Restrictions and Bounds on Parameters" on page 1600 for details.

MAXITER=*n*

specifies the maximum number of iterations allowed. The default is MAXITER=100.

MAXSUBITER=*n*

specifies the maximum number of subiterations allowed for an iteration. For the GAUSS method, the MAXSUBITER= option limits the number of step halvings. For the MARQUARDT method, the MAXSUBITER= option limits the number of times λ can be increased. The default is MAXSUBITER=30. See the section "Minimization Methods" on page 1555 for details.

METHOD=GAUSS | MARQUARDT

specifies the iterative minimization method to use. METHOD=GAUSS specifies the Gauss-Newton method, and METHOD=MARQUARDT specifies the Marquardt-Levenberg method. The default is METHOD=GAUSS. If the default GAUSS method fails to converge, the procedure switches to the MARQUARDT method. See the section "Minimization Methods" on page 1555 for details.

MINTIMESTEP=*n*

specifies the smallest allowed time step to be used in the integration. This option is ignored if no ODEs are specified.

NESTIT

changes the way the iterations are performed for estimation methods that iterate the estimate of the equation covariance (**S** matrix). The NESTIT option is relevant only for the methods that iterate the estimate of the covariance matrix (ITGMM, ITOLS, ITSUR, IT2SLS, and IT3SLS). See the section "Details on the Covariance of Equation Errors" on page 1553 for an explanation of NESTIT.

SINGULAR=*value*

specifies the smallest pivot value allowed. The default 1.0E–12.

STARTITER=*n*

specifies the number of minimization iterations to perform at each grid point. The default is STARTITER=0, which implies that no minimization is performed at the grid points. See the section "Using the STARTITER Option" on page 1562 for more details.

Other Options

Other options that can be used on the FIT statement include the following that list and analyze the model: BLOCK, GRAPH, LIST, LISTCODE, LISTDEP, LISTDER, and XREF. The following printing control options are also available: DETAILS, FLOW, INTGPRINT, MAXERRORS=, NOPRINT, PRINTALL, and TRACE. For complete descriptions of these options, see the discussion of the PROC MODEL statement options earlier in this chapter.

ID Statement

ID *variables* **;**

The ID statement specifies variables to identify observations in error messages or other listings and in the OUT= data set. The ID variables are normally SAS date or datetime variables. If more than one ID variable is used, the first variable is used to identify the observations; the remaining variables are added to the OUT= data set.

INCLUDE Statement

INCLUDE *model-names . . .* **;**

The INCLUDE statement reads model files and inserts their contents into the current model. However, instead of replacing the current model as the RESET MODEL= option does, the contents of included model files are inserted into the model program at the position that the INCLUDE statement appears.

INSTRUMENTS Statement

INSTRUMENTS *variables* < *_EXOG_* > **;**

INSTRUMENTS < *variables-list* > < *_EXOG_* > < *EXCLUDE =(parameters)* > < */ options* > **;**

INSTRUMENTS *(equation, variables) (equation, variables) . . .* **;**

The INSTRUMENTS statement specifies the instrumental variables to be used in the N2SLS, N3SLS, IT2SLS, IT3SLS, GMM, and ITGMM estimation methods.

There are three ways of specifying the INSTRUMENTS statement. The first form of the INSTRUMENTS statement is declared before a FIT statement and defines the default instruments list. The items specified as instruments can be variables or the special keyword _EXOG_. The keyword _EXOG_ indicates that all the model variables declared EXOGENOUS are to be added to the instruments list. If a single INSTRUMENTS statement of the first form is declared before multiple FIT statements, then it serves as the default instruments list for each of the FIT statements. However, if any of these FIT statements are followed by separate

INSTRUMENTS statement, then the latter take precedence over the default list. Hence, in the case of multiple FIT statements, the INSTRUMENTS statement for a particular FIT statement is written below the FIT statement if instruments other than the default are required. For a single FIT statement, you can declare the INSTRUMENTS statement of the first form either preceding or following the FIT statement.

The second form of the INSTRUMENTS statement is used only after the FIT statement and before the next RUN statement. The items specified as instruments for the second form can be variables, names of parameters to be estimated, or the special keyword _EXOG_. If you specify the name of a parameter in the instruments list, the partial derivatives of the equations with respect to the parameter (that is, the columns of the Jacobian matrix associated with the parameter) are used as instruments. The parameter itself is not used as an instrument. These partial derivatives should not depend on any of the parameters to be estimated. Only the names of parameters to be estimated can be specified.

Note that an INSTRUMENTS statement of only the first form declared before multiple FIT statements serves as the default instruments list. Hence, in the cases of multiple as well as single FIT statements, you can declare the second form of INSTRUMENTS statements only following the FIT statements.

In the case where a FIT statement is preceded by an INSTRUMENTS statement of the second form in error and not followed by any INSTRUMENTS statement, then the default list is used. This default list is given by the INSTRUMENTS statement of the first form as explained above. If such a list is not declared, all the model variables declared EXOGENOUS comprise the default.

A third form of the INSTRUMENTS statement is used to specify instruments for each equation. No explicit intercept is added, parameters cannot be specified to represent instruments, and the _EXOG_ keyword is not allowed. Equations not explicitly assigned instruments use all the instruments specified for the other equations as well as instruments not assigned specific equations. In the following statements, z1, z2, and z3 are instruments used with equation y1, and z2, z3, and z4 are instruments used with equation y2.

```
proc model data=data_sim;
   exogenous x1 x2;
   parms a b c d e f;

   y1 =a*x1**2 + b*x2**2 + c*x1*x2   ;
   y2 =d*x1**2 + e*x2**2 + f*x1*x2**2;

   fit y1 y2 / 3sls ;
   instruments (y1, z1 z2 z3) (y2,z2 z3 z4);
run;
```

EXCLUDE=*(parameters)*

specifies that the derivatives of the equations with respect to all of the parameters to be estimated (except the parameters listed in the EXCLUDE list) be used as instruments, in addition to the other instruments specified. If you use the EXCLUDE= option, you should be sure that the derivatives with respect to the nonexcluded parameters in the estimation are independent of the endogenous variables and not functions of the parameters estimated.

The following options can be specified on the INSTRUMENTS statement following a slash (/):

NOINTERCEPT

NOINT

excludes the constant of 1.0 (intercept) from the instruments list. An intercept is included as an instrument while using the first or second form of the INSTRUMENTS statement unless NOINTERCEPT is specified.

When a FIT statement specifies an instrumental variables estimation method and no INSTRUMENTS statement accompanies the FIT statement, the default instruments are used. If no default instruments list has been specified, all the model variables declared EXOGENOUS are used as instruments. See the section "Choice of Instruments" on page 1607 for more details.

INTONLY

specifies that only the intercept be used as an instrument. This option is used for GMM estimation where the moments have been specified explicitly.

LABEL Statement

LABEL *variable='label' . . .* **;**

The LABEL statement specifies a label of up to 255 characters for parameters and other variables used in the model program. Labels are used to identify parts of the printout of FIT and SOLVE tasks. The labels are displayed in the output if the LINESIZE= option is large enough.

MOMENT Statement

MOMENT *variables = moment specification* **;**

In many scenarios, endogenous variables are observed from data. From the models, you can simulate these endogenous variables based on a fixed set of parameters. The goal of simulated method of moments (SMM) is to find a set of parameters such that the moments of the simulated data match the moments of the observed variables. If there are many moments to match, the code might be tedious. The following MOMENT statement provides a way to generate some commonly used moments automatically. Multiple MOMENT statements can be used.

variables can be one or more endogenous variables.

moment specification can have the following four types:

- (*number list*) specifies that the endogenous variable is raised to the power specified by each number in *number list*. For example,

```
moment y = (2 3);
```

adds the following two equations to be estimated:

```
eq._moment_1 = y**2 - pred.y**2;
eq._moment_2 = y**3 - pred.y**3;
```

- ABS(*number list*) specifies that the absolute value of the endogenous variable is raised to the power specified by each number in *number list*. For example,

```
moment y = ABS(3);
```

  adds the following equation to be estimated:

```
eq._moment_2 = abs(y)**3 - abs(pred.y)**3;
```

- LAG*n* (*number list*) specifies that the endogenous variable is multiplied by the *n*th lag of the endogenous variable, and this product is raised to the power specified by each number in *number list*. For example,

```
moment y = LAG4(3);
```

  adds the following equation to be estimated:

```
eq._moment_3 = (y*lag4(y))**3 - (pred.y*lag4(pred.y))**3;
```

- ABS_LAG*n* (*number list*) specifies that the endogenous variable is multiplied by the *n*th lag of the endogenous variable, and the absolute value of this product is raised to the power specified by each number in *number list*. For example,

```
moment y = ABS_LAG4(3);
```

  adds the following equation to be estimated:

```
eq._moment_4 = abs(y*lag4(y))**3 - abs(pred.y*lag4(pred.y))**3;
```

The following PROC MODEL statements use the MOMENT statement to generate 24 moments and fit these moments using SMM.

```
proc model data=_tmpdata list;
   parms a b .5 s 1;
   instrument _exog_ / intonly;

   u = rannor( 10091 );
   z = rannor( 97631 );

   lsigmasq = xlag(sigmasq,exp(a));

   lnsigmasq = a + b * log(lsigmasq) + s * u;
   sigmasq = exp( lnsigmasq );

   y = sqrt(sigmasq) * z;

   moment y = (2 4) abs(1 3) abs_lag1(1 2) abs_lag2(1 2);
   moment y = abs_lag3(1 2) abs_lag4(1 2)
              abs_lag5(1 2) abs_lag6(1 2)
              abs_lag7(1 2) abs_lag8(1 2)
              abs_lag9(1 2) abs_lag10(1 2);
```

```
   fit y  / gmm npreobs=20 ndraw=10;
   bound s > 0, 1>b>0;

run;
```

OUTVARS Statement

OUTVARS *variables* **;**

The OUTVARS statement specifies additional variables defined in the model program to be output to the OUT= data sets. The OUTVARS statement is not needed unless the variables to be added to the output data set are not referred to by the model, or unless you want to include parameters or other special variables in the OUT= data set. The OUTVARS statement includes additional variables, whereas the KEEP statement excludes variables.

PARAMETERS Statement

PARAMETERS *variable* < *value* > < *variable* < *value* > > . . . **;**

The PARAMETERS statement declares the parameters of a model and optionally sets their initial values. Valid abbreviations are PARMS and PARM.

Each parameter has a single value associated with it, which is the same for all observations. Lagging is not relevant for parameters. If a value is not specified in the PARMS statement (or by the PARMS= option of a FIT statement), the value defaults to 0.0001 for FIT tasks and to a missing value for SOLVE tasks.

Programming Statements

To define the model, you can use most of the programming statements that are allowed in the SAS DATA step. See the *SAS Language Reference: Dictionary* for more information.

RANGE Statement

RANGE *variable* < = *first* > < *TO last* > **;**

The RANGE statement specifies the range of observations to be read from the DATA= data set. For FIT tasks, the RANGE statement controls the period of fit for the estimation. For SOLVE tasks, the RANGE statement controls the simulation period or forecast horizon.

The RANGE variable must be a numeric variable in the DATA= data set that identifies the observations, and the data set must be sorted by the RANGE variable. The first observation in the range is identified by *first*, and the last observation is identified by *last*.

PROC MODEL uses the first *l* observations prior to *first* to initialize the lags, where *l* is the maximum number of lags needed to evaluate any of the equations to be fit or solved, or the maximum number of lags needed to

compute any of the instruments when an instrumental variables estimation method is used. There should be at least *l* observations in the data set before *first*. If *last* is not specified, all the nonmissing observations starting with *first* are used.

If *first* is omitted, the first *l* observations are used to initialize the lags, and the rest of the data, until *last*, is used. If a RANGE statement is used but both *first* and *last* are omitted, the RANGE statement variable is used to report the range of observations processed.

The RANGE variable should be nonmissing for all observations. Observations that contain missing RANGE values are deleted.

The following are examples of RANGE statements:

```
range year = 1971 to 1988;               /* yearly  data  */
range date = '1feb73'd to '1nov82'd;     /* monthly data  */
range time = 60.5;                       /* time in years */
range year to 1977;         /* use all years through 1977 */
range date; /* use values of date to report period-of-fit */
```

If no RANGE statements follow multiple FIT statements and if a single RANGE statement is declared before all the FIT statements, estimation in each of the multiple FIT statements is based on the data specified in the single RANGE statement. A single RANGE statement that follows multiple FIT statements affects only the fit immediately preceding it.

If the FIT statement is both followed by and preceded by RANGE statements, the following RANGE statement takes precedence over the preceding RANGE statement.

In the case where a range of data is to be used for a particular SOLVE task, the RANGE statement should be specified following the SOLVE statement in the case of either single or multiple SOLVE statements.

RESET Statement

RESET *options* **;**

All the options of the PROC MODEL statement can be reset by the RESET statement. In addition, the RESET statement supports one additional option:

PURGE
deletes the current model so that a new model can be defined.

When the MODEL= option is used in the RESET statement, the current model is deleted before the new model is read.

RESTRICT Statement

RESTRICT *restriction1* < *, restriction2 . . .* > **;**

The RESTRICT statement is used to impose linear and nonlinear restrictions either on the parameters in an estimation or on the solution variables that are specified in a solve operation.

Each *restriction* is written as an optional name, followed by an expression, followed by an equality operator (=) or an inequality operator (<, >, <=, >=), followed by a second expression:

< *"name"* **>** *expression operator expression*

The optional *"name"* is a string used to identify the restriction. The *operator* can be =, <, >, <= , or >=. The *operator* and second *expression* are optional. When they are omitted, the default *operator* is > and the default second *expression* is 0.

Each RESTRICT statement is associated with the preceding FIT statement or SOLVE statement. When there is no preceding FIT or SOLVE statement, the RESTRICT statement is associated with the following FIT or SOLVE statement. You can specify any number of RESTRICT statements.

Parameter Estimates

Expressions in RESTRICT statements that apply to the parameters estimated by a FIT statement can be composed of parameter names, arithmetic operators, functions, and constants. Comparison operators (such as = or <) and logical operators (such as &) cannot be used in RESTRICT statement expressions. Parameters that are named in restriction expressions must be among the parameters estimated by the associated FIT statement. Expressions can refer to variables defined in the program.

The restriction expressions can be linear or nonlinear functions of the parameters.

The optional *"name"* is a string used to identify the restriction in the printed output and in the OUTEST= data set.

The following example shows how to use the RESTRICT statement:

```
proc model data=one;
   endogenous y1 y2;
   exogenous x1 x2;
   parms a b c;
   restrict b*(b+c) <= a;

   eq.one = -y1/c + a/x2 + b * x1**2 + c * x2**2;
   eq.two = -y2 * y1 + b * x2**2 - c/(2 * x1);

   fit one two / fiml;
run;
```

Solution Variables

Expressions in RESTRICT statements that apply to the solution variables in a SOLVE statement can be composed of any variables in the model. Unlike restriction expressions that are used in parameter estimation, exogenous model variables can be used in restriction expressions that involve solution variables because each observation is solved independently in a SOLVE statement. To include constraints that are imposed by RESTRICT inequalities in a solution, you must specify the OPTIMIZE option in the SOLVE statement.

The following example illustrates how multiple solutions to a nonlinear system of equations can be found by using a RESTRICT expression that depends on exogenous variables. Two of the four possible solutions are presented in Figure 26.22.

```
data d;
   do i = 0 to 1;
      date=i;
      if i = 0 then r = -1;
      else          r = +1;
      output;
   end;
run;


proc model data=d ;
   endo x y;

   eq.a = x*x - 4;
   eq.b = y*y - 9;

   restrict x*y*r > 1;

   solve /  optimize out=o outall;
quit;

proc print data = o; run;
```

Figure 26.22 Listing of OUT= Data Set Created by a Nonlinear Restriction

Obs	_TYPE_	_MODE_	_ERRORS_	_OBJVAL_	x	y	r
1	ACTUAL	SIMULATE	0	0	2	-3	-1
2	PREDICT	SIMULATE	0	0	2	-3	-1
3	RESIDUAL	SIMULATE	0	0	.	.	-1
4	ERROR	SIMULATE	0	0	.	.	-1
5	VIOL	SIMULATE	0	0	.	.	-1
6	ACTUAL	SIMULATE	0	0	-2	-3	1
7	PREDICT	SIMULATE	0	0	-2	-3	1
8	RESIDUAL	SIMULATE	0	0	.	.	1
9	ERROR	SIMULATE	0	0	.	.	1
10	VIOL	SIMULATE	0	0	.	.	1

SOLVE Statement

SOLVE *variables* < *SATISFY= equations* > < */options* > **;**

The SOLVE statement specifies that the model be simulated or forecast for input data values and, optionally, selects the variables to be solved. If the list of variables is omitted, all of the model variables declared ENDOGENOUS are solved. If no model variables are declared ENDOGENOUS, then all model variables are solved.

The following specification can be used in the SOLVE statement:

SATISFY=*equation*

SATISFY=*(equations)*

specifies a subset of the model equations that the solution values are to satisfy. If the SATISFY= option is not used, the solution is computed to satisfy all the model equations. Note that the number of equations must equal the number of variables solved.

Data Set Options

DATA=*SAS-data-set*

names the input data set. The model is solved for each observation read from the DATA= data set. If the DATA= option is not specified on the SOLVE statement, the data set specified by the DATA= option in the PROC MODEL statement is used.

ESTDATA=*SAS-data-set*

names a data set whose first observation provides values for some or all of the parameters and whose additional observations (if any) give the covariance matrix of the parameter estimates. The covariance matrix read from the ESTDATA= data set is used to generate multivariate normal pseudo-random shocks to the model parameters when the RANDOM= option requests Monte Carlo simulation.

OUT=*SAS-data-set*

outputs the predicted (solution) values, residual values, actual values, or equation errors from the solution to a data set. The residual values are the $actual - predicted$ values, which is the negative of RESID.*variable* as defined in the section "Equation Translations" on page 1677. Only the solution values are output by default.

OUTACTUAL

outputs the actual values of the solved variables read from the input data set to the OUT= data set. This option is applicable only if the OUT= option is specified.

OUTALL

specifies the OUTACTUAL, OUTERRORS, OUTLAGS, OUTPREDICT, and OUTRESID options.

OUTERRORS

writes the equation errors to the OUT= data set. These values are normally very close to 0 when a simultaneous solution is computed; they can be used to double-check the accuracy of the solution process. This option applies only if the OUT= option is specified.

OUTLAGS

writes the observations that are used to start the lags to the OUT= data set. This option applies only if the OUT= option is specified.

OUTOBJVALS

writes the objective function value to the OBJVALS variable in the OUT= data set. The objective function value is computed only when the OPTIMIZE solution method is specified. This value is close to 0 when an unbounded simultaneous solution is computed and can be greater than 0 when bounds are active in the solution. This option applies only if the OUT= option is specified.

OUTPREDICT
writes the solution values to the OUT= data set. This option applies only if the OUT= option is specified.

The OUTPREDICT option is the default unless one of the other output options is specified.

OUTRESID
writes the residual values that are computed as the $actual - predicted$ values and is not the same as the RESID.*variable* values. This option applies only if the OUT= option is specified.

OUTVIOLATIONS
writes the equation violations to the OUT= data set. The equation violations are computed only when the OPTIMIZE solution method is specified. The violations provide information about how much each equation contributes to the objective function value when bounds are active in the solution. This option applies only if the OUT= option is specified.

PARMSDATA=*SAS-data-set*
specifies a data set that contains the parameter estimates. See the section "Input Data Sets" on page 1627 for more details.

RESIDDATA=*SAS-data-set*
specifies a data set that contains the residuals to be used in the empirical distribution. This data set can be created using the OUT= option in the FIT statement.

SDATA=*SAS-data-set*
specifies a data set that provides the covariance matrix of the equation errors. The covariance matrix that is read from the SDATA= data set is used to generate multivariate normal pseudo-random shocks to the equations when the RANDOM= option requests Monte Carlo simulation.

TIME=*name*
specifies the name of the time variable. This variable must be in the data set.

TYPE=*name*
specifies the estimation type. The name that is specified in the TYPE= option is compared to the _TYPE_ variable in the ESTDATA= and SDATA= data sets to select observations to use in constructing the covariance matrices. When TYPE= is omitted, the last estimation type in the data set is used.

Solution Mode Options: Lag Processing

DYNAMIC
specifies a dynamic solution. In the dynamic solution mode, solved values are used by the lagging functions. DYNAMIC is the default.

NAHEAD=*n*
specifies a simulation of *n*-period-ahead dynamic forecasting. The NAHEAD= option is used to simulate the process of using the model to produce successive forecasts to a fixed forecast horizon, in which each forecast uses the historical data available at the time the forecast is made.

Note that NAHEAD=1 produces a static (one-step-ahead) solution. NAHEAD=2 produces a solution that uses one-step-ahead solutions for the first lag (LAG1 functions return static predicted values) and actual values for longer lags. NAHEAD=3 produces a solution that uses NAHEAD=2 solutions for

the first lags, NAHEAD=1 solutions for the second lags, and actual values for longer lags. In general, NAHEAD=*n* solutions use NAHEAD=*n*–1 solutions for LAG1, NAHEAD=*n*–2 solutions for LAG2, and so forth.

START=*s*
specifies static solutions until the *s*th observation and then changes to dynamic solutions. If the START=*s* option is specified, the first observation in the range in which LAG*n* delivers solved predicted values is *s*+*n*, while LAG*n* returns actual values for earlier observations.

STATIC
specifies a static solution. In static solution mode, actual values of the solved variables from the input data set are used by the lagging functions.

Solution Mode Options: Use of Available Data

FORECAST
specifies that the actual value of a solved variable is used as the solution value (instead of the predicted value from the model equations) whenever nonmissing data are available in the input data set. That is, in FORECAST mode, PROC MODEL solves only for those variables that are missing in the input data set.

SIMULATE
specifies that PROC MODEL always solves for all solution variables as a function of the input values of the other variables, even when actual data for some of the solution variables are available in the input data set. SIMULATE is the default.

Solution Mode Options: Numerical Solution Method

JACOBI
computes a simultaneous solution using a Jacobi iteration.

NEWTON
computes a simultaneous solution by using Newton's method. When the NEWTON option is selected, the analytic derivatives of the equation errors with respect to the solution variables are computed, and memory-efficient sparse matrix techniques are used for factoring the Jacobian matrix.

The NEWTON option can be used to solve both normalized-form and general-form equations and can compute goal-seeking solutions. NEWTON is the default.

OPTIMIZE
computes a simultaneous solution by minimizing a norm of the equation errors with respect to the solution variables. The OPTIMIZE method obeys constraints on the solution variables that are imposed by the BOUNDS and RESTRICT statements.

SEIDEL
computes a simultaneous solution by using a Gauss-Seidel method.

SINGLE

ONEPASS

specifies a single-equation (nonsimultaneous) solution. The model is executed once to compute predicted values for the variables from the actual values of the other endogenous variables. The SINGLE option can be used only for normalized-form equations and cannot be used for goal-seeking solutions.

For more information about these options, see the section "Solution Modes" on page 1639.

Monte Carlo Simulation Options

COPULA=(*copula-options***)**

specifies the copula to be used in the simulation. You can specify the following *copula-options*:

- CLAYTON(θ), where θ is the Clayton copula parameter
- FRANK(θ), where θ is the Frank copula parameter
- GUMBEL(θ), where θ is the Gumbel copula parameter
- NORMAL
- NORMALMIX($n, p_1 \ldots p_n, v_1 \ldots v_n$), where p_i are the probabilities and v_i are the variances
- T(df) < ASYM >, where df is the degrees-of-freedom parameter

The normal (Gaussian) copula is the default. The copula applies to covariance of equation errors.

PSEUDO=DEFAULT | TWISTER

specifies which pseudo-number generator to use in generating draws for Monte Carlo simulation. The two pseudo-random number generators that are supported by the MODEL procedure are a default congruential generator that has period $2^{31} - 1$ and a Mersenne-Twister pseudo-random number generator that has an extraordinarily long period $2^{19937} - 1$.

QUASI=NONE|SOBOL|FAURE

specifies a pseudo- or quasi-random number generator. Two quasi-random number generators are supported by the MODEL procedure: the Sobol sequence (QUASI=SOBOL) and the Faure sequence (QUASI=FAURE). The default is QUASI=NONE, which is the pseudo-random number generator.

RANDOM=*n*

repeats the solution n times for each BY group, with different random perturbations of the equation errors if the SDATA= option is specified; with different random perturbations of the parameters if the ESTDATA= option is specified and the ESTDATA= data set contains a parameter covariance matrix; and with different values returned from the random number generator functions, if any are used in the model program. If RANDOM=0, the random number generator functions always return zero. See the section "Monte Carlo Simulation" on page 1642 for details. The default is RANDOM=0.

SEED=*n*

specifies an integer to use as the seed in generating pseudo-random numbers to shock the parameters and equations when the ESTDATA= or SDATA= option is specified. If n is negative or 0, the time of day from the computer's clock is used as the seed. The SEED= option is relevant only if the RANDOM= option is specified. The default is SEED=0.

WISHART=*df*
specifies that a Wishart distribution with degrees of freedom *df* be used in place of the normal error covariance matrix. This option is used to model the variance of the error covariance matrix when Monte Carlo simulation is selected.

Options for Controlling the Numerical Solution Process

The following options are useful when you have difficulty converging to the simultaneous solution.

CONVERGE=*value*
specifies the convergence criterion for the simultaneous solution. Convergence of the solution is judged by comparing the CONVERGE= value to the maximum over the equations of

$$\frac{|\epsilon_i|}{|y_i| + 1E-6}$$

if they are computable, otherwise

$$|\epsilon_i|$$

where ϵ_i represents the equation error and y_i represents the solution variable that corresponds to the *i*th equation for normalized-form equations. The default is CONVERGE=1E–8.

MAXITER=*n*
specifies the maximum number of iterations allowed for computing the simultaneous solution for any observation. The default is MAXITER=50.

MAXSUBITER=*n*
specifies the maximum number of damping subiterations that are performed in solving a nonlinear system when using the NEWTON solution method. Damping is disabled by setting MAXSUBITER=0. The default is MAXSUBITER=10.

Printing Options

INTGPRINT
prints between data points integration values for the DERT. variables and the auxiliary variables. If you specify the DETAILS option, the integrated derivative variables are printed as well.

ITPRINT
prints the solution approximation and equation errors at each iteration for each observation. This option can produce voluminous output.

PRINTALL
specifies the printing control options DETAILS, ITPRINT, SOLVEPRINT, STATS, and THEIL.

SOLVEPRINT
prints the solution values and residuals at each observation.

STATS
prints various summary statistics for the solution values.

THEIL
prints tables of Theil inequality coefficients and Theil relative change forecast error measures for the solution values. See the section "Summary Statistics" on page 1657 for more information.

Other Options

Other options that can be used on the SOLVE statement include the following that list and analyze the model: BLOCK, GRAPH, LIST, LISTCODE, LISTDEP, LISTDER, and XREF. The LTEBOUND= and MINTIMESTEP= options can be used to control the integration process. The following printing-control options are also available: DETAILS, FLOW, MAXERRORS=, NOPRINT, and TRACE. For complete descriptions of these options, see the PROC MODEL and FIT statement options described earlier in this chapter.

TEST Statement

TEST < *"name"*> *test1* < *, test2 . . .* > < *,/ options* > **;**

The TEST statement performs tests of nonlinear hypotheses on the model parameters.

Each TEST statement applies to the parameters estimated by one FIT statement. TEST statements that appear before or after the first FIT statement are associated with the first FIT statement. Subsequent TEST statements are associated with the FIT statement that precedes them. TEST statements that are separated from a FIT statement by an intervening RUN, SOLVE, or RESET statement are ignored. You can specify any number of TEST statements.

If you specify options on the TEST statement, a comma is required before the "/" character that separates the test expressions from the options, because the "/" character can also be used within test expressions to indicate division.

The label lengths for tests and estimate statements are 256 characters. If the labels exceed this length, the label is truncated to 256 characters with a note printed to the log.

Each test is written as an expression optionally followed by an equal sign (=) and a second expression:

< expression > < = expression >

Test expressions can be composed of parameter names, arithmetic operators, functions, and constants. Comparison operators (such as =) and logical operators (such as &) cannot be used in TEST statement expressions. Parameters named in test expressions must be among the parameters estimated by the associated FIT statement.

If you specify only one expression in a test, that expression is tested against zero. For example, the following two TEST statements are equivalent:

```
test a + b;

test a + b = 0;
```

When you specify multiple tests in the same TEST statement, a joint test is performed. For example, the following TEST statement tests the joint hypothesis that both A and B are equal to zero.

```
test a, b;
```

To perform separate tests rather than a joint test, use separate TEST statements. For example, the following TEST statements test the two separate hypotheses that A is equal to zero and that B is equal to zero.

```
test a;
test b;
```

You can use the following options in the TEST statement.

WALD
specifies that a Wald test be computed. By default, the Wald test is computed.

LM
RAO
LAGRANGE
specifies that a Lagrange multiplier test be computed.

LR
LIKE
specifies that a likelihood ratio test be computed.

ALL
requests all three types of tests.

OUT=*SAS-data-set*
specifies the name of an output *SAS data set* that contains the test results. The format of the OUT= data set that is produced by the TEST statement is similar to that of the OUTEST= data set produced by the FIT statement.

VAR Statement

VAR *variables* < *initial-values* > . . . **;**

The VAR statement declares model variables and optionally provides initial values for the lags of the variables. See the section "Lag Logic" on page 1682 for more information.

VARGROUP Statement

VARGROUP *label*=*variable*. . . **;**

The VARGROUP statement applies a group label to the specified list of variables in the model program. Variable groups are used to identify sets of related solve variables. The variable groups can be used by the ANALYZEDEP= option in a subsequent SOLVE statement to help specify and understand the role of groups of solve variables in a SOLVE step. If a variable appears in more than one VARGROUP statement, the label that is specified in the last VARGROUP statement is applied to that variable.

WEIGHT Statement

WEIGHT *variable* **;**

The WEIGHT statement specifies a variable to supply weighting values to use for each observation in estimating parameters.

If the weight of an observation is nonpositive, that observation is not used for the estimation. *variable* must be a numeric variable in the input data set.

An alternative weighting method is to use an assignment statement to give values to the special variable _WEIGHT_. The _WEIGHT_ variable must not depend on the parameters being estimated. If both weighting specifications are given, the weights are multiplied together.

Details: Estimation by the MODEL Procedure

Estimation Methods

Consider the general nonlinear model:

$$\begin{aligned} \boldsymbol{\epsilon}_t &= \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \boldsymbol{\theta}) \\ \mathbf{z}_t &= Z(\mathbf{x}_t) \end{aligned}$$

where $\mathbf{q} \in R^g$ is a real vector valued function of $\mathbf{y}_t \in R^g$, $\mathbf{x}_t \in R^l$, $\boldsymbol{\theta} \in R^p$, where g is the number of equations, l is the number of exogenous variables (lagged endogenous variables are considered exogenous here), p is the number of parameters, and t ranges from 1 to n. $\mathbf{z}_t \in R^k$ is a vector of instruments. $\boldsymbol{\epsilon}_t$ is an unobservable disturbance vector with the following properties:

$$\begin{aligned} E(\boldsymbol{\epsilon}_t) &= 0 \\ E(\boldsymbol{\epsilon}_t \boldsymbol{\epsilon}_t^{'}) &= \boldsymbol{\Sigma} \end{aligned}$$

All of the methods implemented in PROC MODEL aim to minimize an *objective function*. The following table summarizes the objective functions that define the estimators and the corresponding estimator of the covariance of the parameter estimates for each method.

Table 26.2 Summary of PROC MODEL Estimation Methods

Method	Instruments	Objective Function	Covariance of θ
OLS	no	$\mathbf{r}'\mathbf{r}/n$	$(\mathbf{X}'(\mathrm{diag}(\mathbf{S})^{-1}\otimes\mathbf{I})\mathbf{X})^{-1}$
ITOLS	no	$\mathbf{r}'(\mathrm{diag}(\mathbf{S})^{-1}\otimes\mathbf{I})\mathbf{r}/n$	$(\mathbf{X}'(\mathrm{diag}(\mathbf{S})^{-1}\otimes\mathbf{I})\mathbf{X})^{-1}$
SUR	no	$\mathbf{r}'(\mathbf{S}_{\mathrm{OLS}}^{-1}\otimes\mathbf{I})\mathbf{r}/n$	$(\mathbf{X}'(\mathbf{S}^{-1}\otimes\mathbf{I})\mathbf{X})^{-1}$
ITSUR	no	$\mathbf{r}'(\mathbf{S}^{-1}\otimes\mathbf{I})\mathbf{r}/n$	$(\mathbf{X}'(\mathbf{S}^{-1}\otimes\mathbf{I})\mathbf{X})^{-1}$
N2SLS	yes	$\mathbf{r}'(\mathbf{I}\otimes\mathbf{W})\mathbf{r}/n$	$(\mathbf{X}'(\mathrm{diag}(\mathbf{S})^{-1}\otimes\mathbf{W})\mathbf{X})^{-1}$
IT2SLS	yes	$\mathbf{r}'(\mathrm{diag}(\mathbf{S})^{-1}\otimes\mathbf{W})\mathbf{r}/n$	$(\mathbf{X}'(\mathrm{diag}(\mathbf{S})^{-1}\otimes\mathbf{W})\mathbf{X})^{-1}$
N3SLS	yes	$\mathbf{r}'(\mathbf{S}_{\mathrm{N2SLS}}^{-1}\otimes\mathbf{W})\mathbf{r}/n$	$(\mathbf{X}'(\mathbf{S}^{-1}\otimes\mathbf{W})\mathbf{X})^{-1}$
IT3SLS	yes	$\mathbf{r}'(\mathbf{S}^{-1}\otimes\mathbf{W})\mathbf{r}/n$	$(\mathbf{X}'(\mathbf{S}^{-1}\otimes\mathbf{W})\mathbf{X})^{-1}$
GMM	yes	$[n\mathbf{m}_n(\theta)]'\hat{\mathbf{V}}_{\mathrm{N2SLS}}^{-1}[n\mathbf{m}_n(\theta)]/n$	$[(\mathbf{YX})'\hat{\mathbf{V}}^{-1}(\mathbf{YX})]^{-1}$
ITGMM	yes	$[n\mathbf{m}_n(\theta)]'\hat{\mathbf{V}}^{-1}[n\mathbf{m}_n(\theta)]/n$	$[(\mathbf{YX})'\hat{\mathbf{V}}^{-1}(\mathbf{YX})]^{-1}$
FIML	no	$constant + \frac{n}{2}\ln(\det(\mathbf{S}))$ $-\sum_1^n \ln\lvert(\mathbf{J}_t)\rvert$	$[\hat{\mathbf{Z}}'(\mathbf{S}^{-1}\otimes\mathbf{I})\hat{\mathbf{Z}}]^{-1}$

The column labeled "Instruments" identifies the estimation methods that require instruments. The variables used in this table and the remainder of this chapter are defined as follows:

n = is the number of nonmissing observations.

g = is the number of equations.

k = is the number of instrumental variables.

$\mathbf{r} = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_g \end{bmatrix}$ is the $ng \times 1$ vector of residuals for the g equations stacked together.

$\mathbf{r}_i = \begin{bmatrix} q_i(\mathbf{y}_1, \mathbf{x}_1, \boldsymbol{\theta}) \\ q_i(\mathbf{y}_2, \mathbf{x}_2, \boldsymbol{\theta}) \\ \vdots \\ q_i(\mathbf{y}_n, \mathbf{x}_n, \boldsymbol{\theta}) \end{bmatrix}$ is the $n \times 1$ column vector of residuals for the ith equation.

S is a $g \times g$ matrix that estimates $\mathbf{\Sigma}$, the covariances of the errors across equations (referred to as the **S** matrix).

X is an $ng \times p$ matrix of partial derivatives of the residual with respect to the parameters.

W is an $n \times n$ matrix, $\mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'$.

Z is an $n \times k$ matrix of instruments.

Y is a $gk \times ng$ matrix of instruments. $\mathbf{Y} = \mathbf{I}_g \otimes \mathbf{Z}'$.

$\hat{\mathbf{Z}}$ $\hat{\mathbf{Z}} = (\hat{Z}_1, \hat{Z}_2, \ldots, \hat{Z}_p)$ is an $ng \times p$ matrix. $\hat{Z}_i$ is a $ng \times 1$ column vector obtained from stacking the columns of

$$\mathbf{U}\frac{1}{n}\sum_{t=1}^{n}\left(\frac{\partial \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \boldsymbol{\theta})'}{\partial y_t}\right)^{-1}\frac{\partial^2 \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \boldsymbol{\theta})'}{\partial y_t \partial \theta_i} - \mathbf{Q}_i$$

U is an $n \times g$ matrix of residual errors. $\mathbf{U} = \boldsymbol{\epsilon}_1, \boldsymbol{\epsilon}_2, \ldots, \boldsymbol{\epsilon}_n{}'$.

$\mathbf{Q}$	is the $n \times g$ matrix $\mathbf{q}(\mathbf{y}_1, \mathbf{x}_1, \boldsymbol{\theta}), \mathbf{q}(\mathbf{y}_2, \mathbf{x}_2, \boldsymbol{\theta}), \ldots, \mathbf{q}(\mathbf{y}_n, \mathbf{x}_n, \boldsymbol{\theta})$.
$\mathbf{Q}_i$	is an $n \times g$ matrix $\frac{\partial \mathbf{Q}}{\partial \theta_i}$.
$\mathbf{I}$	is an $n \times n$ identity matrix.
$\mathbf{J}_t$	is $\frac{\partial \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \boldsymbol{\theta})}{\partial \mathbf{y}_t'}$, which is a $g \times g$ Jacobian matrix.
$\mathbf{m}_n$	is first moment of the crossproduct $\mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \boldsymbol{\theta}) \otimes \mathbf{z}_t$, $m_n = \frac{1}{n} \sum_{t=1}^{n} \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \boldsymbol{\theta}) \otimes \mathbf{z}_t$
$\mathbf{z}_t$	is a k column vector of instruments for observation t. $\mathbf{z}_t'$ is also the tth row of $\mathbf{Z}$.
$\hat{\mathbf{V}}$	is the $gk \times gk$ matrix that represents the variance of the moment functions.
k	is the number of instrumental variables used.
constant	is the constant $\frac{ng}{2}(1 + \ln(2\pi))$.
$\otimes$	is the notation for a Kronecker product.

All vectors are column vectors unless otherwise noted. Other estimates of the covariance matrix for FIML are also available.

Dependent Regressors and Two-Stage Least Squares

Ordinary regression analysis is based on several assumptions. A key assumption is that the independent variables are in fact statistically independent of the unobserved error component of the model. If this assumption is not true (if the regressor varies systematically with the error), then ordinary regression produces inconsistent results. The parameter estimates are *biased*.

Regressors might fail to be independent variables because they are dependent variables in a larger simultaneous system. For this reason, the problem of dependent regressors is often called *simultaneous equation bias*. For example, consider the following two-equation system:

$$y_1 = a_1 + b_1 y_2 + c_1 x_1 + \epsilon_1$$

$$y_2 = a_2 + b_2 y_1 + c_2 x_2 + \epsilon_2$$

In the first equation, y_2 is a dependent, or *endogenous*, variable. As shown by the second equation, y_2 is a function of y_1, which by the first equation is a function of ϵ_1, and therefore y_2 depends on ϵ_1. Likewise, y_1 depends on ϵ_2 and is a dependent regressor in the second equation. This is an example of a *simultaneous equation* system; y_1 and y_2 are a function of all the variables in the system.

Using the ordinary least squares (OLS) estimation method to estimate these equations produces biased estimates. One solution to this problem is to replace y_1 and y_2 on the right-hand side of the equations with predicted values, thus changing the regression problem to the following:

$$y_1 = a_1 + b_1 \hat{y}_2 + c_1 x_1 + \epsilon_1$$

$$y_2 = a_2 + b_2 \hat{y}_1 + c_2 x_2 + \epsilon_2$$

This method requires estimating the predicted values $\hat{y}_1$ and $\hat{y}_2$ through a preliminary, or "first stage," *instrumental regression*. An instrumental regression is a regression of the dependent regressors on a set of *instrumental variables*, which can be any independent variables useful for predicting the dependent regressors.

In this example, the equations are linear and the exogenous variables for the whole system are known. Thus, the best choice for instruments (of the variables in the model) are the variables x_1 and x_2.

This method is known as *two-stage least squares* or 2SLS, or more generally as the *instrumental variables method*. The 2SLS method for linear models is discussed in Pindyck and Rubinfeld (1981, pp. 191–192). For nonlinear models this situation is more complex, but the idea is the same. In nonlinear 2SLS, the derivatives of the model with respect to the parameters are replaced with predicted values. See the section "Choice of Instruments" on page 1607 for further discussion of the use of instrumental variables in nonlinear regression.

To perform nonlinear 2SLS estimation with PROC MODEL, specify the instrumental variables with an INSTRUMENTS statement and specify the 2SLS or N2SLS option in the FIT statement. The following statements show how to estimate the first equation in the preceding example with PROC MODEL:

```
proc model data=in;
   y1 = a1 + b1 * y2 + c1 * x1;
   fit y1 / 2sls;
   instruments x1 x2;
run;
```

The 2SLS or instrumental variables estimator can be computed by using a first-stage regression on the instrumental variables as described previously. However, PROC MODEL actually uses the equivalent but computationally more appropriate technique of projecting the regression problem into the linear space defined by the instruments. Thus, PROC MODEL does not produce any "first stage" results when you use 2SLS. If you specify the FSRSQ option in the FIT statement, PROC MODEL prints "First-Stage R^2" statistic for each parameter estimate.

Formally, the $\hat{\boldsymbol{\theta}}$ that minimizes

$$\hat{S}_n = \frac{1}{n}\left(\sum_{t=1}^{n}(\mathbf{q}(\mathbf{y}_t,\mathbf{x}_t,\theta)\otimes\mathbf{z}_t)\right)'\left(\sum_{t=1}^{n}I\otimes\mathbf{z}_t\mathbf{z}_t'\right)^{-1}\left(\sum_{t=1}^{n}(\mathbf{q}(\mathbf{y}_t,\mathbf{x}_t,\boldsymbol{\theta})\otimes\mathbf{z}_t)\right)$$

is the N2SLS estimator of the parameters. The estimate of $\boldsymbol{\Sigma}$ at the final iteration is used in the covariance of the parameters given in Table 26.2. See Amemiya (1985, p. 250) for details on the properties of nonlinear two-stage least squares.

Seemingly Unrelated Regression

If the regression equations are not simultaneous (so there are no dependent regressors), *seemingly unrelated regression* (SUR) can be used to estimate systems of equations with correlated random errors. The large-sample efficiency of an estimation can be improved if these cross-equation correlations are taken into account. SUR is also known as *joint generalized least squares* or *Zellner regression*. Formally, the $\hat{\boldsymbol{\theta}}$ that minimizes

$$\hat{S}_n = \frac{1}{n}\sum_{t=1}^{n}\mathbf{q}(\mathbf{y}_t,\mathbf{x}_t,\boldsymbol{\theta})'\hat{\boldsymbol{\Sigma}}^{-1}\mathbf{q}(\mathbf{y}_t,\mathbf{x}_t,\boldsymbol{\theta})$$

is the SUR estimator of the parameters.

The SUR method requires an estimate of the cross-equation covariance matrix, $\boldsymbol{\Sigma}$. PROC MODEL first performs an OLS estimation, computes an estimate, $\hat{\boldsymbol{\Sigma}}$, from the OLS residuals, and then performs the SUR estimation based on $\hat{\boldsymbol{\Sigma}}$. The OLS results are not printed unless you specify the OLS option in addition to the SUR option.

You can specify the $\hat{\mathbf{\Sigma}}$ to use for SUR by storing the matrix in a SAS data set and naming that data set in the SDATA= option. You can also feed the $\hat{\mathbf{\Sigma}}$ computed from the SUR residuals back into the SUR estimation process by specifying the ITSUR option. You can print the estimated covariance matrix $\hat{\mathbf{\Sigma}}$ by using the COVS option in the FIT statement.

The SUR method requires estimation of the $\mathbf{\Sigma}$ matrix, and this increases the sampling variability of the estimator for small sample sizes. The efficiency gain that SUR has over OLS is a large sample property, and you must have a reasonable amount of data to realize this gain. For a more detailed discussion of SUR, see Pindyck and Rubinfeld (1981, pp. 331–333).

Three-Stage Least Squares Estimation

If the equation system is simultaneous, you can combine the 2SLS and SUR methods to take into account both dependent regressors and cross-equation correlation of the errors. This is called *three-stage least squares* (3SLS).

Formally, the $\hat{\boldsymbol{\theta}}$ that minimizes

$$\hat{S}_n = \frac{1}{n}\left(\sum_{t=1}^{n}(\mathbf{q}(\mathbf{y}_t,\mathbf{x}_t,\boldsymbol{\theta})\otimes\mathbf{z}_t)\right)' \left(\sum_{t=1}^{n}(\hat{\mathbf{\Sigma}}\otimes\mathbf{z}_t\mathbf{z}_t')\right)^{-1} \left(\sum_{t=1}^{n}(\mathbf{q}(\mathbf{y}_t,\mathbf{x}_t,\boldsymbol{\theta})\otimes\mathbf{z}_t)\right)$$

is the 3SLS estimator of the parameters. For more details on 3SLS, see Gallant (1987, p. 435).

Residuals from the 2SLS method are used to estimate the $\mathbf{\Sigma}$ matrix required for 3SLS. The results of the preliminary 2SLS step are not printed unless the 2SLS option is also specified.

To use the three-stage least squares method, specify an INSTRUMENTS statement and use the 3SLS or N3SLS option in either the PROC MODEL statement or a FIT statement.

Generalized Method of Moments (GMM)

For systems of equations with heteroscedastic errors, generalized method of moments (GMM) can be used to obtain efficient estimates of the parameters. See the section "Heteroscedasticity" on page 1576 for alternatives to GMM.

Consider the nonlinear model

$$\begin{aligned}\boldsymbol{\epsilon}_t &= \mathbf{q}(\mathbf{y}_t,\mathbf{x}_t,\boldsymbol{\theta})\\ \mathbf{z}_t &= Z(\mathbf{x}_t)\end{aligned}$$

where $\mathbf{z}_t$ is a vector of instruments and $\boldsymbol{\epsilon}_t$ is an unobservable disturbance vector that can be serially correlated and nonstationary.

In general, the following orthogonality condition is desired:

$$E(\boldsymbol{\epsilon}_t\otimes\mathbf{z}_t) = 0$$

This condition states that the expected crossproducts of the unobservable disturbances, $\boldsymbol{\epsilon}_t$, and functions of the observable variables are set to 0. The first moment of the crossproducts is

$$\begin{aligned} \mathbf{m}_n &= \frac{1}{n}\sum_{t=1}^{n}\mathbf{m}(\mathbf{y}_t, \mathbf{x}_t, \boldsymbol{\theta}) \\ \mathbf{m}(\mathbf{y}_t, \mathbf{x}_t, \boldsymbol{\theta}) &= \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \boldsymbol{\theta}) \otimes \mathbf{z}_t \end{aligned}$$

where $\mathbf{m}(\mathbf{y}_t, \mathbf{x}_t, \boldsymbol{\theta}) \in R^{gk}$.

The case where $gk > p$ is considered here, where p is the number of parameters.

Estimate the true parameter vector θ^0 by the value of $\hat{\theta}$ that minimizes

$$S(\theta, \mathbf{V}) = [n\mathbf{m}_n(\theta)]' \mathbf{V}^{-1} [n\mathbf{m}_n(\theta)]/n$$

where

$$\mathbf{V} = \mathrm{Cov}\left([n\mathbf{m}_n(\theta^0)], [n\mathbf{m}_n(\theta^0)]'\right)$$

The parameter vector that minimizes this objective function is the GMM estimator. GMM estimation is requested in the FIT statement with the GMM option.

The variance of the moment functions, $\mathbf{V}$, can be expressed as

$$\begin{aligned} \mathbf{V} &= E\left(\sum_{t=1}^{n}\boldsymbol{\epsilon}_t \otimes \mathbf{z}_t\right)\left(\sum_{s=1}^{n}\boldsymbol{\epsilon}_s \otimes \mathbf{z}_s\right)' \\ &= \sum_{t=1}^{n}\sum_{s=1}^{n} E\left[(\boldsymbol{\epsilon}_t \otimes \mathbf{z}_t)(\boldsymbol{\epsilon}_s \otimes \mathbf{z}_s)'\right] \\ &= n\mathbf{S}_n^0 \end{aligned}$$

where $\mathbf{S}_n^0$ is estimated as

$$\hat{\mathbf{S}}_n = \frac{1}{n}\sum_{t=1}^{n}\sum_{s=1}^{n}(\mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \boldsymbol{\theta}) \otimes \mathbf{z}_t)(\mathbf{q}(\mathbf{y}_s, \mathbf{x}_s, \boldsymbol{\theta}) \otimes \mathbf{z}_s)'$$

Note that $\hat{\mathbf{S}}_n$ is a $gk \times gk$ matrix. Because Var $(\hat{\mathbf{S}}_n)$ does not decrease with increasing n, you consider estimators of $\mathbf{S}_n^0$ of the form:

$$
\begin{aligned}
\hat{\mathbf{S}}_n(l(n)) &= \sum_{\tau=-n+1}^{n-1} \hat{w}(\frac{\tau}{l(n)})\mathbf{D}\hat{\mathbf{S}}_{n,\tau}\mathbf{D} \\
\hat{\mathbf{S}}_{n,\tau} &= \begin{cases} \sum_{t=1+\tau}^{n} [\mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \boldsymbol{\theta}^{\#}) \otimes \mathbf{z}_t][\mathbf{q}(\mathbf{y}_{t-\tau}, \mathbf{x}_{t-\tau}, \boldsymbol{\theta}^{\#}) \otimes \mathbf{z}_{t-\tau}]' & \tau \geq 0 \\ (\hat{\mathbf{S}}_{n,-\tau})' & \tau < 0 \end{cases} \\
\hat{w}(\frac{\tau}{l(n)}) &= \begin{cases} w(\frac{\tau}{l(n)}) & l(n) > 0 \\ \delta_{\tau,0} & l(n) = 0 \end{cases}
\end{aligned}
$$

where $l(n)$ is a scalar function that computes the bandwidth parameter, $w(\cdot)$ is a scalar valued kernel, and the Kronecker delta function, $\delta_{i,j}$, is 1 if $i = j$ and 0 otherwise. The diagonal matrix $\mathbf{D}$ is used for a small sample degrees of freedom correction (Gallant 1987). The initial $\theta^{\#}$ used for the estimation of $\hat{\mathbf{S}}_n$ is obtained from a 2SLS estimation of the system. The degrees of freedom correction is handled by the VARDEF= option as it is for the **S** matrix estimation.

The following kernels are supported by PROC MODEL. They are listed with their default bandwidth functions.

Bartlett: KERNEL=BART

$$
\begin{aligned}
w(x) &= \begin{cases} 1 - |x| & |x| \leq 1 \\ 0 & \text{otherwise} \end{cases} \\
l(n) &= \frac{1}{2} n^{1/3}
\end{aligned}
$$

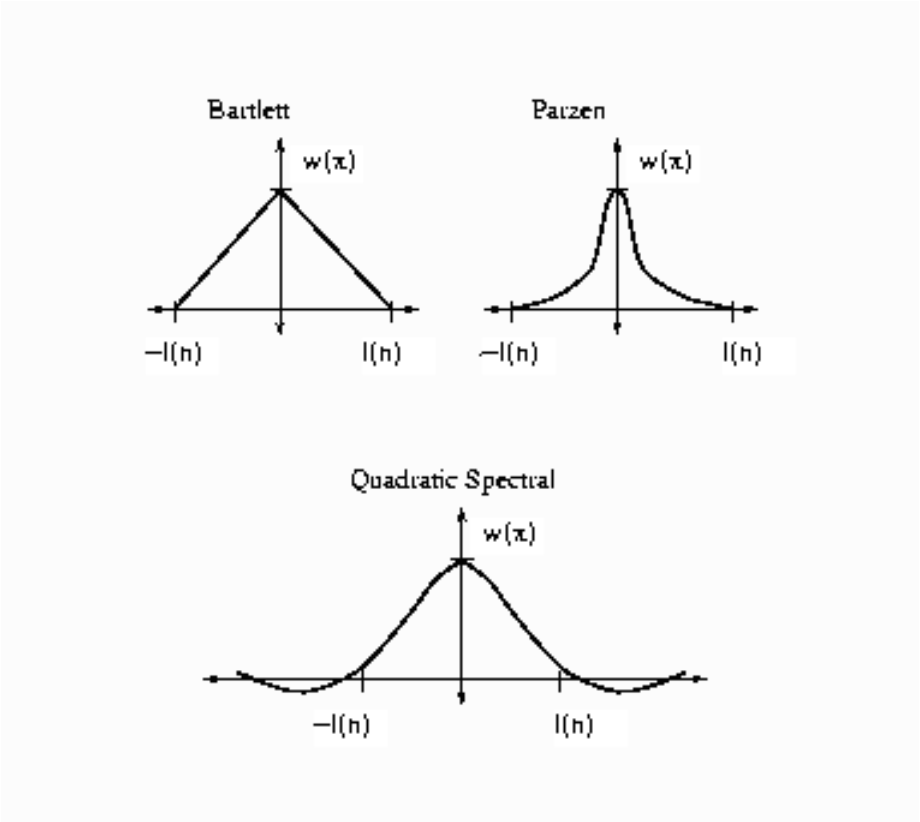
Parzen: KERNEL=PARZEN

$$
\begin{aligned}
w(x) &= \begin{cases} 1 - 6|x|^2 + 6|x|^3 & 0 \leq |x| \leq \frac{1}{2} \\ 2(1 - |x|)^3 & \frac{1}{2} \leq |x| \leq 1 \\ 0 & \text{otherwise} \end{cases} \\
l(n) &= n^{1/5}
\end{aligned}
$$

Quadratic spectral: KERNEL=QS

$$
\begin{aligned}
w(x) &= \frac{25}{12\pi^2 x^2} \left(\frac{\sin(6\pi x/5)}{6\pi x/5} - \cos(6\pi x/5) \right) \\
l(n) &= \frac{1}{2} n^{1/5}
\end{aligned}
$$

Figure 26.23 Kernels for Smoothing



Details of the properties of these and other kernels are given in Andrews (1991). Kernels are selected with the KERNEL= option; KERNEL=PARZEN is the default. The general form of the KERNEL= option is

```
KERNEL=( PARZEN | QS | BART, c, e )
```

where the $e \geq 0$ and $c \geq 0$ are used to compute the bandwidth parameter as

$$l(n) = cn^e$$

The bias of the standard error estimates increases for large bandwidth parameters. A warning message is produced for bandwidth parameters greater than $n^{\frac{1}{3}}$. For a discussion of the computation of the optimal $l(n)$, refer to Andrews (1991).

The "Newey-West" kernel (Newey and West 1987) corresponds to the Bartlett kernel with bandwidth parameter $l(n) = L + 1$. That is, if the "lag length" for the Newey-West kernel is L, then the corresponding MODEL procedure syntax is KERNEL=(bart, L+1, 0).

Andrews and Monahan (1992) show that using prewhitening in combination with GMM can improve confidence interval coverage and reduce over rejection of t statistics at the cost of inflating the variance and MSE of the estimator. Prewhitening can be performed by using the %AR macros.

For the special case that the errors are not serially correlated—that is,

$$E(e_t \otimes \mathbf{z}_t)(e_s \otimes \mathbf{z}_s) = 0 \qquad t \neq s$$

the estimate for $\mathbf{S}_n^0$ reduces to

$$\hat{\mathbf{S}}_n = \frac{1}{n}\sum_{t=1}^{n}[\mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \boldsymbol{\theta})\otimes\mathbf{z}_t][\mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \boldsymbol{\theta})\otimes\mathbf{z}_t]'$$

The option KERNEL=(*kernel*,0,) is used to select this type of estimation when using GMM.

Covariance of GMM estimators

The covariance of GMM estimators, given a general weighting matrix $\mathbf{V}_{\mathrm{G}}^{-1}$, is

$$[(\mathbf{YX})'\mathbf{V}_{\mathrm{G}}^{-1}(\mathbf{YX})]^{-1}(\mathbf{YX})'\mathbf{V}_{\mathrm{G}}^{-1}\hat{\mathbf{V}}\mathbf{V}_{\mathrm{G}}^{-1}(\mathbf{YX})[(\mathbf{YX})'\mathbf{V}_{\mathrm{G}}^{-1}(\mathbf{YX})]^{-1}$$

By default or when GENGMMV is specified, this is the covariance of GMM estimators.

If the weighting matrix is the same as $\hat{\mathbf{V}}$, then the covariance of GMM estimators becomes

$$[(\mathbf{YX})'\hat{\mathbf{V}}^{-1}(\mathbf{YX})]^{-1}$$

If NOGENGMMV is specified, this is used as the covariance estimators.

Testing Overidentifying Restrictions

Let r be the number of unique instruments times the number of equations. The value r represents the number of orthogonality conditions imposed by the GMM method. Under the assumptions of the GMM method, $r - p$ linearly independent combinations of the orthogonality should be close to zero. The GMM estimates are computed by setting these combinations to zero. When r exceeds the number of parameters to be estimated, the OBJECTIVE*N, reported at the end of the estimation, is an asymptotically valid statistic to test the null hypothesis that the overidentifying restrictions of the model are valid. The OBJECTIVE*N is distributed as a chi-square with $r - p$ degrees of freedom (Hansen 1982, p. 1049). When the GMM method is selected, the value of the overidentifying restrictions test statistic, also known as Hansen's J test statistic, and its associated number of degrees of freedom are reported together with the probability under the null hypothesis.

Iterated Generalized Method of Moments (ITGMM)

Iterated generalized method of moments is similar to the iterated versions of 2SLS, SUR, and 3SLS. The variance matrix for GMM estimation is reestimated at each iteration with the parameters determined by the GMM estimation. The iteration terminates when the variance matrix for the equation errors change less than the CONVERGE= value. Iterated generalized method of moments is selected by the ITGMM option on the FIT statement. For some indication of the small sample properties of ITGMM, see Ferson and Foerster (1993).

Simulated Method of Moments (SMM)

The SMM method uses simulation techniques in model inference and estimation. It is appropriate for estimating models in which integrals appear in the objective function, and these integrals can be approximated by simulation. There might be various reasons for integrals to appear in an objective function (for example, transformation of a latent model into an observable model, missing data, random coefficients, heterogeneity, and so on).

This simulation method can be used with all the estimation methods except full information maximum likelihood (FIML) in PROC MODEL. SMM, also known as simulated generalized method of moments (SGMM), is the default estimation method because of its nice properties.

Estimation Details

A general nonlinear model can be described as

$$\boldsymbol{\epsilon}_t = \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \boldsymbol{\theta})$$

where $\mathbf{q} \in R^g$ is a real vector valued function of $\mathbf{y}_t \in R^g$, $\mathbf{x}_t \in R^l$, $\boldsymbol{\theta} \in R^p$; g is the number of equations; l is the number of exogenous variables (lagged endogenous variables are considered exogenous here); p is the number of parameters; and t ranges from 1 to n. $\boldsymbol{\epsilon}_t$ is an unobservable disturbance vector with the following properties:

$$\begin{aligned} E(\boldsymbol{\epsilon}_t) &= 0 \\ E(\boldsymbol{\epsilon}_t \boldsymbol{\epsilon}_t^{'}) &= \boldsymbol{\Sigma} \end{aligned}$$

In many cases, it is not possible to write $\mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \boldsymbol{\theta})$ in a closed form. Instead $\mathbf{q}$ is expressed as an integral of a function $\mathbf{f}$; that is,

$$\mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \boldsymbol{\theta}) = \int \mathbf{f}(\mathbf{y}_t, \mathbf{x}_t, \boldsymbol{\theta}, \mathbf{u}_t) dP(\mathbf{u})$$

where $\mathbf{f} \in R^g$ is a real vector valued function of $\mathbf{y}_t \in R^g$, $\mathbf{x}_t \in R^l$, $\boldsymbol{\theta} \in R^p$, and $\mathbf{u}_t \in R^m$, m is the number of stochastic variables with a known distribution $P(\mathbf{u})$. Since the distribution of $\mathbf{u}$ is completely known, it is possible to simulate artificial draws from this distribution. Using such independent draws $\mathbf{u}_{ht}$, $h = 1, \ldots, H$, and the strong law of large numbers, $\mathbf{q}$ can be approximated by

$$\frac{1}{H} \sum_{h=1}^{H} \mathbf{f}(\mathbf{y}_t, \mathbf{x}_t, \boldsymbol{\theta}, \mathbf{u}_{ht}).$$

Simulated Generalized Method of Moments (SGMM)

Generalized method of moments (GMM) is widely used to obtain efficient estimates for general model systems. When the moment conditions are not readily available in closed forms but can be approximated by simulation, simulated generalized method of moments (SGMM) can be used. The SGMM estimators have the nice property of being asymptotically consistent and normally distributed even if the number of draws H is fixed (see McFadden 1989; Pakes and Pollard 1989).

Consider the nonlinear model

$$\begin{aligned} \boldsymbol{\epsilon}_t &= \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \boldsymbol{\theta}) = \frac{1}{H} \sum_{h=1}^{H} \mathbf{f}(\mathbf{y}_t, \mathbf{x}_t, \boldsymbol{\theta}, \mathbf{u}_{ht}) \\ \mathbf{z}_t &= Z(\mathbf{x}_t) \end{aligned}$$

where $\mathbf{z}_t \in R^k$ is a vector of k instruments and $\boldsymbol{\epsilon}_t$ is an unobservable disturbance vector that can be serially correlated and nonstationary. In the case of no instrumental variables, $\mathbf{z}_t$ is 1. $\mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \boldsymbol{\theta})$ is the vector of moment conditions, and it is approximated by simulation.

In general, theory suggests the following orthogonality condition

$$E(\boldsymbol{\epsilon}_t \otimes \mathbf{z}_t) = 0$$

which states that the expected crossproducts of the unobservable disturbances, $\boldsymbol{\epsilon}_t$, and functions of the observable variables are set to 0. The sample means of the crossproducts are

$$\begin{aligned} \mathbf{m}_n &= \frac{1}{n}\sum_{t=1}^{n}\mathbf{m}(\mathbf{y}_t, \mathbf{x}_t, \boldsymbol{\theta}) \\ \mathbf{m}(\mathbf{y}_t, \mathbf{x}_t, \boldsymbol{\theta}) &= \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \boldsymbol{\theta})\otimes\mathbf{z}_t \end{aligned}$$

where $\mathbf{m}(\mathbf{y}_t, \mathbf{x}_t, \boldsymbol{\theta})\in R^{gk}$. The case where $gk > p$, where p is the number of parameters, is considered here. An estimate of the true parameter vector θ^0 is the value of $\hat{\theta}$ that minimizes

$$S(\theta, V) = [n\mathbf{m}_n(\theta)]'\mathbf{V}^{-1}[n\mathbf{m}_n(\theta)]/n$$

where

$$\mathbf{V} = \mathrm{Cov}\left(\mathbf{m}(\theta^0), \mathbf{m}(\theta^0)'\right).$$

The steps for SGMM are as follows:

1. Start with a positive definite $\hat{\mathbf{V}}$ matrix. This $\hat{\mathbf{V}}$ matrix can be estimated from a consistent estimator of θ. If $\hat{\theta}$ is a consistent estimator, then $\mathbf{u}_t$ for $t = 1, \ldots, n$ can be simulated H' number of times. A consistent estimator of $\mathbf{V}$ is obtained as

$$\hat{\mathbf{V}} = \frac{1}{n}\sum_{t=1}^{n}[\frac{1}{H'}\sum_{h=1}^{H'}\mathbf{f}(\mathbf{y}_t, \mathbf{x}_t, \hat{\boldsymbol{\theta}}, \mathbf{u}_{ht})\otimes\mathbf{z}_t][\frac{1}{H'}\sum_{h=1}^{H'}\mathbf{f}(\mathbf{y}_t, \mathbf{x}_t, \hat{\boldsymbol{\theta}}, \mathbf{u}_{ht})\otimes\mathbf{z}_t]'$$

H' must be large so that this is an consistent estimator of $\mathbf{V}$.

2. Simulate H number of $\mathbf{u}_t$ for $t = 1, \ldots, n$. As shown by Gourieroux and Monfort (1993), the number of simulations H does not need to be very large. For $H = 10$, the SGMM estimator achieves 90% of the efficiency of the corresponding GMM estimator. Find $\hat{\theta}$ that minimizes the quadratic product of the moment conditions again with the weight matrix being $\hat{\mathbf{V}}^{-1}$.

$$\min_{\theta}[n\mathbf{m}_n(\theta)]'\hat{\mathbf{V}}^{-1}[n\mathbf{m}_n(\theta)]/n$$

3. The covariance matrix of $\sqrt{n}\theta$ is given as (Gourieroux and Monfort 1993)

$$\boldsymbol{\Sigma}_1^{-1}\mathbf{D}\hat{\mathbf{V}}^{-1}\mathbf{V}(\hat{\theta})\hat{\mathbf{V}}^{-1}\mathbf{D}'\boldsymbol{\Sigma}_1^{-1} + \frac{1}{H}\boldsymbol{\Sigma}_1^{-1}\mathbf{D}\hat{\mathbf{V}}^{-1}E[\mathbf{z}\otimes Var(\mathbf{f}|\mathbf{x})\otimes\mathbf{z}]\hat{\mathbf{V}}^{-1}\mathbf{D}'\boldsymbol{\Sigma}_1^{-1}$$

where $\boldsymbol{\Sigma}_1 = \mathbf{D}\hat{\mathbf{V}}^{-1}\mathbf{D}$, $\mathbf{D}$ is the matrix of partial derivatives of the residuals with respect to the parameters, $\mathbf{V}(\hat{\theta})$ is the covariance of moments from estimated parameters $\hat{\theta}$, and $Var(\mathbf{f}|\mathbf{x})$ is the covariance of moments for each observation from simulation. The first term is the variance-covariance matrix of the exact GMM estimator, and the second term accounts for the variation contributed by simulating the moments.

Implementation in PROC MODEL

In PROC MODEL, if the user specifies the GMM and NDRAW options in the FIT statement, PROC MODEL first fits the model by using N2SLS and computes $\hat{\mathbf{V}}$ by using the estimates from N2SLS and H' simulation. If NO2SLS is specified in the FIT statement, $\hat{\mathbf{V}}$ is read from VDATA= data set. If the user does not provide a $\hat{\mathbf{V}}$ matrix, the initial starting value of θ is used as the estimator for computing the $\hat{\mathbf{V}}$ matrix in step 1. If ITGMM option is specified instead of GMM, then PROC MODEL iterates from step 1 to step 3 until the $\mathbf{V}$ matrix converges.

The consistency of the parameter estimates is not affected by the variance correction shown in the second term in step 3. The correction on the variance of parameter estimates is not computed by default. To add the adjustment, use ADJSMMV option on the FIT statement. This correction is of the order of $\frac{1}{H}$ and is small even for moderate H.

The following example illustrates how to use SMM to estimate a simple regression model. Suppose the model is

$$y = a + bx + u, u \sim iid\ N(0, s^2).$$

First, consider the problem in a GMM context. The first two moments of y are easily derived:

$$\begin{aligned} E(y) &= a + bx \\ E(y^2) &= (a + bx)^2 + s^2 \end{aligned}$$

Rewrite the moment conditions in the form similar to the discussion above:

$$\begin{aligned} \epsilon_{1t} &= y_t - (a + bx_t) \\ \epsilon_{2t} &= y_t^2 - (a + bx_t)^2 - s^2 \end{aligned}$$

Then you can estimate this model by using GMM with following statements:

```
proc model data=a;
   parms a b s;
   instrument x;
   eq.m1 = y-(a+b*x);
   eq.m2 = y*y - (a+b*x)**2 - s*s;
   bound s > 0;
   fit m1 m2 / gmm;
run;
```

Now suppose you do not have the closed form for the moment conditions. Instead you can simulate the moment conditions by generating H number of simulated samples based on the parameters. Then the simulated moment conditions are

$$\begin{aligned} \epsilon_{1t} &= \frac{1}{H}\sum_{h=1}^{H}\{y_t - (a + bx_t + su_{t,h})\} \\ \epsilon_{2t} &= \frac{1}{H}\sum_{h=1}^{H}\{y_t^2 - (a + bx_t + su_{t,h})^2\} \end{aligned}$$

This model can be estimated by using SGMM with the following statements:

```
proc model data=_tmpdata;
   parms a b s;
   instrument x;
   ysim = (a+b*x) + s * rannor( 98711 );
   eq.m1 = y-ysim;
   eq.m2 = y*y - ysim*ysim;
   bound s > 0;
   fit m1 m2 / gmm ndraw=10;
run;
```

You can use the following MOMENT statement instead of specifying the two moment equations above:

```
moment ysim=(1, 2);
```

In cases where you require a large number of moment equations, using the MOMENT statement to specify them is more efficient.

Note that the NDRAW= option tells PROC MODEL that this is a simulation-based estimation. Thus, the random number function RANNOR returns random numbers in estimation process. During the simulation, 10 draws of $m1$ and $m2$ are generated for each observation, and the averages enter the objective functions just as the equations specified previously.

Other Estimation Methods

The simulation method can be used not only with GMM and ITGMM, but also with OLS, ITOLS, SUR, ITSUR, N2SLS, IT2SLS, N3SLS, and IT3SLS. These simulation-based methods are similar to the corresponding methods in PROC MODEL; the only difference is that the objective functions include the average of the H simulations.

Full Information Maximum Likelihood Estimation (FIML)

A different approach to the simultaneous equation bias problem is the full information maximum likelihood (FIML) estimation method (Amemiya 1977).

Compared to the instrumental variables methods (2SLS and 3SLS), the FIML method has these advantages and disadvantages:

- FIML does not require instrumental variables.
- FIML requires that the model include the full equation system, with as many equations as there are endogenous variables. With 2SLS or 3SLS, you can estimate some of the equations without specifying the complete system.
- FIML assumes that the equations errors have a multivariate normal distribution. If the errors are not normally distributed, the FIML method might produce poor results. 2SLS and 3SLS do not assume a specific distribution for the errors.
- The FIML method is computationally expensive.

The full information maximum likelihood estimators of θ and σ are the $\hat{\theta}$ and $\hat{\sigma}$ that minimize the negative log-likelihood function:

$$\begin{aligned}\mathbf{l}_n(\boldsymbol{\theta},\boldsymbol{\sigma}) = \quad & \tfrac{ng}{2} \quad \ln(2\pi) - \sum_{t=1}^{n} \ln\left(\left|\frac{\partial \mathbf{q}(\mathbf{y}_t,\mathbf{x}_t,\boldsymbol{\theta})}{\partial \mathbf{y}_t'}\right|\right) + \frac{n}{2}\ln\left(|\boldsymbol{\Sigma}(\sigma)|\right) \\ + \quad & \frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Sigma}(\sigma)^{-1}\sum_{t=1}^{n}\mathbf{q}(\mathbf{y}_t,\mathbf{x}_t,\boldsymbol{\theta})\mathbf{q}'(\mathbf{y}_t,\mathbf{x}_t,\boldsymbol{\theta})\right)\end{aligned}$$

The option FIML requests full information maximum likelihood estimation. If the errors are distributed normally, FIML produces efficient estimators of the parameters. If instrumental variables are not provided, the starting values for the estimation are obtained from a SUR estimation. If instrumental variables are provided, then the starting values are obtained from a 3SLS estimation. The log-likelihood value and the l_2 norm of the gradient of the negative log-likelihood function are shown in the estimation summary.

FIML Details

To compute the minimum of $\mathbf{l}_n(\boldsymbol{\theta},\boldsymbol{\sigma})$, this function is *concentrated* using the relation

$$\boldsymbol{\Sigma}(\theta) = \frac{1}{n}\sum_{t=1}^{n}\mathbf{q}(\mathbf{y}_t,\mathbf{x}_t,\boldsymbol{\theta})\mathbf{q}'(\mathbf{y}_t,\mathbf{x}_t,\boldsymbol{\theta})$$

This results in the concentrated negative log-likelihood function discussed in Davidson and MacKinnon (1993):

$$\mathbf{l}_n(\boldsymbol{\theta}) = \frac{ng}{2}(1+\ln(2\pi)) - \sum_{t=1}^{n}\ln\left|\frac{\partial}{\partial \mathbf{y}_t'}\mathbf{q}(\mathbf{y}_t,\mathbf{x}_t,\boldsymbol{\theta})\right| + \frac{n}{2}\ln|\boldsymbol{\Sigma}(\theta)|$$

The gradient of the negative log-likelihood function is

$$\frac{\partial}{\partial\theta_i}\mathbf{l}_n(\boldsymbol{\theta}) = \sum_{t=1}^{n}\nabla_i(t)$$

$$\begin{aligned}\nabla_i(t) \quad = \quad & -\mathrm{tr}\left(\left(\frac{\partial \mathbf{q}(\mathbf{y}_t,\mathbf{x}_t,\boldsymbol{\theta})}{\partial \mathbf{y}_t'}\right)^{-1}\frac{\partial^2\mathbf{q}(\mathbf{y}_t,\mathbf{x}_t,\boldsymbol{\theta})}{\partial \mathbf{y}_t'\partial\theta_i}\right) \\ + \quad & \frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Sigma}(\theta)^{-1}\frac{\partial\boldsymbol{\Sigma}(\theta)}{\partial\theta_i}\right. \\ & \left.\left[I - \boldsymbol{\Sigma}(\theta)^{-1}\mathbf{q}(\mathbf{y}_t,\mathbf{x}_t,\boldsymbol{\theta})\mathbf{q}(\mathbf{y}_t,\mathbf{x}_t,\boldsymbol{\theta})'\right]\right) \\ + \quad & \mathbf{q}(\mathbf{y}_t,\mathbf{x}_t,\boldsymbol{\theta}')\boldsymbol{\Sigma}(\theta)^{-1}\frac{\partial\mathbf{q}(\mathbf{y}_t,\mathbf{x}_t,\boldsymbol{\theta})}{\partial\theta_i}\end{aligned}$$

where

$$\frac{\partial\boldsymbol{\Sigma}(\theta)}{\partial\theta_i} = \frac{2}{n}\sum_{t=1}^{n}\mathbf{q}(\mathbf{y}_t,\mathbf{x}_t,\boldsymbol{\theta})\frac{\partial\mathbf{q}(\mathbf{y}_t,\mathbf{x}_t,\boldsymbol{\theta})'}{\partial\theta_i}$$

The estimator of the variance-covariance of $\hat{\theta}$ (COVB) for FIML can be selected with the COVBEST= option with the following arguments:

CROSS selects the crossproducts estimator of the covariance matrix (Gallant 1987, p. 473):

$$C = \left(\frac{1}{n} \sum_{t=1}^{n} \nabla(t) \nabla'(t) \right)^{-1}$$

where $\nabla(t) = [\nabla_1(t), \nabla_2(t), \ldots, \nabla_p(t)]'$. This is the default.

GLS selects the generalized least squares estimator of the covariance matrix. This is computed as (Dagenais 1978)

$$C = [\hat{\mathbf{Z}}'(\mathbf{\Sigma}(\theta)^{-1} \otimes I)\hat{\mathbf{Z}}]^{-1}$$

where $\hat{\mathbf{Z}} = (\hat{Z}_1, \hat{Z}_2, \ldots, \hat{Z}_p)$ is $ng \times p$ and each $\hat{Z}_i$ column vector is obtained from stacking the columns of

$$\mathbf{U} \frac{1}{n} \sum_{t=1}^{n} \left(\frac{\partial \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \boldsymbol{\theta})'}{\partial y} \right)^{-1} \frac{\partial^2 \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \boldsymbol{\theta})'}{\partial \mathbf{y}_n' \partial \theta_i} - Q_i$$

$\mathbf{U}$ is an $n \times g$ matrix of residuals and q_i is an $n \times g$ matrix $\frac{\partial \mathbf{Q}}{\partial \theta_i}$.

FDA selects the inverse of concentrated likelihood Hessian as an estimator of the covariance matrix. The Hessian is computed numerically, so for a large problem this is computationally expensive.

The HESSIAN= option controls which approximation to the Hessian is used in the minimization procedure. Alternate approximations are used to improve convergence and execution time. The choices are as follows:

CROSS The crossproducts approximation is used.

GLS The generalized least squares approximation is used (default).

FDA The Hessian is computed numerically by finite differences.

HESSIAN=GLS has better convergence properties in general, but COVBEST=CROSS produces the most pessimistic standard error bounds. When the HESSIAN= option is used, the default estimator of the variance-covariance of $\hat{\theta}$ is the inverse of the Hessian selected.

Multivariate t Distribution Estimation

The multivariate *t* distribution is specified by using the ERRORMODEL statement with the T option. Other method specifications (FIML and OLS, for example) are ignored when the ERRORMODEL statement is used for a distribution other than normal.

The probability density function for the multivariate *t* distribution is

$$P_q = \frac{\Gamma(\frac{df+m}{2})}{(\pi * df)^{\frac{m}{2}} * \Gamma(\frac{df}{2}) |\mathbf{\Sigma}(\sigma)|^{\frac{1}{2}}} * \left(1 + \frac{\mathbf{q}'(\mathbf{y}_t, \mathbf{x}_t, \boldsymbol{\theta}) \mathbf{\Sigma}(\sigma)^{-1} \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \boldsymbol{\theta})}{df} \right)^{-\frac{df+m}{2}}$$

where m is the number of equations and df is the degrees of freedom.

The maximum likelihood estimators of θ and σ are the $\hat{\theta}$ and $\hat{\sigma}$ that minimize the negative log-likelihood function:

$$\begin{aligned}\mathbf{l}_n(\boldsymbol{\theta},\boldsymbol{\sigma}) &= -\sum_{t=1}^{n}\ln\left(\frac{\Gamma(\frac{df+m}{2})}{(\pi * df)^{\frac{m}{2}} * \Gamma(\frac{df}{2})} * \left(1+\frac{q_t'\boldsymbol{\Sigma}^{-1}q_t}{df}\right)^{-\frac{df+m}{2}}\right)\\ &\quad +\frac{n}{2} * \ln(|\boldsymbol{\Sigma}|) - \sum_{t=1}^{n}\ln\left(\left|\frac{\partial q_t}{\partial y_t'}\right|\right)\end{aligned}$$

The ERRORMODEL statement is used to request the *t* distribution maximum likelihood estimation. An OLS estimation is done to obtain initial parameter estimates and MSE.*var* estimates. Use NOOLS to turn off this initial estimation. If the errors are distributed normally, *t* distribution estimation produces results similar to FIML.

The multivariate model has a single shared degrees-of-freedom parameter, which is estimated. The degrees-of-freedom parameter can also be set to a fixed value. The log-likelihood value and the l_2 norm of the gradient of the negative log-likelihood function are shown in the estimation summary.

t Distribution Details

Since a variance term is explicitly specified by using the ERRORMODEL statement, $\boldsymbol{\Sigma}(\theta)$ is estimated as a correlation matrix and $\mathbf{q}(\mathbf{y}_t,\mathbf{x}_t,\boldsymbol{\theta})$ is normalized by the variance. The gradient of the negative log-likelihood function with respect to the degrees of freedom is

$$\begin{aligned}\frac{\partial l_n}{\partial df} &= \frac{nm}{2\,df} - \frac{n}{2}\frac{\Gamma'(\frac{df+m}{2})}{\Gamma(\frac{df+m}{2})} + \frac{n}{2}\frac{\Gamma'(\frac{df}{2})}{\Gamma(\frac{df}{2})} +\\ &\quad 0.5\log(1+\frac{\mathbf{q}'\boldsymbol{\Sigma}^{-1}\mathbf{q}}{df}) - \frac{0.5(df+m)}{(1+\frac{\mathbf{q}'\boldsymbol{\Sigma}^{-1}\mathbf{q}}{df})}\frac{\mathbf{q}'\boldsymbol{\Sigma}^{-1}\mathbf{q}}{df^2}\end{aligned}$$

The gradient of the negative log-likelihood function with respect to the parameters is

$$\frac{\partial l_n}{\partial\theta_i} = \frac{0.5(df+m)}{(1+\mathbf{q}'\boldsymbol{\Sigma}^{-1}\mathbf{q}/df)}\left[\frac{(2\,\mathbf{q}'\boldsymbol{\Sigma}^{-1}\frac{\partial\mathbf{q}}{\partial\theta_i})}{df} + \mathbf{q}'\boldsymbol{\Sigma}^{-1}\frac{\partial\boldsymbol{\Sigma}}{\partial\theta_i}\boldsymbol{\Sigma}^{-1}\mathbf{q}\right] - \frac{n}{2}\mathrm{trace}(\boldsymbol{\Sigma}^{-1}\frac{\partial\boldsymbol{\Sigma}}{\partial\theta_i})$$

where

$$\frac{\partial\boldsymbol{\Sigma}(\theta)}{\partial\theta_i} = \frac{2}{n}\sum_{t=1}^{n}\mathbf{q}(\mathbf{y}_t,\mathbf{x}_t,\boldsymbol{\theta})\frac{\partial\mathbf{q}(\mathbf{y}_t,\mathbf{x}_t,\boldsymbol{\theta})'}{\partial\theta_i}$$

and

$$\mathbf{q}(\mathbf{y}_t,\mathbf{x}_t,\boldsymbol{\theta}) = \frac{\epsilon(\theta)}{\sqrt{h(\theta)}} \in R^{m\times n}$$

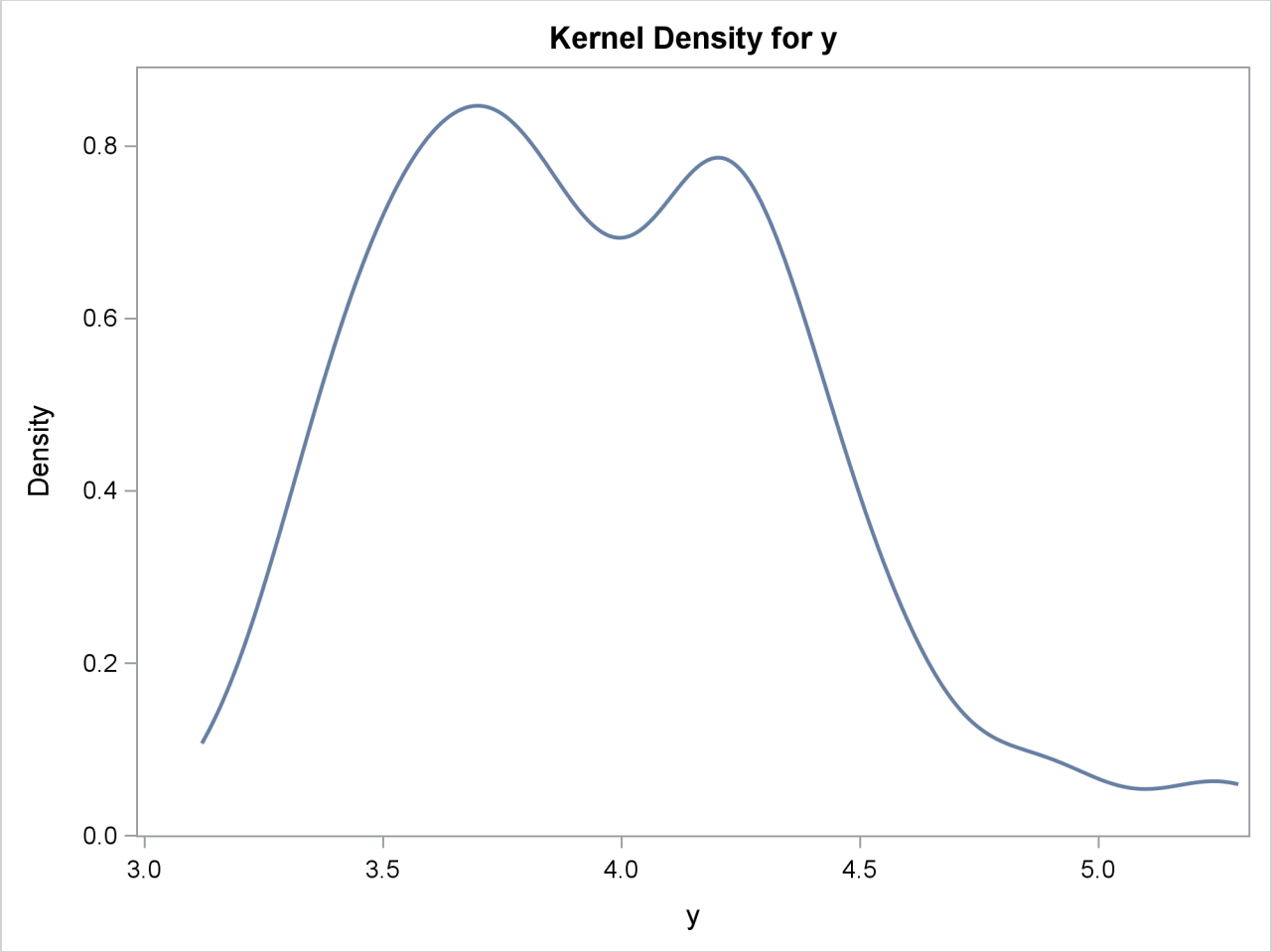
The estimator of the variance-covariance of $\hat{\theta}$ (COVB) for the *t* distribution is the inverse of the likelihood Hessian. The gradient is computed analytically, and the Hessian is computed numerically.

Empirical Distribution Estimation and Simulation

The following SAS statements fit a model that uses least squares as the likelihood function, but represent the distribution of the residuals with an empirical cumulative distribution function (CDF). The plot of the empirical probability distribution is shown in Figure 26.24.

```
data t;  /* Sum of two normals  */
   format date monyy.;
   do t = 0 to 9.9 by 0.1;
      date = intnx( 'month', '1jun90'd, (t*10)-1 );
      y =  0.1 * (rannor(123)-10) +
            .5 * (rannor(123)+10);
      output;
   end;
run;

ods select Model.Liklhood.ResidSummary
           Model.Liklhood.ParameterEstimates;

proc model data=t time=t itprint;
   dependent y;
   parm a 5;

   y = a;
   obj = resid.y * resid.y;
   errormodel y ~ general( obj )
   cdf=(empirical=(tails=( normal percent=10)));

   fit y / outsn=s out=r;
   id  date;

   solve y / data=t(where=(date='1aug98'd))
             residdata=r sdata=s
             random=200 seed=6789 out=monte ;
run;


proc kde data=monte;
   univar y / plots=density;
run;
```

Figure 26.24 Empirical PDF Plot



For simulation, if the CDF for the model is not built in to the procedure, you can use the CDF=EMPIRICAL() option. This uses the sorted residual data to create an empirical CDF. For computing the inverse CDF, the program needs to know how to handle the tails. For continuous data, the tail distribution is generally poorly determined. To counter this, the PERCENT= option specifies the percentage of the observations to use in constructing each tail. The default for the PERCENT= option is 10.

A normal distribution or a *t* distribution is used to extrapolate the tails to infinity. The standard errors for this extrapolation are obtained from the data so that the empirical CDF is continuous.

Properties of the Estimates

All of the methods are consistent. Small sample properties might not be good for nonlinear models. The tests and standard errors reported are based on the convergence of the distribution of the estimates to a normal distribution in large samples.

These nonlinear estimation methods reduce to the corresponding linear systems regression methods if the model is linear. If this is the case, PROC MODEL produces the same estimates as PROC SYSLIN.

Except for GMM, the estimation methods assume that the equation errors for each observation are identically and independently distributed with a 0 mean vector and positive definite covariance matrix $\mathbf{\Sigma}$ consistently estimated by $\mathbf{S}$. For FIML, the errors need to be normally distributed. There are no other assumptions concerning the distribution of the errors for the other estimation methods.

The consistency of the parameter estimates relies on the assumption that the $\mathbf{S}$ matrix is a consistent estimate of $\mathbf{\Sigma}$. These standard error estimates are asymptotically valid, but for nonlinear models they might not be reliable for small samples.

The $\mathbf{S}$ matrix used for the calculation of the covariance of the parameter estimates is the best estimate available for the estimation method selected. For $\mathbf{S}$-iterated methods, this is the most recent estimation of $\mathbf{\Sigma}$. For OLS and 2SLS, an estimate of the $\mathbf{S}$ matrix is computed from OLS or 2SLS residuals and used for the calculation of the covariance matrix. For a complete list of the $\mathbf{S}$ matrix used for the calculation of the covariance of the parameter estimates, see Table 26.2.

Missing Values

An observation is excluded from the estimation if any variable used for FIT tasks is missing, if the weight for the observation is not greater than 0 when weights are used, or if a DELETE statement is executed by the model program. Variables used for FIT tasks include the equation errors for each equation, the instruments, if any, and the derivatives of the equation errors with respect to the parameters estimated. Note that variables can become missing as a result of computational errors or calculations with missing values.

The number of usable observations can change when different parameter values are used; some parameter values can be invalid and cause execution errors for some observations. PROC MODEL keeps track of the number of usable and missing observations at each pass through the data, and if the number of missing observations counted during a pass exceeds the number that was obtained using the previous parameter vector, the pass is terminated and the new parameter vector is considered infeasible. PROC MODEL never takes a step that produces more missing observations than the current estimate does.

The values used to compute the Durbin-Watson, R^2, and other statistics of fit are from the observations used in calculating the objective function and do not include any observation for which any needed variable was missing (residuals, derivatives, and instruments).

Details on the Covariance of Equation Errors

There are several $\mathbf{S}$ matrices that can be involved in the various estimation methods and in forming the estimate of the covariance of parameter estimates. These $\mathbf{S}$ matrices are estimates of $\mathbf{\Sigma}$, the true covariance of the equation errors. Apart from the choice of instrumental or noninstrumental methods, many of the methods provided by PROC MODEL differ in the way the various $\mathbf{S}$ matrices are formed and used.

All of the estimation methods result in a final estimate of $\mathbf{\Sigma}$, which is included in the output if the COVS option is specified. The final $\mathbf{S}$ matrix of each method provides the initial $\mathbf{S}$ matrix for any subsequent estimation.

This estimate of the covariance of equation errors is defined as

$$\mathbf{S} = \mathbf{D}(\mathbf{R}'\mathbf{R})\mathbf{D}$$

where $\mathbf{R} = (\mathbf{r}_1, \ldots, \mathbf{r}_g)$ is composed of the equation residuals computed from the current parameter estimates in an $n \times g$ matrix and $\mathbf{D}$ is a diagonal matrix that depends on the VARDEF= option.

For VARDEF=N, the diagonal elements of **D** are $1/\sqrt{n}$, where n is the number of nonmissing observations. For VARDEF=WGT, n is replaced with the sum of the weights. For VARDEF=WDF, n is replaced with the sum of the weights minus the model degrees of freedom. For the default VARDEF=DF, the ith diagonal element of **D** is $1/\sqrt{n - df_i}$, where df_i is the degrees of freedom (number of parameters) for the ith equation. Binkley and Nelson (1984) show the importance of using a degrees-of-freedom correction in estimating $\mathbf{\Sigma}$. Their results indicate that the DF method produces more accurate confidence intervals for N3SLS parameter estimates in the linear case than the alternative approach they tested. VARDEF=N is always used for the computation of the FIML estimates.

For the fixed **S** methods, the OUTSUSED= option writes the **S** matrix used in the estimation to a data set. This **S** matrix is either the estimate of the covariance of equation errors matrix from the preceding estimation, or a prior $\mathbf{\Sigma}$ estimate read in from a data set when the SDATA= option is specified. For the diagonal **S** methods, all of the off-diagonal elements of the **S** matrix are set to 0 for the estimation of the parameters and for the OUTSUSED= data set, but the output data set produced by the OUTS= option contains the off-diagonal elements. For the OLS and N2SLS methods, there is no previous estimate of the covariance of equation errors matrix, and the option OUTSUSED= saves an identity matrix unless a prior $\mathbf{\Sigma}$ estimate is supplied by the SDATA= option. For FIML, the OUTSUSED= data set contains the **S** matrix computed with VARDEF=N. The OUTS= data set contains the **S** matrix computed with the selected VARDEF= option. Both versions of the **S** matrix appear in the printed output for FIML.

If the COVS option is used, the method is not **S**-iterated, **S** is not an identity, and the OUTSUSED= matrix is included in the printed output.

For the methods that iterate the covariance of equation errors matrix, the **S** matrix is iteratively re-estimated from the residuals produced by the current parameter estimates. This **S** matrix estimate iteratively replaces the previous estimate until both the parameter estimates and the estimate of the covariance of equation errors matrix converge. The final OUTS= matrix and OUTSUSED= matrix are thus identical for the **S**-iterated methods.

Nested Iterations

By default, for **S**-iterated methods, the **S** matrix is held constant until the parameters converge once. Then the **S** matrix is reestimated. One iteration of the parameter estimation algorithm is performed, and the **S** matrix is again reestimated. This latter process is repeated until convergence of both the parameters and the **S** matrix. Since the objective of the minimization depends on the **S** matrix, this has the effect of chasing a moving target.

When the NESTIT option is specified, iterations are performed to convergence for the structural parameters with a fixed **S** matrix. The **S** matrix is then reestimated, the parameter iterations are repeated to convergence, and so on until both the parameters and the **S** matrix converge. This has the effect of fixing the objective function for the inner parameter iterations. It is more reliable, but usually more expensive, to nest the iterations.

R-Square Statistic

For unrestricted linear models with an intercept successfully estimated by OLS, R^2 is always between 0 and 1. However, nonlinear models do not necessarily encompass the dependent mean as a special case and can produce negative R^2 statistics. Negative R^2 statistics can also be produced even for linear models when an estimation method other than OLS is used and no intercept term is in the model.

R^2 is defined for normalized equations as

$$R^2 = 1 - \frac{SSE}{SSA - \bar{y}^2 \times n}$$

where SSA is the sum of the squares of the actual y's and $\bar{y}$ are the actual means. R^2 cannot be computed for models in general form because of the need for an actual Y.

Minimization Methods

PROC MODEL currently supports two methods for minimizing the objective function. These methods are described in the following sections.

GAUSS

The Gauss-Newton parameter-change vector for a system with g equations, n nonmissing observations, and p unknown parameters is

$$\mathbf{\Delta} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{r}$$

where $\mathbf{\Delta}$ is the change vector, $\mathbf{X}$ is the stacked $ng \times p$ Jacobian matrix of partial derivatives of the residuals with respect to the parameters, and $\mathbf{r}$ is an $ng \times 1$vector of the stacked residuals. The components of $\mathbf{X}$ and $\mathbf{r}$ are weighted by the $\mathbf{S}^{-1}$ matrix. When instrumental methods are used, $\mathbf{X}$ and $\mathbf{r}$ are the projections of the Jacobian matrix and residuals vector in the instruments space and not the Jacobian and residuals themselves. In the preceding formula, $\mathbf{S}$ and $\mathbf{W}$ are suppressed. If instrumental variables are used, then the change vector becomes:

$$\mathbf{\Delta} = (\mathbf{X}'(\mathbf{S}^{-1}\otimes\mathbf{W})\mathbf{X})^{-1}\mathbf{X}'(\mathbf{S}^{-1}\otimes\mathbf{W})\mathbf{r}$$

This vector is computed at the end of each iteration. The objective function is then computed at the changed parameter values at the start of the next iteration. If the objective function is not improved by the change, the $\mathbf{\Delta}$ vector is reduced by one-half and the objective function is reevaluated. The change vector will be halved up to MAXSUBITER= times until the objective function is improved. If the objective function cannot be improved after MAXSUBITER= times, the procedure switches to the MARQUARDT method described in the next section to further improve the objective function.

For FIML, the $\mathbf{X}'\mathbf{X}$ matrix is substituted with one of three choices for approximations to the Hessian. See the section "Full Information Maximum Likelihood Estimation (FIML)" on page 1547 in this chapter.

MARQUARDT

The Marquardt-Levenberg parameter change vector is

$$\mathbf{\Delta} = (\mathbf{X}'\mathbf{X} + \lambda\mathrm{diag}(\mathbf{X}'\mathbf{X}))^{-1}\mathbf{X}'\mathbf{r}$$

where $\mathbf{\Delta}$ is the change vector, and $\mathbf{X}$ and $\mathbf{r}$ are the same as for the Gauss-Newton method, described in the preceding section. Before the iterations start, λ is set to a small value (1E–6). At each iteration, the objective function is evaluated at the parameters changed by $\mathbf{\Delta}$. If the objective function is not improved, λ is increased to 10λ and the step is tried again. λ can be increased up to MAXSUBITER= times to a maximum of 1E15 (whichever comes first) until the objective function is improved. For the start of the next iteration, λ is reduced to max(λ/10,1E–10).

Convergence Criteria

There are a number of measures that could be used as convergence or stopping criteria. PROC MODEL computes five convergence measures labeled R, S, PPC, RPC, and OBJECT.

When an estimation technique that iterates estimates of $\mathbf{\Sigma}$ is used (that is, IT3SLS), two convergence criteria are used. The termination values can be specified with the CONVERGE=(p,s) option in the FIT statement. If the second value, s, is not specified, it defaults to p. The criterion labeled S (described later in the section) controls the convergence of the **S** matrix. When S is less than s, the **S** matrix has converged. The criterion labeled R is compared to the p-value to test convergence of the parameters.

The R convergence measure cannot be computed accurately in the special case of singular residuals (when all the residuals are close to 0) or in the case of a 0 objective value. When either the trace of the **S** matrix computed from the current residuals (trace(S)) or the objective value is less than the value of the SINGULAR= option, convergence is assumed.

The various convergence measures are explained in the following:

R is the primary convergence measure for the parameters. It measures the degree to which the residuals are orthogonal to the Jacobian columns, and it approaches 0 as the gradient of the objective function becomes small. R is defined as the square root of

$$\frac{(r'(\mathbf{S}^{-1}\otimes\mathbf{W})\mathbf{X}(\mathbf{X}'(\mathbf{S}^{-1}\otimes\mathbf{W})\mathbf{X})^{-1}\mathbf{X}'(\mathbf{S}^{-1}\otimes\mathbf{W})r)}{(r'(\mathbf{S}^{-1}\otimes\mathbf{W})r)}$$

where **X** is the Jacobian matrix and **r** is the residuals vector. R is similar to the relative offset orthogonality convergence criterion proposed by Bates and Watts (1981).

In the univariate case, the R measure has several equivalent interpretations:

- the cosine of the angle between the residuals vector and the column space of the Jacobian matrix. When this cosine is 0, the residuals are orthogonal to the partial derivatives of the predicted values with respect to the parameters, and the gradient of the objective function is 0.
- the square root of the R^2 for the current linear pseudo-model in the residuals
- a norm of the gradient of the objective function, where the normalizing matrix is proportional to the current estimate of the covariance of the parameter estimates. Thus, using R, convergence is judged when the gradient becomes small in this norm.
- the prospective relative change in the objective function value expected from the next GAUSS step, assuming that the current linearization of the model is a good local approximation.

In the multivariate case, R is somewhat more complicated but is designed to go to 0 as the gradient of the objective becomes small and can still be given the previous interpretations for the aggregation of the equations weighted by $\mathbf{S}^{-1}$.

PPC is the prospective parameter change measure. PPC measures the maximum relative change in the parameters implied by the parameter-change vector computed for the next iteration.

At the *k*th iteration, PPC is the maximum over the parameters

$$\frac{\left|\theta_i^{k+1} - \theta_i^k\right|}{\left|\theta_i^k\right| + 10^{-6}}$$

where θ_i^k is the current value of the *i*th parameter and θ_i^{k+1} is the prospective value of this parameter after adding the change vector computed for the next iteration. The parameter with the maximum prospective relative change is printed with the value of PPC, unless the PPC is nearly 0.

RPC is the retrospective parameter change measure. RPC measures the maximum relative change in the parameters from the previous iteration. At the *k*th iteration, RPC is the maximum over *i* of

$$\frac{\left|\theta_i^k - \theta_i^{k-1}\right|}{\left|\theta_i^{k-1}\right| + 10^{-6}}$$

where θ_i^k is the current value of the *i*th parameter and θ_i^{k-1} is the previous value of this parameter. The name of the parameter with the maximum retrospective relative change is printed with the value of RPC, unless the RPC is nearly 0.

OBJECT measures the relative change in the objective function value between iterations:

$$\frac{\left|O^k - O^{k-1}\right|}{\left|O^{k-1}\right| + 10^{-6}}$$

where O^{k-1} is the value of the objective function (O^k) from the previous iteration.

S measures the relative change in the **S** matrix. S is computed as the maximum over *i, j* of

$$\frac{\left|S_{ij}^k - S_{ij}^{k-1}\right|}{\left|S_{ij}^{k-1}\right| + 10^{-6}}$$

where S^{k-1} is the previous **S** matrix. The S measure is relevant only for estimation methods that iterate the **S** matrix.

An example of the convergence criteria output is shown in Figure 26.25.

Figure 26.25 Convergence Criteria Output

Final Convergence Criteria	
R	0.000737
PPC(b)	0.003943
RPC(b)	0.00968
Object	4.784E-6
Trace(S)	0.533325
Objective Value	0.522214

The Trace(S) is the trace (the sum of the diagonal elements) of the **S** matrix computed from the current residuals. This row is labeled MSE if there is only one equation.

Troubleshooting Convergence Problems

As with any nonlinear estimation routine, there is no guarantee that the estimation will be successful for a given model and data. If the equations are linear with respect to the parameters, the parameter estimates always converge in one iteration. The methods that iterate the **S** matrix must iterate further for the **S** matrix to converge. Nonlinear models might not necessarily converge.

Convergence can be expected only with fully identified parameters, adequate data, and starting values sufficiently close to solution estimates.

Convergence and the rate of convergence might depend primarily on the choice of starting values for the estimates. This does not mean that a great deal of effort should be invested in choosing starting values. First, try the default values. If the estimation fails with these starting values, examine the model and data and rerun the estimation using reasonable starting values. It is usually not necessary that the starting values be very good, just that they not be very bad; choose values that seem plausible for the model and data.

An Example of Requiring Starting Values

Suppose you want to regress a variable Y on a variable X, assuming that the variables are related by the following nonlinear equation:

$$y = a + bx^c + \epsilon$$

In this equation, Y is linearly related to a power transformation of X. The unknown parameters are a, b, and c. ϵ is an unobserved random error. The following SAS statements generate simulated data. In this simulation, $a = 10$, $b = 2$, and the use of the SQRT function corresponds to $c = .5$.

```
data test;
  do i = 1 to 20;
     x = 5 * ranuni(1234);
     y = 10 + 2 * sqrt(x) + .5 * rannor(1234);
     output;
     end;
run;
```

The following statements specify the model and give descriptive labels to the model parameters. Then the FIT statement attempts to estimate a, b, and c by using the default starting value 0.0001.

```
proc model data=test;
   y = a + b * x ** c;
   label a = "Intercept"
         b = "Coefficient of Transformed X"
         c = "Power Transformation Parameter";
   fit y;
run;
```

PROC MODEL prints model summary and estimation problem summary reports and then prints the output shown in Figure 26.26.

Figure 26.26 Diagnostics for Convergence Failure

The MODEL Procedure
OLS Estimation

ERROR: The parameter estimates failed to converge for OLS after 100 iterations using CONVERGE=0.001 as the convergence criteria.

The MODEL Procedure
OLS Estimation

	Iteration	N Obs	R	Objective	N Subit	a	b	c
OLS	100	20	0.9627	3.9678	2	137.3822	-126.533	-0.00213

Gauss Method Parameter Change Vector

a	b	c
-69369.08	69368.01	-1.16

By using the default starting values, PROC MODEL is unable to take even the first step in iterating to the solution. The change in the parameters that the Gauss-Newton method computes is very extreme and makes the objective values worse instead of better. Even when this step is shortened by a factor of a million, the objective function is still worse, and PROC MODEL is unable to estimate the model parameters.

The problem is caused by the starting value of C. Using the default starting value C=0.0001, the first iteration attempts to compute better values of A and B by what is, in effect, a linear regression of Y on the 10,000th root of X, which is almost the same as the constant 1. Thus the matrix that is inverted to compute the changes is nearly singular and affects the accuracy of the computed parameter changes.

This is also illustrated by the next part of the output, which displays collinearity diagnostics for the crossproducts matrix of the partial derivatives with respect to the parameters, shown in Figure 26.27.

Figure 26.27 Collinearity Diagnostics

Collinearity Diagnostics

			Proportion of Variation		
Number	Eigenvalue	Condition Number	a	b	c
1	2.376793	1.0000	0.0000	0.0000	0.0000
2	0.623207	1.9529	0.0000	0.0000	0.0000
3	1.68475E-12	1187758	1.0000	1.0000	1.0000

This output shows that the matrix is singular and that the partials of A, B, and C with respect to the residual are collinear at the point (0.0001, 0.0001, 0.0001) in the parameter space. See the section "Linear Dependencies" on page 1566 for a full explanation of the collinearity diagnostics.

The MODEL procedure next prints the note shown in Figure 26.28, which suggests that you try different starting values.

Figure 26.28 Estimation Failure Note

Note: The parameter estimation is abandoned. Check your model and data. If the model is correct and the input data are appropriate, try rerunning the parameter estimation using different starting values for the parameter estimates.
PROC MODEL continues as if the parameter estimates had converged.

PROC MODEL then produces the usual printout of results for the nonconverged parameter values. The estimation summary is shown in Figure 26.29. The heading includes the reminder "(Not Converged)".

Figure 26.29 Nonconverged Estimation Summary

The MODEL Procedure
OLS Estimation Summary (Not Converged)

Data Set Options	
DATA=	TEST

Minimization Summary	
Parameters Estimated	3
Method	Gauss
Iterations	100
Subiterations	239
Average Subiterations	2.39

Final Convergence Criteria	
R	0.962666
PPC(b)	548.2193
RPC(b)	540.3066
Object	2.654E-6
Trace(S)	4.667946
Objective Value	3.967754

Observations Processed	
Read	20
Solved	20

The nonconverged estimation results are shown in Figure 26.30.

Figure 26.30 Nonconverged Results

The MODEL Procedure

Nonlinear OLS Summary of Residual Errors (Not Converged)

Equation	DF Model	DF Error	SSE	MSE	Root MSE	R-Square	Adj R-Sq
y	3	17	79.3551	4.6679	2.1605	-1.6812	-1.9966

Note that the R^2 statistic is negative. An $R^2 < 0$ results when the residual mean squared error for the model is larger than the variance of the dependent variable. Negative R^2 statistics might be produced when either

the parameter estimates fail to converge correctly, as in this case, or when the correctly estimated model fits the data very poorly.

Controlling Starting Values

To fit the preceding model you must specify a better starting value for C. Avoid starting values of C that are either very large or close to 0. For starting values of A and B, you can specify values, use the default, or have PROC MODEL fit starting values for them conditional on the starting value for C.

Starting values are specified with the START= option of the FIT statement or in a PARMS statement. In PROC MODEL, you have several options to specify starting values for the parameters to be estimated. When more than one option is specified, the options are implemented in the following order of precedence (from highest to lowest): the START= option, the PARMS statement initialization value, the ESTDATA= option, and the PARMSDATA= option. When no starting values for the parameter estimates are specified with BY group processing, the default start value 0.0001 is used for each by group. Again, when no starting values are specified, and a model with a FIT statement is stored by the OUTMODEL=*outmodel-filename* option in a previous step, the *outmodel-filename* can be invoked in a subsequent PROC MODEL step by using the MODEL=*outmodel-filename* option with multiple estimation methods in the second step. In such a case, the parameter estimates from the *outmodel-filename* are used directly as starting values for OLS, and OLS results from the second step provide starting values for the subsequent estimation method such as 2SLS or SUR, provided that NOOLS is not specified.

For example, the following statements estimate the model parameters by using the starting values A=0.0001, B=0.0001, and C=5.

```
proc model data=test;
   y = a + b * x ** c;
   label a = "Intercept"
         b = "Coefficient of Transformed X"
         c = "Power Transformation Parameter";
   fit y start=(c=5);
run;
```

Using these starting values, the estimates converge in 16 iterations. The results are shown in Figure 26.31. Note that since the START= option explicitly declares parameters, the parameter C is placed first in the table.

Figure 26.31 Converged Results

The MODEL Procedure

Nonlinear OLS Summary of Residual Errors

Equation	DF Model	DF Error	SSE	MSE	Root MSE	R-Square	Adj R-Sq
y	3	17	5.7359	0.3374	0.5809	0.8062	0.7834

Nonlinear OLS Parameter Estimates

Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|	Label
c	0.327079	0.2892	1.13	0.2738	Power Transformation Parameter
a	8.384311	3.3775	2.48	0.0238	Intercept
b	3.505391	3.4858	1.01	0.3287	Coefficient of Transformed X

Using the STARTITER Option

PROC MODEL can compute starting values for some parameters conditional on starting values you specify for the other parameters. You supply starting values for some parameters and specify the STARTITER option on the FIT statement.

For example, the following statements set C to 1 and compute starting values for A and B by estimating these parameters conditional on the fixed value of C. With C=1, this is equivalent to computing A and B by linear regression on X. A PARMS statement is used to declare the parameters in alphabetical order. The ITPRINT option is used to print the parameter values at each iteration.

```
proc model data=test;
   parms a b c;
   y = a + b * x ** c;
   label a = "Intercept"
         b = "Coefficient of Transformed X"
         c = "Power Transformation Parameter";
   fit y start=(c=1) / startiter itprint;
run;
```

With better starting values, the estimates converge in only eight iterations. Counting the iteration required to compute the starting values for A and B, this is seven fewer than the 16 iterations required without the STARTITER option. The iteration history listing is shown in Figure 26.32.

Figure 26.32 ITPRINT Listing

The MODEL Procedure
OLS Estimation

	Iteration	N Obs	R	Objective	N Subit	a	b	c
GRID	0	20	0.9989	162.9	0	0.00010	0.00010	1.00000
GRID	1	20	0.0000	0.3464	0	10.96530	0.77007	1.00000

	Iteration	N Obs	R	Objective	N Subit	a	b	c
OLS	0	20	0.3873	0.3464	0	10.96530	0.77007	1.00000
OLS	1	20	0.3339	0.3282	2	10.75993	0.99433	0.83096
OLS	2	20	0.3244	0.3233	1	10.46894	1.31205	0.66810
OLS	3	20	0.3151	0.3197	1	10.11707	1.69149	0.54626
OLS	4	20	0.2764	0.3110	1	9.74691	2.08492	0.46615
OLS	5	20	0.2379	0.3040	0	9.06175	2.80546	0.36575
OLS	6	20	0.0612	0.2879	0	8.51825	3.36746	0.33201
OLS	7	20	0.0022	0.2868	0	8.39485	3.49449	0.32776
OLS	8	20	0.0001	0.2868	0	8.38467	3.50502	0.32711

NOTE: At OLS Iteration 8 CONVERGE=0.001 Criteria Met.

The results produced in this case are almost the same as the results shown in Figure 26.31, except that the PARMS statement causes the parameter estimates table to be ordered A, B, C instead of C, A, B. They are not exactly the same because the different starting values caused the iterations to converge at a slightly different place. This effect is controlled by changing the convergence criterion with the CONVERGE= option.

By default, the STARTITER option performs one iteration to find starting values for the parameters that are not given values. In this case, the model is linear in A and B, so only one iteration is needed. If A or B were nonlinear, you could specify more than one "starting values" iteration by specifying a number for the STARTITER= option.

Finding Starting Values by Grid Search

PROC MODEL can try various combinations of parameter values and use the combination that produces the smallest objective function value as starting values. (For OLS the objective function is the residual mean square.) This is known as a preliminary *grid search*. You can combine the STARTITER option with a grid search.

For example, the following statements try five different starting values for C: 1, 0.7, 0.5, 0.3, and 0. For each value of C, values for A and B are estimated. The combination of A, B, and C values that produce the smallest residual mean square is then used to start the iterative process.

```
proc model data=test;
   parms a b c;
   y = a + b * x ** c;
   label a = "Intercept"
         b = "Coefficient of Transformed X"
         c = "Power Transformation Parameter";
   fit y start=(c=1 .7 .5 .3 0) / startiter itprint;
run;
```

The iteration history listing is shown in Figure 26.33. Using the best starting values found by the grid search, the OLS estimation only requires two iterations. However, since the grid search required nine iterations, the total iterations in this case is 11.

Figure 26.33 ITPRINT Listing

The MODEL Procedure
OLS Estimation

	Iteration	N Obs	R	Objective	N Subit	a	b	c
GRID	0	20	0.9989	162.9	0	0.00010	0.00010	1.00000
GRID	1	20	0.0000	0.3464	0	10.96530	0.77007	1.00000
GRID	0	20	0.7587	0.7242	0	10.96530	0.77007	0.70000
GRID	1	20	0.0000	0.3073	0	10.41027	1.36141	0.70000
GRID	0	20	0.7079	0.5843	0	10.41027	1.36141	0.50000
GRID	1	20	0.0000	0.2915	0	9.69319	2.13103	0.50000
GRID	0	20	0.7747	0.7175	0	9.69319	2.13103	0.30000
GRID	1	20	0.0000	0.2869	0	8.04397	3.85767	0.30000
GRID	0	20	0.5518	2.1277	0	8.04397	3.85767	0.00000
GRID	1	20	0.0000	1.4799	0	8.04397	4.66255	0.00000

	Iteration	N Obs	R	Objective	N Subit	a	b	c
OLS	0	20	0.0189	0.2869	0	8.04397	3.85767	0.30000
OLS	1	20	0.0158	0.2869	0	8.35023	3.54145	0.32233
OLS	2	20	0.0006	0.2868	0	8.37468	3.51540	0.32622

NOTE: At OLS Iteration 2 CONVERGE=0.001 Criteria Met.

Because no initial values for A or B were provided in the PARAMETERS statement or were read in with a PARMSDATA= or ESTDATA= option, A and B were given the default value of 0.0001 for the first iteration. At the second grid point, C=5, the values of A and B obtained from the previous iterations are used for the initial iteration. If initial values are provided for parameters, the parameters start at those initial values at each grid point.

Guessing Starting Values from the Logic of the Model

Example 26.1, which uses a logistic growth curve model of the U.S. population, illustrates the need for reasonable starting values. This model can be written

$$pop = \frac{a}{1 + \exp(b - c(t - 1790))}$$

where t is time in years. The model is estimated by using decennial census data of the U.S. population in millions. If this simple but highly nonlinear model is estimated by using the default starting values, the estimation fails to converge.

To find reasonable starting values, first consider the meaning of a and c. Taking the limit as time increases, a is the limiting or maximum possible population. So, as a starting value for a, several times the most recent population known can be used—for example, one billion (1000 million).

Dividing the time derivative by the function to find the growth rate and taking the limit as t moves into the past, you can determine that c is the initial growth rate. You can examine the data and compute an estimate of the growth rate for the first few decades, or you can pick a number that sounds like a plausible population growth rate figure, such as 2%.

To find a starting value for b, let t equal the base year used, 1790, which causes c to drop out of the formula for that year, and then solve for the value of b that is consistent with the known population in 1790 and with the starting value of a. This yields $b = \ln(a/3.9 - 1)$ or about 5.5, where a is 1000 and 3.9 is roughly the population for 1790 given in the data. The estimates converge using these starting values.

Convergence Problems

When estimating nonlinear models, you might encounter some of the following convergence problems.

Unable to Improve

The optimization algorithm might be unable to find a step that improves the objective function. If this happens in the Gauss-Newton method, the step size is halved to find a change vector for which the objective improves. In the Marquardt method, λ is increased to find a change vector for which the objective improves. If, after MAXSUBITER= step-size halvings or increases in λ, the change vector still does not produce a better objective value, the iterations are stopped and an error message is printed.

Failure of the algorithm to improve the objective value can be caused by a CONVERGE= value that is too small. Look at the convergence measures reported at the point of failure. If the estimates appear to be approximately converged, you can accept the NOT CONVERGED results reported, or you can try rerunning the FIT task with a larger CONVERGE= value.

If the procedure fails to converge because it is unable to find a change vector that improves the objective value, check your model and data to ensure that all parameters are identified and data values are reasonably scaled. Then, rerun the model with different starting values. Also, consider using the Marquardt method if the Gauss-Newton method fails; the Gauss-Newton method can get into trouble if the Jacobian matrix is nearly

singular or ill-conditioned. Keep in mind that a nonlinear model may be well-identified and well-conditioned for parameter values close to the solution values but unidentified or numerically ill-conditioned for other parameter values. The choice of starting values can make a big difference.

Nonconvergence

The estimates might diverge into areas where the program overflows or the estimates might go into areas where function values are illegal or too badly scaled for accurate calculation. The estimation might also take steps that are too small or that make only marginal improvement in the objective function and thus fail to converge within the iteration limit.

When the estimates fail to converge, collinearity diagnostics for the Jacobian crossproducts matrix are printed if there are 20 or fewer parameters estimated. See the section "Linear Dependencies" on page 1566 for an explanation of these diagnostics.

Inadequate Convergence Criterion

If convergence is obtained, the resulting estimates approximate only a minimum point of the objective function. The statistical validity of the results is based on the exact minimization of the objective function, and for nonlinear models the quality of the results depends on the accuracy of the approximation of the minimum. This is controlled by the convergence criterion used.

There are many nonlinear functions for which the objective function is quite flat in a large region around the minimum point so that many quite different parameter vectors might satisfy a weak convergence criterion. By using different starting values, different convergence criteria, or different minimization methods, you can produce very different estimates for such models.

You can guard against this by running the estimation with different starting values and different convergence criteria and checking that the estimates produced are essentially the same. If they are not, use a smaller CONVERGE= value.

Local Minimum

You might have converged to a local minimum rather than a global one. This problem is difficult to detect because the procedure appears to have succeeded. You can guard against this by running the estimation with different starting values or with a different minimization technique. The START= option can be used to automatically perform a grid search to aid in the search for a global minimum.

Discontinuities

The computational methods assume that the model is a continuous and smooth function of the parameters. If this is not the case, the methods might not work.

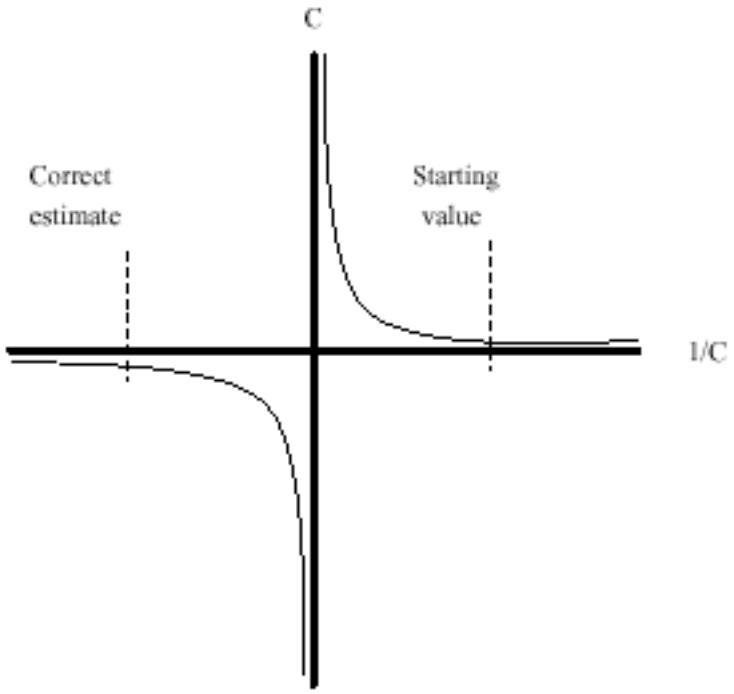
If the model equations or their derivatives contain discontinuities, the estimation usually succeeds, provided that the final parameter estimates lie in a continuous interval and that the iterations do not produce parameter values at points of discontinuity or parameter values that try to cross asymptotes.

One common case of discontinuities causing estimation failure is that of an asymptotic discontinuity between the final estimates and the initial values. For example, consider the following model, which is basically linear but is written with one parameter in reciprocal form:

```
y = a + b * x1 + x2 / c;
```

By placing the parameter C in the denominator, a singularity is introduced into the parameter space at C=0. This is not necessarily a problem, but if the correct estimate of C is negative while the starting value is positive (or vice versa), the asymptotic discontinuity at 0 will lie between the estimate and the starting value. This means that the iterations have to pass through the singularity to get to the correct estimates. The situation is shown in Figure 26.34.

Figure 26.34 Asymptotic Discontinuity



Because of the incorrect sign of the starting value, the C estimate goes off towards positive infinity in a vain effort to get past the asymptote and onto the correct arm of the hyperbola. As the computer is required to work with ever closer approximations to infinity, the numerical calculations break down and an "objective function was not improved" convergence failure message is printed. At this point, the iterations terminate with an extremely large positive value for C. When the sign of the starting value for C is changed, the estimates converge quickly to the correct values.

Linear Dependencies

In some cases, the Jacobian matrix might not be of full rank; parameters might not be fully identified for the current parameter values with the current data. When linear dependencies occur among the derivatives of the model, some parameters appear with a standard error of 0 and with the word BIASED printed in place of the *t* statistic. When this happens, collinearity diagnostics for the Jacobian crossproducts matrix are printed if the DETAILS option is specified and there are twenty or fewer parameters estimated. Collinearity diagnostics are also printed out automatically when a minimization method fails, or when the COLLIN option is specified.

For each parameter, the proportion of the variance of the estimate accounted for by each *principal component* is printed. The principal components are constructed from the eigenvalues and eigenvectors of the correlation matrix (scaled covariance matrix). When collinearity exists, a principal component is associated with proportion of the variance of more than one parameter. The numbers reported are proportions so they remain between 0 and 1. If two or more parameters have large proportion values associated with the same principal component, then two problems can occur: the computation of the parameter estimates are slow or nonconvergent; and the parameter estimates have inflated variances (Belsley, Kuh, and Welsch 1980, pp. 105–117).

For example, the following cubic model is fit to a quadratic data set:

```
proc model data=test3;
   exogenous x1;
   parms b1 a1 c1 ;
   y1 = a1 * x1 + b1 * x1 * x1  + c1 * x1 * x1 *x1;
   fit y1 / collin;
run;
```

The collinearity diagnostics are shown in Figure 26.35.

Figure 26.35 Collinearity Diagnostics

The MODEL Procedure

Collinearity Diagnostics

			Proportion of Variation		
Number	Eigenvalue	Condition Number	b1	a1	c1
1	2.942920	1.0000	0.0001	0.0004	0.0002
2	0.056638	7.2084	0.0001	0.0357	0.0148
3	0.000442	81.5801	0.9999	0.9639	0.9850

Notice that the proportions associated with the smallest eigenvalue are almost 1. For this model, removing any of the parameters decreases the variances of the remaining parameters.

In many models, the collinearity might not be clear cut. Collinearity is not necessarily something you remove. A model might need to be reformulated to remove the redundant parameterization, or the limitations on the estimability of the model can be accepted. The GINV=G4 option can be helpful to avoid problems with convergence for models containing collinearities.

Collinearity diagnostics are also useful when an estimation does not converge. The diagnostics provide insight into the numerical problems and can suggest which parameters need better starting values. These diagnostics are based on the approach of Belsley, Kuh, and Welsch (1980).

Iteration History

The options ITPRINT, ITDETAILS, XPX, I, and ITALL specify a detailed listing of each iteration of the minimization process.

ITPRINT Option

The ITPRINT information is selected whenever any iteration information is requested.

The following information is displayed for each iteration:

N	is the number of usable observations.
Objective	is the corrected objective function value.
Trace(S)	is the trace of the **S** matrix.
subit	is the number of subiterations required to find a λ or a damping factor that reduces the objective function.
R	is the R convergence measure.

The estimates for the parameters at each iteration are also printed.

ITDETAILS Option

The additional values printed for the ITDETAILS option are:

Theta	is the angle in degrees between Δ, the parameter change vector, and the negative gradient of the objective function.
Phi	is the directional derivative of the objective function in the Δ direction scaled by the objective function.
Stepsize	is the value of the damping factor used to reduce Δ if the Gauss-Newton method is used.
Lambda	is the value of λ if the Marquardt method is used.
Rank(XPX)	is the rank of the $\mathbf{X}'\mathbf{X}$ matrix (output if the projected Jacobian crossproducts matrix is singular).

The definitions of PPC and R are explained in the section "Convergence Criteria" on page 1556. When the values of PPC are large, the parameter associated with the criteria is displayed in parentheses after the value.

XPX and I Options

The XPX and the I options select the printing of the augmented $\mathbf{X}'\mathbf{X}$ matrix and the augmented $\mathbf{X}'\mathbf{X}$ matrix after a *sweep* operation (Goodnight 1979) has been performed on it. An example of the output from the following statements is shown in Figure 26.36.

```
proc model data=test2;
   y1 = a1 * x2 * x2 - exp( d1*x1);
   y2 = a2 * x1 * x1 + b2 * exp( d2*x2);
   fit y1 y2 / itall XPX I ;
run;
```

Figure 26.36 XPX and I Options Output

The MODEL Procedure
OLS Estimation

Cross Products for System At OLS Iteration 0

	a1	d1	a2	b2	d2	Residual
a1	1839468	-33818.35	0.0	0.00	0.000000	3879959
d1	-33818	1276.45	0.0	0.00	0.000000	-76928
a2	0	0.00	42925.0	1275.15	0.154739	470686
b2	0	0.00	1275.2	50.01	0.003867	16055
d2	0	0.00	0.2	0.00	0.000064	2
Residual	3879959	-76928.14	470686.3	16055.07	2.329718	24576144

XPX Inverse for System At OLS Iteration 0

	a1	d1	a2	b2	d2	Residual
a1	0.000001	0.000028	0.000000	0.0000	0.00	2
d1	0.000028	0.001527	0.000000	0.0000	0.00	-9
a2	0.000000	0.000000	0.000097	-0.0025	-0.08	6
b2	0.000000	0.000000	-0.002455	0.0825	0.95	172
d2	0.000000	0.000000	-0.084915	0.9476	15746.71	11931
Residual	1.952150	-8.546875	5.823969	171.6234	11930.89	10819902

The first matrix, labeled "Cross Products," for OLS estimation is

$$\begin{bmatrix} \mathbf{X}'\mathbf{X} & \mathbf{X}'\mathbf{r} \\ \mathbf{r}'\mathbf{X} & \mathbf{r}'\mathbf{r} \end{bmatrix}$$

The column labeled Residual in the output is the vector $\mathbf{X}'\mathbf{r}$, which is the gradient of the objective function. The diagonal scalar value $\mathbf{r}'\mathbf{r}$ is the objective function uncorrected for degrees of freedom. The second matrix, labeled "XPX Inverse," is created through a sweep operation on the augmented $\mathbf{X}'\mathbf{X}$ matrix to get:

$$\begin{bmatrix} (\mathbf{X}'\mathbf{X})^{-1} & (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{r} \\ (\mathbf{X}'\mathbf{r})'(\mathbf{X}'\mathbf{X})^{-1} & \mathbf{r}'\mathbf{r} - (\mathbf{X}'\mathbf{r})'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{r} \end{bmatrix}$$

Note that the residual column is the change vector used to update the parameter estimates at each iteration. The corner scalar element is used to compute the R convergence criteria.

ITALL Option

The ITALL option, in addition to causing the output of all of the preceding options, outputs the **S** matrix, the inverse of the **S** matrix, the CROSS matrix, and the swept CROSS matrix. An example of a portion of the CROSS matrix for the preceding example is shown in Figure 26.37.

Figure 26.37 ITALL Option Crossproducts Matrix Output

The MODEL Procedure
OLS Estimation

Crossproducts Matrix At OLS Iteration 0					
	1	@PRED.y1/@a1	@PRED.y1/@d1	@PRED.y2/@a2	@PRED.y2/@b2
1	50.00	6409	-239.16	1275.0	50.00
@PRED.y1/@a1	6409.08	1839468	-33818.35	187766.1	6409.88
@PRED.y1/@d1	-239.16	-33818	1276.45	-7253.0	-239.19
@PRED.y2/@a2	1275.00	187766	-7253.00	42925.0	1275.15
@PRED.y2/@b2	50.00	6410	-239.19	1275.2	50.01
@PRED.y2/@d2	0.00	1	-0.03	0.2	0.00
RESID.y1	14699.97	3879959	-76928.14	420582.9	14701.77
RESID.y2	16052.76	4065028	-85083.68	470686.3	16055.07

Crossproducts Matrix At OLS Iteration 0			
	@PRED.y2/@d2	RESID.y1	RESID.y2
1	0.003803	14700	16053
@PRED.y1/@a1	0.813934	3879959	4065028
@PRED.y1/@d1	-0.026177	-76928	-85084
@PRED.y2/@a2	0.154739	420583	470686
@PRED.y2/@b2	0.003867	14702	16055
@PRED.y2/@d2	0.000064	2	2
RESID.y1	1.820356	11827102	12234106
RESID.y2	2.329718	12234106	12749042

Computer Resource Requirements

If you are estimating large systems, you need to be aware of how PROC MODEL uses computer resources (such as memory and the CPU) so they can be used most efficiently.

Saving Time with Large Data Sets

If your input data set has many observations, the FIT statement performs a large number of model program executions. A pass through the data is made at least once for each iteration and the model program is executed once for each observation in each pass. If you refine the starting estimates by using a smaller data set, the final estimation with the full data set might require fewer iterations.

For example, you could use

```
proc model;
   /* Model goes here */
   fit / data=a(obs=25);
   fit / data=a;
```

where OBS=25 selects the first 25 observations in A. The second FIT statement produces the final estimates using the full data set and starting values from the first run.

Fitting the Model in Sections to Save Space and Time

If you have a very large model (with several hundred parameters, for example), the procedure uses considerable space and time. You might be able to save resources by breaking the estimation process into several steps and estimating the parameters in subsets.

You can use the FIT statement to select for estimation only the parameters for selected equations. Do not break the estimation into too many small steps; the total computer time required is minimized by compromising between the number of FIT statements that are executed and the size of the crossproducts matrices that must be processed.

When the parameters are estimated for selected equations, the entire model program must be executed even though only a part of the model program might be needed to compute the residuals for the equations selected for estimation. If the model itself can be broken into sections for estimation (and later combined for simulation and forecasting), then more resources can be saved.

For example, to estimate the following four equation model in two steps, you could use

```
proc model data=a outmodel=part1;
   parms a0-a2 b0-b2 c0-c3 d0-d3;
   y1 = a0 + a1*y2 + a2*x1;
   y2 = b0 + b1*y1 + b2*x2;
   y3 = c0 + c1*y1 + c2*y4 + c3*x3;
   y4 = d0 + d1*y1 + d2*y3 + d3*x4;
   fit y1 y2;
   fit y3 y4;
   fit y1 y2 y3 y4;
run;
```

You should try estimating the model in pieces to save time only if there are more than 14 parameters; the preceding example takes more time, not less, and the difference in memory required is trivial.

Memory Requirements for Parameter Estimation

PROC MODEL is a large program, and it requires much memory. Memory is also required for the SAS System, various data areas, the model program and associated tables and data vectors, and a few crossproducts matrices. For most models, the memory required for PROC MODEL itself is much larger than that required for the model program, and the memory required for the model program is larger than that required for the crossproducts matrices.

The number of bytes needed for two crossproducts matrices, four **S** matrices, and three parameter covariance matrices is

$$8 \times (2 + k + m + g)^2 + 16 \times g^2 + 12 \times (p + 1)^2$$

plus lower-order terms, where m is the number of unique nonzero derivatives of each residual with respect to each parameter, g is the number of equations, k is the number of instruments, and p is the number of parameters. This formula is for the memory required for 3SLS. If you are using OLS, a reasonable estimate of the memory required for large problems (greater than 100 parameters) is to divide the value obtained from the formula in half.

Consider the following model program:

```
proc model data=test2 details;
   exogenous x1 x2;
   parms b1 100 a1 a2 b2 2.5 c2 55;
   y1 = a1 * y2 + b1 * x1 * x1;
   y2 = a2 * y1 + b2 * x2 * x2 + c2 / x2;
   fit y1 y2 / n3sls memoryuse;
   inst b1 b2 c2 x1 ;
run;
```

The DETAILS option prints the storage requirements information shown in Figure 26.38.

Figure 26.38 Storage Requirements Information

The MODEL Procedure

Storage Requirements for this Problem	
Order of XPX Matrix	6
Order of S Matrix	2
Order of Cross Matrix	13
Total Nonzero Derivatives	5
Distinct Variable Derivatives	5
Size of Cross matrix	728

The matrix $\mathbf{X}'\mathbf{X}$ augmented by the residual vector is called the XPX matrix in the output, and it has the size $m + 1$. The order of the $\mathbf{S}$ matrix, 2 for this example, is the value of g. The CROSS matrix is made up of the k unique instruments, a constant column that represents the intercept terms, followed by the m unique Jacobian variables plus a constant column that represents the parameters with constant derivatives, followed by the g residuals.

The size of two CROSS matrices in bytes is

$$8 \times (2 + k + m + g)^2 + 2 + k + m + g$$

Note that the CROSS matrix is symmetric, so only the diagonal and the upper triangular part of the matrix is stored. For examples of the CROSS and XPX matrices see the section "Iteration History" on page 1567.

The MEMORYUSE Option

The MEMORYUSE option in the FIT, SOLVE, MODEL, or RESET statement can be used to request a comprehensive memory usage summary.

Figure 26.39 shows an example of the output produced by the MEMORYUSE option.

Figure 26.39 MEMORYUSE Option Output for FIT Task

Memory Usage Summary (in bytes)	
Symbols	17060
Strings	2593
Lists	2428
Arrays	2300
Statements	2704
Opcodes	1840
Parsing	6180
Executable	12845
Block option	0
Cross reference	0
Flow analysis	336
Derivatives	28640
Data vector	368
Cross matrix	1480
X'X matrix	590
S matrix	144
GMM memory	0
Jacobian	0
Work vectors	702
Overhead	14246
----------	----------
Total	94456

Definitions of the memory components follow:

symbols	memory used to store information about variables in the model
strings	memory used to store the variable names and labels
lists	space used to hold lists of variables
arrays	memory used by ARRAY statements
statements	memory used for the list of programming statements in the model
opcodes	memory used to store the code compiled to evaluate the expression in the model program
parsing	memory used in parsing the SAS statements
executable	the compiled model program size
block option	memory used by the BLOCK option
cross ref.	memory used by the XREF option
flow analysis	memory used to compute the interdependencies of the variables
derivatives	memory used to compute and store the analytical derivatives
data vector	memory used for the program data vector
cross matrix	memory used for one or more copies of the CROSS matrix
$\mathbf{X}'\mathbf{X}$ matrix	memory used for one or more copies of the $\mathbf{X}'\mathbf{X}$ matrix
$\mathbf{S}$ matrix	memory used for the covariance matrix
GMM memory	additional memory used for the GMM and ITGMM methods
Jacobian	memory used for the Jacobian matrix for SOLVE and FIML
work vectors	memory used for miscellaneous work vectors
overhead	other miscellaneous memory

Testing for Normality

The NORMAL option in the FIT statement performs multivariate and univariate tests of normality.

The three multivariate tests provided are Mardia's skewness test and kurtosis test (Mardia 1970) and the Henze-Zirkler $T_{n,\beta}$ test (Henze and Zirkler 1990). The two univariate tests provided are the Shapiro-Wilk W test and the Kolmogorov-Smirnov test. (For details on the univariate tests, refer to "Goodness-of-Fit Tests" section in "The UNIVARIATE Procedure" chapter in the *Base SAS Procedures Guide*.) The null hypothesis for all these tests is that the residuals are normally distributed.

For a random sample $X_1, \ldots, X_n$, $X_i \in \mathrm{R}^d$, where d is the dimension of X_i and n is the number of observations, a measure of multivariate skewness is

$$b_{1,d} = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} [(X_i - \mu)' \mathbf{S}^{-1} (X_j - \mu)]^3$$

where $\mathbf{S}$ is the sample covariance matrix of $\mathbf{X}$. For weighted regression, both $\mathbf{S}$ and $(X_i - \mu)$ are computed by using the weights supplied by the WEIGHT statement or the _WEIGHT_ variable.

Mardia showed that under the null hypothesis $\frac{n}{6} b_{1,d}$ is asymptotically distributed as $\chi^2(d(d+1)(d+2)/6)$. For small samples, Mardia's skewness test statistic is calculated with a small sample correction formula, given by $\frac{nk}{6} b_{1,d}$ where the correction factor k is given by $k = (d+1)(n+1)(n+3)/n(((n+1)(d+1)) - 6)$. Mardia's skewness test statistic in PROC MODEL uses this small sample corrected formula.

A measure of multivariate kurtosis is given by

$$b_{2,d} = \frac{1}{n} \sum_{i=1}^{n} [(X_i - \mu)' \mathbf{S}^{-1} (X_i - \mu)]^2$$

Mardia showed that under the null hypothesis, $b_{2,d}$ is asymptotically normally distributed with mean $d(d+2)$ and variance $8d(d+2)/n$.

The Henze-Zirkler test is based on a nonnegative functional $D(.,.)$ that measures the distance between two distribution functions and has the property that

$$D(\mathrm{N}_d(0, I_d), Q) = 0$$

if and only if

$$Q = \mathrm{N}_d(0, I_d)$$

where $\mathrm{N}_d(\mu, \Sigma_d)$ is a d-dimensional normal distribution.

The distance measure $D(.,.)$ can be written as

$$D_\beta(P, Q) = \int_{\mathrm{R}^d} |\hat{P}(t) - \hat{Q}(t)|^2 \varphi_\beta(t) dt$$

where $\hat{P}(t)$ and $\hat{Q}(t)$ are the Fourier transforms of P and Q, and $\varphi_\beta(t)$ is a weight or a kernel function. The density of the normal distribution $\mathrm{N}_d(0, \beta^2 I_d)$ is used as $\varphi_\beta(t)$

$$\varphi_\beta(t) = (2\pi\beta^2)^{\frac{-d}{2}} \exp(\frac{-|t|^2}{2\beta^2}), t \in \mathrm{R}^d$$

where $|t| = (t't)^{0.5}$.

The parameter β depends on n as

$$\beta_d(n) = \frac{1}{\sqrt{2}}(\frac{2d+1}{4})^{1/(d+4)} n^{1/(d+4)}$$

The test statistic computed is called $T_\beta(d)$ and is approximately distributed as a lognormal. The lognormal distribution is used to compute the null hypothesis probability.

$$\begin{aligned} T_\beta(d) = \quad & \frac{1}{n} \quad \sum_{j=1}^{n}\sum_{k=1}^{n} \exp(-\frac{\beta^2}{2}|Y_j - Y_k|^2) \\ & - \quad 2(1+\beta^2)^{-d/2} \sum_{j=1}^{n} \exp(-\frac{\beta^2}{2(1+\beta^2)}|Y_j|^2) + n(1+2\beta^2)^{-d/2} \end{aligned}$$

where

$$|Y_j - Y_k|^2 = (X_j - X_k)'\mathbf{S}^{-1}(X_j - X_k)$$

$$|Y_j|^2 = (X_j - \bar{X})'\mathbf{S}^{-1}(X_j - \bar{X})$$

Monte Carlo simulations suggest that $T_\beta(d)$ has good power against distributions with heavy tails.

The Shapiro-Wilk W test is computed only when the number of observations (n) is less than 2000 while computation of the Kolmogorov-Smirnov test statistic requires at least 2000 observations.

The following is an example of the output produced by the NORMAL option.

```
proc model data=test2;
   y1 = a1 * x2 * x2 - exp( d1*x1);
   y2 = a2 * x1 * x1 + b2 * exp( d2*x2);
   fit y1 y2 / normal ;
run;
```

Figure 26.40 Normality Test Output

The MODEL Procedure

Normality Test			
Equation	**Test Statistic**	**Value**	**Prob**
y1	Shapiro-Wilk W	0.34	<.0001
y2	Shapiro-Wilk W	0.82	<.0001
System	Mardia Skewness	286.4	<.0001
	Mardia Kurtosis	31.28	<.0001
	Henze-Zirkler T	6.65	<.0001

Heteroscedasticity

One of the key assumptions of regression is that the variance of the errors is constant across observations. If the errors have constant variance, the errors are called *homoscedastic*. Typically, residuals are plotted to assess this assumption. Standard estimation methods are inefficient when the errors are *heteroscedastic* or have nonconstant variance.

Heteroscedasticity Tests

The MODEL procedure provides two tests for heteroscedasticity of the errors: White's test and the modified Breusch-Pagan test.

Both White's test and the Breusch-Pagan are based on the residuals of the fitted model. For systems of equations, these tests are computed separately for the residuals of each equation.

The residuals of an estimation are used to investigate the heteroscedasticity of the true disturbances.

The WHITE option tests the null hypothesis

$$H_0 : \sigma_i^2 = \sigma^2 \quad \text{for all } i$$

White's test is general because it makes no assumptions about the form of the heteroscedasticity (White 1980). Because of its generality, White's test might identify specification errors other than heteroscedasticity (Thursby 1982). Thus, White's test might be significant when the errors are homoscedastic but the model is misspecified in other ways.

White's test is equivalent to obtaining the error sum of squares for the regression of squared residuals on a constant and all the unique variables in $\mathbf{J}\otimes\mathbf{J}$, where the matrix $\mathbf{J}$ is composed of the partial derivatives of the equation residual with respect to the estimated parameters. White's test statistic W is computed as follows:

$$W = nR^2$$

where R^2 is the correlation coefficient obtained from the above regression. The statistic is asymptotically distributed as chi-squared with P–1 degrees of freedom, where P is the number of regressors in the regression, including the constant and n is the total number of observations. In the example given below, the regressors are constant, income, income*income, income*income*income, and income*income*income*income. income*income occurs twice and one is dropped. Hence, P=5 with degrees of freedom, P–1=4.

Note that White's test in the MODEL procedure is different than White's test in the REG procedure requested by the SPEC option. The SPEC option produces the test from Theorem 2 on page 823 of White (1980). The WHITE option, on the other hand, produces the statistic discussed in Greene (1993).

The null hypothesis for the modified Breusch-Pagan test is homoscedasticity. The alternate hypothesis is that the error variance varies with a set of regressors, which are listed in the BREUSCH= option.

Define the matrix $\mathbf{Z}$ to be composed of the values of the variables listed in the BREUSCH= option, such that $z_{i,j}$ is the value of the jth variable in the BREUSCH= option for the ith observation. The null hypothesis of the Breusch-Pagan test is

$$\sigma_i^2 = \sigma^2(\alpha_0 + \boldsymbol{\alpha}' z_i) \qquad H_0 : \boldsymbol{\alpha} = \mathbf{0}$$

where σ_i^2 is the error variance for the ith observation and α_0 and $\boldsymbol{\alpha}$ are regression coefficients.

The test statistic for the Breusch-Pagan test is

$$bp = \frac{1}{v}(\mathbf{u} - \bar{u}\mathbf{i})'\mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'(\mathbf{u} - \bar{u}\mathbf{i})$$

where $\mathbf{u} = (e_1^2, e_2^2, \ldots, e_n^2)$, $\mathbf{i}$ is a $n \times 1$ vector of ones, and

$$v = \frac{1}{n}\sum_{i=1}^{n}(e_i^2 - \frac{\mathbf{e}'\mathbf{e}}{n})^2$$

This is a modified version of the Breusch-Pagan test, which is less sensitive to the assumption of normality than the original test (Greene 1993, p. 395).

The statements in the following example produce the output in Figure 26.41:

```
proc model data=schools;
   parms const inc inc2;

   exp = const + inc * income + inc2 * income * income;
   incsq = income * income;

   fit exp / white breusch=(1 income incsq);
run;
```

Figure 26.41 Output for Heteroscedasticity Tests

The MODEL Procedure

Heteroscedasticity Test					
Equation	**Test**	**Statistic**	**DF**	**Pr > ChiSq**	**Variables**
exp	White's Test	21.16	4	0.0003	Cross of all vars
	Breusch-Pagan	15.83	2	0.0004	1, income, incsq

Correcting for Heteroscedasticity

There are two methods for improving the efficiency of the parameter estimation in the presence of heteroscedastic errors. If the error variance relationships are known, weighted regression can be used or an error model can be estimated. For details about error model estimation, see the section "Error Covariance Structure Specification" on page 1586. If the error variance relationship is unknown, GMM estimation can be used.

Weighted Regression

The WEIGHT statement can be used to correct for the heteroscedasticity. Consider the following model, which has a heteroscedastic error term:

$$y_t = 250(e^{-0.2t} - e^{-0.8t}) + \sqrt{(9/t)}\epsilon_t$$

The data for this model is generated with the following SAS statements:

```
data test;
   do t=1 to 25;
      y = 250 * (exp( -0.2 * t ) - exp( -0.8 * t )) +
          sqrt( 9 / t ) * rannor(1);
      output;
   end;
run;
```

If this model is estimated with OLS, as shown in the following statements, the estimates shown in Figure 26.42 are obtained for the parameters.

```
proc model data=test;
   parms b1 0.1 b2 0.9;
   y = 250 * ( exp( -b1 * t ) - exp( -b2 * t ) );
   fit y;
run;
```

Figure 26.42 Unweighted OLS Estimates

The MODEL Procedure

Nonlinear OLS Parameter Estimates

Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|
b1	0.200977	0.00101	198.60	<.0001
b2	0.826236	0.00853	96.82	<.0001

If both sides of the model equation are multiplied by $\sqrt{t}$, the model has a homoscedastic error term. This multiplication or weighting is done through the WEIGHT statement. The WEIGHT statement variable operates on the squared residuals as

$$\boldsymbol{\epsilon}_t' \boldsymbol{\epsilon}_t = \text{weight} \times \mathbf{q}_t' \mathbf{q}_t$$

so that the WEIGHT statement variable represents the square of the model multiplier. The following PROC MODEL statements corrects the heteroscedasticity with a WEIGHT statement:

```
proc model data=test;
   parms b1 0.1 b2 0.9;
   y = 250 * ( exp( -b1 * t ) - exp( -b2 * t ) );
   fit y;
   weight t;
run;
```

Note that the WEIGHT statement follows the FIT statement. The weighted estimates are shown in Figure 26.43.

Figure 26.43 Weighted OLS Estimates

The MODEL Procedure

Nonlinear OLS Parameter Estimates

Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|
b1	0.200503	0.000844	237.53	<.0001
b2	0.816701	0.0139	58.71	<.0001

The weighted OLS estimates are identical to the output produced by the following PROC MODEL example:

```
proc model data=test;
   parms b1 0.1 b2 0.9;
   y = 250 * ( exp( -b1 * t ) - exp( -b2 * t ) );
  _weight_ = t;
   fit y;
run;
```

If the WEIGHT statement is used in conjunction with the _WEIGHT_ variable, the two values are multiplied together to obtain the weight used.

The WEIGHT statement and the _WEIGHT_ variable operate on all the residuals in a system of equations. If a subset of the equations needs to be weighted, the residuals for each equation can be modified through the RESID. variable for each equation. The following example demonstrates the use of the RESID. variable to make a homoscedastic error term:

```
proc model data=test;
   parms b1 0.1 b2 0.9;
   y = 250 * ( exp( -b1 * t ) - exp( -b2 * t ) );
   resid.y = resid.y * sqrt(t);
   fit y;
run;
```

These statements produce estimates of the parameters and standard errors that are identical to the weighted OLS estimates. The reassignment of the RESID.Y variable must be done after Y is assigned; otherwise it would have no effect. Also, note that the residual (RESID.Y) is multiplied by $\sqrt{t}$. Here the multiplier is acting on the residual before it is squared.

GMM Estimation

If the form of the heteroscedasticity is unknown, generalized method of moments estimation (GMM) can be used. The following PROC MODEL statements use GMM to estimate the example model used in the preceding section:

```
proc model data=test;
   parms b1 0.1 b2 0.9;
   y = 250 * ( exp( -b1 * t ) - exp( -b2 * t ) );
   fit y / gmm;
   instruments b1 b2;
run;
```

GMM is an instrumental method, so instrument variables must be provided.

GMM estimation generates estimates for the parameters shown in Figure 26.44.

Figure 26.44 GMM Estimation for Heteroscedasticity

The MODEL Procedure

Nonlinear GMM Parameter Estimates

Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|
b1	0.200487	0.000800	250.69	<.0001
b2	0.822148	0.0148	55.39	<.0001

Heteroscedasticity-Consistent Covariance Matrix Estimation

Homoscedasticity is required for ordinary least squares regression estimates to be efficient. A nonconstant error variance, heteroscedasticity, causes the OLS estimates to be inefficient, and the usual OLS covariance matrix, $\hat{\mathbf{\Sigma}}$, is generally invalid:

$$\hat{\mathbf{\Sigma}} = \sigma^2(\mathbf{X}'\mathbf{X})^{-1}$$

When the variance of the errors of a classical linear model

$$Y = \mathbf{X}\beta + \epsilon$$

is not constant across observations (heteroscedastic), so that $\sigma_i^2 \neq \sigma_j^2$ for some $j > 1$, the OLS estimator

$$\hat{\beta}_{OLS} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'Y$$

is unbiased but it is inefficient. Models that take into account the changing variance can make more efficient use of the data. When the variances, σ_t^2, are known, generalized least squares (GLS) can be used and the estimator

$$\hat{\beta}_{GLS} = (\mathbf{X}'\mathbf{\Omega}\mathbf{X})^{-1}\mathbf{X}'\mathbf{\Omega}^{-1}Y$$

where

$$\mathbf{\Omega} = \begin{bmatrix} \sigma_1^2 & 0 & 0 & 0 \\ 0 & \sigma_2^2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \sigma_T^2 \end{bmatrix}$$

is unbiased and efficient. However, GLS is unavailable when the variances, σ_t^2, are unknown.

To solve this problem White (1980) proposed a heteroscedastic consistent-covariance matrix estimator (HCCME)

$$\hat{\boldsymbol{\Sigma}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\hat{\boldsymbol{\Omega}}\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}$$

that is consistent as well as unbiased, where

$$\hat{\boldsymbol{\Omega}}_0 = \begin{bmatrix} \epsilon_1^2 & 0 & 0 & 0 \\ 0 & \epsilon_2^2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \epsilon_T^2 \end{bmatrix}$$

and $\epsilon_t = Y_t - \mathbf{X}_t\hat{\beta_{OLS}}$.

This estimator is considered somewhat unreliable in finite samples. Therefore, Davidson and MacKinnon (1993) propose three different modifications to estimating $\hat{\boldsymbol{\Omega}}$. The first solution is to simply multiply ϵ_t^2 by $\frac{n}{n-df}$, where n is the number of observations and df is the number of explanatory variables, so that

$$\hat{\boldsymbol{\Omega}}_1 = \begin{bmatrix} \frac{n}{n-df}\,\epsilon_1^2 & 0 & 0 & 0 \\ 0 & \frac{n}{n-df}\,\epsilon_2^2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \frac{n}{n-df}\,\epsilon_n^2 \end{bmatrix}$$

The second solution is to define

$$\hat{\boldsymbol{\Omega}}_2 = \begin{bmatrix} \frac{\epsilon_1^2}{1-\hat{h}_1} & 0 & 0 & 0 \\ 0 & \frac{\epsilon_2^2}{1-\hat{h}_2} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \frac{\epsilon_n^2}{1-\hat{h}_n} \end{bmatrix}$$

where $\hat{h}_t = \mathbf{X}_t(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}_t'$.

The third solution, called the "jackknife," is to define

$$\hat{\boldsymbol{\Omega}}_3 = \begin{bmatrix} \frac{\epsilon_1^2}{(1-\hat{h}_1)^2} & 0 & 0 & 0 \\ 0 & \frac{\epsilon_2^2}{(1-\hat{h}_2)^2} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \frac{\epsilon_n^2}{(1-\hat{h}_T)^2} \end{bmatrix}$$

MacKinnon and White (1985) investigated these three modified HCCMEs, including the original HCCME, based on finite-sample performance of pseudo-t statistics. The original HCCME performed the worst. The first modification performed better. The second modification performed even better than the first, and the third modification performed the best. They concluded that the original HCCME should never be used in finite sample estimation, and that the second and third modifications should be used over the first modification if the diagonals of $\hat{\boldsymbol{\Omega}}$ are available.

Seemingly Unrelated Regression HCCME

Extending the discussion to systems of g equations, the HCCME for SUR estimation is

$$(\tilde{\mathbf{X}}'\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}'\hat{\mathbf{\Omega}}\tilde{\mathbf{X}}(\tilde{\mathbf{X}}'\tilde{\mathbf{X}})^{-1}$$

where $\tilde{\mathbf{X}}$ is a $n\ g \times k$ matrix with the first g rows representing the first observation, the next g rows representing the second observation, and so on. $\hat{\mathbf{\Omega}}$ is now a $n\ g \times n\ g$ block diagonal matrix with typical block $g \times g$

$$\hat{\mathbf{\Omega}}_i = \begin{bmatrix} \psi_{1,i}\,\psi_{1,i} & \psi_{1,i}\,\psi_{2,i} & \cdots & \psi_{1,i}\,\psi_{g,i} \\ \psi_{2,i}\,\psi_{1,i} & \psi_{2,i}\,\psi_{2,i} & \cdots & \psi_{2,i}\,\psi_{g,i} \\ \vdots & \vdots & \vdots & \vdots \\ \psi_{g,i}\,\psi_{1,i} & \psi_{g,i}\,\psi_{2,i} & \cdots & \psi_{g,i}\,\psi_{g,i} \end{bmatrix}$$

where

$$\psi_{j,i} = \epsilon_{j,i} \quad HC_0$$

or

$$\psi_{j,i} = \sqrt{\frac{n}{n-df}}\epsilon_{j,i} \quad HC_1$$

or

$$\psi_{j,i} = \epsilon_{j,i}/\sqrt{1-\hat{h}_i} \quad HC_2$$

or

$$\psi_{j,i} = \epsilon_{j,i}/(1-\hat{h}_i) \quad HC_3$$

Two- and Three-Stage Least Squares HCCME

For two- and three-stage least squares, the HCCME for a g equation system is

$$\mathrm{Cov}\,F(\hat{\mathbf{\Omega}})\mathrm{Cov}$$

where

$$\mathrm{Cov} = \left(\frac{1}{n}\mathbf{X}'(\mathbf{I}\otimes\mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}')\mathbf{X}\right)^{-1}$$

is the normal covariance matrix without the $\mathbf{S}$ matrix and

$$F(\mathbf{\Omega}) = \frac{1}{n}\sum_{i}^{g}\sum_{j}^{g}\mathbf{X}_i'\mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'\hat{\mathbf{\Omega}}_{ij}\mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'\mathbf{X}_j$$

where $\mathbf{X}_j$ is a $n \times p$ matrix with the jth equations regressors in the appropriate columns and zeros everywhere else.

$$\hat{\mathbf{\Omega}}_{ij} = \begin{bmatrix} \psi_{i,1}\psi_{j,1} & 0 & 0 & 0 \\ 0 & \psi_{i,2}\psi_{j,2} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \psi_{i,n}\psi_{j,n} \end{bmatrix}$$

For 2SLS $\hat{\mathbf{\Omega}}_{ij} = 0$ when $i \neq j$. The ϵ_t used in $\hat{\mathbf{\Omega}}$ is computed by using the parameter estimates obtained from the instrumental variables estimation.

The leverage value for the ith equation used in the HCCME=2 and HCCME=3 methods is computed as conditional on the first stage as

$$h_{ti} = \mathbf{Z}_t(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{X}_i(\mathbf{X}'(\mathbf{I} \otimes \mathbf{Z}(\mathbf{Z}' * \mathbf{Z})^{-1}\mathbf{Z}')\mathbf{X})^{-1}\mathbf{X}_i'\mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}_t'$$

for 2SLS and

$$h_{ti} = \mathbf{Z}_t(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{X}_i(\mathbf{X}'(\mathbf{S}^{-1} \otimes \mathbf{Z}(\mathbf{Z}' * \mathbf{Z})^{-1}\mathbf{Z}')\mathbf{X})^{-1}\mathbf{X}_i'\mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}_t'/\mathbf{S}_{ii}$$

for 3SLS.

Testing for Autocorrelation

The GODFREY= option in the FIT statement produces the Godfrey Lagrange multiplier test for serially correlated residuals for each equation (Godfrey 1978a, b). n is the maximum autoregressive order, and specifies that Godfrey's tests be computed for lags 1 through n. The default number of lags is four.

The tests are performed separately for each equation estimated by the FIT statement. When a nonlinear model is estimated, the test is computed by using a linearized model.

The following is an example of the output produced by the GODFREY=3 option:

Figure 26.45 Autocorrelation Test Output

Godfrey Test Output

The MODEL Procedure

Godfrey's Serial Correlation Test			
Equation	Alternative	LM	Pr > LM
y	1	6.63	0.0100
	2	6.89	0.0319
	3	6.96	0.0732

The three variations of the test reported by the GODFREY=3 option are designed to have power against different alternative hypothesis. Thus, if the residuals in fact have only first-order autocorrelation, the lag 1 test has the most power for rejecting the null hypothesis of uncorrelated residuals. If the residuals have second- but not higher-order autocorrelation, the lag 2 test might be more likely to reject; the same is true for third-order autocorrelation and the lag 3 test.

The null hypothesis of Godfrey's tests is that the equation residuals are white noise. However, if the equation includes autoregressive error model of order p (AR(p),) then the lag i test, when considered in terms of the structural error, is for the null hypothesis that the structural errors are from an AR(p) process versus the alternative hypothesis that the errors are from an AR($p + i$) process.

The alternative ARMA(p, i) process is locally equivalent to the alternative AR($p + i$) process with respect to the null model AR(p). Thus, the GODFREY= option results are also a test of AR(p) errors against the alternative hypothesis of ARMA(p, i) errors. See Godfrey (1978a, b) for more detailed information.

Transformation of Error Terms

In PROC MODEL you can control the form of the error term. By default, the error term is assumed to be additive. This section demonstrates how to specify nonadditive error terms and discusses the effects of these transformations.

Models with Nonadditive Errors

The estimation methods used by PROC MODEL assume that the error terms of the equations are independently and identically distributed with zero means and finite variances. Furthermore, the methods assume that the RESID.*name* equation variable for normalized form equations or the EQ.*name* equation variable for general form equations contains an estimate of the error term of the true stochastic model whose parameters are being estimated. Details on RESID.*name* and EQ.*name* equation variables are in the section "Equation Translations" on page 1677.

To illustrate these points, consider the common loglinear model

$$y = \alpha x^{\beta} \tag{1}$$

$$\ln y = a + b\ln(x) \tag{2}$$

where a =log(α) and b =β. Equation (2) is called the *log form* of the equation in contrast to equation (1), which is called the *level form* of the equation. Using the SYSLIN procedure, you can estimate equation (2) by specifying

```
proc syslin data=in;
   model logy=logx;
run;
```

where LOGY and LOGX are the logs of Y and X computed in a preceding DATA step. The resulting values for INTERCEPT and LOGX correspond to a and b in equation (2).

Using the MODEL procedure, you can try to estimate the parameters in the level form (and avoid the DATA step) by specifying

```
proc model data=in;
   parms alpha beta;
   y = alpha * x ** beta;
   fit y;
run;
```

where ALPHA and BETA are the parameters in equation (1).

Unfortunately, at least one of the preceding is wrong; an ambiguity results because equations (1) and (2) contain no explicit error term. The SYSLIN and MODEL procedures both deal with additive errors; the residual used (the estimate of the error term in the equation) is the difference between the predicted and actual values (of LOGY for PROC SYSLIN and of Y for PROC MODEL in this example). If you perform the regressions discussed previously, PROC SYSLIN estimates equation (3) while PROC MODEL estimates equation (4).

$$\ln y = a + b\ln(x) + \epsilon \quad (3)$$

$$y = \alpha x^{\beta} + \xi \quad (4)$$

These are different statistical models. Equation (3) is the log form of equation (5)

$$y = \alpha x^{\beta}\mu \quad (5)$$

where μ=e$^{\epsilon}$. Equation (4), on the other hand, cannot be linearized because the error term ξ (different from μ) is additive in the level form.

You must decide whether your model is equation (4) or (5). If the model is equation (4), you should use PROC MODEL. If you linearize equation (1) without considering the error term and apply SYSLIN to MODEL LOGY=LOGX, the results will be wrong. On the other hand, if your model is equation (5) (in practice it usually is), and you want to use PROC MODEL to estimate the parameters in the *level* form, you must do something to account for the multiplicative error.

PROC MODEL estimates parameters by minimizing an objective function. The objective function is computed using either the RESID.-prefixed equation variable or the EQ.-prefixed equation variable. You must make sure that these prefixed equation variables are assigned an appropriate error term. If the model has additive errors that satisfy the assumptions, nothing needs to be done. In the case of equation (5), the error is nonadditive and the equation is in normalized form, so you must alter the value of RESID.Y.

The following assigns a valid estimate of μ to RESID.Y:

```
y = alpha * x ** beta;
resid.y = actual.y / pred.y;
```

However, μ=e$^{\epsilon}$, and therefore μ, cannot have a mean of zero, and you cannot consistently estimate α and β by minimizing the sum of squares of an estimate of μ. Instead, you use $\epsilon = \ln\mu$.

```
proc model data=in;
   parms alpha beta;
   y = alpha * x ** beta;
   resid.y = log( actual.y / pred.y );
   fit y;
run;
```

If the model was expressed in general form, this transformation becomes

```
proc model data=in;
   parms alpha beta;
   EQ.trans = log( y / (alpha * x ** beta));
   fit trans;
run;
```

Both examples produce estimates of α and β of the level form that match the estimates of a and b of the log form. That is, ALPHA=exp(INTERCEPT) and BETA=LOGX, where INTERCEPT and LOGX are the PROC SYSLIN parameter estimates from the MODEL LOGY=LOGX. The standard error reported for ALPHA is different from that for the INTERCEPT in the log form.

The preceding example is not intended to suggest that loglinear models should be estimated in level form but, rather, to make the following points:

- Nonlinear transformations of equations involve the error term of the equation, and this should be taken into account when transforming models.
- The RESID.-prefixed and the EQ.-prefixed equation variables for models estimated by the MODEL procedure must represent additive errors with zero means.
- You can use assignments to RESID.-prefixed and EQ.-prefixed equation variables to transform error terms.
- Some models do not have additive errors or zero means, and many such models can be estimated using the MODEL procedure. The preceding approach applies not only to multiplicative models but to any model that can be manipulated to isolate the error term.

Predicted Values of Transformed Models

Nonadditive or transformed errors affect the distribution of the predicted values, as well as the estimates. For the preceding loglinear example, the MODEL procedure produces consistent parameter estimates. However, the predicted values for Y computed by PROC MODEL are not unbiased estimates of the expected values of Y, although they do estimate the conditional median Y values.

In general, the predicted values produced for a model with nonadditive errors are not unbiased estimates of the conditional means of the endogenous value. If the model can be transformed to a model with additive errors by using a *monotonic* transformation, the predicted values estimate the conditional medians of the endogenous variable.

For transformed models in which the biasing factor is known, you can use programming statements to correct for the bias in the predicted values as estimates of the endogenous means. In the preceding log-linear case, the predicted values are biased by the factor $\exp(\sigma^2/2)$. You can produce approximately unbiased predicted values in this case by writing the model as

```
proc model data=in;
   parms alpha beta;
   y=alpha * x ** beta;
   resid.y = log( actual.y / pred.y );
   fit y;
run;
```

See Miller (1984) for a discussion of bias factors for predicted values of transformed models.

Note that models with transformed errors are not appropriate for Monte Carlo simulation that uses the SDATA= option. PROC MODEL computes the OUTS= matrix from the transformed RESID.-prefixed equation variables, while it uses the SDATA= matrix to generate multivariate normal errors, which are added to the predicted values. This method of computing errors is inconsistent when the equation variables have been transformed.

Error Covariance Structure Specification

One of the key assumptions of regression is that the variance of the errors is constant across observations. Correcting for heteroscedasticity improves the efficiency of the estimates.

Consider the following general form for models:

$$
\begin{aligned}
\mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta) &= \boldsymbol{\varepsilon}_t \\
\boldsymbol{\varepsilon}_t &= H_t * \boldsymbol{\epsilon}_t \\
H_t &= \begin{bmatrix} \sqrt{h_{t,1}} & 0 & \dots & 0 \\ 0 & \sqrt{h_{t,2}} & \dots & 0 \\ & & \ddots & \\ 0 & 0 & \dots & \sqrt{h_{t,g}} \end{bmatrix} \\
\mathbf{h}_t &= \mathbf{g}(\mathbf{y}_t, \mathbf{x}_t, \phi)
\end{aligned}
$$

where $\boldsymbol{\epsilon}_t \sim N(0, \Sigma)$.

For models that are homoscedastic,

$$h_t = 1$$

If you have a model that is heteroscedastic with known form, you can improve the efficiency of the estimates by performing a weighted regression. The weight variable, using this notation, would be $1/\sqrt{h_t}$.

If the errors for a model are heteroscedastic and the functional form of the variance is known, the model for the variance can be estimated along with the regression function.

To specify a functional form for the variance, assign the function to an H.*var* variable where *var* is the equation variable. For example, if you want to estimate the scale parameter for the variance of a simple regression model

$$y = a * x + b$$

you can specify

```
proc model data=s;
   y = a * x + b;
   h.y = sigma**2;
fit y;
```

Consider the same model with the following functional form for the variance:

$$h_t = \sigma^2 * x^{2*\alpha}$$

This would be written as

```
proc model data=s;
   y = a * x + b;
   h.y = sigma**2 * x**(2*alpha);
fit y;
```

There are three ways to model the variance in the MODEL procedure: feasible generalized least squares, generalized method of moments, and full information maximum likelihood.

Feasible GLS

A simple approach to estimating a variance function is to estimate the mean parameters θ by using some auxiliary method, such as OLS, and then use the residuals of that estimation to estimate the parameters ϕ of the variance function. This scheme is called *feasible GLS*. It is possible to use the residuals from an auxiliary method for the purpose of estimating ϕ because in many cases the residuals consistently estimate the error terms.

For all estimation methods except GMM and FIML, using the H.var syntax specifies that feasible GLS is used in the estimation. For feasible GLS, the mean function is estimated by the usual method. The variance function is then estimated using pseudo-likelihood (PL) function of the generated residuals. The objective function for the PL estimation is

$$p_n(\sigma, \theta) = \sum_{i=1}^{n} \left(\frac{(y_i - f(x_i, \hat{\beta}))^2}{\sigma^2 h(z_i, \theta)} + \log[\sigma^2 h(z_i, \theta)] \right)$$

Once the variance function has been estimated, the mean function is reestimated by using the variance function as weights. If an S-iterated method is selected, this process is repeated until convergence (iterated feasible GLS).

Note that feasible GLS does not yield consistent estimates when one of the following is true:

- The variance is unbounded.
- There is too much serial dependence in the errors (the dependence does not fade with time).
- There is a combination of serial dependence and lag dependent variables.

The first two cases are unusual, but the third is much more common. Whether iterated feasible GLS avoids consistency problems with the last case is an unanswered research question. For more information see: Davidson and MacKinnon (1993, pp. 298–301); Gallant (1987, pp. 124–125); Amemiya (1985, pp. 202–203).

One limitation is that parameters cannot be shared between the mean equation and the variance equation. This implies that certain GARCH models, cross-equation restrictions of parameters, or testing of combinations of parameters in the mean and variance component are not allowed.

Generalized Method of Moments

In GMM, normally the first moment of the mean function is used in the objective function.

$$\begin{aligned} \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta) &= \epsilon_t \\ \mathbf{E}(\epsilon_t) &= 0 \end{aligned}$$

To add the second moment conditions to the estimation, add the equation

$$\mathbf{E}(\varepsilon_t * \varepsilon_t - h_t) = 0$$

to the model. For example, if you want to estimate σ for linear example above, you can write

```
proc model data=s;
   y = a * x + b;
   eq.two = resid.y**2 - sigma**2;
fit y two/ gmm;
instruments x;
run;
```

This is a popular way to estimate a continuous-time interest rate processes (see Chan et al. 1992). The H.var syntax automatically generates this system of equations.

To further take advantage of the information obtained about the variance, the moment equations can be modified to

$$\begin{aligned} \mathbf{E}(\varepsilon_t/\sqrt{h_t}) &= 0 \\ \mathbf{E}(\varepsilon_t * \varepsilon_t - h_t) &= 0 \end{aligned}$$

For the above example, this can be written as

```
proc model data=s;
   y = a * x + b;
   eq.two = resid.y**2 - sigma**2;
   resid.y = resid.y / sigma;
fit y two/ gmm;
instruments x;
run;
```

Note that, if the error model is misspecified in this form of the GMM model, the parameter estimates might be inconsistent.

Full Information Maximum Likelihood

For FIML estimation of variance functions, the concentrated likelihood below is used as the objective function. That is, the mean function is coupled with the variance function and the system is solved simultaneously.

$$\begin{aligned} l_n(\phi) &= \frac{ng}{2}(1+\ln(2\pi)) - \sum_{t=1}^{n} \ln\left(\left|\frac{\partial \mathbf{q}(\mathbf{y}_t, \mathbf{x}_t, \theta)}{\partial \mathbf{y}_t}\right|\right) \\ &\quad + \frac{1}{2}\sum_{t=1}^{n}\sum_{i=1}^{g}\left(\ln(h_{t,i}) + \mathbf{q}_i(\mathbf{y}_t, \mathbf{x}_t, \theta)^2/h_{t,i}\right) \end{aligned}$$

where g is the number of equations in the system.

The HESSIAN=GLS option is not available for FIML estimation that involves variance functions. The matrix used when HESSIAN=CROSS is specified is a crossproducts matrix that has been enhanced by the dual quasi-Newton approximation.

Examples

You can specify a GARCH(1,1) model as follows:

```
proc model data=modloc.usd_jpy;

             /* Mean model --------*/
   jpyret = intercept ;

             /* Variance model ----------------*/
   h.jpyret = arch0
            + arch1 * xlag( resid.jpyret ** 2, mse.jpyret  )
            + garch1 * xlag(h.jpyret, mse.jpyret) ;

   bounds arch0 arch1 garch1 >= 0;

fit jpyret / method=marquardt fiml;
run;
```

Note that the BOUNDS statement is used to ensure that the parameters are positive, a requirement for GARCH models.

EGARCH models are used because there are no restrictions on the parameters. You can specify a EGARCH(1,1) model as follows:

```
proc model data=sasuser.usd_dem ;

          /* Mean model ----------*/
  demret = intercept ;

             /* Variance model ----------------*/
  if ( _OBS_ =1 )  then
    h.demret = exp( earch0 + egarch1 * log(mse.demret)  );
  else
    h.demret = exp( earch0 + earch1 * zlag( g)
                          + egarch1 * log(zlag(h.demret)));
  g = - theta * nresid.demret + abs( nresid.demret ) - sqrt(2/3.1415);

fit demret / method=marquardt fiml  maxiter=100 converge=1.0e-6;
run;
```

Ordinary Differential Equations

Ordinary differential equations (ODEs) are also called *initial value problems* because a time zero value for each first-order differential equation is needed. The following is an example of a first-order system of ODEs:

$$
\begin{aligned}
y' &= -0.1y + 2.5z^2 \\
z' &= -z \\
y_0 &= 0 \\
z_0 &= 1
\end{aligned}
$$

Note that you must provide an initial value for each ODE.

As a reminder, any n-order differential equation can be modeled as a system of first-order differential equations. For example, consider the differential equations

$$\begin{aligned} y'' &= by' + cy \\ y_0 &= 0 \\ y'_0 &= 1 \end{aligned}$$

which can be written as the system of differential equations

$$\begin{aligned} y' &= z \\ z' &= by' + cy \\ y_0 &= 0 \\ z_0 &= 1 \end{aligned}$$

This differential system can be simulated as follows:

```
data t;
   time=0; output;
   time=1; output;
   time=2; output;
run;

proc model data=t ;
   dependent y 0 z 1;
   parm b -2 c -4;

   dert.y = z;
   dert.z = b * dert.y + c * y;

   solve y z / dynamic solveprint;
run;
```

The preceding statements produce the output shown in Figure 26.46. These statements produce additional output, which is not shown.

Figure 26.46 Simulation Results for Differential System

The MODEL Procedure
Simultaneous Simulation

Observation	1	**Missing**	2	**CC**	-1.000000
		Iterations	0		

Solution Values	
y	**z**
0.000000	1.000000

Figure 26.46 *continued*

Observation	2	Iterations	0	CC	0.000000	ERROR.y	0.000000

Solution Values	
y	z
0.2096398	-.2687053

Observation	3	Iterations	0	CC	9.464802	ERROR.y	-0.234405

Solution Values	
y	z
-.0247649	-.1035929

The differential variables are distinguished by the derivative with respect to time (DERT.) prefix. Once you define the DERT. variable, you can use it on the right-hand side of another equation. The differential equations must be expressed in normal form; implicit differential equations are not allowed, and other terms on the left-hand side are not allowed.

The TIME variable is the *implied with respect to* variable for all DERT. variables. The TIME variable is also the only variable that must be in the input data set.

You can provide initial values for the differential equations in the data set, in the declaration statement (as in the previous example), or in statements in the program. Using the previous example, you can specify the initial values as

```
proc model data=t ;
   dependent y z ;
   parm b -2 c -4;

   if ( time=0 ) then
      do;
         y=0;
         z=1;
      end;
   else
      do;
         dert.y = z;
         dert.z = b * dert.y + c * y;
      end;
   end;

   solve y z / dynamic solveprint;
run;
```

If you do not provide an initial value, 0 is used.

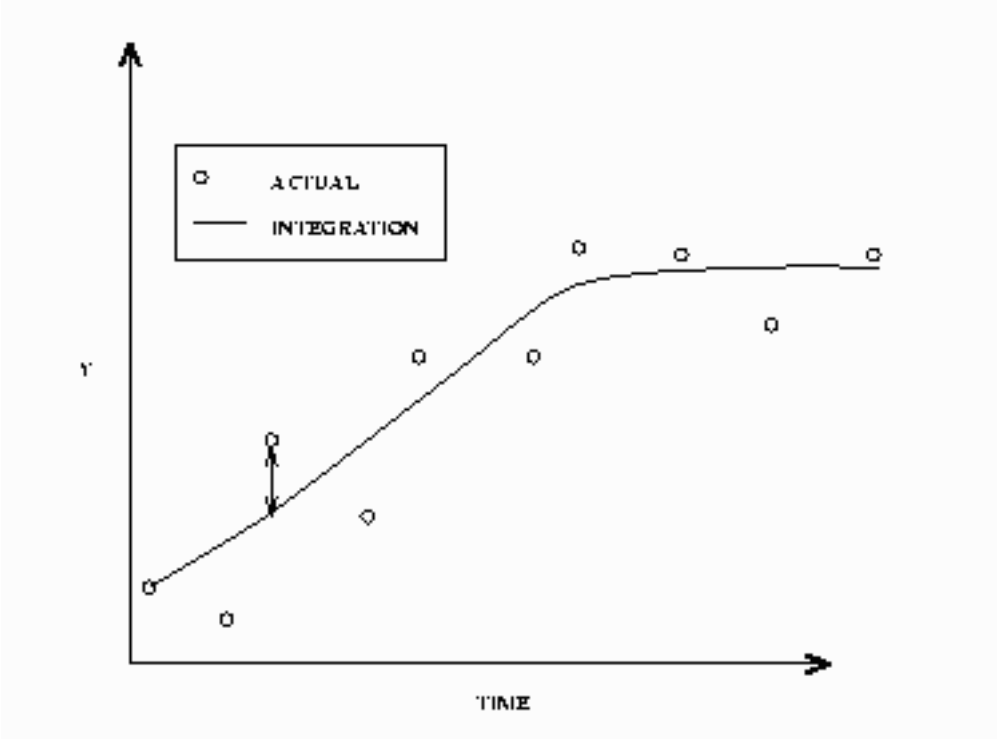
DYNAMIC and STATIC Simulation

Note that, in the previous example, the DYNAMIC option is specified in the SOLVE statement. The DYNAMIC and STATIC options work the same for differential equations as they do for dynamic systems. In the differential equation case, the DYNAMIC option makes the initial value needed at each observation the computed value from the previous iteration. For a static simulation, the data set must contain values for the integrated variables. For example, if DERT.Y and DERT.Z are the differential variables, you must include Y and Z in the input data set in order to do a static simulation of the model.

If the simulation is dynamic, the initial values for the differential equations are obtained from the data set, if they are available. If the variable is not in the data set, you can specify the initial value in a declaration statement. If you do not specify an initial value, the value of 0.0 is used.

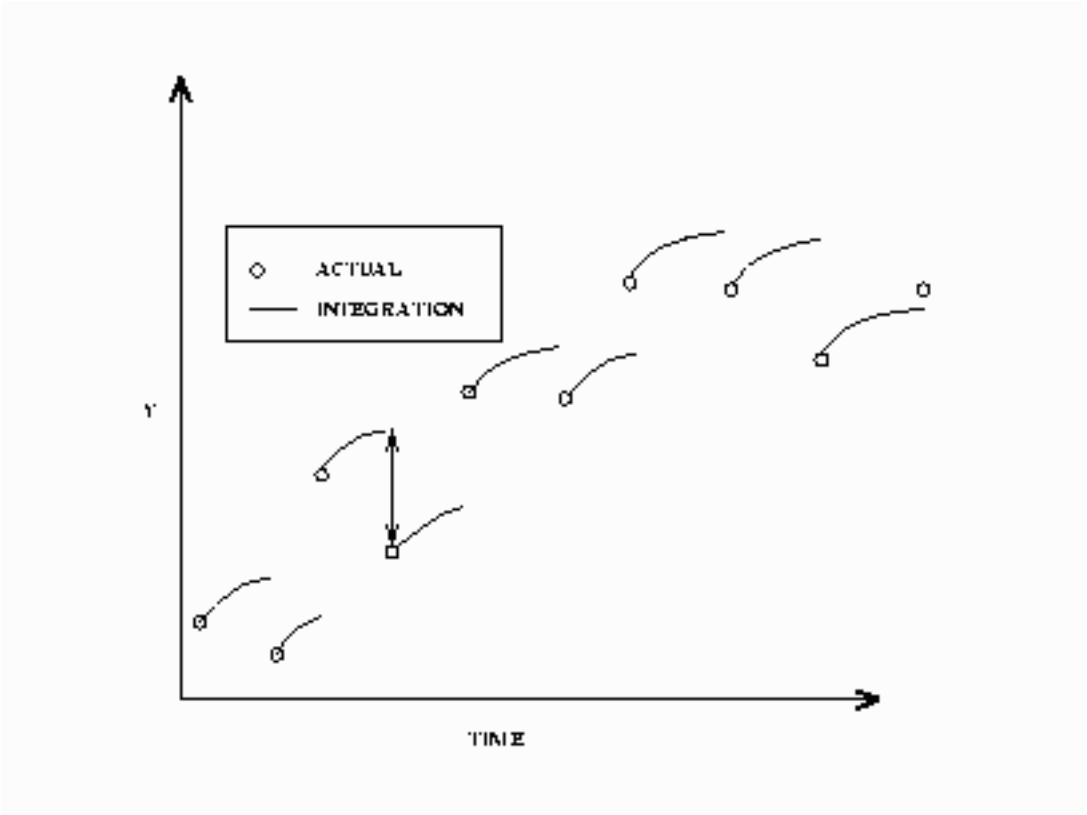
A dynamic solution is obtained by solving one initial value problem for all the data. A graph of a simple dynamic simulation is shown in Figure 26.47. If the time variable for the current observation is less than the time variable for the previous observation, the integration is restarted from this point. This allows for multiple samples in one data file.

Figure 26.47 Dynamic Solution



In a static solution, n–1 initial value problems are solved using the first n–1 data values as initial values. The equations are integrated using the *i* th data value as an initial value to the *i*+1 data value. Figure 26.48 displays a static simulation of noisy data from a simple differential equation. The static solution does not propagate errors in initial values as the dynamic solution does.

Figure 26.48 Static Solution



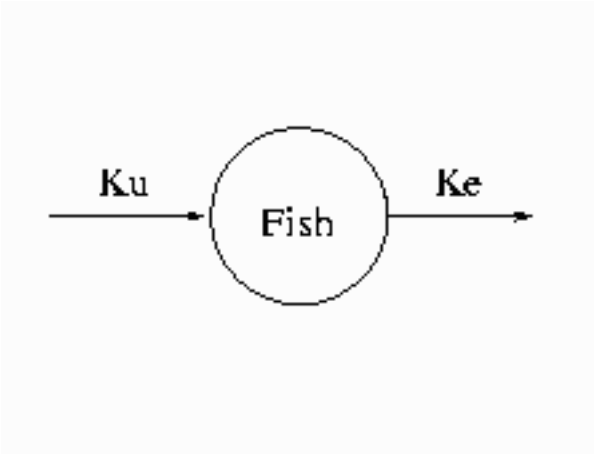
For estimation, the DYNAMIC and STATIC options in the FIT statement perform the same functions as they do in the SOLVE statement. Components of differential systems that have missing values or are not in the data set are simulated dynamically. For example, often in multiple compartment kinetic models, only one compartment is monitored. The differential equations that describe the unmonitored compartments are simulated dynamically.

For estimation, it is important to have accurate initial values for ODEs that are not in the data set. If an accurate initial value is not known, the initial value can be made an unknown parameter and estimated. This allows for errors in the initial values but increases the number of parameters to estimate by the number of equations.

Estimation of Differential Equations

Consider the kinetic model for the accumulation of mercury (Hg) in mosquito fish (Matis, Miller, and Allen 1991, p. 177). The model for this process is the one-compartment constant infusion model shown in Figure 26.49.

Figure 26.49 One-Compartment Constant Infusion Model



The differential equation that models this process is

$$\begin{aligned} \frac{dconc}{dt} &= k_u - k_e conc \\ conc_0 &= 0 \end{aligned}$$

The analytical solution to the model is

$$conc = (k_u/k_e)(1 - \exp(-k_e t))$$

The data for the model are

```
data fish;
   input day conc;
datalines;
0.0    0.0
1.0    0.15
2.0    0.2
3.0    0.26
4.0    0.32
6.0    0.33
;
```

To fit this model in differential form, use the following statements:

```
proc model data=fish;
   parm ku ke;

   dert.conc = ku - ke * conc;

   fit conc / time=day;
run;
```

The results from this estimation are shown in Figure 26.50.

Figure 26.50 Static Estimation Results for Fish Model

The MODEL Procedure

Nonlinear OLS Parameter Estimates

Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|
ku	0.180159	0.0312	5.78	0.0044
ke	0.524661	0.1181	4.44	0.0113

To perform a dynamic estimation of the differential equation, add the DYNAMIC option to the FIT statement.

```
proc model data=fish;
   parm ku .3 ke .3;

   dert.conc = ku - ke * conc;

   fit conc / time = day dynamic;
run;
```

The equation DERT.CONC is integrated from $conc(0) = 0$. The results from this estimation are shown in Figure 26.51.

Figure 26.51 Dynamic Estimation Results for Fish Model

The MODEL Procedure

Nonlinear OLS Parameter Estimates

Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|
ku	0.167109	0.0170	9.84	0.0006
ke	0.469033	0.0731	6.42	0.0030

To perform a dynamic estimation of the differential equation and estimate the initial value, use the following statements:

```
proc model data=fish;
   parm ku .3 ke .3 conc0 0;

   dert.conc = ku - ke * conc;

   fit conc initial=(conc = conc0) / time = day dynamic;
run;
```

The INITIAL= option in the FIT statement is used to associate the initial value of a differential equation with a parameter. The results from this estimation are shown in Figure 26.52.

Figure 26.52 Dynamic Estimation with Initial Value for Fish Model

The MODEL Procedure

Nonlinear OLS Parameter Estimates

Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|
ku	0.164408	0.0230	7.14	0.0057
ke	0.45949	0.0943	4.87	0.0165
conc0	0.003798	0.0174	0.22	0.8414

Finally, to estimate the fish model by using the analytical solution, use the following statements:

```
proc model data=fish;
   parm ku .3  ke .3;

   conc = (ku/ ke)*( 1 -exp(-ke * day));

   fit conc;
run;
```

The results from this estimation are shown in Figure 26.53.

Figure 26.53 Analytical Estimation Results for Fish Model

The MODEL Procedure

Nonlinear OLS Parameter Estimates

Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|
ku	0.167109	0.0170	9.84	0.0006
ke	0.469033	0.0731	6.42	0.0030

A comparison of the results among the four estimations reveals that the two dynamic estimations and the analytical estimation give nearly identical results (identical to the default precision). The two dynamic estimations are identical because the estimated initial value (0.00013071) is very close to the initial value used in the first dynamic estimation (0). Note also that the static model did not require an initial guess for the parameter values. Static estimation, in general, is more forgiving of bad initial values.

The form of the estimation that is preferred depends mostly on the model and data. If a very accurate initial value is known, then a dynamic estimation makes sense. If, additionally, the model can be written analytically, then the analytical estimation is computationally simpler. If only an approximate initial value is known and not modeled as an unknown parameter, the static estimation is less sensitive to errors in the initial value.

The form of the error in the model is also an important factor in choosing the form of the estimation. If the error term is additive and independent of previous error, then the dynamic mode is appropriate. If, on the other hand, the errors are cumulative, a static estimation is more appropriate. See the section "Monte Carlo Simulation" on page 1642 for an example.

Auxiliary Equations

Auxiliary equations can be used with differential equations. These are equations that need to be satisfied with the differential equations at each point between each data value. They are automatically added to the system, so you do not need to specify them in the SOLVE or FIT statement.

Consider the following example.

The Michaelis-Menten equations describe the kinetics of an enzyme-catalyzed reaction. The enzyme is E, and S is called the *substrate*. The enzyme first reacts with the substrate to form the enzyme-substrate complex ES, which then breaks down in a second step to form enzyme and products P.

The reaction rates are described by the following system of differential equations:

$$\frac{d[ES]}{dt} = k_1([E] - [ES])[S] - k_2[ES] - k_3[ES]$$
$$\frac{d[S]}{dt} = -k_1([E] - [ES])[S] + k_2[ES]$$
$$[E] = [E]_{tot} - [ES]$$

The first equation describes the rate of formation of ES from E + S. The rate of formation of ES from E + P is very small and can be ignored. The enzyme is in either the complexed or the uncomplexed form. So if the total ($[E]_{tot}$) concentration of enzyme and the amount bound to the substrate is known, $[E]$ can be obtained by conservation.

In this example, the conservation equation is an auxiliary equation and is coupled with the differential equations for integration.

Time Variable

You must provide a time variable in the data set. The name of the time variable defaults to TIME. You can use other variables as the time variable by specifying the TIME= option in the FIT or SOLVE statement. The time intervals need not be evenly spaced. If the time variable for the current observation is less than the time variable for the previous observation, the integration is restarted.

Differential Equations and Goal Seeking

Consider the following differential equation

$$y' = a * x$$

and the data set

```
data t2;
   y=0; time=0; output;
   y=2; time=1; output;
   y=3; time=2; output;
run;
```

The problem is to find values for X that satisfy the differential equation and the data in the data set. Problems of this kind are sometimes referred to as *goal-seeking problems* because they require you to search for values of X that satisfy the goal of Y.

This problem is solved with the following statements:

```
proc model data=t2;
   independent x 0;
   dependent y;
   parm a 5;
   dert.y = a * x;
   solve x / out=goaldata;
run;

proc print data=goaldata;
run;
```

The output from the PROC PRINT statement is shown in Figure 26.54.

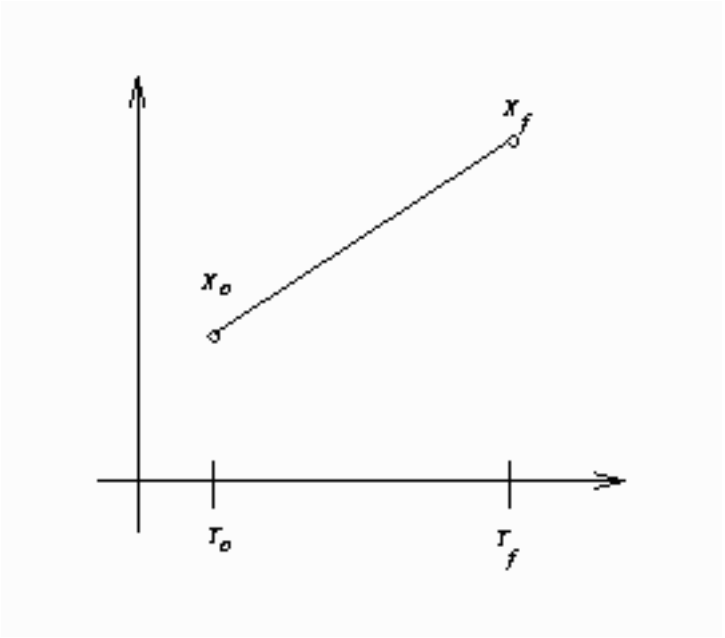
Figure 26.54 Dynamic Solution

Obs	_TYPE_	_MODE_	_ERRORS_	x	y	time
1	PREDICT	SIMULATE	0	0.0	0	0
2	PREDICT	SIMULATE	0	0.8	2	1
3	PREDICT	SIMULATE	0	-0.4	3	2

Note that an initial value of 0 is provided for the X variable because it is undetermined at TIME = 0.

In the preceding goal-seeking example, X is treated as a linear function between each set of data points (see Figure 26.55).

Figure 26.55 Form of X Used for Integration in Goal Seeking



If you integrate $y' = ax$ manually, you have

$$\begin{aligned} x(t) &= \frac{t_f - t}{t_f - t_o} x_o + \frac{t - t_o}{t_f - t_o} x_f \\ y_f - y_o &= \int_{t_o}^{t_f} ax(t)\, dt \\ &= a \frac{1}{t_f - t_o} (t(t_f x_o - t_o x_f) + \frac{1}{2} t^2 (x_f - x_o)) \Big|_{t_o}^{t_f} \end{aligned}$$

For observation 2, this reduces to

$$\begin{aligned} y_f - y_o &= \frac{1}{2} a * x_f \\ 2 &= 2.5 * x_f \end{aligned}$$

So $x = 0.8$ for this observation.

Goal seeking for the TIME variable is not allowed.

Restrictions and Bounds on Parameters

Using the BOUNDS and RESTRICT statements, PROC MODEL can compute optimal estimates subject to equality or inequality constraints on the parameter estimates.

Equality restrictions can be written as a vector function:

$$\mathbf{h}(\theta) = 0$$

Inequality restrictions are either active or inactive. When an inequality restriction is active, it is treated as an equality restriction. All inactive inequality restrictions can be written as a vector function:

$$F(\theta) \geq 0$$

Strict inequalities, such as $(f(\theta) > 0)$, are transformed into inequalities as $f(\theta) \times (1 - \epsilon) - \epsilon \geq 0$, where the tolerance ϵ is controlled by the EPSILON= option in the FIT statement and defaults to 10^{-8}. The ith inequality restriction becomes active if $F_i < 0$ and remains active until its Lagrange multiplier becomes negative. Lagrange multipliers are computed for all the nonredundant equality restrictions and all the active inequality restrictions.

For the following, assume the vector $\mathbf{h}(\theta)$ contains all the current active restrictions. The constraint matrix $\mathbf{A}$ is

$$\mathbf{A}(\hat{\theta}) = \frac{\partial \mathbf{h}(\hat{\theta})}{\partial \hat{\theta}}$$

The covariance matrix for the restricted parameter estimates is computed as

$$\mathbf{Z}(\mathbf{Z}'\mathbf{H}\mathbf{Z})^{-1}\mathbf{Z}'$$

where **H** is Hessian or approximation to the Hessian of the objective function ($(\mathbf{X}'(\mathrm{diag}(\mathbf{S})^{-1}\otimes\mathbf{I})\mathbf{X})$ for OLS), and **Z** is the last $(np - nc)$ columns of **Q**. **Q** is from an LQ factorization of the constraint matrix, nc is the number of active constraints, and np is the number of parameters. For more details on LQ factorization, see Gill, Murray, and Wright (1981). The covariance column in Table 26.2 summarizes the Hessian approximation used for each estimation method.

The covariance matrix for the Lagrange multipliers is computed as

$$(\mathbf{A}\mathbf{H}^{-1}\mathbf{A}')^{-1}$$

The p-value reported for a restriction is computed from a beta distribution rather than a t distribution because the numerator and the denominator of the t ratio for an estimated Lagrange multiplier are not independent.

The Lagrange multipliers for the active restrictions are printed with the parameter estimates. The Lagrange multiplier estimates are computed using the relationship

$$\mathbf{A}'\lambda = \mathrm{g}$$

where the dimensions of the constraint matrix **A** are the number of constraints by the number of parameters, λ is the vector of Lagrange multipliers, and g is the gradient of the objective function at the final estimates.

The final gradient includes the effects of the estimated **S** matrix. For example, for OLS the final gradient would be:

$$\mathrm{g} = \mathbf{X}'(\mathrm{diag}(\mathbf{S})^{-1}\otimes\mathbf{I})r$$

where r is the residual vector. Note that when nonlinear restrictions are imposed, the convergence measure R might have values greater than one for some iterations.

Tests on Parameters

In general, the hypothesis tested can be written as

$$H_0 : \mathbf{h}(\theta) = 0$$

where $\mathbf{h}(\theta)$ is a vector-valued function of the parameters θ given by the r expressions specified on the TEST statement.

Let $\hat{\mathbf{V}}$ be the estimate of the covariance matrix of $\hat{\theta}$. Let $\hat{\theta}$ be the unconstrained estimate of θ and $\tilde{\theta}$ be the constrained estimate of θ such that $h(\tilde{\theta}) = 0$. Let

$$\mathbf{A}(\theta) = \partial h(\theta)/\partial\theta \mid_{\hat{\theta}}$$

Let r be the dimension of $h(\theta)$ and n be the number of observations. Using this notation, the test statistics for the three kinds of tests are computed as follows.

The Wald test statistic is defined as

$$W = h'(\hat{\theta})\left(\mathbf{A}(\hat{\theta})\hat{\mathbf{V}}\mathbf{A}'(\hat{\theta})\right)^{-1}h(\hat{\theta})$$

The Wald test is not invariant to reparameterization of the model (Gregory and Veall 1985; Gallant 1987, p. 219). For more information about the theoretical properties of the Wald test, see Phillips and Park (1988).

The Lagrange multiplier test statistic is

$$R = \lambda' \mathbf{A}(\tilde{\theta})\tilde{\mathbf{V}}\mathbf{A}'(\tilde{\theta})\lambda$$

where λ is the vector of Lagrange multipliers from the computation of the restricted estimate $\tilde{\theta}$.

The Lagrange multiplier test statistic is equivalent to Rao's efficient score test statistic:

$$R = (\partial L(\tilde{\theta})/\partial\theta)' \tilde{\mathbf{V}} (\partial L(\tilde{\theta})/\partial\theta)$$

where L is the log-likelihood function for the estimation method used. For SUR, 3SLS, GMM, and iterated versions of these methods, the likelihood function is computed as

$$L = \text{Objective} \times Nobs/2$$

For OLS and 2SLS, the Lagrange multiplier test statistic is computed as:

$$R = [(\partial \hat{S}(\tilde{\theta})/\partial\theta)' \tilde{\mathbf{V}} (\partial \hat{S}(\tilde{\theta})/\partial\theta)]/\hat{S}(\tilde{\theta})$$

where $\hat{S}(\tilde{\theta})$ is the corresponding objective function value at the constrained estimate.

The likelihood ratio test statistic is

$$T = 2\left(L(\hat{\theta}) - L(\tilde{\theta})\right)$$

where $\tilde{\theta}$ represents the constrained estimate of θ and L is the concentrated log-likelihood value.

For OLS and 2SLS, the likelihood ratio test statistic is computed as:

$$T = (n - nparms) \times (\hat{S}(\tilde{\theta}) - \hat{S}(\hat{\theta}))/\hat{S}(\hat{\theta})$$

This test statistic is an approximation from

$$T = n \times log\left(1 + \frac{rF}{n - nparms}\right)$$

when the value of $rF/(n - nparms)$ is small (Greene 2004, p. 421).

The likelihood ratio test is not appropriate for models with nonstationary serially correlated errors (Gallant 1987, p. 139). The likelihood ratio test should not be used for dynamic systems, for systems with lagged dependent variables, or with the FIML estimation method unless certain conditions are met (see Gallant 1987, p. 479).

For each kind of test, under the null hypothesis the test statistic is asymptotically distributed as a χ^2 random variable with *r* degrees of freedom, where *r* is the number of expressions in the TEST statement. The *p*-values reported for the tests are computed from the $\chi^2(r)$ distribution and are only asymptotically valid. When both RESTRICT and TEST statements are used in a PROC MODEL step, test statistics are computed by taking into account the constraints imposed by the RESTRICT statement.

Monte Carlo simulations suggest that the asymptotic distribution of the Wald test is a poorer approximation to its small sample distribution than the other two tests. However, the Wald test has the least computational cost, since it does not require computation of the constrained estimate $\tilde{\theta}$.

The following is an example of using the TEST statement to perform a likelihood ratio test for a compound hypothesis.

```
test a*exp(-k) = 1-k, d = 0 ,/ lr;
```

It is important to keep in mind that although individual *t* tests for each parameter are printed by default into the parameter estimates table, they are only asymptotically valid for nonlinear models. You should be cautious in drawing any inferences from these *t* tests for small samples.

Hausman Specification Test

Hausman's specification test, or *m*-statistic, can be used to test hypotheses in terms of bias or inconsistency of an estimator. This test was also proposed by Wu (1973). Hausman's *m*-statistic is as follows.

Given two estimators, $\hat{\beta}_0$ and $\hat{\beta}_1$, where under the null hypothesis both estimators are consistent but only $\hat{\beta}_0$ is asymptotically efficient and under the alternative hypothesis only $\hat{\beta}_1$ is consistent, the *m*-statistic is

$$m = \hat{q}'(\hat{\mathbf{V}}_1 - \hat{\mathbf{V}}_0)^{-}\hat{q}$$

where $\hat{\mathbf{V}}_1$ and $\hat{\mathbf{V}}_0$ represent consistent estimates of the asymptotic covariance matrices of $\hat{\beta}_1$ and $\hat{\beta}_0$ respectively, and

$$q = \hat{\beta}_1 - \hat{\beta}_0$$

The *m*-statistic is then distributed χ^2 with k degrees of freedom, where k is the rank of the matrix $(\hat{\mathbf{V}}_1 - \hat{\mathbf{V}}_0)$. A generalized inverse is used, as recommended by Hausman and Taylor (1982).

In the MODEL procedure, Hausman's *m*-statistic can be used to determine if it is necessary to use an instrumental variables method rather than a more efficient OLS estimation. Hausman's *m*-statistic can also be used to compare 2SLS with 3SLS for a class of estimators for which 3SLS is asymptotically efficient (similarly for OLS and SUR).

Hausman's *m*-statistic can also be used, in principle, to test the null hypothesis of normality when comparing 3SLS to FIML. Because of the poor performance of this form of the test, it is not offered in the MODEL procedure. See Fair (1984, pp. 246–247) for a discussion of why Hausman's test fails for common econometric models.

To perform a Hausman's specification test, specify the HAUSMAN option in the FIT statement. The selected estimation methods are compared using Hausman's *m*-statistic.

In the following example, Hausman's test is used to check the presence of measurement error. Under H_0 of no measurement error, OLS is efficient, while under H_1, 2SLS is consistent. In the following code, OLS and 2SLS are used to estimate the model, and Hausman's test is requested.

```
proc model data=one out=fiml2;
   endogenous y1 y2;

   y1 = py2 * y2 + px1 * x1 + interc;
   y2 = py1* y1 + pz1 * z1 + d2;

   fit y1 y2 / ols 2sls hausman;
   instruments x1 z1;
run;
```

The output specified by the HAUSMAN option produces the results shown in Figure 26.56.

Figure 26.56 Hausman's Specification Test Results

The MODEL Procedure

Hausman's Specification Test Results				
Efficient under H0	**Consistent under H1**	**DF**	**Statistic**	**Pr > ChiSq**
OLS	2SLS	6	13.86	0.0313

Figure 26.56 indicates that 2SLS is preferred over OLS at 5% level of significance. In this case, the null hypothesis of no measurement error is rejected. Hence, the instrumental variable estimator is required for this example due to the presence of measurement error.

Chow Tests

The Chow test is used to test for break points or structural changes in a model. The problem is posed as a partitioning of the data into two parts of size n_1 and n_2. The null hypothesis to be tested is

$$H_o : \beta_1 = \beta_2 = \beta$$

where β_1 is estimated by using the first part of the data and β_2 is estimated by using the second part.

The test is performed as follows (see Davidson and MacKinnon 1993, p. 380).

1. The p parameters of the model are estimated.
2. A second linear regression is performed on the residuals, $\hat{u}$, from the nonlinear estimation in step one.

   $$\hat{u} = \hat{\mathbf{X}}b + \text{residuals}$$

   where $\hat{\mathbf{X}}$ is Jacobian columns that are evaluated at the parameter estimates. If the estimation is an instrumental variables estimation with matrix of instruments $\mathbf{W}$, then the following regression is performed:

   $$\hat{u} = \mathbf{P}_{W*}\hat{\mathbf{X}}b + \text{residuals}$$

   where $\mathbf{P}_{W*}$ is the projection matrix.
3. The restricted SSE (RSSE) from this regression is obtained. An SSE for each subsample is then obtained by using the same linear regression.
4. The F statistic is then

   $$f = \frac{(RSSE - SSE_1 - SSE_2)/p}{(SSE_1 + SSE_2)/(n - 2p)}$$

   This test has p and $n - 2p$ degrees of freedom.

Chow's test is not applicable if $\min(n_1, n_2) < p$, since one of the two subsamples does not contain enough data to estimate β. In this instance, the *predictive Chow test* can be used. The predictive Chow test is defined as

$$f = \frac{(RSSE - SSE_1)\times(n_1 - p)}{SSE_1 * n_2}$$

where $n_1 > p$. This test can be derived from the Chow test by noting that the $SSE_2 = 0$ when $n_2 <= p$ and by adjusting the degrees of freedom appropriately.

You can select the Chow test and the predictive Chow test by specifying the CHOW=*arg* and the PCHOW=*arg* options in the FIT statement, where *arg* is either the number of observations in the first sample or a parenthesized list of first sample sizes. If the size of the one of the two groups in which the sample is partitioned is less than the number of parameters, then a predictive Chow test is automatically used. These tests statistics are not produced for GMM and FIML estimations.

The following is an example of the use of the Chow test.

```
data exp;
   x=0;
   do time=1 to 100;
      if time=50 then x=1;
      y = 35 * exp( 0.01 * time ) + rannor( 123 ) + x * 5;
      output;
   end;
run;

proc model data=exp;
   parm zo 35 b;
      dert.z = b * z;
      y=z;
   fit y init=(z=zo) / chow =(40 50 60) pchow=90;
run;
```

The data set introduces an artificial structural change into the model (the structural change effects the intercept parameter). The output from the requested Chow tests are shown in Figure 26.57.

Figure 26.57 Chow's Test Results

The MODEL Procedure

Structural Change Test					
Test	Break Point	Num DF	Den DF	F Value	Pr > F
Chow	40	2	96	12.95	<.0001
Chow	50	2	96	101.37	<.0001
Chow	60	2	96	26.43	<.0001
Predictive Chow	90	11	87	1.86	0.0566

Profile Likelihood Confidence Intervals

Wald-based and likelihood-ratio-based confidence intervals are available in the MODEL procedure for computing a confidence interval on an estimated parameter. A confidence interval on a parameter θ can be constructed by inverting a Wald-based or a likelihood-ratio-based test.

The approximate $100(1-\alpha)$ % Wald confidence interval for a parameter θ is

$$\hat{\theta} \pm z_{1-\alpha/2}\hat{\sigma}$$

where z_p is the $100p$th percentile of the standard normal distribution, $\hat{\theta}$ is the maximum likelihood estimate of θ, and $\hat{\sigma}$ is the standard error estimate of $\hat{\theta}$.

A likelihood-ratio-based confidence interval is derived from the χ^2 distribution of the generalized likelihood ratio test. The approximate $1-\alpha$ confidence interval for a parameter θ is

$$\theta : 2[l(\hat{\theta}) - l(\theta)] \leq q_{1,1-\alpha} = 2l^*$$

where $q_{1,1-\alpha}$ is the $(1-\alpha)$ quantile of the χ^2 with one degree of freedom, and $l(\theta)$ is the log likelihood as a function of one parameter. The endpoints of a confidence interval are the zeros of the function $l(\theta) - l^*$. Computing a likelihood-ratio-based confidence interval is an iterative process. This process must be performed twice for each parameter, so the computational cost is considerable. Using a modified form of the algorithm recommended by Venzon and Moolgavkar (1988), you can determine that the cost of each endpoint computation is approximately the cost of estimating the original system.

To request confidence intervals on estimated parameters, specify the PRL= option in the FIT statement. By default, the PRL option produces 95% likelihood ratio confidence limits. The coverage of the confidence interval is controlled by the ALPHA= option in the FIT statement.

The following is an example of the use of the confidence interval options.

```
data exp;
   do time = 1 to 20;
      y = 35 * exp( 0.01 * time ) + 5*rannor( 123 );
   output;
   end;
run;

proc model data=exp;
   parm zo 35 b;
      dert.z = b * z;
      y=z;
   fit y init=(z=zo) / prl=both;
   test zo = 40.475437 ,/ lr;
run;
```

The output from the requested confidence intervals and the TEST statement are shown in Figure 26.58

Figure 26.58 Confidence Interval Estimation

The MODEL Procedure

Nonlinear OLS Parameter Estimates

Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|
zo	36.58933	1.9471	18.79	<.0001
b	0.006497	0.00464	1.40	0.1780

Test Results

Test	Type	Statistic	Pr > ChiSq	Label
Test0	L.R.	3.81	0.0509	zo = 40.475437

Figure 26.58 *continued*

Parameter Wald 95% Confidence Intervals			
Parameter	**Value**	**Lower**	**Upper**
zo	36.5893	32.7730	40.4056
b	0.00650	-0.00259	0.0156

Parameter Likelihood Ratio 95% Confidence Intervals			
Parameter	**Value**	**Lower**	**Upper**
zo	36.5893	32.8381	40.4921
b	0.00650	-0.00264	0.0157

In this example the parameter value used in the likelihood ratio test, $zo = 40.475437$, is close to the upper bound computed for the likelihood ratio confidence interval, $zo \leq 40.4921$. This coincidence is not germane to the analysis however, since the likelihood ratio test is a test of the null hypothesis $H_0 : zo = 40.475437$ and the confidence interval can be viewed as a test of the null hypothesis $H_0 : 32.8381 \leq zo \leq 40.4921$.

Choice of Instruments

Several of the estimation methods supported by PROC MODEL are instrumental variables methods. There is no standard method for choosing instruments for nonlinear regression. Few econometric textbooks discuss the selection of instruments for nonlinear models. See Bowden and Turkington (1984, pp. 180–182) for more information.

The purpose of the instrumental projection is to purge the regressors of their correlation with the residual. For nonlinear systems, the regressors are the partials of the residuals with respect to the parameters.

Possible instrumental variables include the following:

- any variable in the model that is independent of the errors
- lags of variables in the system
- derivatives with respect to the parameters, if the derivatives are independent of the errors
- low-degree polynomials in the exogenous variables
- variables from the data set or functions of variables from the data set

Selected instruments must not have any of the following characteristics:

- depend on any variable endogenous with respect to the equations estimated
- depend on any of the parameters estimated
- be lags of endogenous variables if there is serial correlation of the errors

If the preceding rules are satisfied and there are enough observations to support the number of instruments used, the results should be consistent and the efficiency loss held to a minimum.

You need at least as many instruments as the maximum number of parameters in any equation, or some of the parameters cannot be estimated. Note that *number of instruments* means linearly independent instruments. If you add an instrument that is a linear combination of other instruments, it has no effect and does not increase the effective number of instruments.

You can, however, use too many instruments. In order to get the benefit of instrumental variables, you must have more observations than instruments. Thus, there is a trade-off; the instrumental variables technique completely eliminates the simultaneous equation bias only in large samples. In finite samples, the larger the excess of observations over instruments, the more the bias is reduced. Adding more instruments might improve the efficiency, but after some point efficiency declines as the excess of observations over instruments becomes smaller and the bias grows.

The instruments used in an estimation are printed out at the beginning of the estimation. For example, the following statements produce the instruments list shown in Figure 26.59.

```
proc model data=test2;
   exogenous x1 x2;
   parms b1 a1 a2 b2 2.5 c2 55;
   y1 = a1 * y2 + b1 * exp(x1);
   y2 = a2 * y1 + b2 * x2 * x2 + c2 / x2;
   fit y1 y2 / n2sls;
   inst b1 b2 c2 x1 ;
run;
```

Figure 26.59 Instruments Used Message

The MODEL Procedure

The 2 Equations to Estimate	
y1 =	F(b1, a1(y2))
y2 =	F(a2(y1), b2, c2)
Instruments	1 x1 @y1/@b1 @y2/@b2 @y2/@c2

This states that an intercept term, the exogenous variable X1, and the partial derivatives of the equations with respect to B1, B2, and C2, were used as instruments for the estimation.

Examples

Suppose that Y1 and Y2 are endogenous variables, that X1 and X2 are exogenous variables, and that A, B, C, D, E, F, and G are parameters. Consider the following model:

```
y1 = a + b * x1 + c * y2 + d * lag(y1);
y2 = e + f * x2 + g * y1;
fit y1 y2;
instruments exclude=(c g);
```

The INSTRUMENTS statement produces X1, X2, LAG(Y1), and an intercept as instruments.

In order to estimate the Y1 equation by itself, it is necessary to include X2 explicitly in the instruments since F, in this case, is not included in the following estimation:

```
y1 = a + b * x1 + c * y2 + d * lag(y1);
y2 = e + f * x2 + g * y1;
fit y1;
instruments x2 exclude=(c);
```

This produces the same instruments as before. You can list the parameter associated with the lagged variable as an instrument instead of using the EXCLUDE= option. Thus, the following is equivalent to the previous example:

```
y1 = a + b * x1 + c * y2 + d * lag(y1);
y2 = e + f * x2 + g * y1;
fit y1;
instruments x1 x2 d;
```

For an example of declaring instruments when estimating a model involving identities, consider Klein's Model I:

```
proc model data=klien;
   endogenous c p w i x wsum k y;
   exogenous  wp g t year;
   parms c0-c3 i0-i3 w0-w3;
   a: c = c0 + c1 * p + c2 * lag(p) + c3 * wsum;
   b: i = i0 + i1 * p + i2 * lag(p) + i3 * lag(k);
   c: w = w0 + w1 * x + w2 * lag(x) + w3 * year;
   x = c + i + g;
   y = c + i + g-t;
   p = x-w-t;
   k = lag(k) + i;
   wsum = w + wp;
run;
```

The three equations to estimate are identified by the labels A, B, and C. The parameters associated with the predetermined terms are C2, I2, I3, W2, and W3 (and the intercepts, which are automatically added to the instruments). In addition, the system includes five identities that contain the predetermined variables G, T, LAG(K), and WP. Thus, the INSTRUMENTS statement can be written as

```
lagk = lag(k);
instruments c2 i2 i3 w2 w3 g t wp lagk;
```

where LAGK is a program variable used to hold LAG(K). However, this is more complicated than it needs to be. Except for LAG(K), all the predetermined terms in the identities are exogenous variables, and LAG(K) is already included as the coefficient of I3. There are also more parameters for predetermined terms than for endogenous terms, so you might prefer to use the EXCLUDE= option. Thus, you can specify the same instruments list with the simpler statement

```
instruments _exog_ exclude=(c1 c3 i1 w1);
```

To illustrate the use of polynomial terms as instrumental variables, consider the following model:

```
y1 = a + b * exp( c * x1 ) + d * log( x2 ) + e * exp( f * y2 );
```

The parameters are A, B, C, D, E, and F, and the right-hand-side variables are X1, X2, and Y2. Assume that X1 and X2 are exogenous (independent of the error), while Y2 is endogenous. The equation for Y2 is not specified, but assume that it includes the variables X1, X3, and Y1, with X3 exogenous, so the exogenous variables of the full system are X1, X2, and X3. Using as instruments quadratic terms in the exogenous variables, the model is specified to PROC MODEL as follows:

```
proc model;
   parms a b c d e f;
   y1 = a + b * exp( c * x1 ) + d * log( x2 ) + e * exp( f * y2 );
   instruments inst1-inst9;
   inst1 = x1; inst2 = x2; inst3 = x3;
   inst4 = x1 * x1; inst5 = x1 * x2; inst6 = x1 * x3;
   inst7 = x2 * x2; inst8 = x2 * x3; inst9 = x3 * x3;
   fit y1 / 2sls;
run;
```

It is not clear what degree polynomial should be used. There is no way to know how good the approximation is for any degree chosen, although the first-stage R^2s might help the assessment.

First-Stage R-Squares

When the FSRSQ option is used on the FIT statement, the MODEL procedure prints a column of first-stage R^2 (FSRSQ) statistics along with the parameter estimates. The FSRSQ measures the fraction of the variation of the derivative column associated with the parameter that remains after projection through the instruments.

Ideally, the FSRSQ should be very close to 1.00 for exogenous derivatives. If the FSRSQ is small for an endogenous derivative, it is unclear whether this reflects a poor choice of instruments or a large influence of the errors in the endogenous right-hand-side variables. When the FSRSQ for one or more parameters is small, the standard errors of the parameter estimates are likely to be large.

Note that you can make all the FSRSQs larger (or 1.00) by including more instruments, because of the disadvantage discussed previously. The FSRSQ statistics reported are unadjusted R^2s and do not include a degrees-of-freedom correction.

Autoregressive Moving-Average Error Processes

Autoregressive moving-average error processes (ARMA errors) and other models that involve lags of error terms can be estimated by using FIT statements and simulated or forecast by using SOLVE statements. ARMA models for the error process are often used for models with autocorrelated residuals. The %AR macro can be used to specify models with autoregressive error processes. The %MA macro can be used to specify models with moving-average error processes.

Autoregressive Errors

A model with first-order autoregressive errors, AR(1), has the form

$$y_t = f(x_t, \theta) + \mu_t$$

$$\mu_t = \phi \mu_{t-1} + \epsilon_t$$

while an AR(2) error process has the form

$$\mu_t = \phi_1 \mu_{t-1} + \phi_2 \mu_{t-2} + \epsilon_t$$

and so forth for higher-order processes. Note that the ϵ_t's are independent and identically distributed and have an expected value of 0.

An example of a model with an AR(2) component is

$$y = \alpha + \beta x_1 + \mu_t$$

$$\mu_t = \phi_1 \mu_{t-1} + \phi_2 \mu_{t-2} + \epsilon_t$$

You would write this model as

```
proc model data=in;
   parms a b p1 p2;
   y = a + b * x1 + p1 * zlag1(y - (a + b * x1)) +
         p2 * zlag2(y - (a + b * x1));
   fit y;
run;
```

or equivalently using the %AR macro as

```
proc model data=in;
   parms a b;
   y = a + b * x1;
   %ar( y, 2 );
   fit y;
run;
```

Moving-Average Models

A model with first-order moving-average errors, MA(1), has the form

$$y_t = f(x_t) + \mu_t$$

$$\mu_t = \epsilon_t - \theta_1 \epsilon_{t-1}$$

where ϵ_t is identically and independently distributed with mean zero. An MA(2) error process has the form

$$\mu_t = \epsilon_t - \theta_1 \epsilon_{t-1} - \theta_2 \epsilon_{t-2}$$

and so forth for higher-order processes.

For example, you can write a simple linear regression model with MA(2) moving-average errors as

```
proc model data=inma2;
   parms a b ma1 ma2;
   y = a + b * x + ma1 * zlag1( resid.y ) +
       ma2 * zlag2( resid.y );
   fit;
run;
```

where MA1 and MA2 are the moving-average parameters.

Note that RESID.Y is automatically defined by PROC MODEL as

```
pred.y = a + b * x + ma1 * zlag1( resid.y ) +
       ma2 * zlag2( resid.y );
resid.y = pred.y - actual.y;
```

Note that RESID.Y is negative of ϵ_t.

The ZLAG function must be used for MA models to truncate the recursion of the lags. This ensures that the lagged errors start at zero in the lag-priming phase and do not propagate missing values when lag-priming period variables are missing, and it ensures that the future errors are zero rather than missing during simulation or forecasting. For details about the lag functions, see the section "Lag Logic" on page 1682.

This model written using the %MA macro is as follows:

```
proc model data=inma2;
   parms a b;
   y =  a + b * x;
   %ma(y, 2);
   fit;
run;
```

General Form for ARMA Models

The general ARMA(p,q) process has the following form

$$\mu_t = \phi_1 \mu_{t-1} + \ldots + \phi_p \mu_{t-p} + \epsilon_t - \theta_1 \epsilon_{t-1} - \ldots - \theta_q \epsilon_{t-q}$$

An ARMA(p,q) model can be specified as follows:

```
yhat =  ... compute structural predicted value here ... ;
yarma = ar1 * zlag1( y - yhat ) + ...  /* ar part */
                                + ar(p) * zlag(p)( y - yhat )
     + ma1 * zlag1( resid.y )   + ...  /* ma part */
                                + ma(q) * zlag(q)( resid.y );
y = yhat + yarma;
```

where ARi and MAj represent the autoregressive and moving-average parameters for the various lags. You can use any names you want for these variables, and there are many equivalent ways that the specification could be written.

Vector ARMA processes can also be estimated with PROC MODEL. For example, a two-variable AR(1) process for the errors of the two endogenous variables Y1 and Y2 can be specified as follows:

```
y1hat =  ... compute structural predicted value here ... ;

y1    = y1hat + ar1_1 * zlag1( y1 - y1hat )   /* ar part y1,y1 */
              + ar1_2 * zlag1( y2 - y2hat );  /* ar part y1,y2 */

y21hat =  ... compute structural predicted value here ... ;

y2     = y2hat + ar2_2 * zlag1( y2 - y2hat )   /* ar part y2,y2 */
               + ar2_1 * zlag1( y1 - y1hat );  /* ar part y2,y1 */
```

Convergence Problems with ARMA Models

ARMA models can be difficult to estimate. If the parameter estimates are not within the appropriate range, a moving-average model's residual terms grow exponentially. The calculated residuals for later observations can be very large or can overflow. This can happen either because improper starting values were used or because the iterations moved away from reasonable values.

Care should be used in choosing starting values for ARMA parameters. Starting values of 0.001 for ARMA parameters usually work if the model fits the data well and the problem is well-conditioned. Note that an MA model can often be approximated by a high-order AR model, and vice versa. This can result in high collinearity in mixed ARMA models, which in turn can cause serious ill-conditioning in the calculations and instability of the parameter estimates.

If you have convergence problems while estimating a model with ARMA error processes, try to estimate in steps. First, use a FIT statement to estimate only the structural parameters with the ARMA parameters held to zero (or to reasonable prior estimates if available). Next, use another FIT statement to estimate the ARMA parameters only, using the structural parameter values from the first run. Since the values of the structural parameters are likely to be close to their final estimates, the ARMA parameter estimates might now converge. Finally, use another FIT statement to produce simultaneous estimates of all the parameters. Since the initial values of the parameters are now likely to be quite close to their final joint estimates, the estimates should converge quickly if the model is appropriate for the data.

AR Initial Conditions

The initial lags of the error terms of AR(p) models can be modeled in different ways. The autoregressive error startup methods supported by SAS/ETS procedures are the following:

CLS	conditional least squares (ARIMA and MODEL procedures)
ULS	unconditional least squares (AUTOREG, ARIMA, and MODEL procedures)
ML	maximum likelihood (AUTOREG, ARIMA, and MODEL procedures)
YW	Yule-Walker (AUTOREG procedure only)
HL	Hildreth-Lu, which deletes the first p observations (MODEL procedure only)

See Chapter 9, "The AUTOREG Procedure," for an explanation and discussion of the merits of various AR(p) startup methods.

The CLS, ULS, ML, and HL initializations can be performed by PROC MODEL. For AR(1) errors, these initializations can be produced as shown in Table 26.3. These methods are equivalent in large samples.

Table 26.3 Initializations Performed by PROC MODEL: AR(1) ERRORS

Method	Formula
conditional least squares	Y=YHAT+AR1*ZLAG1(Y-YHAT);
unconditional least squares	Y=YHAT+AR1*ZLAG1(Y-YHAT); IF _OBS_=1 THEN RESID.Y=SQRT(1-AR1**2)*RESID.Y;
maximum likelihood	Y=YHAT+AR1*ZLAG1(Y-YHAT); W=(1-AR1**2)**(-1/(2*_NUSED_)); IF _OBS_=1 THEN W=W*SQRT(1-AR1**2); RESID.Y=W*RESID.Y;
Hildreth-Lu	Y=YHAT+AR1*LAG1(Y-YHAT);

MA Initial Conditions

The initial lags of the error terms of MA(q) models can also be modeled in different ways. The following moving-average error start-up paradigms are supported by the ARIMA and MODEL procedures:

ULS	unconditional least squares
CLS	conditional least squares
ML	maximum likelihood

The conditional least squares method of estimating moving-average error terms is not optimal because it ignores the start-up problem. This reduces the efficiency of the estimates, although they remain unbiased. The initial lagged residuals, extending before the start of the data, are assumed to be 0, their unconditional expected value. This introduces a difference between these residuals and the generalized least squares residuals for the moving-average covariance, which, unlike the autoregressive model, persists through the data set. Usually this difference converges quickly to 0, but for nearly noninvertible moving-average processes the convergence is quite slow. To minimize this problem, you should have plenty of data, and the moving-average parameter estimates should be well within the invertible range.

This problem can be corrected at the expense of writing a more complex program. Unconditional least squares estimates for the MA(1) process can be produced by specifying the model as follows:

```
yhat =  ... compute structural predicted value here ... ;
if _obs_ = 1 then do;
   h = sqrt( 1 + ma1 ** 2 );
   y = yhat;
   resid.y = ( y - yhat ) / h;
   end;
else do;
   g = ma1 / zlag1( h );
   h = sqrt( 1 + ma1 ** 2 - g ** 2 );
   y = yhat + g * zlag1( resid.y );
   resid.y = ( ( y - yhat) - g * zlag1( resid.y ) ) / h;
   end;
```

Moving-average errors can be difficult to estimate. You should consider using an AR(p) approximation to the moving-average process. A moving-average process can usually be well-approximated by an autoregressive process if the data have not been smoothed or differenced.

The %AR Macro

The SAS macro %AR generates programming statements for PROC MODEL for autoregressive models. The %AR macro is part of SAS/ETS software, and no special options need to be set to use the macro. The autoregressive process can be applied to the structural equation errors or to the endogenous series themselves.

The %AR macro can be used for the following types of autoregression:

- univariate autoregression
- unrestricted vector autoregression
- restricted vector autoregression

Univariate Autoregression

To model the error term of an equation as an autoregressive process, use the following statement after the equation:

```
%ar( varname, nlags )
```

For example, suppose that Y is a linear function of X1, X2, and an AR(2) error. You would write this model as follows:

```
proc model data=in;
   parms a b c;
   y = a + b * x1 + c * x2;
   %ar( y, 2 )
   fit y / list;
run;
```

The calls to %AR must come *after* all of the equations that the process applies to.

The preceding macro invocation, %AR(y,2), produces the statements shown in the LIST output in Figure 26.60.

Figure 26.60 LIST Option Output for an AR(2) Model

The MODEL Procedure

Listing of Compiled Program Code

Stmt	Line:Col	Statement as Parsed
1	2524:4	PRED.y = a + b * x1 + c * x2;
1	2524:4	RESID.y = PRED.y - ACTUAL.y;
1	2524:4	ERROR.y = PRED.y - y;
2	2525:14	_PRED__y = PRED.y;
3	2525:15	_OLD_PRED.y = PRED.y + y_l1 * ZLAG1(y - _PRED__y) + y_l2 * ZLAG2(y - _PRED__y);
3	2525:15	PRED.y = _OLD_PRED.y;
3	2525:15	RESID.y = PRED.y - ACTUAL.y;
3	2525:15	ERROR.y = PRED.y - y;

The _PRED__ prefixed variables are temporary program variables used so that the lags of the residuals are the correct residuals and not the ones redefined by this equation. Note that this is equivalent to the statements explicitly written in the section "General Form for ARMA Models" on page 1612.

You can also restrict the autoregressive parameters to zero at selected lags. For example, if you wanted autoregressive parameters at lags 1, 12, and 13, you can use the following statements:

```
proc model data=in;
   parms a b c;
   y = a + b * x1 + c * x2;
   %ar( y, 13, , 1 12 13 )
   fit y / list;
run;
```

These statements generate the output shown in Figure 26.61.

Figure 26.61 LIST Option Output for an AR Model with Lags at 1, 12, and 13

The MODEL Procedure

Listing of Compiled Program Code

Stmt	Line:Col	Statement as Parsed
1	2533:4	PRED.y = a + b * x1 + c * x2;
1	2533:4	RESID.y = PRED.y - ACTUAL.y;
1	2533:4	ERROR.y = PRED.y - y;
2	2534:14	_PRED__y = PRED.y;
3	2534:15	_OLD_PRED.y = PRED.y + y_l1 * ZLAG1(y - _PRED__y) + y_l12 * ZLAG12(y - _PRED__y) + y_l13 * ZLAG13(y - _PRED__y);
3	2534:15	PRED.y = _OLD_PRED.y;
3	2534:15	RESID.y = PRED.y - ACTUAL.y;
3	2534:15	ERROR.y = PRED.y - y;

There are variations on the conditional least squares method, depending on whether observations at the start of the series are used to "warm up" the AR process. By default, the %AR conditional least squares method uses all the observations and assumes zeros for the initial lags of autoregressive terms. By using the M= option, you can request that %AR use the unconditional least squares (ULS) or maximum-likelihood (ML) method instead. For example,

```
proc model data=in;
   y = a + b * x1 + c * x2;
   %ar( y, 2, m=uls )
   fit y;
run;
```

Discussions of these methods is provided in the section "AR Initial Conditions" on page 1613.

By using the M=CLS*n* option, you can request that the first *n* observations be used to compute estimates of the initial autoregressive lags. In this case, the analysis starts with observation *n* +1. For example:

```
proc model data=in;
   y = a + b * x1 + c * x2;
   %ar( y, 2, m=cls2 )
   fit y;
run;
```

You can use the %AR macro to apply an autoregressive model to the endogenous variable, instead of to the error term, by using the TYPE=V option. For example, if you want to add the five past lags of Y to the equation in the previous example, you could use %AR to generate the parameters and lags by using the following statements:

```
proc model data=in;
   parms a b c;
   y = a + b * x1 + c * x2;
   %ar( y, 5, type=v )
   fit y / list;
run;
```

The preceding statements generate the output shown in Figure 26.62.

Figure 26.62 LIST Option Output for an AR model of Y

The MODEL Procedure

Listing of Compiled Program Code

Stmt	Line:Col	Statement as Parsed
1	2556:4	PRED.y = a + b * x1 + c * x2;
1	2556:4	RESID.y = PRED.y - ACTUAL.y;
1	2556:4	ERROR.y = PRED.y - y;
2	2557:15	_OLD_PRED.y = PRED.y + y_l1 * ZLAG1(y) + y_l2 * ZLAG2(y) + y_l3 * ZLAG3(y) + y_l4 * ZLAG4(y) + y_l5 * ZLAG5(y);
2	2557:15	PRED.y = _OLD_PRED.y;
2	2557:15	RESID.y = PRED.y - ACTUAL.y;
2	2557:15	ERROR.y = PRED.y - y;

This model predicts Y as a linear combination of X1, X2, an intercept, and the values of Y in the most recent five periods.

Unrestricted Vector Autoregression

To model the error terms of a set of equations as a vector autoregressive process, use the following form of the %AR macro after the equations:

```
%ar( process_name, nlags, variable_list )
```

The *process_name* value is any name that you supply for %AR to use in making names for the autoregressive parameters. You can use the %AR macro to model several different AR processes for different sets of equations by using different process names for each set. The process name ensures that the variable names used are unique. Use a short *process_name* value for the process if parameter estimates are to be written to an output data set. The %AR macro tries to construct parameter names less than or equal to eight characters, but this is limited by the length of *process_name*, which is used as a prefix for the AR parameter names.

The *variable_list* value is the list of endogenous variables for the equations.

For example, suppose that errors for equations Y1, Y2, and Y3 are generated by a second-order vector autoregressive process. You can use the following statements:

```
proc model data=in;
   y1 = ... equation for y1 ...;
   y2 = ... equation for y2 ...;
   y3 = ... equation for y3 ...;
   %ar( name, 2, y1 y2 y3 )
   fit y1 y2 y3;
run;
```

which generate the following for Y1 and similar code for Y2 and Y3:

```
y1 = pred.y1 + name1_1_1*zlag1(y1-name_y1) +
     name1_1_2*zlag1(y2-name_y2) +
     name1_1_3*zlag1(y3-name_y3) +
     name2_1_1*zlag2(y1-name_y1) +
     name2_1_2*zlag2(y2-name_y2) +
     name2_1_3*zlag2(y3-name_y3) ;
```

Only the conditional least squares (M=CLS or M=CLS*n*) method can be used for vector processes.

You can also use the same form with restrictions that the coefficient matrix be 0 at selected lags. For example, the following statements apply a third-order vector process to the equation errors with all the coefficients at lag 2 restricted to 0 and with the coefficients at lags 1 and 3 unrestricted:

```
proc model data=in;
   y1 = ... equation for y1 ...;
   y2 = ... equation for y2 ...;
   y3 = ... equation for y3 ...;
   %ar( name, 3, y1 y2 y3, 1 3 )
   fit y1 y2 y3;
```

You can model the three series Y1–Y3 as a vector autoregressive process in the variables instead of in the errors by using the TYPE=V option. If you want to model Y1–Y3 as a function of past values of Y1–Y3 and some exogenous variables or constants, you can use %AR to generate the statements for the lag terms. Write an equation for each variable for the nonautoregressive part of the model, and then call %AR with the TYPE=V option. For example,

```
proc model data=in;
   parms a1-a3 b1-b3;
   y1 = a1 + b1 * x;
   y2 = a2 + b2 * x;
   y3 = a3 + b3 * x;
   %ar( name, 2, y1 y2 y3, type=v )
   fit y1 y2 y3;
run;
```

The nonautoregressive part of the model can be a function of exogenous variables, or it can be intercept parameters. If there are no exogenous components to the vector autoregression model, including no intercepts, then assign zero to each of the variables. There must be an assignment to each of the variables before %AR is called.

```
proc model data=in;
   y1=0;
   y2=0;
   y3=0;
   %ar( name, 2, y1 y2 y3, type=v )
   fit y1 y2 y3;
run;
```

This example models the vector Y=(Y1 Y2 Y3)′ as a linear function only of its value in the previous two periods and a white noise error vector. The model has 18=(3 × 3 + 3 × 3) parameters.

Syntax of the %AR Macro

There are two cases of the syntax of the %AR macro. When restrictions on a vector AR process are not needed, the syntax of the %AR macro has the general form

%AR *(* name *,* nlag *<,*endolist *<,* laglist *>> <,M=* method *> <,TYPE= V>)* **;**

where

name specifies a prefix for %AR to use in constructing names of variables needed to define the AR process. If the *endolist* is not specified, the endogenous list defaults to *name*, which must be the name of the equation to which the AR error process is to be applied. The *name* value cannot exceed 32 characters.

nlag is the order of the AR process.

endolist specifies the list of equations to which the AR process is to be applied. If more than one name is given, an unrestricted vector process is created with the structural residuals of all the equations included as regressors in each of the equations. If not specified, *endolist* defaults to *name*.

laglist specifies the list of lags at which the AR terms are to be added. The coefficients of the terms at lags not listed are set to 0. All of the listed lags must be less than or equal to *nlag*,

and there must be no duplicates. If not specified, the *laglist* defaults to all lags 1 through *nlag*.

M=*method* specifies the estimation method to implement. Valid values of M= are CLS (conditional least squares estimates), ULS (unconditional least squares estimates), and ML (maximum likelihood estimates). M=CLS is the default. Only M=CLS is allowed when more than one equation is specified. The ULS and ML methods are not supported for vector AR models by %AR.

TYPE=V specifies that the AR process is to be applied to the endogenous variables themselves instead of to the structural residuals of the equations.

Restricted Vector Autoregression

You can control which parameters are included in the process, restricting to 0 those parameters that you do not include. First, use %AR with the DEFER option to declare the variable list and define the dimension of the process. Then, use additional %AR calls to generate terms for selected equations with selected variables at selected lags. For example,

```
proc model data=d;
   y1 = ... equation for y1 ...;
   y2 = ... equation for y2 ...;
   y3 = ... equation for y3 ...;
   %ar( name, 2, y1 y2 y3, defer )
   %ar( name, y1, y1 y2 )
   %ar( name, y2 y3, , 1 )
   fit y1 y2 y3;
run;
```

The error equations produced are as follows:

```
y1 = pred.y1 + name1_1_1*zlag1(y1-name_y1) +
     name1_1_2*zlag1(y2-name_y2) + name2_1_1*zlag2(y1-name_y1) +
     name2_1_2*zlag2(y2-name_y2) ;
y2 = pred.y2 + name1_2_1*zlag1(y1-name_y1) +
     name1_2_2*zlag1(y2-name_y2) + name1_2_3*zlag1(y3-name_y3) ;
y3 = pred.y3 + name1_3_1*zlag1(y1-name_y1) +
     name1_3_2*zlag1(y2-name_y2) + name1_3_3*zlag1(y3-name_y3) ;
```

This model states that the errors for Y1 depend on the errors of both Y1 and Y2 (but not Y3) at both lags 1 and 2, and that the errors for Y2 and Y3 depend on the previous errors for all three variables, but only at lag 1.

%AR Macro Syntax for Restricted Vector AR

An alternative use of %AR is allowed to impose restrictions on a vector AR process by calling %AR several times to specify different AR terms and lags for different equations.

The first call has the general form

%AR(name, nlag, endolist *, DEFER)* **;**

where

name specifies a prefix for %AR to use in constructing names of variables needed to define the vector AR process.

nlag specifies the order of the AR process.

endolist specifies the list of equations to which the AR process is to be applied.

DEFER specifies that %AR is not to generate the AR process but is to wait for further information specified in later %AR calls for the same *name* value.

The subsequent calls have the general form

```
%AR( name, eqlist, varlist, laglist,TYPE= )
```

where

name is the same as in the first call.

eqlist specifies the list of equations to which the specifications in this %AR call are to be applied. Only names specified in the *endolist* value of the first call for the *name* value can appear in the list of equations in *eqlist*.

varlist specifies the list of equations whose lagged structural residuals are to be included as regressors in the equations in *eqlist*. Only names in the *endolist* of the first call for the *name* value can appear in *varlist*. If not specified, *varlist* defaults to *endolist*.

laglist specifies the list of lags at which the AR terms are to be added. The coefficients of the terms at lags not listed are set to 0. All of the listed lags must be less than or equal to the value of *nlag*, and there must be no duplicates. If not specified, *laglist* defaults to all lags 1 through *nlag*.

The %MA Macro

The SAS macro %MA generates programming statements for PROC MODEL for moving-average models. The %MA macro is part of SAS/ETS software, and no special options are needed to use the macro. The moving-average error process can be applied to the structural equation errors. The syntax of the %MA macro is the same as the %AR macro except there is no TYPE= argument.

When you are using the %MA and %AR macros combined, the %MA macro must follow the %AR macro. The following SAS/IML statements produce an ARMA(1, (1 3)) error process and save it in the data set MADAT2.

```
proc iml;
   phi = { 1 .2 };
   theta = { 1 .3 0 .5 };
   y = armasim( phi, theta, 0, .1, 200, 32565 );
   create madat2 from y[colname='y'];
   append from y;
quit;
```

The following PROC MODEL statements are used to estimate the parameters of this model by using maximum likelihood error structure:

```
title 'Maximum Likelihood ARMA(1, (1 3))';
proc model data=madat2;
   y=0;
   %ar( y, 1, ,  M=ml )
   %ma( y, 3, , 1 3,  M=ml )  /* %MA always after %AR */
   fit y;
run;
title;
```

The estimates of the parameters produced by this run are shown in Figure 26.63.

Figure 26.63 Estimates from an ARMA(1, (1 3)) Process

Maximum Likelihood ARMA(1, (1 3))

The MODEL Procedure

Nonlinear OLS Summary of Residual Errors

Equation	DF Model	DF Error	SSE	MSE	Root MSE	R-Square	Adj R-Sq
y	3	197	2.6383	0.0134	0.1157	-0.0067	-0.0169
RESID.y		197	1.9957	0.0101	0.1007		

Nonlinear OLS Parameter Estimates

Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|	Label
y_l1	-0.10067	0.1187	-0.85	0.3973	AR(y) y lag1 parameter
y_m1	-0.1934	0.0939	-2.06	0.0408	MA(y) y lag1 parameter
y_m3	-0.59384	0.0601	-9.88	<.0001	MA(y) y lag3 parameter

Syntax of the %MA Macro

There are two cases of the syntax for the %MA macro. When restrictions on a vector MA process are not needed, the syntax of the %MA macro has the general form

%MA *(* name *,* nlag < *,* endolist < *,* laglist >> < *,M=* method > *)* **;**

where

name specifies a prefix for %MA to use in constructing names of variables needed to define the MA process and is the default *endolist*.

nlag is the order of the MA process.

endolist specifies the equations to which the MA process is to be applied. If more than one name is given, CLS estimation is used for the vector process.

laglist specifies the lags at which the MA terms are to be added. All of the listed lags must be less than or equal to *nlag*, and there must be no duplicates. If not specified, the *laglist* defaults to all lags 1 through *nlag*.

M=*method* specifies the estimation method to implement. Valid values of M= are CLS (conditional least squares estimates), ULS (unconditional least squares estimates), and ML (maximum

likelihood estimates). M=CLS is the default. Only M=CLS is allowed when more than one equation is specified in the *endolist*.

%MA Macro Syntax for Restricted Vector Moving-Average

An alternative use of %MA is allowed to impose restrictions on a vector MA process by calling %MA several times to specify different MA terms and lags for different equations.

The first call has the general form

%MA(name , nlag , endolist , *DEFER)* **;**

where

name specifies a prefix for %MA to use in constructing names of variables needed to define the vector MA process.

nlag specifies the order of the MA process.

endolist specifies the list of equations to which the MA process is to be applied.

DEFER specifies that %MA is not to generate the MA process but is to wait for further information specified in later %MA calls for the same *name* value.

The subsequent calls have the general form

```
%MA( name, eqlist, varlist, laglist )
```

where

name is the same as in the first call.

eqlist specifies the list of equations to which the specifications in this %MA call are to be applied.

varlist specifies the list of equations whose lagged structural residuals are to be included as regressors in the equations in *eqlist*.

laglist specifies the list of lags at which the MA terms are to be added.

The %EQAR and%EQMA Marcos for General Form Equations

The %AR and %MA macros are not supported when the structural portion of a model is specified using general form equations. When you want to include AR or MA error terms in a model that is specified using general form equations, use the %EQAR or %EQMA macros.

The following code specifies an AR(2) error process for a normal form equation:

```
proc model data=in;
   parms a b;
   y = a + b * x1;
   %ar( y, 2 );
   fit y;
run;
```

You can use the %EQAR macro as follows to express the same model for a general form equation:

```
proc model data=in;
   parms a b;
   eq.y = y - (a + b * x1);
   %eqar( y, 2, eq.y );
   fit y;
run;
```

Like the %AR and %MA macros, the %EQAR and %EQMA macros support the following types of AR and MA processes:

- univariate processes
- unrestricted vector processes
- restricted vector processes

The %EQAR and %EQMA macros also support the following initial conditions

CLS	conditional least squares
ULS	unconditional least squares
ML	maximum likelihood

Differences between models that are expressed using normal form equations and general form equations lead to differences between the %EQAR and %EQMA macros and their %AR and %MA counterparts. The syntax for the %EQAR macro is

%EQAR *(name , nlag , eqlist < , laglist > <,***M=** *method >)* **;**

where

eqlist specifies the list of equations whose autoregressive errors use the EQ.*var* syntax. The *eqlist* parameter takes the place of the *endolist* parameter in the %AR macro. The *eqlist* parameter is required.

The *name*, *nlag*, *nlaglist*, and M=*method* parameters have the same meanings in the %EQAR macro as in the %AR macro. Unlike the %AR macro, the %EQAR macro does not include a TYPE=V parameter.

The syntax for the %EQMA macro is

%EQMA *(name , nlag , eqlist < , laglist > <,***M=***method >)* **;**

where

eqlist specifies the list of equations whose moving average errors use the EQ.*var* syntax. The *eqlist* parameter takes the place of the *endolist* parameter in the %MA macro. The *eqlist* parameter is required.

The *name*, *nlag*, *nlaglist*, and M=*method* parameters have the same meanings in the %EQMA macro as in the %MA macro.

Both the %EQAR and %EQMA macros support use of the DEFER syntax for specifying restricted autoregressive and moving average vector processes.

Distributed Lag Models and the %PDL Macro

In the following example, the variable y is modeled as a linear function of x, the first lag of x, the second lag of x, and so forth:

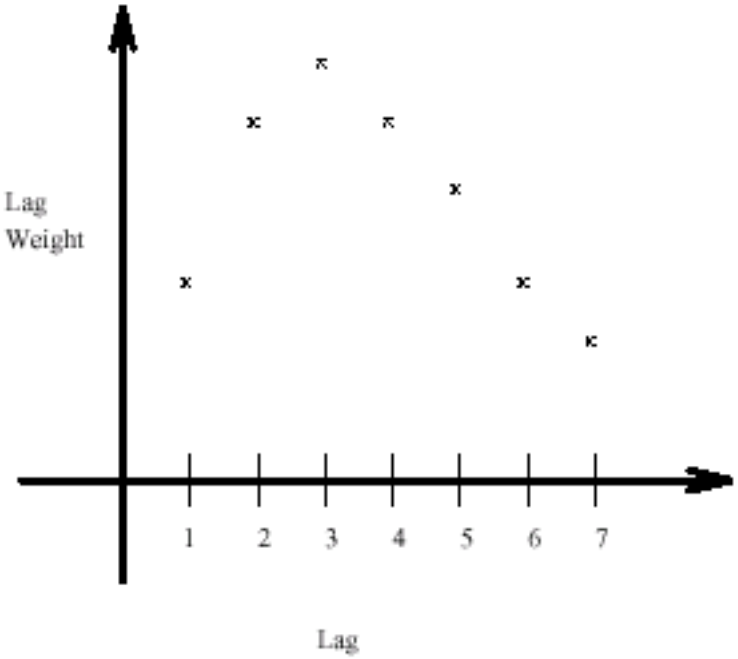
$$y_t = a + b_0 x_t + b_1 x_{t-1} + b_2 x_{t-2} + b_3 x_{t-3} + \ldots + b_n x_{t-n}$$

Models of this sort can introduce a great many parameters for the lags, and there may not be enough data to compute accurate independent estimates for them all. Often, the number of parameters is reduced by assuming that the lag coefficients follow some pattern. One common assumption is that the lag coefficients follow a polynomial in the lag length

$$b_i = \sum_{j=0}^{d} \alpha_j (i)^j$$

where d is the degree of the polynomial used. Models of this kind are called *Almon lag models*, *polynomial distributed lag models*, or *PDLs* for short. For example, Figure 26.64 shows the lag distribution that can be modeled with a low-order polynomial. Endpoint restrictions can be imposed on a PDL to require that the lag coefficients be 0 at the 0th lag, or at the final lag, or at both.

Figure 26.64 Polynomial Distributed Lags



For linear single-equation models, SAS/ETS software includes the PDLREG procedure for estimating PDL models. See Chapter 28, "The PDLREG Procedure," for a more detailed discussion of polynomial distributed lags and an explanation of endpoint restrictions.

Polynomial and other distributed lag models can be estimated and simulated or forecast with PROC MODEL. For polynomial distributed lags, the %PDL macro can generate the needed programming statements automatically.

The %PDL Macro

The SAS macro %PDL generates the programming statements to compute the lag coefficients of polynomial distributed lag models and to apply them to the lags of variables or expressions.

To use the %PDL macro in a model program, you first call it to declare the lag distribution; later, you call it again to apply the PDL to a variable or expression. The first call generates a PARMS statement for the polynomial parameters and assignment statements to compute the lag coefficients. The second call generates an expression that applies the lag coefficients to the lags of the specified variable or expression. A PDL can be declared only once, but it can be used any number of times (that is, the second call can be repeated).

The initial declaratory call has the general form

%PDL *(* pdlname, nlags, degree *, R=*code *, OUTEST=*dataset *)* **;**

where *pdlname* is a name (up to 32 characters) that you give to identify the PDL, *nlags* is the lag length, and *degree* is the degree of the polynomial for the distribution. The R=*code* is optional for endpoint restrictions. The value of *code* can be FIRST (for upper), LAST (for lower), or BOTH (for both upper and lower endpoints). See Chapter 28, "The PDLREG Procedure," for a discussion of endpoint restrictions. The option OUTEST=*dataset* creates a data set that contains the estimates of the parameters and their covariance matrix.

The later calls to apply the PDL have the general form

```
%PDL( pdlname, expression )
```

where *pdlname* is the name of the PDL and *expression* is the variable or expression to which the PDL is to be applied. The *pdlname* given must be the same as the name used to declare the PDL.

The following statements produce the output in Figure 26.65:

```
proc model data=in list;
   parms int pz;
   %pdl(xpdl,5,2);
   y = int + pz * z + %pdl(xpdl,x);
   %ar(y,2,M=ULS);
   id i;
fit y / out=model1 outresid converge=1e-6;
run;
```

Figure 26.65 %PDL Macro Estimates

The MODEL Procedure

Nonlinear OLS Estimates

Term	Estimate	Approx Std Err	t Value	Approx Pr > |t|	Label
XPDL_L0	1.568788	0.0935	16.77	<.0001	PDL(XPDL,5,2) coefficient for lag0
XPDL_L1	0.564917	0.0328	17.22	<.0001	PDL(XPDL,5,2) coefficient for lag1
XPDL_L2	-0.05063	0.0593	-0.85	0.4155	PDL(XPDL,5,2) coefficient for lag2
XPDL_L3	-0.27785	0.0517	-5.37	0.0004	PDL(XPDL,5,2) coefficient for lag3
XPDL_L4	-0.11675	0.0368	-3.17	0.0113	PDL(XPDL,5,2) coefficient for lag4
XPDL_L5	0.43267	0.1362	3.18	0.0113	PDL(XPDL,5,2) coefficient for lag5

This second example models two variables, Y1 and Y2, and uses two PDLs:

```
proc model data=in;
   parms int1 int2;
   %pdl( logxpdl, 5, 3 )
   %pdl( zpdl, 6, 4 )
   y1 = int1 + %pdl( logxpdl, log(x) ) + %pdl( zpdl, z );
   y2 = int2 + %pdl( zpdl, z );
   fit y1 y2;
run;
```

A (5,3) PDL of the log of X is used in the equation for Y1. A (6,4) PDL of Z is used in the equations for both Y1 and Y2. Since the same ZPDL is used in both equations, the lag coefficients for Z are the same for the Y1 and Y2 equations, and the polynomial parameters for ZPDL are shared by the two equations. See Example 26.5 for a complete example and comparison with PDLREG.

Input Data Sets

DATA= Input Data Set

For FIT tasks, the DATA= option specifies which input data set to use in estimating parameters. Variables in the model program are looked up in the DATA= data set and, if found, their attributes (type, length, label, and format) are set to be the same as those in the DATA= data set (if not defined otherwise within PROC MODEL).

ESTDATA= Input Data Set

The ESTDATA= option specifies an input data set that contains an observation that gives values for some or all of the model parameters. The data set can also contain observations that gives the rows of a covariance matrix for the parameters.

Parameter values read from the ESTDATA= data set provide initial starting values for parameters estimated. Observations that provide covariance values, if any are present in the ESTDATA= data set, are ignored.

The ESTDATA= data set is usually created by the OUTEST= option in a previous FIT statement. You can also create an ESTDATA= data set with a SAS DATA step program. The data set must contain a numeric variable for each parameter to be given a value or covariance column. The name of the variable in the

ESTDATA= data set must match the name of the parameter in the model. Parameters with names longer than 32 characters cannot be set from an ESTDATA= data set. The data set must also contain a character variable _NAME_ of length 32. _NAME_ has a blank value for the observation that gives values to the parameters. _NAME_ contains the name of a parameter for observations that define rows of the covariance matrix.

More than one set of parameter estimates and covariances can be stored in the ESTDATA= data set if the observations for the different estimates are identified by the variable _TYPE_. _TYPE_ must be a character variable of length 8. The TYPE= option is used to select for input the part of the ESTDATA= data set for which the _TYPE_ value matches the value of the TYPE= option.

In PROC MODEL, you have several options to specify starting values for the parameters to be estimated. When more than one option is specified, the options are implemented in the following order of precedence (from highest to lowest): the START= option, the PARMS statement initialization value, the ESTDATA= option, and the PARMSDATA= option. If no options are specified for the starting value, the default value of 0.0001 is used.

The following SAS statements generate the ESTDATA= data set shown in Figure 26.66. The second FIT statement uses the TYPE= option to select the estimates from the GMM estimation as starting values for the FIML estimation.

```
/* Generate test data */
data gmm2;
    do t=1 to 50;
       x1 = sqrt(t) ;
       x2 = rannor(10) * 10;
       y1 = -.002 * x2 * x2 - .05 / x2 - 0.001 * x1 * x1;
       y2 = 0.002* y1 + 2 * x2 * x2 + 50 / x2 + 5 * rannor(1);
       y1 = y1 + 5 * rannor(1);
       z1 = 1; z2 = x1 * x1; z3 = x2 * x2; z4 = 1.0/x2;
       output;
    end;
run;

proc model data=gmm2 ;
   exogenous x1 x2;
   parms a1 a2 b1 2.5 b2 c2 55 d1;
   inst b1 b2 c2 x1 x2;
   y1 = a1 * y2 + b1 * x1 * x1 + d1;
   y2 = a2 * y1 + b2 * x2 * x2 + c2 / x2 + d1;

   fit y1 y2 / 3sls gmm kernel=(qs,1,0.2) outest=gmmest;

   fit y1 y2 / fiml type=gmm estdata=gmmest;
run;

proc print data=gmmest;
run;
```

Figure 26.66 ESTDATA= Data Set

Obs	_NAME_	_TYPE_	_STATUS_	_NUSED_	a1	a2	b1	b2	c2	d1
1		3SLS	0 Converged	50	-.002229607	-1.25002	0.025827	1.99609	49.8119	-0.44533
2		GMM	0 Converged	50	-.001772196	-1.02345	0.014025	1.99726	49.8648	-0.87573

MISSING=PAIRWISE | DELETE

When missing values are encountered for any one of the equations in a system of equations, the default action is to drop that observation for all of the equations. The new MISSING=PAIRWISE option in the FIT statement provides a different method of handling missing values that avoids losing data for nonmissing equations for the observation. This is especially useful for SUR estimation on equations with unequal numbers of observations.

The option MISSING=PAIRWISE specifies that missing values are tracked on an equation-by-equation basis. The MISSING=DELETE option specifies that the entire observation is omitted from the analysis when any equation has a missing predicted or actual value for the equation. The default is MISSING=DELETE.

When you specify the MISSING=PAIRWISE option, the **S** matrix is computed as

$$\mathbf{S} = \mathbf{D}(\mathbf{R}'\mathbf{R})\mathbf{D}$$

where **D** is a diagonal matrix that depends on the VARDEF= option, the matrix **R** is $(\mathbf{r}_1, \ldots, \mathbf{r}_g)$, and $\mathbf{r}_i$ is the vector of residuals for the ith equation with r_{ij} replaced with zero when r_{ij} is missing.

For MISSING=PAIRWISE, the calculation of the diagonal element $d_{i,i}$ of **D** is based on n_i, the number of nonmissing observations for the ith equation, instead of on n. Similarly, for VARDEF=WGT or WDF, the calculation is based on the sum of the weights for the nonmissing observations for the ith equation instead of on the sum of the weights for all observations. See the description of the VARDEF= option for the definition of **D**.

The degrees-of-freedom correction for a shared parameter is computed by using the average number of observations used in its estimation.

The MISSING=PAIRWISE option is not valid for the GMM and FIML estimation methods.

For the instrumental variables estimation methods (2SLS, 3SLS), when an instrument is missing for an observation, that observation is dropped for all equations, regardless of the MISSING= option.

PARMSDATA= Input Data Set

The option PARMSDATA= reads values for all parameters whose names match the names of variables in the PARMSDATA= data set. Values for any or all of the parameters in the model can be reset by using the PARMSDATA= option. The PARMSDATA= option goes in the PROC MODEL statement, and the data set is read before any FIT or SOLVE statements are executed.

In PROC MODEL, you have several options to specify starting values for the parameters to be estimated. When more than one option is specified, the options are implemented in the following order of precedence (from highest to lowest): the START= option, the PARMS statement initialization value, the ESTDATA= option, and the PARMSDATA= option. If no options are specified for the starting value, the default value of 0.0001 is used.

Together, the OUTPARMS= and PARMSDATA= options enable you to change part of a model and recompile the new model program without the need to reestimate equations that were not changed.

Suppose you have a large model with parameters estimated and you now want to replace one equation, Y, with a new specification. Although the model program must be recompiled with the new equation, you don't need to reestimate all the equations, just the one that changed.

Using the OUTPARMS= and PARMSDATA= options, you could do the following:

```
proc model model=oldmod outparms=temp; run;
proc model outmodel=newmod parmsdata=temp data=in;
   ...  include new model definition with changed y eq. here ...
   fit y;
run;
```

The model file NEWMOD then contains the new model and its estimated parameters plus the old models with their original parameter values.

SDATA= Input Data Set

The SDATA= option allows a cross-equation covariance matrix to be input from a data set. The **S** matrix read from the SDATA= data set, specified in the FIT statement, is used to define the objective function for the OLS, N2SLS, SUR, and N3SLS estimation methods and is used as the initial **S** for the methods that iterate the **S** matrix.

Most often, the SDATA= data set has been created by the OUTS= or OUTSUSED= option in a previous FIT statement. The OUTS= and OUTSUSED= data sets from a FIT statement can be read back in by a FIT statement in the same PROC MODEL step.

You can create an input SDATA= data set by using the DATA step. PROC MODEL expects to find a character variable _NAME_ in the SDATA= data set as well as variables for the equations in the estimation or solution. For each observation with a _NAME_ value that matches the name of an equation, PROC MODEL fills the corresponding row of the **S** matrix with the values of the names of equations found in the data set. If a row or column is omitted from the data set, a 1 is placed on the diagonal for the row or column. Missing values are ignored, and since the **S** matrix is symmetric, you can include only a triangular part of the **S** matrix in the SDATA= data set with the omitted part indicated by missing values. If the SDATA= data set contains multiple observations with the same _NAME_, the last values supplied for the _NAME_ are used. The structure of the expected data set is further described in the section "OUTS= Data Set" on page 1634.

Use the TYPE= option in the PROC MODEL or FIT statement to specify the type of estimation method used to produce the **S** matrix you want to input.

The following SAS statements are used to generate an **S** matrix from a GMM and a 3SLS estimation and to store that estimate in the data set GMMS:

```
proc model data=gmm2 ;
   exogenous x1 x2;
   parms a1 a2 b1 2.5 b2 c2 55 d1;
   inst b1 b2 c2 x1 x2;
   y1 = a1 * y2 + b1 * x1 * x1 + d1;
   y2 = a2 * y1 + b2 * x2 * x2 + c2 / x2 + d1;

   fit y1 y2 / 3sls gmm kernel=(qs,1,0.2)
               outest=gmmest outs=gmms;
run;

proc print data=gmms;
run;
```

The data set GMMS is shown in Figure 26.67.

Figure 26.67 SDATA= Data Set

Obs	_NAME_	_TYPE_	_NUSED_	y1	y2
1	y1	3SLS	50	27.1032	38.1599
2	y2	3SLS	50	38.1599	74.6253
3	y1	GMM	50	27.6248	32.2811
4	y2	GMM	50	32.2811	58.8387

VDATA= Input data set

The VDATA= option enables a variance matrix for GMM estimation to be input from a data set. When the VDATA= option is used in the PROC MODEL or FIT statement, the matrix that is input is used to define the objective function and is used as the initial **V** for the methods that iterate the **V** matrix.

Normally the VDATA= matrix is created from the OUTV= option in a previous FIT statement. Alternately an input VDATA= data set can be created by using the DATA step. Each row and column of the **V** matrix is associated with an equation and an instrument. The position of each element in the **V** matrix can then be indicated by an equation name and an instrument name for the row of the element and an equation name and an instrument name for the column. Each observation in the VDATA= data set is an element in the **V** matrix. The row and column of the element are indicated by four variables (EQ_ROW, INST_ROW, EQ_COL, and INST_COL) that contain the equation name or instrument name. The variable name for an element is VALUE. Missing values are set to 0. Because the variance matrix is symmetric, only a triangular part of the matrix needs to be input.

The following SAS statements are used to generate a **V** matrix estimation from GMM and to store that estimate in the data set GMMV:

```
proc model data=gmm2;
   exogenous x1 x2;
   parms a1 a2 b2 b1 2.5 c2 55 d1;
   inst b1 b2 c2 x1 x2;
   y1 = a1 * y2 + b1 * x1 * x1 + d1;
   y2 = a2 * y1 + b2 * x2 * x2 + c2 / x2 + d1;

   fit y1 y2 / gmm outv=gmmv;
run;

proc print data=gmmv(obs=15);
run;
```

The data set GMM2 was generated by the example in the preceding ESTDATA= section. The **V** matrix stored in GMMV is selected for use in an additional GMM estimation by the following FIT statement:

```
fit y1 y2 / gmm vdata=gmmv;
run;
```

A partial listing of the GMMV data set is shown in Figure 26.68. There are a total of 78 observations in this data set. The **V** matrix is 12 by 12 for this example.

Figure 26.68 The First 15 Observations in the VDATA= Data Set

Obs	_TYPE_	EQ_ROW	EQ_COL	INST_ROW	INST_COL	VALUE
1	GMM	y1	y1	1	1	1555.78
2	GMM	y1	y1	x1	1	8565.80
3	GMM	y1	y1	x1	x1	49932.47
4	GMM	y1	y1	x2	1	8244.34
5	GMM	y1	y1	x2	x1	51324.21
6	GMM	y1	y1	x2	x2	159913.24
7	GMM	y1	y1	@PRED.y1/@b1	1	49933.61
8	GMM	y1	y1	@PRED.y1/@b1	x1	301270.02
9	GMM	y1	y1	@PRED.y1/@b1	x2	317277.10
10	GMM	y1	y1	@PRED.y1/@b1	@PRED.y1/@b1	1860095.90
11	GMM	y1	y1	@PRED.y2/@b2	1	163855.31
12	GMM	y1	y1	@PRED.y2/@b2	x1	900622.60
13	GMM	y1	y1	@PRED.y2/@b2	x2	1285421.56
14	GMM	y1	y1	@PRED.y2/@b2	@PRED.y1/@b1	5173744.58
15	GMM	y1	y1	@PRED.y2/@b2	@PRED.y2/@b2	30307640.16

Output Data Sets

OUT= Data Set

For normalized form equations, the OUT= data set specified in the FIT statement contains residuals, actuals, and predicted values of the dependent variables computed from the parameter estimates. For general form equations, actual values of the endogenous variables are copied for the residual and predicted values.

The variables in the data set are as follows:

- BY variables
- RANGE variable
- ID variables
- _ESTYPE_, a character variable of length 8 that identifies the estimation method: OLS, SUR, N2SLS, N3SLS, ITOLS, ITSUR, IT2SLS, IT3SLS, GMM, ITGMM, or FIML
- _TYPE_, a character variable of length 8 that identifies the type of observation: RESIDUAL, PREDICT, or ACTUAL
- _WEIGHT_, the weight of the observation in the estimation. The _WEIGHT_ value is 0 if the observation was not used. It is equal to the product of the _WEIGHT_ model program variable and the variable named in the WEIGHT statement, if any, or 1 if weights were not used.
- the WEIGHT statement variable if used
- the model variables. The dependent variables for the normalized form equations in the estimation contain residuals, actuals, or predicted values, depending on the _TYPE_ variable, whereas the model variables that are not associated with estimated equations always contain actual values from the input data set.

- any other variables named in the OUTVARS statement. These can be program variables computed by the model program, CONTROL variables, parameters, or special variables in the model program.

The following SAS statements are used to generate and print an OUT= data set:

```
proc model data=gmm2;
   exogenous x1 x2;
   parms a1 a2 b2 b1 2.5 c2 55 d1;
   inst b1 b2 c2 x1 x2;
   y1 = a1 * y2 + b1 * x1 * x1 + d1;
   y2 = a2 * y1 + b2 * x2 * x2 + c2 / x2 + d1;

   fit y1 y2 / 3sls gmm out=resid outall ;
run;

proc print data=resid(obs=20);
run;
```

The data set GMM2 was generated by the example in the preceding ESTDATA= section above. A partial listing of the RESID data set is shown in Figure 26.69.

Figure 26.69 The OUT= Data Set

Obs	_ESTYPE_	_TYPE_	_WEIGHT_	x1	x2	y1	y2
1	3SLS	ACTUAL	1	1.00000	-1.7339	-3.05812	-23.071
2	3SLS	PREDICT	1	1.00000	-1.7339	-0.36806	-19.351
3	3SLS	RESIDUAL	1	1.00000	-1.7339	-2.69006	-3.720
4	3SLS	ACTUAL	1	1.41421	-5.3046	0.59405	43.866
5	3SLS	PREDICT	1	1.41421	-5.3046	-0.49148	45.588
6	3SLS	RESIDUAL	1	1.41421	-5.3046	1.08553	-1.722
7	3SLS	ACTUAL	1	1.73205	-5.2826	3.17651	51.563
8	3SLS	PREDICT	1	1.73205	-5.2826	-0.48281	41.857
9	3SLS	RESIDUAL	1	1.73205	-5.2826	3.65933	9.707
10	3SLS	ACTUAL	1	2.00000	-0.6878	3.66208	-70.011
11	3SLS	PREDICT	1	2.00000	-0.6878	-0.18592	-76.502
12	3SLS	RESIDUAL	1	2.00000	-0.6878	3.84800	6.491
13	3SLS	ACTUAL	1	2.23607	-7.0797	0.29210	99.177
14	3SLS	PREDICT	1	2.23607	-7.0797	-0.53732	92.201
15	3SLS	RESIDUAL	1	2.23607	-7.0797	0.82942	6.976
16	3SLS	ACTUAL	1	2.44949	14.5284	1.86898	423.634
17	3SLS	PREDICT	1	2.44949	14.5284	-1.23490	421.969
18	3SLS	RESIDUAL	1	2.44949	14.5284	3.10388	1.665
19	3SLS	ACTUAL	1	2.64575	-0.6968	-1.03003	-72.214
20	3SLS	PREDICT	1	2.64575	-0.6968	-0.10353	-69.680

OUTEST= Data Set

The OUTEST= data set contains parameter estimates and, if requested, estimates of the covariance of the parameter estimates.

The variables in the data set are as follows:

- BY variables
- _NAME_, a character variable of length 32, blank for observations that contain parameter estimates or a parameter name for observations that contain covariances
- _TYPE_, a character variable of length 8 that identifies the estimation method: OLS, SUR, N2SLS, N3SLS, ITOLS, ITSUR, IT2SLS, IT3SLS, GMM, ITGMM, or FIML
- _STATUS_, variable that gives the convergence status of estimation. _STATUS_ = 0 when convergence criteria are met, = 1 when estimation converges with a note, = 2 when estimation converges with a warning, and = 3 when estimation fails to converge
- _NUSED_, the number of observations used in estimation
- the parameters estimated

If the COVOUT option is specified, an additional observation is written for each row of the estimate of the covariance matrix of parameter estimates, with the _NAME_ values that contain the parameter names for the rows. Parameter names longer than 32 characters are truncated.

OUTPARMS= Data Set

The option OUTPARMS= writes all the parameter estimates to an output data set. This output data set contains one observation and is similar to the OUTEST= data set, but it contains all the parameters, is not associated with any FIT task, and contains no covariances. The OUTPARMS= option is used in the PROC MODEL statement, and the data set is written at the end, after any FIT or SOLVE steps have been performed.

OUTS= Data Set

The OUTS= SAS data set contains the estimate of the covariance matrix of the residuals across equations. This matrix is formed from the residuals that are computed by using the parameter estimates.

The variables in the OUTS= data set are as follows:

- BY variables
- _NAME_, a character variable that contains the name of the equation
- _TYPE_, a character variable of length 8 that identifies the estimation method: OLS, SUR, N2SLS, N3SLS, ITOLS, ITSUR, IT2SLS, IT3SLS, GMM, ITGMM, or FIML
- variables with the names of the equations in the estimation

Each observation contains a row of the covariance matrix. The data set is suitable for use with the SDATA= option in a subsequent FIT or SOLVE statement. (See the section "Tests on Parameters" on page 1601 in this chapter for an example of the SDATA= option.)

OUTSUSED= Data Set

The OUTSUSED= SAS data set contains the covariance matrix of the residuals across equations that is used to define the objective function. The form of the OUTSUSED= data set is the same as that for the OUTS= data set.

Note that OUTSUSED= is the same as OUTS= for the estimation methods that iterate the **S** matrix (ITOLS, IT2SLS, ITSUR, and IT3SLS). If the SDATA= option is specified in the FIT statement, OUTSUSED= is the same as the SDATA= matrix read in for the methods that do not iterate the **S** matrix (OLS, SUR, N2SLS, and N3SLS).

OUTV= Data Set

The OUTV= data set contains the estimate of the variance matrix, **V**. This matrix is formed from the instruments and the residuals that are computed by using the final parameter estimates obtained from the estimation method chosen.

An estimate of **V** obtained from 2SLS is used in GMM estimation. Hence if you input the dataset obtained from the OUTV statement in 2SLS into the VDATA statement while fitting GMM, you get the same result by fitting GMM directly without specifying the VDATA option.

ODS Table Names

PROC MODEL assigns a name to each table it creates. You can use these names to reference the table when you use the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in the following table.

Table 26.4 ODS Tables Produced in PROC MODEL

ODS Table Name	Description	Option
ODS Tables Created by the FIT Statement		
AugGMMCovariance	Crossproducts matrix	GMM ITALL
ChowTest	Structural change test	CHOW=
CollinDiagnostics	Collinearity diagnostics	
ConfInterval	Profile likelihood confidence intervals	PRL=
ConvCrit	Convergence criteria for estimation	default
ConvergenceStatus	Convergence status	default
CorrB	Correlations of parameters	COVB/CORRB
CorrResiduals	Correlations of residuals	CORRS/COVS
CovB	Covariance of parameters	COVB/CORRB
CovResiduals	Covariance of residuals	CORRS/COVS
Crossproducts	Crossproducts matrix	ITALL/ITPRINT
DatasetOptions	Data sets used	default
DetResidCov	Determinant of the residuals	DETAILS
DWTest	Durbin Watson test	DW=
Equations	Listing of equations to estimate	default
EstSummaryMiss	Model summary statistics for PAIRWISE	MISSING=

Table 26.4 (continued)

ODS Table Name	Description	Option
EstSummaryStats	Objective, objective * N	default
FirstLagrMultEst	First order Lagrange Multiplier estimates	GMM ITALL
GMMCovariance	Crossproducts matrix	GMM DETAILS
GMMTestStats	GMM test statistics	GMM
Godfrey	Godfrey's serial correlation test	GF=
HausmanTest	Hausman's test table	HAUSMAN
HeteroTest	Heteroscedasticity test tables	BREUSCH/PAGEN
InvXPXMat	X′X inverse for system	I
IterInfo	Iteration printing	ITALL/ITPRINT
LagLength	Model lag length	default
MinSummary	Number of parameters, estimation kind	default
ModSummary	Listing of all categorized variables	default
ModVars	Listing of model variables and parameters	default
NormalityTest	Normality test table	NORMAL
ObsSummary	Identifies observations with errors	default
ObsUsed	Observations read, used, and missing.	default
ParameterEstimates	Parameter estimates	default
ParmChange	Parameter change vector	ITALL
ResidSummary	Summary of the SSE, MSE for the equations	default
SecondLagrMultEst	Second order Lagrange Multiplier estimates	GMM ITALL
SizeInfo	Storage requirement for estimation	DETAILS
TermEstimates	Nonlinear OLS and ITOLS Estimates	OLS/ITOLS
TestResults	Test statement table	
WgtVar	The name of the weight variable	
XPXMat	X′X for system	XPX
YkVector	Marquardt iteration vector	GMM ITALL
ODS Tables Created by the SOLVE Statement		
BlockEqsAndVars	Dependency analysis block partitioning	ANALYZEDEPS=
DatasetOptions	Data sets used	default
DescriptiveStatistics	Descriptive statistics	STATS
FitStatistics	Fit statistics for simulation	STATS
LagLength	Model lag length	default
ModSummary	Listing of all categorized variables	default
ObsSummary	Simulation trace output	SOLVEPRINT
ObsUsed	Observations read, used, and missing.	default
SimulationSummary	Number of variables solved for	default
SolutionVarList	Solution variable lists	default
TheilRelStats	Theil relative change error statistics	THEIL
TheilStats	Theil forecast error statistics	THEIL
ErrorVec	Iteration Error vector	ITPRINT
ResidualValues	Iteration residual values	ITPRINT
PredictedValues	Iteration predicted values	ITPRINT
SolutionValues	Iteration solved for variable values	ITPRINT

Table 26.4 (continued)

ODS Table Name	Description	Option
ODS Tables Created by the FIT and SOLVE Statements		
AdjacencyMatrix	Adjacency graph	GRAPH
BlockAnalysis	Block analysis	BLOCK
BlockStructure	Block structure	BLOCK
CodeDependency	Variable cross reference	LISTDEP
CodeList	Listing of programs statements	LISTCODE
CrossReference	Cross-reference listing for program	
DepStructure	Dependency structure of the system	BLOCK
FirstDerivatives	First derivative table	LISTDER
IterIntg	Integration iteration output	INTGPRINT
MemUsage	Memory usage statistics	MEMORYUSE
MissingDependencies	Missing values by dependency	REPORTMISSINGS
MissingObservations	Missing values by observation	REPORTMISSINGS
MissingSymbols	Missing values by symbol	REPORTMISSINGS
ParmReadIn	Parameter estimates read in	ESTDATA=
ProgList	Listing of compiled program code	
RangeInfo	RANGE statement specification	
SortAdjacencyMatrix	Sorted adjacency graph	GRAPH
TransitiveClosure	Transitive closure graph	GRAPH

The AugGMMCovariance table is the **V** matrix augmented with the moment vector at iteration zero, produced when the ITALL option is used with the GMM option. If the **V** matrix to be used in GMM is read in by the VDATA option, then AugGMMCovariance would be the same matrix augmented with the moment vectors. The GMMCovariance ODS output is produced only when you read in a covariance matrix to be used in the GMM method. This table is produced by using DETAILS option with the GMM option.

ODS Graphics

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*).

Before you create graphs, ODS Graphics must be enabled (for example, with the ODS GRAPHICS ON statement). For more information about enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" in that chapter.

The overall appearance of graphs is controlled by ODS styles. Styles and other aspects of using ODS Graphics are discussed in the section "A Primer on ODS Statistical Graphics" in that chapter.

This section describes the use of ODS for creating graphics with the MODEL procedure.

ODS Graph Names

PROC MODEL assigns a name to each graph it creates using ODS. You can use these names to reference the graphs when you use ODS. The names are listed in Table 26.5.

To request these graphs, ODS Graphics must be enabled.

Table 26.5 ODS Graphics Produced by PROC MODEL

ODS Graph Name	Plot Description
ACFPlot	Autocorrelation of residuals
ActualByPredicted	Predicted versus actual plot
BlockDependencyPlot	Simulation dependency analysis
CooksD	Cook's *D* plot
DiagnosticsPanel	Panel of all plots
IACFPlot	Inverse autocorrelation of residuals
QQPlot	Q-Q plot of residuals
PACFPlot	Partial autocorrelation of residuals
ResidualHistogram	Histogram of the residuals
StudentResidualPlot	Studentized residual plot

Details: Simulation by the MODEL Procedure

The *solution*, given the vector $\mathbf{k}$, of the following nonlinear system of equations is the vector $\mathbf{u}$ that satisfies this equation:

$$\mathbf{q}(\mathbf{u}, \mathbf{k}, \boldsymbol{\theta}) = 0$$

A *simulation* is a set of solutions $\mathbf{u}_t$ for a specific sequence of vectors $\mathbf{k}_t$.

Model simulation can be performed to do the following:

- check how well the model predicts the actual values over the historical period
- investigate the sensitivity of the solution to changes in the input values or parameters
- examine the dynamic characteristics of the model
- check the stability of the simultaneous solution
- estimate the statistical distribution of the predicted values of the nonlinear model using Monte Carlo methods

By combining the various solution modes with different input data sets, model simulation can answer many different questions about the model. This section presents details of model simulation and solution.

Solution Modes

The following solution modes are commonly used:

- The *dynamic simultaneous forecast* mode is used for forecasting with the model. Collect the historical data on the model variables, the future assumptions of the exogenous variables, and any prior information on the future endogenous values, and combine them in a SAS data set. Use the FORECAST option in the SOLVE statement.
- The *dynamic simultaneous simulation* mode is often called *ex post simulation*, *historical simulation*, or *ex post forecasting*. Use the DYNAMIC option. This mode is the default.
- The *static simultaneous simulation* mode can be used to examine the within-period performance of the model without the complications of previous period errors. Use the STATIC option.
- The *NAHEAD*=n *dynamic simultaneous simulation* mode can be used to see how well *n*-period-ahead forecasting would have performed over the historical period. Use the NAHEAD=*n* option.

The different solution modes are explained in detail in the following sections.

Dynamic and Static Simulations

In model simulation, either solved values or actual values from the data set can be used to supply lagged values of an endogenous variable. A *dynamic* solution refers to a solution obtained by using only solved values for the lagged values. Dynamic mode is used both for forecasting and for simulating the dynamic properties of the model.

A *static* solution refers to a solution obtained by using the actual values when available for the lagged endogenous values. Static mode is used to simulate the behavior of the model without the complication of previous period errors. Dynamic simulation is the default.

If you want to use static values for lags only for the first *n* observations, and dynamic values thereafter, specify the START=*n* option. For example, if you want a dynamic simulation to start after observation twenty-four, specify START=24 on the SOLVE statement. If the model being simulated had a value lagged for four time periods, then this value would start using dynamic values when the simulation reached observation number 28.

n-Period-Ahead Forecasting

Suppose you want to regularly forecast 12 months ahead and produce a new forecast each month as more data becomes available. *n*-period-ahead forecasting allows you to test how well you would have done over time if you had been using your model to forecast one year ahead.

To see how well a model predicts *n* time periods in the future, perform an *n*-period-ahead forecast on real data and compare the forecast values with the actual values.

n-period-ahead forecasting refers to using dynamic values for the lagged endogenous variables only for lags *1* through *n–1*. For example, one-period-ahead forecasting, specified by the NAHEAD=1 option in the SOLVE statement, is the same as if a static solution had been requested. Specifying NAHEAD=2 produces a solution that uses dynamic values for lag one and static, actual, values for longer lags.

The following example is a two-year-ahead dynamic simulation. The output is shown in Figure 26.70.

```
data yearly;
   input year x1 x2 x3 y1 y2 y3;
   datalines;
84 4 9  0  7  4  5
85 5 6  1  1  27  4
86 3 8  2  5  8  2
87 2 10 3  0  10 10
88 4 7  6  20 60 40
89 5 4  8  40 40 40
90 3 2  10 50 60 60
91 2 5  11 40 50 60
;
run;

proc model data=yearly outmodel=yearlyModel;
   endogenous y1 y2 y3;
   exogenous  x1 x2 x3;

   y1 = 2 + 3*x1 - 2*x2 + 4*x3;
   y2 = 4 + lag2( y3 ) + 2*y1 + x1;
   y3 = lag3( y1 ) + y2 - x2;

   solve y1 y2 y3 / nahead=2 out=c;
run;

proc print data=c;
run;
```

Figure 26.70 NAHEAD Summary Report

The MODEL Procedure
Dynamic Simultaneous 2-Periods-Ahead Forecasting Simulation

Data Set Options	
DATA=	YEARLY
OUT=	C

Solution Summary	
Variables Solved	3
Simulation Lag Length	3
Solution Method	NEWTON
CONVERGE=	1E-8
Maximum CC	0
Maximum Iterations	1
Total Iterations	8
Average Iterations	1

Figure 26.70 *continued*

Observations Processed	
Read	20
Lagged	12
Solved	8
First	5
Last	8

Variables Solved For	y1 y2 y3

The C data set is shown in Figure 26.71:

Figure 26.71 C Data Set

Obs	_TYPE_	_MODE_	_LAG_	_ERRORS_	y1	y2	y3	x1	x2	x3
1	PREDICT	SIMULATE	0	0	0	10	7	2	10	3
2	PREDICT	SIMULATE	1	0	24	58	52	4	7	6
3	PREDICT	SIMULATE	1	0	41	101	102	5	4	8
4	PREDICT	SIMULATE	1	0	47	141	139	3	2	10
5	PREDICT	SIMULATE	1	0	42	130	145	2	5	11

The preceding two-year-ahead simulation can be emulated without using the NAHEAD= option by the following PROC MODEL statements:

```
proc model data=yearly model=yearlyModel;
  range year = 87 to 88;
  solve y1 y2 y3 / dynamic solveprint;
run;

  range year = 88 to 89;
  solve y1 y2 y3 / dynamic solveprint;
run;

  range year = 89 to 90;
  solve y1 y2 y3 / dynamic solveprint;
run;

  range year = 90 to 91;
  solve y1 y2 y3 / dynamic solveprint;
```

The totals shown under "Observations Processed" in Figure 26.70 are equal to the sum of the four individual runs.

Simulation and Forecasting

You can perform a simulation of your model or use the model to produce forecasts. *Simulation* refers to the determination of the endogenous or dependent variables as a function of the input values of the other variables, even when actual data for some of the solution variables are available in the input data set. The simulation mode is useful for verifying the fit of the model parameters. Simulation is selected by the SIMULATE option in the SOLVE statement. Simulation mode is the default.

In forecast mode, PROC MODEL solves only for those endogenous variables that are missing in the data set. The actual value of an endogenous variable is used as the solution value whenever nonmissing data for it is available in the input data set. Forecasting is selected by the FORECAST option in the SOLVE statement.

For example, an econometric forecasting model can contain an equation to predict future tax rates, but tax rates are usually set in advance by law. Thus, for the first year or so of the forecast, the predicted tax rate should really be exogenous. Or, you might want to use a prior forecast of a certain variable from a short-run forecasting model to provide the predicted values for the earlier periods of a longer-range forecast of a long-run model. A common situation in forecasting is when historical data needed to fill the initial lags of a dynamic model are available for some of the variables but have not yet been obtained for others. In this case, the forecast must start in the past to supply the missing initial lags. Clearly, you should use the actual data that are available for the lags. In all the preceding cases, the forecast should be produced by running the model in the FORECAST mode; simulating the model over the future periods would not be appropriate.

Monte Carlo Simulation

The accuracy of the forecasts produced by PROC MODEL depends on four sources of error (Pindyck and Rubinfeld 1981, pp. 405–406):

- The system of equations contains an implicit random error term ϵ

  $$\mathbf{g}(\mathbf{y}, \mathbf{x}, \hat{\boldsymbol{\theta}}) = \boldsymbol{\epsilon}$$

  where $\mathbf{y}$, $\mathbf{x}$, $\mathbf{g}$, $\hat{\boldsymbol{\theta}}$, and $\boldsymbol{\epsilon}$ are vector valued.
- The estimated values of the parameters, $\hat{\boldsymbol{\theta}}$, are themselves random variables.
- The exogenous variables might have been forecast themselves and therefore might contain errors.
- The system of equations might be incorrectly specified; the model only approximates the process modeled.

The RANDOM= option is used to request Monte Carlo (or stochastic) simulations to generate confidence intervals for errors that arise from the first two sources. The Monte Carlo simulations can be performed with $\boldsymbol{\epsilon}$, $\boldsymbol{\theta}$, or both vectors represented as random variables. The SEED= option is used to control the random number generator for the simulations. SEED=0 forces the random number generator to use the system clock as its seed value.

In Monte Carlo simulations, repeated simulations are performed on the model for random perturbations of the parameters and the additive error term. The random perturbations follow a multivariate normal distribution with expected value of 0 and covariance described by a covariance matrix of the parameter estimates in the case of θ, or a covariance matrix of the equation residuals for the case of ϵ. PROC MODEL can generate both covariance matrices or you can provide them.

The ESTDATA= option specifies a data set that contains an estimate of the covariance matrix of the parameter estimates to use for computing perturbations of the parameters. The ESTDATA= data set is usually created by the FIT statement with the OUTEST= and OUTCOV options. When the ESTDATA= option is specified, the matrix read from the ESTDATA= data set is used to compute vectors of random shocks or perturbations for the parameters. These random perturbations are computed at the start of each repetition of the solution and added to the parameter values. The perturbed parameters are fixed throughout the solution range. If the covariance matrix of the parameter estimates is not provided, the parameters are not perturbed.

The SDATA= option specifies a data set that contains the covariance matrix of the residuals to use for computing perturbations of the equations. The SDATA= data set is usually created by the FIT statement with the OUTS= option. When SDATA= is specified, the matrix read from the SDATA= data set is used to compute vectors of random shocks or perturbations for the equations. These random perturbations are computed at each observation. The simultaneous solution satisfies the model equations plus the random shocks. That is, the solution is not a perturbation of a simultaneous solution of the structural equations; rather, it is a simultaneous solution of the stochastic equations by using the simulated errors. If the SDATA= option is not specified, the random shocks are not used.

The different random solutions are identified by the _REP_ variable in the OUT= data set. An unperturbed solution with _REP_=0 is also computed when the RANDOM= option is used. RANDOM=*n* produces *n* +1 solution observations for each input observation in the solution range. If the RANDOM= option is not specified, the SDATA= and ESTDATA= options are ignored, and no Monte Carlo simulation is performed.

PROC MODEL does not have an automatic way of modeling the exogenous variables as random variables for Monte Carlo simulation. If the exogenous variables have been forecast, the error bounds for these variables should be included in the error bounds generated for the endogenous variables. If the models for the exogenous variables are included in PROC MODEL, then the error bounds created from a Monte Carlo simulation contain the uncertainty due to the exogenous variables.

Alternatively, if the distribution of the exogenous variables is known, the built-in random number generator functions can be used to perturb these variables appropriately for the Monte Carlo simulation. For example, if you know the forecast of an exogenous variable, X, has a standard error of 5.2 and the error is normally distributed, then the following statements can be used to generate random values for X:

```
x_new = x + 5.2 * rannor(456);
```

During a Monte Carlo simulation, the random number generator functions produce one value at each observation. It is important to use a different seed value for all the random number generator functions in the model program; otherwise, the perturbations will be correlated. For the unperturbed solution, _REP_=0, the random number generator functions return 0.

PROC UNIVARIATE can be used to create confidence intervals for the simulation (see the Monte Carlo simulation example in the section "Getting Started: MODEL Procedure" on page 1475).

Multivariate *t* Distribution Simulation

To perform a Monte Carlo analysis of models that have residuals distributed as a multivariate *t*, use the ERRORMODEL statement with either the ~ t(*variance*, *df*) option or with the CDF=t(*variance*, *df*) option. The CDF= option specifies the distribution that is used for simulation so that the estimation can be done for one set of distributional assumptions and the simulation for another.

The following is an example of estimating and simulating a system of equations with *t* distributed errors by using the ERRORMODEL statement.

```
/* generate simulation data set */
data five;
   set xfrate end=last;
   if last then do;
```

```
      todate = date +5;
      do date = date to todate;
         output;
      end;
   end;
run;
```

The preceding DATA step generates the data set to request a five-days-ahead forecast. The following statements estimate and forecast the three forward-rate models of the following form.

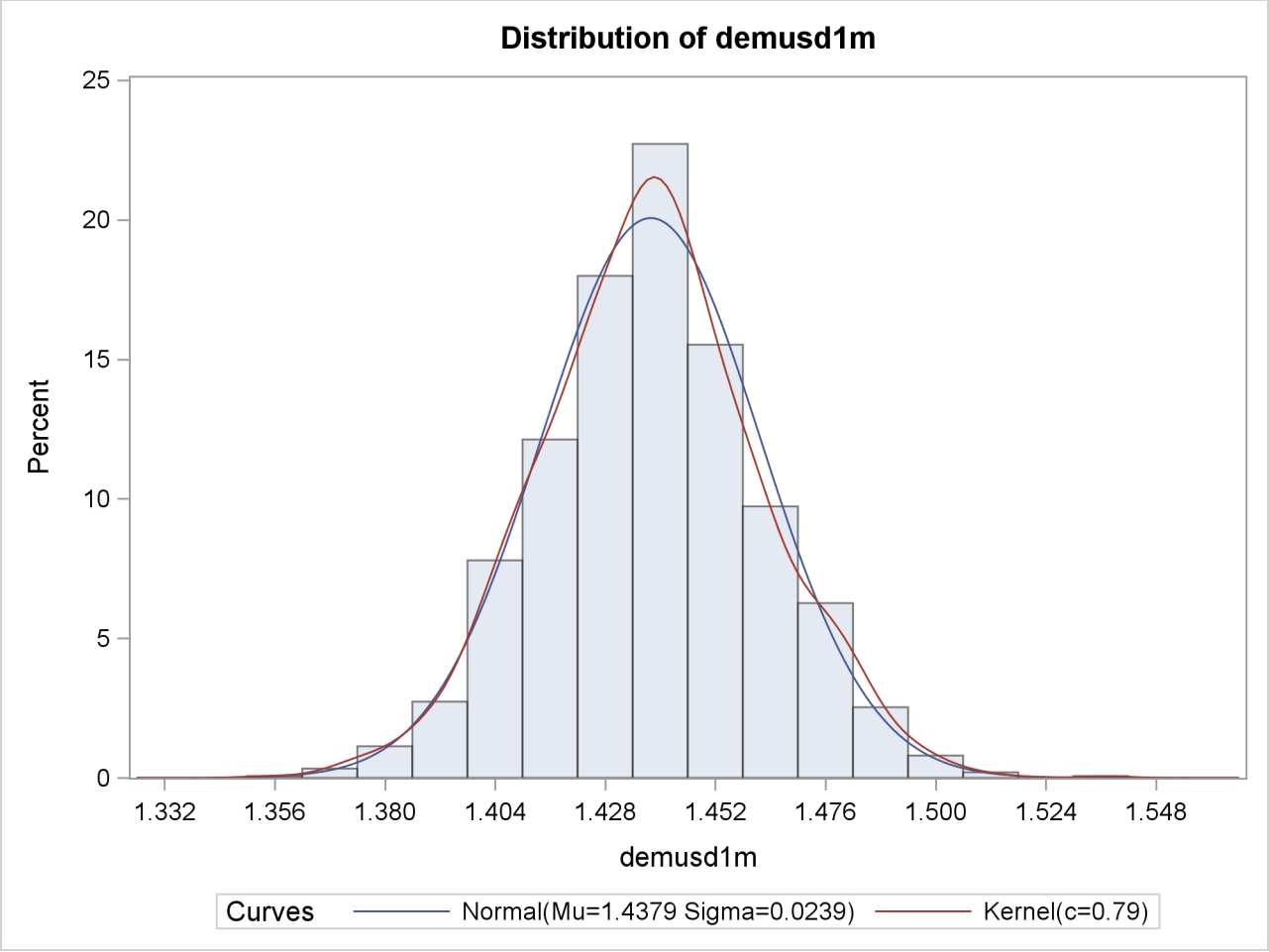
$$
\begin{aligned}
rate_t &= rate_{t-1} + \mu * rate_{t-1} + \nu \\
\nu &= \sigma * rate_{t-1} * \epsilon \\
\epsilon &\sim \mathrm{N}(0,1)
\end{aligned}
$$

```
title "Daily Multivariate Geometric Brownian Motion Model "
         "of D-Mark/USDollar Forward Rates";

proc model data=xfrate;

   parms df 15;       /* Give initial value to df */

   demusd1m = lag(demusd1m) + mu1m * lag(demusd1m);
   var_demusd1m = sigma1m ** 2 * lag(demusd1m **2);
   demusd3m = lag(demusd3m) + mu3m * lag(demusd3m);
   var_demusd3m = sigma3m ** 2 * lag(demusd3m ** 2);
   demusd6m = lag(demusd6m) + mu6m * lag(demusd6m);
   var_demusd6m = sigma6m ** 2 * lag(demusd6m ** 2);

      /* Specify the error distribution */
   errormodel demusd1m demusd3m demusd6m
        ~ t( var_demusd1m var_demusd3m var_demusd6m, df );

      /* output normalized S matrix */
   fit demusd1m demusd3m demusd6m / outsn=s;
run;
      /* forecast five days in advance */
   solve demusd1m demusd3m demusd6m /
         data=five sdata=s seed=1 random=1500 out=monte;
   id date;
run;

   /* select out the last date ---*/
data monte; set monte;
   if date = '10dec95'd then output;
run;

title "Distribution of demusd1m Five Days Ahead";
proc univariate data=monte noprint;
   var demusd1m;
   histogram demusd1m /
             normal(noprint color=red)
             kernel(noprint color=blue) cfill=ligr;
run;
```

The Monte Carlo simulation specified in the preceding example draws from a multivariate t distribution with constant degrees of freedom and forecasted variance, and it computes future states of DEMUSD1M, DEMUSD3M, and DEMUSD6M. The OUTSN= option in the FIT statement is used to specify the data set for the normalized $\mathbf{\Sigma}$ matrix. That is, the $\mathbf{\Sigma}$ matrix is created by crossing the normally distributed residuals. The normally distributed residuals are created from the t distributed residuals by using the normal inverse CDF and the t CDF. This matrix is a correlation matrix.

The distribution of DEMUSD1M on the fifth day is shown in the Figure 26.72. The two curves overlaid on the graph are a kernel density estimation and a normal distribution fit to the results.

Figure 26.72 Distribution of DEMUSD1M



Alternate Distribution Simulation

As an alternate to the normal distribution, the ERRORMODEL statement can be used in a simulation to specify other distributions. The distributions available for simulation are Cauchy, chi-squared, F, Poisson, t, and uniform. An empirical distribution can also be used if the residuals are specified by using the RESIDDATA= option in the SOLVE statement.

Except for the *t* distribution, all of these alternate distributions are univariate but can be used together in a multivariate simulation. The ERRORMODEL statement applies to solved for equations only. That is, the normal form or general form equation referred to by the ERRORMODEL statement must be one of the equations you have selected in the SOLVE statement.

In the following example, two Poisson distributed variables are used to simulate the calls that arrive at and leave a call center.

```
data s;    /* Covariance between arriving and leaving */
   arriving = 1; leaving = 0.7; _name_= "arriving";
   output;
   arriving = 0.7; leaving = 1.0; _name_= "leaving";
   output;
run;

data calls;
   date = '20mar2001'd;
   output;
run;
```

The first DATA step generates a data set that contains a covariance matrix for the ARRIVING and LEAVING variables. The covariance is

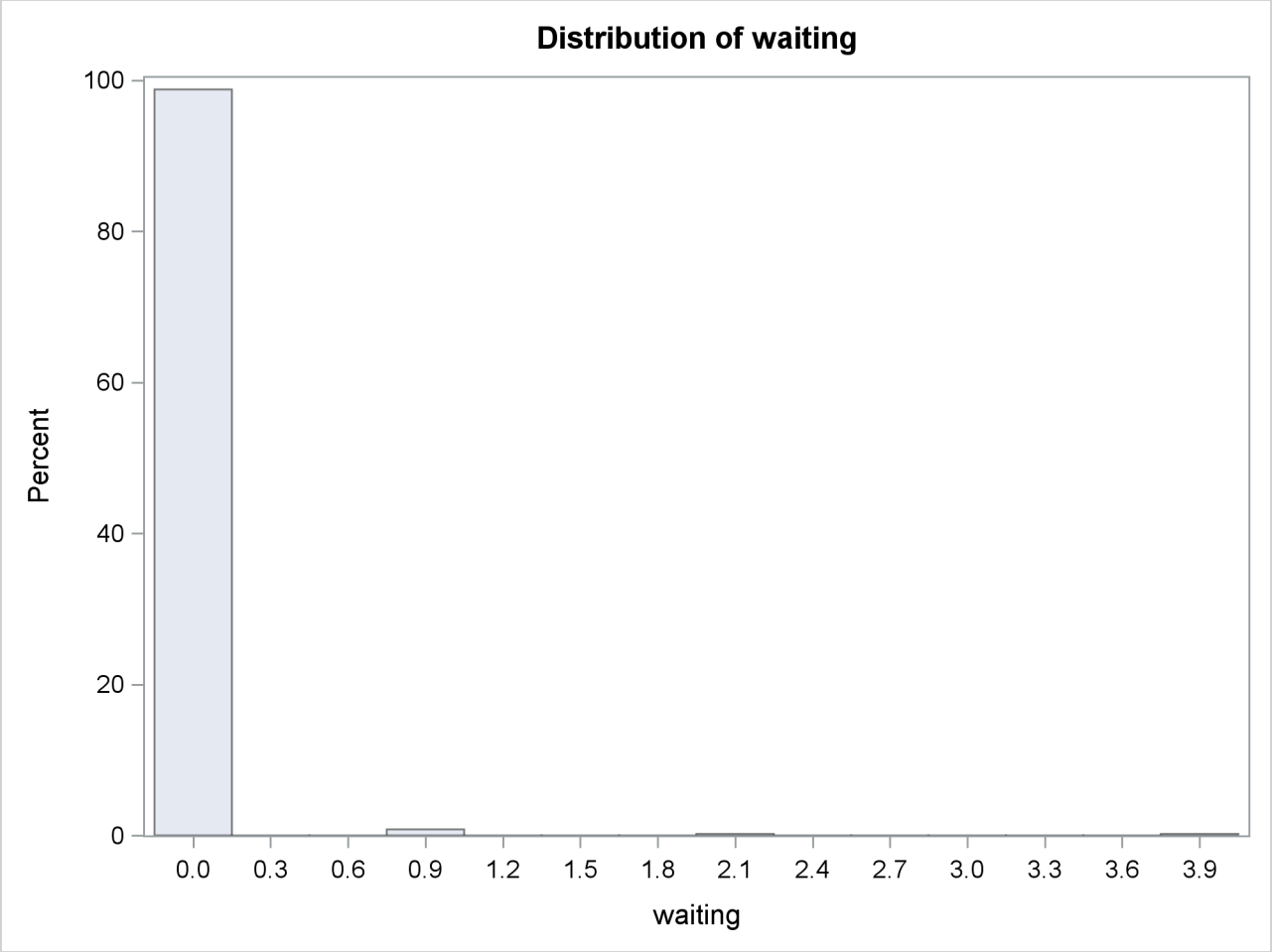
$$\begin{vmatrix} 1 & .7 \\ .7 & 1 \end{vmatrix}$$

The following statements create the number of waiting clients data:

```
proc model data=calls;
   arriving = 0;
   errormodel arriving ~ poisson( 10 );
   leaving  = 4;
   errormodel leaving ~ poisson( 11 );

   waiting = arriving - leaving;
   if waiting < 0 then waiting=0;
   outvars waiting;

   solve arriving leaving / seed=1 random=500 sdata=s out=sim;
run;

title "Distribution of Clients Waiting";
proc univariate data=sim noprint;
   var waiting ;
   histogram waiting / cfill=ligr;
run;
```

The distribution of number of waiting clients is shown in Figure 26.73.

Figure 26.73 Distribution of Number of Clients Waiting



Mixtures of Distributions—Copulas

The theory of copulas is what enables the MODEL procedure to combine and simulate multivariate distributions with different marginals. This section provides a brief overview of copulas.

Modeling a system of variables accurately is a difficult task. The underlying, ideal, distributional assumptions for each variable are usually different from each other. An individual variable might be best modeled as a *t* distribution or as a Poisson process. The correlation of the various variables are very important to estimate as well. A joint estimation of a set of variables would make it possible to estimate a correlation structure but would restrict the modeling to single, simple multivariate distribution (for example, the normal). Even with a simple multivariate distribution, the joint estimation would be computationally difficult and would have to deal with issues of missing data.

By using the MODEL procedure ERRORMODEL statement, you can combine and simulate from models of different distributions. The covariance matrix for the combined model is constructed by using the copula induced by the multivariate normal distribution. A copula is a function that couples joint distributions to their marginal distributions.

By default, the copula used in the MODEL procedure is based on the multivariate normal. This particular multivariate normal has zero mean and covariance matrix **R**. The user provides **R**, which can be created by using the following steps:

1. Each model is estimated separately and their residuals are saved.
2. The residuals for each model are converted to a normal distribution by using their CDFs, $F_i(.)$, using the relationship $\Phi^{-1}(F(\epsilon_{it}))$.
3. These normal residuals are crossed to create a covariance matrix **R**.

If the model of interest can be estimated jointly, such as multivariate T, then the OUTSN= option can be used to generate the correct covariance matrix.

A draw from this mixture of distributions is created by using the following steps that are performed automatically by the MODEL procedure.

1. Independent $N(0, 1)$ variables are generated.
2. These variables are transformed to a correlated set by using the covariance matrix **R**.
3. These correlated normals are transformed to a uniform by using $\Phi()$.
4. $F^{-1}()$ is used to compute the final sample value.

Alternate Copulas

The Gaussian, *t*, and the normal mixture copula are available in the MODEL procedure. These copulas support asymmetric parameters and can use alternate estimation methods for creating the base covariance matrix.

The normal (Gaussian) copula is the default. A draw from a Gaussian copula is obtained from

$$\mathbf{x} = \mathbf{A}\mathbf{z}$$

where $\mathbf{z} \in R^d$ is a vector of independent random normal(0, 1) draws, $\mathbf{A} \in R^{d \times d}$ is the square root of the covariance matrix, **R**. For the normal mixture and *t* copula, a draw is created as

$$\mathbf{x} = w\boldsymbol{\gamma} + \sqrt{w}\mathbf{A}\mathbf{z}$$

where w is a scalar random variable and $\boldsymbol{\gamma} \in R^d$ is a vector of asymmetry parameters. $\boldsymbol{\gamma}$ is specified in the SDATA= data set. If $W \sim$inverse gamma$(df/2, df/2)$, then **x** is multivariate *t* or skewed *t* if $\boldsymbol{\gamma}$ is provided. When NORMALMIX is specified, w is distributed as a step function with each of the n positive variances, $v_1 \ldots v_n$, having probability $p_1 \ldots p_n$.

The covariance matrix $\mathbf{R} = \mathbf{A}'\mathbf{A}$ is specified with the SDATA= option. The vector of asymmetry parameters, $\boldsymbol{\gamma}$, defaults to zero or is specified in the SDATA= data set with _TYPE_=ASYM. The ASYM option specifies that the nonzero asymmetry vector, $\boldsymbol{\gamma}$, is to be used.

In the event the covariance matrix specified with the SDATA= option is not positive semidefinite the matrix is modified to be positive semidefinite. See Rebonato and Jäckel (1999) for more details.

The actual draw for an individual variable, y_i, depends on the marginal distribution of the variable, $\tilde{F}$, and the chosen copula F as

$$y_i = \tilde{F}_i^{-1}\left(F(x_i)\right)$$

Archimedean Copulas

The three Archimedean copulas available in the MODEL procedure are the Clayton, Gumbel, and Frank copulas. Archimedean copulas require only a single parameter, θ, to define the joint distribution's covariance structure for a simulation problem, Therefore, a covariance matrix is not required to perform simulations that use Archimedean copulas, and the SDATA= option does not have to be specified for these simulations. For more information about Archimedean copulas, including the functional forms of the Clayton, Gumbel, and Frank copulas, see the section "Archimedean Copulas" in Chapter 11, "The COPULA Procedure."

Asymmetrical Copula Example

In this example, an asymmetrical *t* copula is used to correlate two uniform distributions. The asymmetrical parameter is varied over a range of values to demonstrate its effect. The resulting graphs is produced by using ODS graphics.

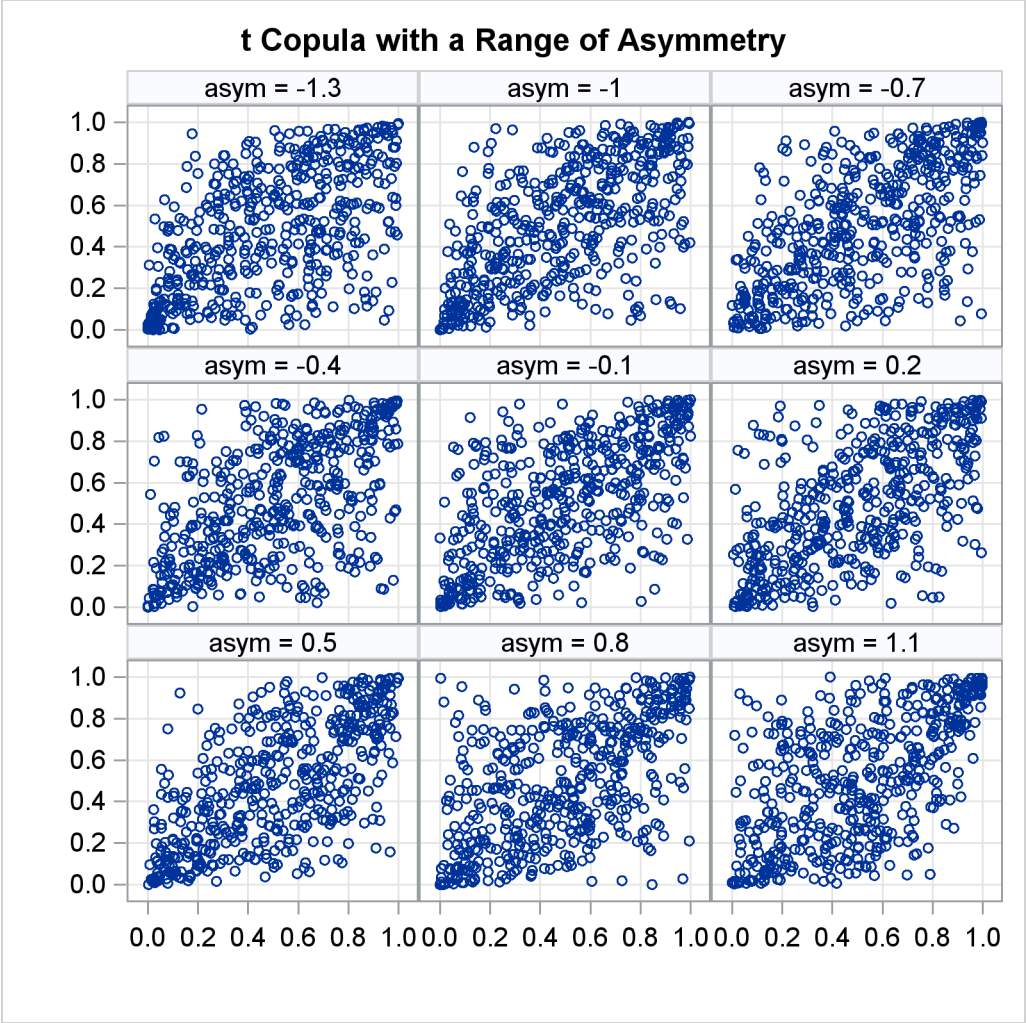
```
data histdata;
   do asym = -1.3 to 1.1 by .3;
      date='01aug2007'd;
      y = .5;
      z = .5;
      output;
   end;
run ;

  /* Add the asymmetric parameter to cov mat */
data asym;
   do asym = -1.3 to 1.1 by .3;
      y = asym;
      z = 0;
      _name_ = "    ";
      _type_ = "asym";
      output;
      y = 1;
      z = .65;
      _name_ = "y";
      _type_ = "cov";
      output;
      y = .65;
      z = 1;
      _name_ = "z";
      _type_ = "cov";
      output;
   end;
run;
```

```
proc model out=sim(where=(_REP_ > 0)) data=histdata sdata=asym;
   y = 0;
   errormodel y ~ Uniform(0,1);

   z = 0;
   errormodel z ~ Uniform(0,1);

   solve y z / random=500 seed=12345 copula=(t(5) asym );
   by asym;
run;
```

To produce a panel plot of this joint distribution, use the following SAS/GRAPH statements.

```
ods graphics on / height=800 width=800;
proc template;
  define statgraph myplot.panel;
  BeginGraph;
    entrytitle halign=left halign=center
          textattrs=GRAPHTITLETEXT "t Copula with a Range of Asymmetry";

    layout datapanel classvars=(asym) / rows=3 columns=3
          order=rowmajor height=1024 width=1420
          rowaxisopts=(griddisplay=on label=' ')
          columnaxisopts=(griddisplay=on label=' ');
       layout prototype;
          scatterplot x=z y=y ;
       endlayout;
    endlayout;
  EndGraph;
  end;
run;

proc sgrender data=sim template='myplot.panel';
run;
```

Figure 26.74 *t* Copula with Asymmetry



Quasi-Random Number Generators

Traditionally high-discrepancy pseudo-random number generators are used to generate innovations in Monte Carlo simulations. Loosely translated, a high-discrepancy pseudo-random number generator is one in which there is very little correlation between the current number generated and the past numbers generated. This property is ideal if indeed independence of the innovations is required. If, on the other hand, the efficient spanning of a multidimensional space is desired, a low discrepancy, quasi-random number generator can be used. A quasi-random number generator produces numbers that have no random component.

A simple one-dimensional quasi-random sequence is the van der Corput sequence. Given a prime number r ($r \geq 2$), any integer has a unique representation in terms of base r. A number in the interval [0,1) can be created by inverting the representation base power by base power. For example, consider r=3 and n=1, 1 in base 3 is

$$1_{10} = 1 \cdot 3^0 = 1_3$$

When the powers of 3 are inverted,

$$\phi(1) = \frac{1}{3}$$

Also, 11 in base 3 is

$$11_{10} = 1 \cdot 3^2 + 2 \cdot 3^0 = 102_3$$

When the powers of 3 are inverted,

$$\phi(11) = \frac{1}{9} + 2 \cdot \frac{1}{3} = \frac{7}{9}$$

The first 10 numbers in this sequence $\phi(1) \ldots \phi(10)$ are provided below

$$0, \frac{1}{3}, \frac{2}{3}, \frac{1}{9}, \frac{4}{9}, \frac{7}{9}, \frac{2}{9}, \frac{5}{9}, \frac{8}{9}, \frac{1}{27}$$

As the sequence proceeds, it fills in the gaps in a uniform fashion.

Several authors have expanded this idea to many dimensions. Two versions supported by the MODEL procedure are the Sobol sequence (QUASI=SOBOL) and the Faure sequence (QUASI=FAURE). The Sobol sequence is based on binary numbers and is generally computationally faster than the Faure sequence. The Faure sequence uses the dimensionality of the problem to determine the number base to use to generate the sequence. The Faure sequence has better distributional properties than the Sobol sequence for dimensions greater than 8.

As an example of the difference between a pseudo-random number and a quasi-random number, consider simulating a bivariate normal with 100 draws.

Figure 26.75 Kernel Density of a Bivariate Normal produced by 100 Faure-Random Draws

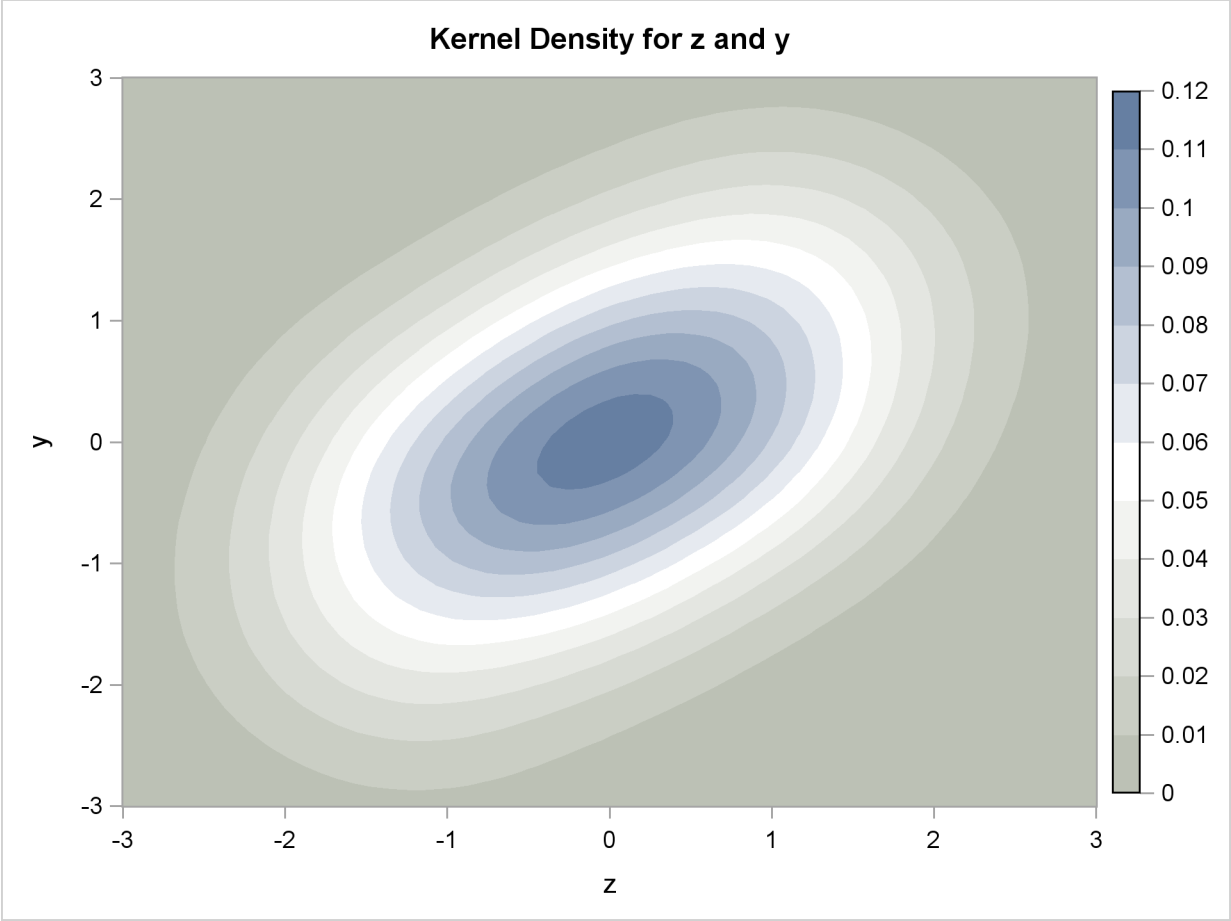
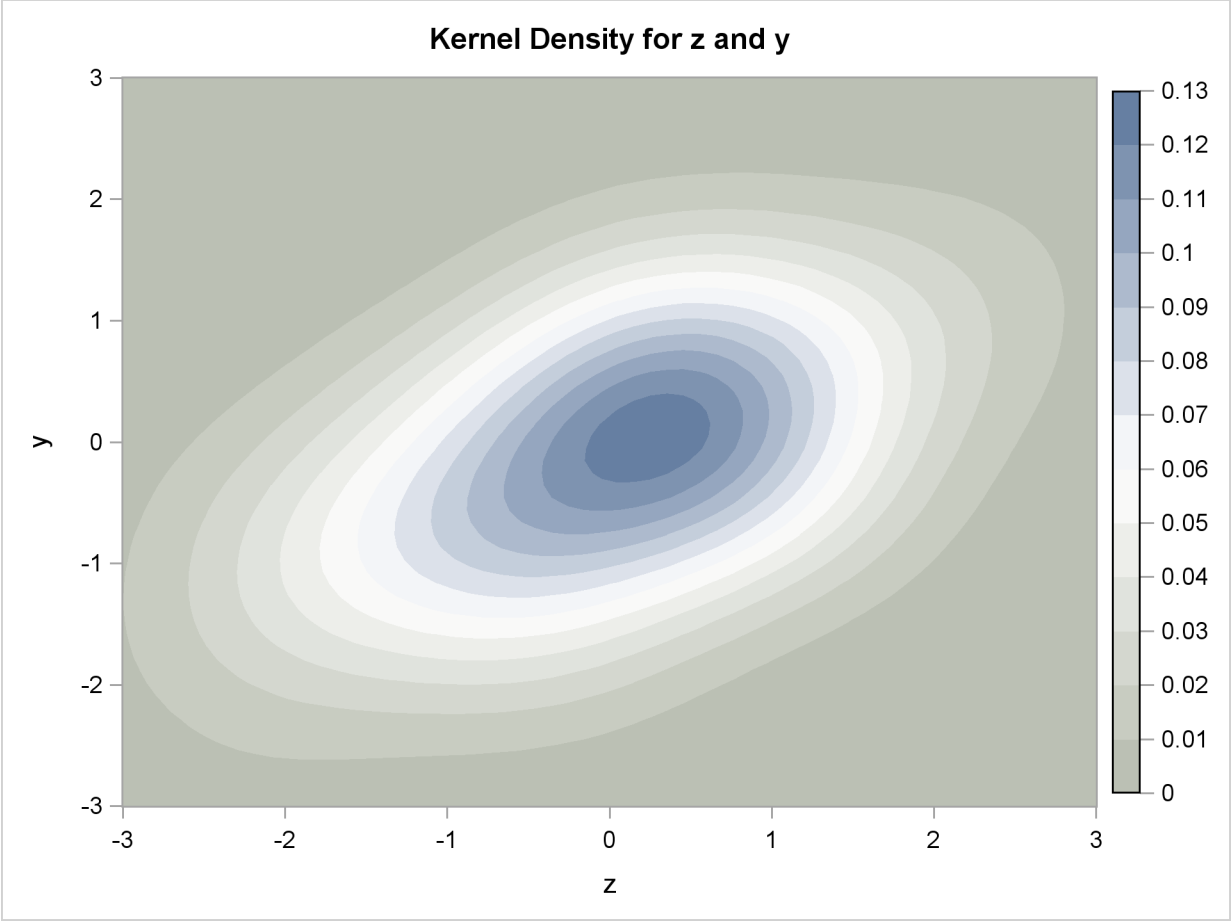


Figure 26.76 Kernel Density of a Bivariate Normal produced by 100 Pseudo-Random Draws



Solution Mode Output

The following SAS statements dynamically forecast the solution to a nonlinear equation:

```
proc model data=sashelp.citimon;
   parameters a 0.010708  b  -0.478849 c 0.929304;
   lhur = 1/(a * ip) + b + c * lag(lhur);
   solve lhur / out=sim forecast dynamic;
run;
```

The first page of output produced by the SOLVE step is shown in Figure 26.77. This is the summary description of the model. The error message states that the simulation was aborted at observation 144 because of missing input values.

Figure 26.77 Solve Step Summary Output

The MODEL Procedure

Model Summary	
Model Variables	1
Parameters	3
Equations	1
Number of Statements	1
Program Lag Length	1

Model Variables	LHUR
Parameters(Value)	a(0.010708) b(-0.478849) c(0.929304)
Equations	LHUR

The second page of output, shown in Figure 26.78, gives more information on the failed observation.

Figure 26.78 Solve Step Error Message

The MODEL Procedure
Dynamic Single-Equation Forecast

Error: Solution values are missing because of missing input values for observation 144 at NEWTON iteration 0.

Note: Additional information on the values of the variables at this observation, which may be helpful in determining the cause of the failure of the solution process, is printed below.

Observation	144	**Iteration**	0	**CC**	-1.000000
		Missing	1		

Iteration Errors - Missing.

The MODEL Procedure
Dynamic Single-Equation Forecast

--- Listing of Program Data Vector ---

_N_:	144	ACTUAL.LHUR:	.	ERROR.LHUR:	.
IP:	.	LHUR:	7.10000	PRED.LHUR:	.
a:	0.01071	b:	-0.47885	c:	0.92930

Note: Simulation aborted.

From the program data vector, you can see the variable IP is missing for observation 144. LHUR could not be computed, so the simulation aborted.

The solution summary table is shown in Figure 26.79.

Figure 26.79 Solution Summary Report

The MODEL Procedure
Dynamic Single-Equation Forecast

Data Set Options	
DATA=	SASHELP.CITIMON
OUT=	SIM

Solution Summary	
Variables Solved	1
Forecast Lag Length	1
Solution Method	NEWTON
CONVERGE=	1E-8
Maximum CC	0
Maximum Iterations	1
Total Iterations	143
Average Iterations	1

Observations Processed	
Read	145
Lagged	1
Solved	143
First	2
Last	145
Failed	1

Variables Solved For	LHUR

This solution summary table includes the names of the input data set and the output data set followed by a description of the model. The table also indicates that the solution method defaulted to Newton's method. The remaining output is defined as follows.

Maximum CC	is the maximum convergence value accepted by the Newton procedure. This number is always less than the value for the CONVERGE= option.
Maximum Iterations	is the maximum number of Newton iterations performed at each observation and each replication of Monte Carlo simulations.
Total Iterations	is the sum of the number of iterations required for each observation and each Monte Carlo simulation.
Average Iterations	is the average number of Newton iterations required to solve the system at each step.
Solved	is the number of observations used times the number of random replications selected plus one, for Monte Carlo simulations. The one additional simulation is the original unperturbed solution. For simulations that do not involve Monte Carlo, this number is the number of observations used.

Summary Statistics

The STATS and THEIL options are used to select goodness-of-fit statistics. Actual values must be provided in the input data set for these statistics to be printed. When the RANDOM= option is specified, the statistics do not include the unperturbed (_REP_=0) solution.

STATS Option Output

The following statements show the addition of the STATS and THEIL options to the model in the previous section:

```
proc model data=sashelp.citimon;
   parameters a 0.010708  b  -0.478849 c 0.929304;
   lhur= 1/(a * ip) + b + c * lag(lhur) ;
   solve lhur / out=sim dynamic stats theil;
   range date to '01nov91'd;
run;
```

The STATS output in Figure 26.80 and the THEIL output in Figure 26.81 are generated.

Figure 26.80 STATS Output

The MODEL Procedure
Dynamic Single-Equation Simulation

Solution Range DATE = FEB1980 To NOV1991

Descriptive Statistics

			Actual		Predicted		
Variable	N Obs	N	Mean	Std Dev	Mean	Std Dev	Label
LHUR	142	142	7.0887	1.4509	7.2473	1.1465	UNEMPLOYMENT RATE: ALL WORKERS, 16 YEARS

Statistics of fit

Variable	N	Mean Error	Mean % Error	Mean Abs Error	Mean Abs % Error	RMS Error	RMS % Error	R-Square	Label
LHUR	142	0.1585	3.5289	0.6937	10.0001	0.7854	11.2452	0.7049	UNEMPLOYMENT RATE: ALL WORKERS, 16 YEARS

The number of observations (Nobs), the number of observations with both predicted and actual values nonmissing (N), and the mean and standard deviation of the actual and predicted values of the determined variables are printed first. The next set of columns in the output are defined as follows:

Mean Error	$\frac{1}{N}\sum_{j=1}^{N}(\hat{y}_j - y_j)$
Mean % Error	$\frac{100}{N}\sum_{j=1}^{N}(\hat{y}_j - y_j)/y_j$
Mean Abs Error	$\frac{1}{N}\sum_{j=1}^{N}\lvert\hat{y}_j - y_j\rvert$
Mean Abs % Error	$\frac{100}{N}\sum_{j=1}^{N}\lvert(\hat{y}_j - y_j)/y_j\rvert$
RMS Error	$\sqrt{\frac{1}{N}\sum_{j=1}^{N}(\hat{y}_j - y_j)^2}$
RMS % Error	$100\sqrt{\frac{1}{N}\sum_{j=1}^{N}((\hat{y}_j - y_j)/y_j)^2}$
R-square	$1 - SSE/CSSA$
SSE	$\sum_{j=1}^{N}(\hat{y}_j - y_j)^2$
SSA	$\sum_{j=1}^{N}(y_j)^2$
CSSA	$SSA - \left(\sum_{j=1}^{N} y_j\right)^2$
$\hat{y}$	predicted value
y	actual value

When the RANDOM= option is specified, the statistics do not include the unperturbed (_REP_=0) solution.

THEIL Option Output

The THEIL option specifies that Theil forecast error statistics be computed for the actual and predicted values and for the relative changes from lagged values. Mathematically, the quantities are

$$\hat{yc} = (\hat{y} - lag(y))/lag(y)$$

$$yc = (y - lag(y))/lag(y)$$

where $\hat{yc}$ is the relative change for the predicted value and yc is the relative change for the actual value.

Figure 26.81 THEIL Output

Theil Forecast Error Statistics

			MSE Decomposition Proportions						Inequality Coef		
Variable	N	MSE	Corr (R)	Bias (UM)	Reg (UR)	Dist (UD)	Var (US)	Covar (UC)	U1	U	Label
LHUR	142	0.6168	0.85	0.04	0.01	0.95	0.15	0.81	0.1086	0.0539	UNEMPLOYMENT RATE: ALL WORKERS, 16 YEARS

Figure 26.81 *continued*

Theil Relative Change Forecast Error Statistics											
		Relative Change			MSE Decomposition Proportions				Inequality Coef		
Variable	N	MSE	Corr (R)	Bias (UM)	Reg (UR)	Dist (UD)	Var (US)	Covar (UC)	U1	U	Label
LHUR	142	0.0126	-0.08	0.09	0.85	0.06	0.43	0.47	4.1226	0.8348	UNEMPLOYMENT RATE: ALL WORKERS, 16 YEARS

The columns have the following meaning:

Corr (R) is the correlation coefficient, ρ, between the actual and predicted values.

$$\rho = \frac{\text{cov}(y, \hat{y})}{\sigma_a \sigma_p}$$

where σ_p and σ_a are the standard deviations of the predicted and actual values.

Bias (UM) is an indication of systematic error and measures the extent to which the average values of the actual and predicted deviate from each other.

$$\frac{(\text{E}(y) - \text{E}(\hat{y}))^2}{\frac{1}{N}\sum_{t=1}^{N}(y_t - \hat{y}_t)^2}$$

Reg (UR) is defined as $(\sigma_p - \rho * \sigma_a)^2 / MSE$. Consider the regression

$$y = \alpha + \beta \hat{y}$$

If $\hat{\beta} = 1$, UR will equal zero.

Dist (UD) is defined as $(1 - \rho^2)\sigma_a \sigma_a / MSE$ and represents the variance of the residuals obtained by regressing yc on $\hat{yc}$.

Var (US) is the variance proportion. US indicates the ability of the model to replicate the degree of variability in the endogenous variable.

$$US = \frac{(\sigma_p - \sigma_a)^2}{MSE}$$

Covar (UC) represents the remaining error after deviations from average values and average variabilities have been accounted for.

$$UC = \frac{2(1 - \rho)\sigma_p \sigma_a}{MSE}$$

U1 is a statistic that measures the accuracy of a forecast defined as follows:

$$U1 = \frac{\sqrt{MSE}}{\sqrt{\frac{1}{N}\sum_{t=1}^{N}(y_t)^2}}$$

U is the Theil's inequality coefficient defined as follows:

$$U = \frac{\sqrt{MSE}}{\sqrt{\frac{1}{N}\sum_{t=1}^{N}(y_t)^2} + \sqrt{\frac{1}{N}\sum_{t=1}^{N}(\hat{y}_t)^2}}$$

MSE is the mean square error. In the case of the relative change Theil statistics, the MSE is computed as follows:

$$MSE = \frac{1}{N}\sum_{t=1}^{N}(\hat{yc}_t - yc_t)^2$$

More information about these statistics can be found in the references Maddala (1977, pp. 344–347) and Pindyck and Rubinfeld (1981, pp. 364–365).

Goal Seeking: Solving for Right-Hand-Side Variables

The process of computing input values that are needed to produce target results is often called *goal seeking*. To compute a goal-seeking solution, use a SOLVE statement that lists the variables you want to solve for and provide a data set that contains values for the remaining variables.

Consider the following demand model for packaged rice

$$quantity\ demanded = \alpha_1 + \alpha_2 price^{2/3} + \alpha_3 income$$

where *price* is the price of the package and *income* is disposable personal income. The only variable the company has control over is the price it charges for rice. This model is estimated by using the following simulated data and PROC MODEL statements:

```
data demand;
   do t=1 to 40;
      price = (rannor(10) +5) * 10;
      income = 8000 * t ** (1/8);
      demand = 7200 - 1054 * price ** (2/3) +
               7 * income + 100 * rannor(1);
      output;
   end;
run;

data goal;
   demand = 85000;
   income = 12686;
run;
```

The goal is to find the price the company would have to charge to meet a sales target of 85,000 units. To do this, a data set is created with a DEMAND variable set to 85000 and with an INCOME variable set to 12686, the last income value.

The desired price is then determined by using the following PROC MODEL statements:

```
proc model data=demand
           outmodel=demandModel;
   demand = a1 - a2 * price ** (2/3) + a3 * income;
   fit demand / outest=demest;
   solve price / estdata=demest data=goal solveprint;
run;
```

The SOLVEPRINT option prints the solution values, number of iterations, and final residuals at each observation. The SOLVEPRINT output from this solve is shown in Figure 26.82.

Figure 26.82 Goal Seeking, SOLVEPRINT Output

The MODEL Procedure
Single-Equation Simulation

Observation 1	**Iterations** 6	**CC** 0.000000	**ERROR.demand** 0.000000

Solution Values
price
33.59016

The output indicates that it took six Newton iterations to determine the PRICE of 33.5902, which makes the DEMAND value within 16E–11 of the goal of 85,000 units.

Consider a more ambitious goal of 100,000 units. The output shown in Figure 26.83 indicates that the sales target of 100,000 units is not attainable according to this model.

```
data goal;
   demand = 100000;
   income = 12686;
run;

proc model model=demandModel;
   solve price / estdata=demest data=goal solveprint;
run;
```

Figure 26.83 Goal Seeking, Convergence Failure

The MODEL Procedure
Single-Equation Simulation

Error: Could not reduce norm of residuals in 10 subiterations.

Error: The solution failed because 1 equations are missing or have extreme values for observation 1 at NEWTON iteration 1.

Observation 1	**Iteration** 1	**CC** -1.000000
	Missing 1	

```
   The MODEL Procedure
Single-Equation Simulation

                --- Listing of Program Data Vector ---
_N_:                 12   ACTUAL.demand:    100000   ERROR.demand:          .
PRED.demand:          .   a1:          7126.437997   a2:          1040.841492
a3:            6.992694   demand:          100000   income:            12686
price:        -0.000172
@PRED.demand/@pri:     .
```

The program data vector with the error note indicates that even after 10 subiterations, the norm of the residuals could not be reduced. The sales target of 100,000 units are unattainable with the given model. You might need to reformulate your model or collect more data to more accurately reflect the market response.

Numerical Solution Methods

If the SINGLE option is not used, PROC MODEL computes values that simultaneously satisfy the model equations for the variables named in the SOLVE statement. PROC MODEL provides three iterative methods, Newton, Jacobi, and Seidel, for computing a simultaneous solution of the system of nonlinear equations.

Single-Equation Solution

For normalized form equation systems, the solution either can simultaneously satisfy all the equations or can be computed for each equation separately, by using the actual values of the solution variables in the current period to compute each predicted value. By default, PROC MODEL computes a simultaneous solution. The SINGLE option in the SOLVE statement selects single-equation solutions.

Single-equation simulations are often used to produce residuals (which estimate the random terms of the stochastic equations) rather than the predicted values themselves. If the input data and range are the same as those used for parameter estimation, a static single-equation simulation reproduces the residuals of the estimation.

Newton's Method

The NEWTON option in the SOLVE statement requests Newton's method to simultaneously solve the equations for each observation. Newton's method is the default solution method. Newton's method is an iterative scheme that uses the derivatives of the equations with respect to the solution variables, $\mathbf{J}$, to compute a change vector as

$$\Delta \mathbf{y}^i = \mathbf{J}^{-1}\mathbf{q}(\mathbf{y}^i, \mathbf{x}, \boldsymbol{\theta})$$

PROC MODEL builds and solves $\mathbf{J}$ by using efficient sparse matrix techniques. The solution variables $\boldsymbol{y}^i$ at the ith iteration are then updated as

$$\mathbf{y}^{i+1} = \mathbf{y}^i + d \times \Delta \mathbf{y}^i$$

where d is a damping factor between 0 and 1 chosen iteratively so that

$$\|\mathbf{q}(\mathbf{y}^{i+1}, \mathbf{x}, \boldsymbol{\theta})\| < \|\mathbf{q}(\mathbf{y}^i, \mathbf{x}, \boldsymbol{\theta})\|$$

The number of subiterations that are allowed for finding a suitable d is controlled by the MAXSUBITER= option. The number of iterations of Newton's method that are allowed for each observation is controlled by MAXITER= option. See Ortega and Rheinbolt (1970) for more details.

Optimization Method

The OPTIMIZE option in the SOLVE statement requests that an optimization algorithm be used to minimize a norm of the errors in equations subject to constraints on the solution variables. The OPTIMIZE method is the only solution method that supports constraints on solution variables that are specified using the BOUNDS and RESTRICT statements. Constraints are ignored by the other solution methods. The OPTIMIZE method performs the following optimization:

$$\begin{aligned} \text{minimize} \quad & \|\mathbf{q}(\mathbf{y}, \mathbf{x}, \boldsymbol{\theta})\| \\ \text{subject to} \quad & \mathbf{y}_l \le \mathbf{y} \le \mathbf{y}_u \\ \text{and} \quad & f(\mathbf{y}) \ge 0) \end{aligned}$$

The norm used in the minimization process is

$$\|\mathbf{q}(\mathbf{y}, \mathbf{x}, \boldsymbol{\theta})\| = \mathbf{q}(\mathbf{y}, \mathbf{x}, \boldsymbol{\theta})' \text{diag}(\mathbf{S})^{-1} \mathbf{q}(\mathbf{y}, \mathbf{x}, \boldsymbol{\theta})$$

where the $\mathbf{S}$ matrix is the covariance of equation errors that is specified by the SDATA= option in the SOLVE statement. If no SDATA= option is specified, the identity matrix is used. Both strict inequality and inequality constraints on the solution variables can be imposed using the BOUNDS or RESTRICT statement. For bounded problems, each lower and upper strict inequality is transformed into an inequality by using the equations

$$y_l = (y_{\text{lower strict}} + \epsilon)/(1 - \epsilon)$$
$$y_u = (y_{\text{upper strict}} - \epsilon)/(1 + \epsilon)$$

When strict inequality expressions are imposed using the RESTRICT statement, these expressions are transformed into an inequality by using the equation

$$f(\mathbf{y}) = (f_{\text{strict}}(\mathbf{y}) + \epsilon)/(1 - \epsilon)$$

where $f_{\text{strict}}(\mathbf{y})$ is a nonlinear strict inequality constraint. The tolerance ϵ is controlled by the EPSILON= option in the SOLVE statement and defaults to 10^{-8}. To achieve the best performance from the minimization algorithm, both the first and second analytic derivatives of the equation errors with respect to the solution variables are used to compute the gradient and second derivatives of the objective function, $\|\mathbf{q}(\mathbf{y}, \mathbf{x}, \boldsymbol{\theta})\|$. Analytic derivatives of the restriction expressions that are used to specify constraints are also used in the minimization. The gradient of the objective function is

$$\nabla \|\mathbf{q}(\mathbf{y}, \mathbf{x}, \boldsymbol{\theta})\| = 2\,\mathbf{J}' \text{diag}(\mathbf{S})^{-1} \mathbf{q}(\mathbf{y}, \mathbf{x}, \boldsymbol{\theta})$$

The matrix of second derivatives of the objective function with respect to the solution variables is

$$\frac{\partial^2 \|\mathbf{q}(\mathbf{y}, \mathbf{x}, \boldsymbol{\theta})\|}{\partial \mathbf{y}^2} = 2 \left(\mathbf{J}' \text{diag}(\mathbf{S})^{-1} \mathbf{J} + \sum_{k=1}^{d} \frac{\partial^2 q_k(\mathbf{y}, \mathbf{x}, \boldsymbol{\theta})}{\partial \mathbf{y}^2} \text{diag}(\mathbf{S})^{-1} q_k(\mathbf{y}, \mathbf{x}, \boldsymbol{\theta}) \right)$$

where d is the number of equations.

The algorithm that is used to find a minimum of $\|\mathbf{q}(\mathbf{y}, \mathbf{x}, \boldsymbol{\theta})\|$ subject to bounds on the solution variables employs the interior point technique for nonlinear optimization problems. For further information about this optimization method, see Chapter 10, "The Nonlinear Programming Solver" (*SAS/OR User's Guide: Mathematical Programming*).

When constraints are active in a solution, the minimum value of the objective function, $\|\mathbf{q}(\mathbf{y}, \mathbf{x}, \boldsymbol{\theta})\|$, is typically greater than 0. The diagnostic quantities that are produced by the OUTOBJVALS and OUTVIOLATIONS options are available to help identify and characterize solutions that have active bounds constraints. The following program contains a boundary constraint that becomes active in steps 6, 8, 10, 12, 13, and 16 of a Monte Carlo simulation:

```
proc model data=d sdata=s;
   dependent rate stock;
   parms theta   0.2
         kappa   0.002
         sigma   0.4
         sinit   1
         vol     .1;
   id i;

   bounds rate >= 0;

   rate   = zlag(rate) + kappa*(theta - zlag(rate));
   h.rate = sigma**2 * zlag(rate);
   eq.stock = log(stock/sinit) - (rate + vol*vol/2);
   h.stock = vol**2;

   solve / optimize converge=1e-6 seed=1 random=1 out=o outobjvals outviolations;
quit;

proc print data=o(where=(_objval_>1e-6));
run;
```

Figure 26.84 shows how the OUTOBJVALS option can be used to identify simulation steps with an active bounds constraint, and how the OUTVIOLATIONS option can be used to determine that the RATE equation is not satisfied for those steps.

Figure 26.84 Objective Function and Violation Values

Obs	i	_TYPE_	_MODE_	_REP_	_ERRORS_	_OBJVAL_	rate	stock
51	6	PREDICT	SIMULATE	1	0	.000363415	0.000027	1.03050
52	6	VIOL	SIMULATE	1	0	.000363415	-0.019073	0.00000
55	8	PREDICT	SIMULATE	1	0	.000123866	0.000045	1.08828
56	8	VIOL	SIMULATE	1	0	.000123866	-0.011151	0.00000
59	10	PREDICT	SIMULATE	1	0	.000330766	0.000028	0.96248
60	10	VIOL	SIMULATE	1	0	.000330766	-0.018207	-0.00000
63	12	PREDICT	SIMULATE	1	0	.000034095	0.000086	0.85526
64	12	VIOL	SIMULATE	1	0	.000034095	-0.005895	-0.00000
65	13	PREDICT	SIMULATE	1	0	.000011997	0.000141	1.10514
66	13	VIOL	SIMULATE	1	0	.000011997	-0.003573	-0.00000
71	16	PREDICT	SIMULATE	1	0	.000118982	0.000046	1.07103
72	16	VIOL	SIMULATE	1	0	.000118982	-0.010931	0.00000

Jacobi Method

The JACOBI option in the SOLVE statement selects a matrix-free alternative to Newton's method. This method is the traditional nonlinear Jacobi method found in the literature. The Jacobi method as implemented in PROC MODEL substitutes predicted values for the endogenous variables and iterates until a fixed point is reached. Then necessary derivatives are computed only for the diagonal elements of the jacobian, $\mathbf{J}$.

If the normalized form equation is

$$\mathbf{y} = \mathbf{f}(\mathbf{y}, \mathbf{x}, \boldsymbol{\theta})$$

the Jacobi iteration has the form

$$\mathbf{y}^{i+1} = \mathbf{f}(\mathbf{y}^i, \mathbf{x}, \boldsymbol{\theta})$$

Seidel Method

The Seidel method is an order-dependent alternative to the Jacobi method. You select the Seidel method by specifying the SEIDEL option in the SOLVE statement. The Seidel method is like the Jacobi method, except that in the Seidel method the model is further edited to substitute the predicted values into the solution variables immediately after they are computed. The Seidel method thus differs from the other methods in that the values of the solution variables are not fixed within an iteration. With the other methods, the order of the equations in the model program makes no difference, but the Seidel method might work much differently when the equations are specified in a different sequence. This fixed-point method is the traditional nonlinear Seidel method found in the literature.

The iteration has the form

$$\mathbf{y}_j^{i+1} = \mathbf{f}(\hat{\mathbf{y}}^i, \mathbf{x}, \boldsymbol{\theta})$$

where $\mathbf{y}_j^{i+1}$ is the *j*th equation variable at the *i*th iteration and

$$\hat{\mathbf{y}}^i = (y_1^{i+1}, y_2^{i+1}, y_3^{i+1}, \ldots, y_{j-1}^{i+1}, y_j^i, y_{j+1}^i, \ldots, y_g^i)'$$

If the model is recursive, and if the equations are in recursive order, the Seidel method converges at once. If the model is block-recursive, the Seidel method might converge faster if the equations are grouped by block and the blocks are placed in block-recursive order. The BLOCK option can be used to determine the block-recursive form.

Jacobi and Seidel Methods with General Form Equations

Jacobi and Seidel solution methods support general form equations.

There are two cases where derivatives are (automatically) computed. The first case is for equations with the solution variable on the right-hand side and on the left-hand side of the equation

$$y^i = f(\mathbf{x}, y^i)$$

In this case the derivative of ERROR.y with respect to y is computed, and the new y approximation is computed as

$$y^{i+1} = y^i - \frac{f(\mathbf{x}, y^i) - y^i}{\partial(f(\mathbf{x}, y^i) - y^i)/\partial y}$$

The second case is a system of equations that contains one or more EQ.var equations. In this case, the MODEL procedure assigns a unique solution variable to each equation if such an assignment exists. Use the DETAILS option in the SOLVE statement to print a listing of the assigned variables.

Once the assignment is made, the new y approximation is computed as

$$y^{i+1} = y^i - \frac{f(\mathbf{x}, \mathbf{y}^i) - y^i}{\partial(f(\mathbf{x}, \mathbf{y}^i) - y^i)/\partial y}$$

If k is the number of general form equations, then k derivatives are required.

The convergence properties of the Jacobi and Seidel solution methods remain significantly poorer than the default Newton's method.

Comparison of Methods

Newton's method is the default and should work better than the others for most small- to medium-sized models. The Seidel method is always faster than the Jacobi for recursive models with equations in recursive order. For very large models and some highly nonlinear smaller models, the Jacobi or Seidel methods can sometimes be faster. Newton's method uses more memory than the Jacobi or Seidel methods.

Both the Newton's method and the Jacobi method are order-invariant in the sense that the order in which equations are specified in the model program has no effect on the operation of the iterative solution process. In order-invariant methods, the values of the solution variables are fixed for the entire execution of the model program. Assignments to model variables are automatically changed to assignments to corresponding equation variables. Only after the model program has completed execution are the results used to compute the new solution values for the next iteration.

Troubleshooting Problems

In solving a simultaneous nonlinear dynamic model you might encounter some of the following problems.

Missing Values

For SOLVE tasks, there can be no missing parameter values. Missing right-hand-side variables result in missing left-hand-side variables for that observation.

Unstable Solutions

A solution might exist but be unstable. An unstable system can cause the Jacobi and Seidel methods to diverge.

Explosive Dynamic Systems

A model might have well-behaved solutions at each observation but be dynamically unstable. The solution might oscillate wildly or grow rapidly with time.

Propagation of Errors

During the solution process, solution variables can take on values that cause computational errors. For example, a solution variable that appears in a LOG function might be positive at the solution but might be given a negative value during one of the iterations. When computational errors occur, missing values are generated and propagated, and the solution process might collapse.

Convergence Problems

The following items can cause convergence problems:

- There are illegal function values (for example $\sqrt{-1}$).
- There are local minima in the model equation.
- No solution exists.

- Multiple solutions exist.
- Initial values are too far from the solution.
- The CONVERGE= value is too small.

When PROC MODEL fails to find a solution to the system, the current iteration information and the program data vector are printed. The simulation halts if actual values are not available for the simulation to proceed. Consider the following program, which produces the output shown in Figure 26.85:

```
data test1;
   do t=1 to 50;
      x1 = sqrt(t) ;
      y = .;
      output;
   end;

proc model data=test1;
   exogenous x1 ;
   control a1 -1 b1 -29 c1 -4 ;
   y = a1 * sqrt(y) + b1 * x1 * x1 + c1 * lag(x1);
   solve y / out=sim forecast dynamic ;
run;
```

Figure 26.85 SOLVE Convergence Problems

The MODEL Procedure
Dynamic Single-Equation Forecast

Error: Could not reduce norm of residuals in 10 subiterations.

Error: The solution failed because 1 equations are missing or have extreme values for observation 1 at NEWTON iteration 1.

Note: Additional information on the values of the variables at this observation, which may be helpful in determining the cause of the failure of the solution process, is printed below.

Observation	1	**Iteration**	1	**CC**	-1.000000		
		Missing	1				

Iteration Errors - Missing.

The MODEL Procedure
Dynamic Single-Equation Forecast

--- Listing of Program Data Vector ---

_N_:	12	ACTUAL.x1:	1.41421	ACTUAL.y:	.
ERROR.y:	.	PRED.y:	.	a1:	-1
b1:	-29	c1:	-4	x1:	1.41421
y:	-0.00109				
@PRED.y/@y:	.	@ERROR.y/@y:	.		

Figure 26.85 *continued*

Note: Check for missing input data or uninitialized lags.

(Note that the LAG and DIF functions return missing values for the initial lag starting observations. This is a change from the 1982 and earlier versions of SAS/ETS which returned zero for uninitialized lags.)

Note: Simulation aborted.

At the first observation, a solution to the following equation is attempted:

$$y = -\sqrt{y} - 62$$

There is no solution to this problem. The iterative solution process got as close as it could to making Y negative while still being able to evaluate the model. This problem can be avoided in this case by altering the equation.

In other models, the problem of missing values can be avoided by either altering the data set to provide better starting values for the solution variables or by altering the equations.

You should be aware that, in general, a nonlinear system can have any number of solutions and the solution found might not be the one that you want. When multiple solutions exist, the solution that is found is usually determined by the starting values for the iterations. If the value from the input data set for a solution variable is missing, the starting value for it is taken from the solution of the last period (if nonmissing) or else the solution estimate is started at 0.

Iteration Output

The iteration output, produced by the ITPRINT option, is useful in determining the cause of a convergence problem. The ITPRINT option forces the printing of the solution approximation and equation errors at each iteration for each observation. A portion of the ITPRINT output from the following statements is shown in Figure 26.86.

```
proc model data=test1;
   exogenous x1 ;
   control a1 -1 b1 -29 c1 -4 ;
   y = a1 * sqrt(abs(y)) + b1 * x1 * x1 + c1 * lag(x1);
   solve y / out=sim forecast dynamic itprint;
run;
```

For each iteration, the equation with the largest error is listed in parentheses after the Newton convergence criteria measure. From this output you can determine which equation or equations in the system are not converging well.

Figure 26.86 SOLVE, ITPRINT Output

The MODEL Procedure
Dynamic Single-Equation Forecast

Observation 1 **Iteration** 0 **CC** 613961.39 **ERROR.y** -62.01010

Predicted Values
y
0.0001000

Figure 26.86 *continued*

Iteration Errors
y
-62.01010

Observation 1	Iteration 1	CC 50.902771	ERROR.y -61.88684

Predicted Values
y
-1.215784

Iteration Errors
y
-61.88684

Observation 1	Iteration 2	CC 0.364806	ERROR.y 41.752112

Predicted Values
y
-114.4503

Iteration Errors
y
41.75211

Numerical Integration

The differential equation system is numerically integrated to obtain a solution for the derivative variables at each data point. The integration is performed by evaluating the provided model at multiple points between each data point. The integration method used is a variable order, variable step-size backward difference scheme; for more detailed information, see Aiken (1985); Byrne and Hindmarsh (1975). The step size or time step is chosen to satisfy a *local truncation error* requirement. The term *truncation error* comes from the fact that the integration scheme uses a truncated series expansion of the integrated function to do the integration. Because the series is truncated, the integration scheme is within the truncation error of the true value.

To further improve the accuracy of the integration, the total integration time is broken up into small intervals (time steps or step sizes), and the integration scheme is applied to those intervals. The integration at each time step uses the values computed at the previous time step so that the truncation error tends to accumulate. It is usually not possible to estimate the global error with much precision. The best that can be done is to monitor and to control the local truncation error, which is the truncation error committed at each time step relative to

$$d = \max_{0 \le t \le T} (\|y(t)\|_\infty, 1)$$

where $y(t)$ is the integrated variable. Furthermore, the $y(t)$s are dynamically scaled to within two orders of magnitude of one to keep the error monitoring well-behaved.

The local truncation error requirement defaults to 1.0E–9. You can specify the LTEBOUND= option to modify that requirement. The LTEBOUND= option is a relative measure of accuracy, so a value smaller than 1.0E–10 is usually not practical. A larger bound increases the speed of the simulation and estimation but decreases the accuracy of the results. If the LTEBOUND= option is set too small, the integrator is not able to take time steps small enough to satisfy the local truncation error requirement and still have enough machine precision to compute the results. Since the integrations are scaled to within 1.0E–2 of one, the simulated values should be correct to at least seven decimal places.

There is a default minimum time step of 1.0E–14. This minimum time step is controlled by the MINTIMESTEP= option and the machine epsilon. If the minimum time step is smaller than the machine epsilon times the final time value, the minimum time step is increased automatically.

For the points between each observation in the data set, the values for nonintegrated variables in the data set are obtained from a linear interpolation from the two closest points. Lagged variables can be used with integrations, but their values are discrete and are not interpolated between points. Lagging, therefore, can then be used to input step functions into the integration.

The derivatives necessary for estimation (the gradient with respect to the parameters) and goal seeking (the Jacobian) are computed by numerically integrating analytical derivatives. The accuracy of the derivatives is controlled by the same integration techniques mentioned previously.

Limitations

There are limitations to the types of differential equations that can be solved or estimated. One type is an explosive differential equation (finite escape velocity) for which the following differential equation is an example:

$$y' = a \times y, a > 0$$

If this differential equation is integrated too far in time, y exceeds the maximum value allowed on the computer, and the integration terminates.

Likewise, differential systems that are singular cannot be solved or estimated in general. For example, consider the following differential system:

$$\begin{aligned} x' &= -y' + 2x + 4y + \exp(t) \\ y' &= -x' + y + \exp(4*t) \end{aligned}$$

This system has an analytical solution, but an accurate numerical solution is very difficult to obtain. The reason is that y' and x' cannot be isolated on the left-hand side of the equation. If the equation is modified slightly to

$$\begin{aligned} x' &= -y' + 2x + 4y + \exp(t) \\ y' &= x' + y + \exp(4t) \end{aligned}$$

the system is nonsingular, but the integration process could still fail or be extremely slow. If the MODEL procedure encounters either system, a warning message is issued.

This system can be rewritten as the following recursive system, which can be estimated and simulated successfully with the MODEL procedure:

$$\begin{aligned} x' &= 0.5y + 0.5\exp(4t) + x + 1.5y - 0.5\exp(t) \\ y' &= x' + y + \exp(4t) \end{aligned}$$

Petzold (1982) mentions a class of differential algebraic equations that, when integrated numerically, could produce incorrect or misleading results. An example of such a system is

$$\begin{aligned} y_2'(t) &= y_1(t) + g_1(t) \\ 0 &= y_2(t) + g_2(t) \end{aligned}$$

The analytical solution to this system depends on g and its derivatives at the current time only and not on its initial value or past history. You should avoid systems of this and other similar forms mentioned in Petzold (1982).

SOLVE Data Sets

SDATA= Input Data Set

The SDATA= option reads a cross-equation covariance matrix from a data set. The covariance matrix read from the SDATA= data set specified in the SOLVE statement is used to generate random equation errors when the RANDOM= option specifies Monte Carlo simulation.

Typically, the SDATA= data set is created by the OUTS= option in a previous FIT statement. (The OUTS= data set from a FIT statement can be read back in by a SOLVE statement in the same PROC MODEL step.)

You can create an input SDATA= data set by using the DATA step. PROC MODEL expects to find a character variable _NAME_ in the SDATA= data set as well as variables for the equations in the estimation or solution. For each observation with a _NAME_ value that matches the name of an equation, PROC MODEL fills the corresponding row of the **S** matrix with the values of the names of equations found in the data set. If a row or column is omitted from the data set, an identity matrix row or column is assumed. Missing values are ignored. Since the **S** matrix is symmetric, you can include only a triangular part of the **S** matrix in the SDATA= data set with the omitted part indicated by missing values. If the SDATA= data set contains multiple observations with the same _NAME_, the last values supplied for the _NAME_ variable are used. The section "OUTS= Data Set" on page 1634 contains more details on the format of this data set.

Use the TYPE= option to specify the type of estimation method used to produce the **S** matrix you want to input.

ESTDATA= Input Data Set

The ESTDATA= option specifies an input data set that contains an observation with values for some or all of the model parameters. It can also contain observations with the rows of a covariance matrix for the parameters.

When the ESTDATA= option is used, parameter values are set from the first observation. If the RANDOM= option is used and the ESTDATA= data set contains a covariance matrix, the covariance matrix of the

parameter estimates is read and used to generate pseudo-random shocks to the model parameters for Monte Carlo simulation. These random perturbations have a multivariate normal distribution with the covariance matrix read from the ESTDATA= data set.

The ESTDATA= data set is usually created by the OUTEST= option in a FIT statement. The OUTEST= data set contains the parameter estimates produced by the FIT statement and also contains the estimated covariance of the parameter estimates if the OUTCOV option is used. This OUTEST= data set can be read in by the ESTDATA= option in a SOLVE statement.

You can also create an ESTDATA= data set with a SAS DATA step program. The data set must contain a numeric variable for each parameter to be given a value or covariance column. The name of the variable in the ESTDATA= data set must match the name of the parameter in the model. Parameters with names longer than 32 characters cannot be set from an ESTDATA= data set. The data set must also contain a character variable _NAME_ of length 32. _NAME_ has a blank value for the observation that gives values to the parameters. _NAME_ contains the name of a parameter for observations that define rows of the covariance matrix.

More than one set of parameter estimates and covariances can be stored in the ESTDATA= data set if the observations for the different estimates are identified by the variable _TYPE_. _TYPE_ must be a character variable of length eight. The TYPE= option is used to select for input the part of the ESTDATA= data set for which the value of the _TYPE_ variable matches the value of the TYPE= option.

OUT= Data Set

The OUT= data set contains solution values, residual values, and actual values of the solution variables.

The OUT= data set contains the following variables:

- BY variables
- RANGE variable
- ID variables
- _TYPE_, a character variable of length eight that identifies the type of observation. The _TYPE_ variable can be PREDICT, RESIDUAL, ACTUAL, or ERROR.
- _MODE_, a character variable of length eight that identifies the solution mode. _MODE_ takes the value FORECAST or SIMULATE.
- if lags are used, a numeric variable, _LAG_, that contains the number of dynamic lags that contribute to the solution. The value of _LAG_ is always zero for STATIC mode solutions. _LAG_ is set to a missing value for lag-starting observations.
- if the RANDOM= option is used, _REP_, a numeric variable that contains the replication number. For example, if RANDOM=10, each input observation results in eleven output observations with _REP_ values 0 through 10. The observations with _REP_=0 are from the unperturbed solution. (The random-number generator functions are suppressed, and the parameter and endogenous perturbations are zero when _REP_=0.)
- _ERRORS_, a numeric variable that contains the number of errors that occurred during the execution of the program for the last iteration for the observation. If the solution failed to converge, this is counted as one error, and the _ERRORS_ variable is made negative.

- solution and other variables. The solution variables contain solution or predicted values for _TYPE_=PREDICT observations, residuals for _TYPE_=RESIDUAL observations, or actual values for _TYPE_=ACTUAL observations. The other model variables, and any other variables read from the input data set, are always actual values from the input data set.
- any other variables named in the OUTVARS statement. These can be program variables computed by the model program, CONTROL variables, parameters, or special variables in the model program. Compound variable names longer than 32 characters are truncated in the OUT= data set.

By default, only the predicted values are written to the OUT= data set. The OUTRESID, OUTACTUAL, and OUTERROR options are used to add the residual, actual, and ERROR. values, respectively, to the data set.

For examples of the OUT= data set, see Example 26.6.

DATA= Input Data Set

The input data set should contain all of the exogenous variables and should supply nonmissing values for them for each period to be solved.

Solution variables can be supplied in the input data set and are used as follows:

- to supply initial lags. For example, if the lag length of the model is three, three observations are read in to feed the lags before any solutions are computed.
- to evaluate the goodness of fit. Goodness-of-fit measures are computed based on the difference between the solved values and the actual values supplied from the data set.
- to supply starting values for the iterative solution. If the value from the input data set for a solution variable is missing, the starting value for it is taken from the solution of the last period (if nonmissing) or else the solution estimate is started at zero.
- for STATIC mode solutions, actual values from the data set are used by the lagging functions for the solution variables.
- for FORECAST mode solutions, actual values from the data set are used as the solution values when nonmissing.

Programming Language Overview: MODEL Procedure

Variables in the Model Program

Variable names are alphanumeric but must start with a letter. The length is limited to 32 characters.

PROC MODEL uses several classes of variables, and different variable classes are treated differently. The variable class is controlled by *declaration statements*: the VAR, ENDOGENOUS, and EXOGENOUS statements for model variables, the PARAMETERS statement for parameters, and the CONTROL statement

for control class variables. These declaration statements have several valid abbreviations. Various *internal variables* are also made available to the model program to allow communication between the model program and the procedure. RANGE, ID, and BY variables are also available to the model program. Those variables not declared as any of the preceding classes are *program variables*.

Some classes of variables can be lagged; that is, their value at each observation is remembered, and previous values can be referred to by the lagging functions. Other classes have only a single value and are not affected by lagging functions. For example, parameters have only one value and are not affected by lagging functions; therefore, if P is a parameter, DIF*n* (P) is always 0, and LAG*n* (P) is always the same as P for all values of *n*.

The different variable classes and their roles in the model are described in the following.

Model Variables

Model variables are declared by VAR, ENDOGENOUS, or EXOGENOUS statements, or by FIT and SOLVE statements. The model variables are the variables that the model is intended to explain or predict.

PROC MODEL enables you to use expressions on the left-hand side of the equal sign to define model equations. For example, a log-linear model for Y can be written as

```
log( y ) = a + b * x;
```

Previously, only a variable name was allowed on the left-hand side of the equal sign.

The text on the left-hand side of the equation serves as the equation name used to identify the equation in printed output, in the OUT= data sets, and in FIT or SOLVE statements. To refer to equations specified by using left-hand side expressions (in the FIT statement, for example), place the left-hand side expression in quotes. For example, the following statements fit a log-linear model to the dependent variable Y:

```
proc model data=in;
   log( y ) = a + b * x;
   fit "log(y)";
run;
```

The estimation and simulation is performed by transforming the models into general form equations. No actual or predicted value is available for general form equations, so no R^2 or adjusted R^2 is computed.

Equation Variables

An equation variable is one of several special variables used by PROC MODEL to control the evaluation of model equations. An equation variable name consists of one of the prefixes EQ, RESID, ERROR, PRED, or ACTUAL, followed by a period and the name of a model equation.

Equation variable names can appear in parts of the PROC MODEL printed output, and they can be used in the model program. For example, RESID-prefixed variables can be used in LAG functions to define equations with moving-average error terms. See the section "Autoregressive Moving-Average Error Processes" on page 1610 for details.

The meaning of these prefixes is detailed in the section "Equation Translations" on page 1677.

Parameters

Parameters are variables that have the same value for each observation. Parameters can be given values or can be estimated by fitting the model to data. During the SOLVE stage, parameters are treated as constants. If no estimation is performed, the SOLVE stage uses the initial value provided in the ESTDATA= data set, the MODEL= file, or in the PARAMETER statement, as the value of the parameter.

The PARAMETERS statement declares the parameters of the model. Parameters are not lagged, and they cannot be changed by the model program.

Control Variables

Control variables supply constant values to the model program that can be used to control the model in various ways. The CONTROL statement declares control variables and specifies their values. A control variable is like a parameter except that it has a fixed value and is not estimated from the data.

Control variables are not reinitialized before each pass through the data and can thus be used to retain values between passes. You can use control variables to vary the program logic. Control variables are not affected by lagging functions.

For example, if you have two versions of an equation for a variable Y, you could put both versions in the model and, by using a CONTROL statement to select one of them, produce two different solutions to explore the effect the choice of equation has on the model, as shown in the following statements:

```
select (case);
   when (1) y =  ...first version of equation... ;
   when (2) y =  ...second version of equation... ;
end;

control case 1;
solve / out=case1;
run;

control case 2;
solve / out=case2;
run;
```

RANGE, ID, and BY Variables

The RANGE statement controls the range of observations in the input data set that is processed by PROC MODEL. The ID statement lists variables in the input data set that are used to identify observations in the printout and in the output data set. The BY statement can be used to make PROC MODEL perform a separate analysis for each BY group. The variable in the RANGE statement, the ID variables, and the BY variables are available for the model program to examine, but their values should not be changed by the program. The BY variables are not affected by lagging functions.

Internal Variables

You can use several internal variables in the model program to communicate with the procedure. For example, if you want PROC MODEL to list the values of all the variables when more than 10 iterations are performed and the procedure is past the 20th observation, you can write

```
if _obs_ > 20 then if _iter_ > 10 then _list_ = 1;
```

Internal variables are not affected by lagging functions, and they cannot be changed by the model program except as noted. The following internal variables are available. The variables are all numeric except where noted.

ERRORS is a flag that is set to 0 at the start of program execution and is set to a nonzero value whenever an error occurs. The program can also set the _ERRORS_ variable.

ITER is the iteration number. For FIT tasks, the value of _ITER_ is negative for preliminary grid-search passes. The iterative phase of the estimation starts with iteration 0. After the estimates have converged, a final pass is made to collect statistics with _ITER_ set to a missing value. Note that at least one pass, and perhaps several subiteration passes as well, is made for each iteration. For SOLVE tasks, _ITER_ counts the iterations used to compute the simultaneous solution of the system.

LAG is the number of dynamic lags that contribute to the solution at the current observation. _LAG_ is always 0 for FIT tasks and for STATIC solutions. _LAG_ is set to a missing value during the lag starting phase.

LIST is a list flag that is set to 0 at the start of program execution. The program can set _LIST_ to a nonzero value to request a listing of the values of all the variables in the program after the program has finished executing.

METHOD is the solution method in use for SOLVE tasks. _METHOD_ is set to a blank value for FIT tasks. _METHOD_ is a character-valued variable. Values are NEWTON, JACOBI, SIEDEL, or ONEPASS.

MODE takes the value ESTIMATE for FIT tasks and the value SIMULATE or FORECAST for SOLVE tasks. _MODE_ is a character-valued variable.

NMISS is the number of missing or otherwise unusable observations during the model estimation. For FIT tasks, _NMISS_ is initially set to 0; at the start of each iteration, _NMISS_ is set to the number of unusable observations for the previous iteration. For SOLVE tasks, _NMISS_ is set to a missing value.

NUSED is the number of nonmissing observations used in the estimation. For FIT tasks, PROC MODEL initially sets _NUSED_ to the number of parameters; at the start of each iteration, _NUSED_ is reset to the number of observations used in the previous iteration. For SOLVE tasks, _NUSED_ is set to a missing value.

OBS counts the observations being processed. _OBS_ is negative or 0 for observations in the lag starting phase.

REP is the replication number for Monte Carlo simulation when the RANDOM= option is specified in the SOLVE statement. _REP_ is 0 when the RANDOM= option is not used and for FIT tasks. When _REP_=0, the random-number generator functions always return 0.

WEIGHT is the weight of the observation. For FIT tasks, _WEIGHT_ provides a weight for the observation in the estimation. _WEIGHT_ is initialized to 1.0 at the start of execution for FIT tasks. For SOLVE tasks, _WEIGHT_ is ignored.

Program Variables

Variables not in any of the other classes are called program variables. Program variables are used to hold intermediate results of calculations. Program variables are reinitialized to missing values before each observation is processed. Program variables can be lagged. The RETAIN statement can be used to give program variables initial values and enable them to keep their values between observations.

Character Variables

PROC MODEL supports both numeric and character variables. Character variables are not involved in the model specification but can be used to label observations, to write debugging messages, or for documentation purposes. All variables are numeric unless they are the following.

- character variables in a DATA= SAS data set
- program variables assigned a character value
- declared to be character by a LENGTH or ATTRIB statement

Equation Translations

Equations written in normalized form are always automatically converted to general form equations. For example, when a normalized form equation such as

```
y = a + b*x;
```

is encountered, it is translated into the equations

```
PRED.y = a + b*x;
RESID.y = PRED.y - ACTUAL.y;
ERROR.y = PRED.y - y;
```

If the same system is expressed as the following general form equation, then this equation is used unchanged.

```
EQ.y = y - (a + b*x);
```

This makes it easy to solve for arbitrary variables and to modify the error terms for autoregressive or moving average models.

Use the LIST option to see how this transformation is performed. For example, the following statements produce the listing shown in Figure 26.87.

```
proc model data=line list;
   y = a1 + b1*x1 + c1*x2;
   fit y;
run;
```

Figure 26.87 LIST Output

The MODEL Procedure

Listing of Compiled Program Code		
Stmt	Line:Col	Statement as Parsed
1	4310:4	PRED.y = a1 + b1 * x1 + c1 * x2;
1	4310:4	RESID.y = PRED.y - ACTUAL.y;
1	4310:4	ERROR.y = PRED.y - y;

PRED.Y is the predicted value of Y, and ACTUAL.Y is the value of Y in the data set. The predicted value minus the actual value, RESID.Y, is then the error term, ϵ, for the original Y equation. Note that the residuals obtained from the OUTRESID option in the OUT=dataset for both the FIT and SOLVE statements are defined as $actual - predicted$, the negative of RESID.Y. See the section "Syntax: MODEL Procedure" on page 1489 for details. ACTUAL.Y and Y have the same value for parameter estimation. For solve tasks, ACTUAL.Y is still the value of Y in the data set but Y becomes the solved value; the value that satisfies PRED.Y – Y = 0.

The following are the equation variable definitions.

EQ. The value of an EQ.-prefixed equation variable (normally used to define a general form equation) represents the failure of the equation to hold. When the EQ.*name* variable is 0, the *name* equation is satisfied.

RESID. The RESID.*name* variables represent the stochastic parts of the equations and are used to define the objective function for the estimation process. A RESID.-prefixed equation variable is like an EQ.-prefixed variable but makes it possible to use or transform the stochastic part of the equation. The RESID. equation is used in place of the ERROR. equation for model solutions if it has been reassigned or used in the equation.

ERROR. An ERROR.*name* variable is like an EQ.-prefixed variable, except that it is used only for model solution and does not affect parameter estimation.

PRED. For a normalized form equation (specified by assignment to a model variable), the PRED.*name* equation variable holds the predicted value, where *name* is the name of both the model variable and the corresponding equation. (PRED.-prefixed variables are not created for general form equations.)

ACTUAL. For a normalized form equation (specified by assignment to a model variable), the ACTUAL.*name* equation variable holds the value of the *name* model variable read from the input data set.

DERT. The DERT.*name* variable defines a differential equation. Once defined, it might be used on the right-hand side of another equation.

H. The H.*name* variable specifies the functional form for the variance of the named equation.

GMM_H. This is created for H.*vars* and is the moment equation for the variance for GMM. This variable is used only for GMM.

```
GMM_H.name = RESID.name**2 - H.name;
```

MSE. The MSE.*y* variable contains the value of the mean squared error for *y* at each iteration. An MSE. variable is created for each dependent/endogenous variable in the model. These

variables can be used to specify the missing lagged values in the estimation and simulation of GARCH type models.

```
demret = intercept ;
h.demret = arch0 +
           arch1 * xlag( resid.demret ** 2, mse.demret) +
           garch1 * xlag(h.demret, mse.demret) ;
```

NRESID. This is created for H.*vars* and is the normalized residual of the variable <*name* >. The formula is

```
NRESID.name = RESID.name/ sqrt(H.name);
```

The three equation variable prefixes, RESID., ERROR., and EQ. allow for control over the objective function for the FIT, the SOLVE, or both the FIT and the SOLVE stages. For FIT tasks, PROC MODEL looks first for a RESID.*name* variable for each equation. If defined, the RESID.-prefixed equation variable is used to define the objective function for the parameter estimation process. Otherwise, PROC MODEL looks for an EQ.-prefixed variable for the equation and uses it instead.

For SOLVE tasks, PROC MODEL looks first for an ERROR.*name* variable for each equation. If defined, the ERROR.-prefixed equation variable is used for the solution process. Otherwise, PROC MODEL looks for an EQ.-prefixed variable for the equation and uses it instead. To solve the simultaneous equation system, PROC MODEL computes values of the solution variables (the model variables being solved for) that make all of the ERROR.name and EQ.*name* variables close to 0.

Derivatives

Nonlinear modeling techniques require the calculation of derivatives of certain variables with respect to other variables. The MODEL procedure includes an analytic differentiator that determines the model derivatives and generates program code to compute these derivatives. When parameters are estimated, the MODEL procedure takes the derivatives of the equation with respect to the parameters. When the model is solved, Newton's method requires the derivatives of the equations with respect to the variables solved for.

PROC MODEL uses exact mathematical formulas for derivatives of non-user-defined functions. For other functions, numerical derivatives are computed and used.

The differentiator differentiates the entire model program, including the conditional logic and flow of control statements. Delayed definitions, as when the LAG of a program variable is referred to before the variable is assigned a value, are also differentiated correctly.

The differentiator includes optimization features that produce efficient code for the calculation of derivatives. However, when flow of control statements such as GOTO statements are used, the optimization process is impeded, and less efficient code for derivatives might be produced. Optimization is also reduced by conditional statements, iterative DO loops, and multiple assignments to the same variable.

The table of derivatives is printed with the LISTDER option. The code generated for the computation of the derivatives is printed with the LISTCODE option.

Derivative Variables

When the differentiator needs to generate code to evaluate the expression for the derivative of a variable, the result is stored in a special derivative variable. Derivative variables are not created when the derivative expression reduces to a previously computed result, a variable, or a constant. The names of derivative variables, which might sometimes appear in the printed output, have the form @*obj* /@*wrt*, where *obj* is the variable whose derivative is being taken and *wrt* is the variable that the differentiation is with respect to. For example, the derivative variable for the derivative of *Y* with respect to *X* is named @*Y*/@*X*.

The derivative variables can be accessed or used as part of the model program using the GETDER() function.

GETDER(*x*, *a*) the derivative of *x* with respect to *a*.

GETDER(*x*, *a*, *b*) the second derivative of *x* with respect to *a* and *b*.

The main purpose of the GETDER() function is for surfacing the derivatives so they can be stored in a data set for further processing. Only derivatives that are implied by the problem are available to the GETDER() function. When derivatives are requested that aren't already created, a missing value will be returned. The derivative of the GETDER() function is always zero so the results of the GETDER() function shouldn't be used in any of the equations in the FIT or the SOLVE statement.

The following example adds the gradient of the PRED.*y* value with respect to the parameters to the OUT= data set.

```
proc model data=line ;
   y = a1 + b1**2 *x1 + c1*x2;
   Dy_a1 = getder(PRED.y,a1);
   Dy_b1 = getder(PRED.y,b1);
   Dy_c1 = getder(PRED.y,c1);
   outvars Dy_a1 Dy_b1 Dy_c1;
   fit y / out=grad;
run;
```

Mathematical Functions

The following is a brief summary of SAS functions that are useful for defining models. Additional functions and details are in *SAS Language: Reference*. Information about creating new functions can be found in *SAS/BASE Software: Procedure Reference*, Chapter 18, "The FCMP Procedure."

ABS(*x*) the absolute value of *x*

ARCOS(*x*) the arccosine in radians of *x*; *x* should be between −1 and 1.

ARSIN(*x*) the arcsine in radians of *x*; *x* should be between −1 and 1.

ATAN(*x*) the arctangent in radians of *x*

COS(*x*) the cosine of *x*; *x* is in radians.

COSH(*x*) the hyperbolic cosine of *x*

EXP(*x*) e^x

LOG(*x*) the natural logarithm of *x*

LOG10(x) the log base ten of x

LOG2(x) the log base two of x

SIN(x) the sine of x; x is in radians.

SINH(x) the hyperbolic sine of x

SQRT(x) the square root of x

TAN(x) the tangent of x; x is in radians and is not an odd multiple of $\pi/2$.

TANH(x) the hyperbolic tangent of x

Random-Number Functions

The MODEL procedure provides several functions for generating random numbers for Monte Carlo simulation. These functions use the same generators as the corresponding SAS DATA step functions.

The following random number functions are supported: RANBIN, RANCAU, RAND, RANEXP, RANGAM, RANNOR, RANPOI, RANTBL, RANTRI, and RANUNI. For more information, refer to *SAS Language: Reference*.

Each reference to a random number function sets up a separate pseudo-random sequence. Note that this means that two calls to the same random function with the same seed produce identical results. This is different from the behavior of the random number functions used in the SAS DATA step. For example, the following statements produce identical values for X and Y, but Z is from an independent pseudo-random sequence:

```
x=rannor(123);
y=rannor(123);
z=rannor(567);
q=rand('BETA', 1, 12 );
```

For FIT tasks, all random number functions always return 0. For SOLVE tasks, when Monte Carlo simulation is requested, a random number function computes a new random number on the first iteration for an observation (if it is executed on that iteration) and returns that same value for all later iterations of that observation. When Monte Carlo simulation is not requested, random number functions always return 0.

Functions across Time

PROC MODEL provides four types of special built-in functions that refer to the values of variables and expressions in previous time periods. These functions have the following forms where n represents the number of periods, x is any expression, and the argument i is a variable or expression that gives the lag length ($0 <= i <= n$). If the index value i is omitted, the maximum lag length n is used.

LAGn ($< i, > x$) returns the ith lag of x, where n is the maximum lag;

DIFn (x) is the difference of x at lag n

ZLAGn ($< i, > x$) returns the ith lag of x, where n is the maximum lag, with missing lags replaced with zero

XLAG*n* (*x*, *y*) returns the *n*th lag of *x* if *x* is nonmissing, or *y* if x is missing

ZDIF*n* (*x*) is the difference with lag length truncated and missing values converted to zero; *x* is the variable or expression to compute the moving average of

MOVAVG*n*(*x*) is the moving average if X_t denotes the observation at time point *t*, to ensure compatibility with the number *n* of observations used to calculate the moving average MOVAVGn, the following definition is used:

$$MOVAVGn(X_t) = \frac{X_t + X_{t-1} + X_{t-2} + \ldots + X_{t-n+1}}{n}$$

The moving average calculation for SAS 9.1 and earlier releases is as follows:

$$MOVAVGn(X_t) = \frac{X_t + X_{t-1} + X_{t-2} + \ldots + X_{t-n}}{n+1}$$

Missing values of *x* are omitted in computing the average.

If you do not specify *n*, the number of periods is assumed to be one. For example, LAG(X) is the same as LAG1(X). No more than four digits can be used with a lagging function; that is, LAG9999 is the greatest LAG function, ZDIF9999 is the greatest ZDIF function, and so on.

The LAG functions get values from previous observations and make them available to the program. For example, LAG(X) returns the value of the variable X as it was computed in the execution of the program for the preceding observation. The expression LAG2(X+2*Y) returns the value of the expression X+2*Y, computed by using the values of the variables X and Y that were computed by the execution of the program for the observation two periods ago.

The DIF functions return the difference between the current value of a variable or expression and the value of its LAG. For example, DIF2(X) is a short way of writing X–LAG2(X), and DIF15(SQRT(2*Z)) is a short way of writing SQRT(2*Z)–LAG15(SQRT(2*Z)).

The ZLAG and ZDIF functions are like the LAG and DIF functions, but they are not counted in the determination of the program lag length, and they replace missing values with 0s. The ZLAG function returns the lagged value if the lagged value is nonmissing, or 0 if the lagged value is missing. The ZDIF function returns the differenced value if the differenced value is nonmissing, or 0 if the value of the differenced value is missing. The ZLAG function is especially useful for models with ARMA error processes. See the next section for details.

Lag Logic

The LAG and DIF lagging functions in the MODEL procedure are different from the queuing functions with the same names in the DATA step. Lags are determined by the final values that are set for the program variables by the execution of the model program for the observation. This can have upsetting consequences for programs that take lags of program variables that are given different values at various places in the program, as shown in the following statements:

```
temp = x + w;
t    = lag( temp );
temp = q - r;
s    = lag( temp );
```

The expression LAG(TEMP) always refers to LAG(Q–R), never to LAG(X+W), since Q–R is the final value assigned to the variable TEMP by the model program. If LAG(X+W) is wanted for T, it should be computed as T=LAG(X+W) and not T=LAG(TEMP), as in the preceding example.

Care should also be exercised in using the DIF functions with program variables that might be reassigned later in the program. For example, the program

```
temp =  x ;
s    = dif( temp );
temp = 3 * y;
```

computes values for S equivalent to

```
s =  x  - lag( 3 * y );
```

Note that in the preceding examples, TEMP is a program variable, *not* a model variable. If it were a model variable, the assignments to it would be changed to assignments to a corresponding equation variable.

Note that whereas LAG1(LAG1(X)) is the same as LAG2(X), DIF1(DIF1(X)) is *not* the same as DIF2(X). The DIF2 function is the difference between the current period value at the point in the program where the function is executed and the final value at the end of execution two periods ago; DIF2 is not the second difference. In contrast, DIF1(DIF1(X)) is equal to DIF1(X)-LAG1(DIF1(X)), which equals X–2*LAG1(X)+LAG2(X), which is the second difference of X.

More information about the differences between PROC MODEL and the DATA step LAG and DIF functions is found in Chapter 4, "Working with Time Series Data."

Lag Lengths

The lag length of the model program is the number of lags needed for any relevant equation. The program lag length controls the number of observations used to initialize the lags.

PROC MODEL keeps track of the use of lags in the model program and automatically determines the lag length of each equation and of the model as a whole. PROC MODEL sets the program lag length to the maximum number of lags needed to compute any equation to be estimated, solved, or needed to compute any instrument variable used.

In determining the lag length, the ZLAG and ZDIF functions are treated as always having a lag length of 0. For example, if Y is computed as

```
y = lag2( x + zdif3( temp ) );
```

then Y has a lag length of 2 (regardless of how TEMP is defined). If Y is computed as

```
y = zlag2( x + dif3( temp ) );
```

then Y has a lag length of 0.

This is so that ARMA errors can be specified without causing the loss of additional observations to the lag starting phase and so that recursive lag specifications, such as moving-average error terms, can be used. Recursive lags are not permitted unless the ZLAG or ZDIF functions are used to truncate the lag length. For example, the following statement produces an error message:

```
t = a + b * lag( t );
```

The program variable T depends recursively on its own lag, and the lag length of T is therefore undefined.

In the following equation RESID.Y depends on the predicted value for the Y equation but the predicted value for the Y equation depends on the LAG of RESID.Y, and thus, the predicted value for the Y equation depends recursively on its own lag.

```
y = yhat + ma * lag( resid.y );
```

The lag length is infinite, and PROC MODEL prints an error message and stops. Since this kind of specification is allowed, the recursion must be truncated at some point. The ZLAG and ZDIF functions do this.

The following equation is valid and results in a lag length for the Y equation equal to the lag length of YHAT:

```
y = yhat + ma * zlag( resid.y );
```

Initially, the lags of RESID.Y are missing, and the ZLAG function replaces the missing residuals with 0s, their unconditional expected values.

The ZLAG0 function can be used to zero out the lag length of an expression. ZLAG0(x) returns the current period value of the expression x, if nonmissing, or else returns 0, and prevents the lag length of x from contributing to the lag length of the current statement.

Initializing Lags

At the start of each pass through the data set or BY group, the lag variables are set to missing values and an initialization is performed to fill the lags. During this phase, observations are read from the data set, and the model variables are given values from the data. If necessary, the model is executed to assign values to program variables that are used in lagging functions. The results for variables used in lag functions are saved. These observations are not included in the estimation or solution.

If, during the execution of the program for the lag starting phase, a lag function refers to lags that are missing, the lag function returns missing. Execution errors that occur while starting the lags are not reported unless requested. The modeling system automatically determines whether the program needs to be executed during the lag starting phase.

If L is the maximum lag length of any equation being fit or solved, then the first L observations are used to prime the lags. If a BY statement is used, the first L observations in the BY group are used to prime the lags. If a RANGE statement is used, the first L observations prior to the first observation requested in the RANGE statement are used to prime the lags. Therefore, there should be at least L observations in the data set.

Initial values for the lags of model variables can also be supplied in VAR, ENDOGENOUS, and EXOGENOUS statements. This feature provides initial lags of solution variables for dynamic solution when initial values for the solution variable are not available in the input data set. For example, the statement

```
var x 2 3 y 4 5 z 1;
```

feeds the initial lags exactly like these values in an input data set:

Lag	X	Y	Z
2	3	5	.
1	2	4	1

If initial values for lags are available in the input data set and initial lag values are also given in a declaration statement, the values in the VAR, ENDOGENOUS, or EXOGENOUS statements take priority.

The RANGE statement is used to control the range of observations in the input data set that are processed by PROC MODEL. In the following statement, '01jan1924' specifies the starting period of the range, and '01dec1943' specifies the ending period:

```
range date = '01jan1924'd to '01dec1943'd;
```

The observations in the data set immediately prior to the start of the range are used to initialize the lags.

Language Differences

For the most part, PROC MODEL programming statements work the same as they do in the DATA step as documented in *SAS Language: Reference*. However, there are several differences that should be noted.

DO Statement Differences

The DO statement in PROC MODEL does not allow a character index variable. Thus, the following DO statement is not valid in PROC MODEL, although it is supported in the DATA step:

```
do i = 'A', 'B', 'C';  /* invalid PROC MODEL code */
```

IF Statement Differences

The IF statement in PROC MODEL does not allow a character-valued condition. For example, the following IF statement is not supported by PROC MODEL:

```
if 'this' then  statement;
```

Comparisons of character values are supported in IF statements, so the following IF statement is acceptable:

```
if 'this' < 'that' then  statement;
```

PROC MODEL allows for embedded conditionals in expressions. For example the following two statements are equivalent:

```
flag = if time = 1 or time = 2 then conc+30/5 + dose*time
       else if time > 5 then (0=1) else (patient * flag);

if time = 1 or time = 2 then flag= conc+30/5 + dose*time;
else if time > 5 then flag=(0=1); else flag=patient*flag;
```

Note that the ELSE operator involves only the first object or token after it so that the following assignments are not equivalent:

```
total = if sum > 0 then sum else sum + reserve;
total = if sum > 0 then sum else (sum + reserve);
```

The first assignment makes TOTAL always equal to SUM plus RESERVE.

PUT Statement Differences

The PUT statement, mostly used in PROC MODEL for program debugging, supports only some of the features of the DATA step PUT statement. It also has some new features that the DATA step PUT statement does not support.

The PROC MODEL PUT statement does not support line pointers, factored lists, iteration factors, overprinting, the _INFILE_ option, or the colon (:) format modifier.

The PROC MODEL PUT statement does support expressions, but an expression must be enclosed in parentheses. For example, the following statement prints the square root of x:

```
put (sqrt(x));
```

Subscripted array names must be enclosed in parentheses. For example, the following statement prints the *i*th element of the array A:

```
put (a i);
```

However, the following statement is an error:

```
put a i;
```

The PROC MODEL PUT statement supports the print item _PDV_ to print a formatted listing of all the variables in the program. For example, the following statement prints a much more readable listing of the variables than does the _ALL_ print item:

```
put _pdv_;
```

To print all the elements of the array A, use the following statement:

```
put a;
```

To print all the elements of A with each value labeled by the name of the element variable, use the following statement:

```
put a=;
```

ABORT Statement Difference

In the MODEL procedure, the ABORT statement does not allow any arguments.

SELECT/WHEN/OTHERWISE Statement Differences

The WHEN and OTHERWISE statements allow more than one target statement. That is, DO groups are not necessary for multiple statement WHENs. For example in PROC MODEL, the following syntax is valid:

```
select;
   when(exp1)
      stmt1;
      stmt2;
   when(exp2)
      stmt3;
      stmt4;
end;
```

The ARRAY Statement

ARRAY *arrayname* < *{dimensions}* > < *$ [length]* > < *variables and constants* > *;* **;**

The ARRAY statement is used to associate a name with a list of variables and constants. The array name can then be used with subscripts in the model program to refer to the items in the list.

In PROC MODEL, the ARRAY statement does not support all the features of the DATA step ARRAY statement. Implicit indexing cannot be used; all array references must have explicit subscript expressions. Only exact array dimensions are allowed; lower-bound specifications are not supported. A maximum of six dimensions is allowed.

On the other hand, the ARRAY statement supported by PROC MODEL does allow both variables and constants to be used as array elements. You cannot make assignments to constant array elements. Both dimension specification and the list of elements are optional, but at least one must be supplied. When the list of elements is not given or fewer elements than the size of the array are listed, array variables are created by suffixing element numbers to the array name to complete the element list.

The following are valid PROC MODEL array statements:

```
array x[120];           /* array X of length 120            */
array q[2,2];           /* Two dimensional array Q          */
array b[4] va vb vc vd; /* B[2] = VB, B[4] = VD             */
array x x1-x30;         /* array X of length 30, X[7] = X7  */
array a[5] (1 2 3 4 5); /* array A initialized to 1,2,3,4,5 */
```

RETAIN Statement

RETAIN *variables initial-values* **;**

The RETAIN statement causes a program variable to hold its value from a previous observation until the variable is reassigned. The RETAIN statement can be used to initialize program variables.

The RETAIN statement does not work for model variables, parameters, or control variables because the values of these variables are under the control of PROC MODEL and not programming statements. Use the PARMS and CONTROL statements to initialize parameters and control variables. Use the VAR, ENDOGENOUS, or EXOGENOUS statement to initialize model variables.

Storing Programs in Model Files

Models can be saved in and recalled from SAS catalog files as well as XML-based data sets. SAS catalogs are special files that can store many kinds of data structures as separate units in one SAS file. Each separate unit is called an entry, and each entry has an entry type that identifies its structure to the SAS system. Starting with SAS 9.2, model files are being stored as SAS data sets instead of being stored as members of a SAS catalog as in earlier releases. This makes MODEL files more readily extendable in the future and enables Java-based applications to read the MODEL files directly. You can choose between the two formats by specifying a global CMPMODEL option in an OPTIONS statement. Details are given below.

In general, to save a model, use the OUTMODEL=*name* option in the PROC MODEL statement, where *name* is specified as *libref.catalog.entry*, *libref.entry*, or *entry* for catalog entry and, starting with SAS 9.2, *libref.datasetname* or *datasetname* for XML-based SAS data sets. The *libref*, *catalog*, *datasetnames* and *entry* names must be valid SAS names no more than 32 characters long. The *catalog* name is restricted to seven characters on the CMS operating system. If not given, the *catalog* name defaults to MODELS, and the *libref* defaults to WORK. The entry type is always MODEL. Thus, OUTMODEL=X writes the model to the file WORK.MODELS.X.MODEL in the SAS catalog or creates a WORK.X XML-based dataset in the WORK library depending on the format chosen by using the CMPMODEL= option. By default, both these formats are chosen.

The CMPMODEL= option can be used in an OPTIONS statement to modify the behavior when reading and writing MODEL files. The values allowed are CMPMODEL= BOTH | XML | CATALOG. For example, the following statements restore the previous behavior:

```
options cmpmodel=catalog;
```

The CMPMODEL= option defaults to BOTH in SAS 9.2 and is intended for transitional use. If CMPMODEL=BOTH, the MODEL procedure writes both formats; when loading model files PROC MODEL attempts to load the XML version first and the CATALOG version second (if the XML version is not found). If CMPMODEL=XML, the MODEL procedure reads and writes only the XML format. If CMPMODEL=CATALOG, only the catalog format is used.

The MODEL= option is used to read in a model. A list of model files can be specified in the MODEL= option, and a range of names with numeric suffixes can be given, as in MODEL=(MODEL1–MODEL10). When more than one model file is given, the list must be placed in parentheses, as in MODEL=(A B C), except in case of a single name. If more than one model file is specified, the files are combined in the order listed in the MODEL= option.

The MODEL procedure continues to read and write catalog MODEL files, and model files created by previous releases of SAS/ETS continue to work, so you should experience no direct impact from this change.

When the MODEL= option is specified in the PROC MODEL statement and model definition statements are also given later in the PROC MODEL step, the model files are read in first, in the order listed, and the model program specified in the PROC MODEL step is appended after the model program read from the MODEL= files. The class that is assigned to a variable, when multiple model files are used, is the last declaration of that variable. For example, if Y1 is declared endogenous in the model file M1 and exogenous in the model file M2, the following statement causes Y1 to be declared exogenous:

```
proc model model=(m1 m2);
```

The INCLUDE statement can be used to append model code to the current model code. In contrast, when the MODEL= option is specified in the RESET statement, the current model is deleted before the new model is read.

By default, no model file is output if the PROC MODEL step performs any FIT or SOLVE tasks, or if the MODEL= option or the NOSTORE option is specified. However, to ensure compatibility with previous versions of SAS/ETS software, if the PROC MODEL step does nothing but compile the model program, no input model file is read, and the NOSTORE option is not used, then a model file is written. This model file is the default input file for a later PROC SYSLIN or PROC SIMLIN step. The default output model filename in this case is WORK.MODELS._MODEL_.MODEL.

If FIT statements are used to estimate model parameters, the parameter estimates that are written to the output model file are the estimates from the last estimation performed for each parameter.

Macro Return Codes (SYSINFO)

The MODEL procedure stores a return code in the automatic macro variable SYSINFO upon completion of the PROC MODEL step. In the event any FIT or SOLVE task fails to converge during the completion of a PROC MODEL step, the value 1 is stored in the SYSINFO macro variable. Any subsequent SAS step resets the value of SYSINFO.

Diagnostics and Debugging

PROC MODEL provides several features to aid in finding errors in the model program. These debugging features are not usually needed; most models can be developed without them.

The example model program that follows is used in the following sections to illustrate the diagnostic and debugging capabilities. This example is the estimation of a segmented model.

```
/*--- Diagnostics and Debugging ---*/

*---------Fitting a Segmented Model using MODEL----*
|      |                                           |
|   y  | quadratic          plateau                |
|      | y=a+b*x+c*x*x      y=p                    |
|      |                     ..................... |
|      |               .     :                     |
|      |          .          :                     |
|      |       .             :                     |
|      |     .               :                     |
|      |    .                :                     |
|      +-------------------------------------X     |
|                          x0                       |
|                                                   |
| continuity restriction: p=a+b*x0+c*x0**2          |
| smoothness restriction: 0=b+2*c*x0 so x0=-b/(2*c)|
*---------------------------------------------------*;
title 'QUADRATIC MODEL WITH PLATEAU';
data a;
   input y x @@;
datalines;
.46 1  .47  2 .57  3 .61  4 .62  5 .68  6 .69  7
.78 8  .70  9 .74 10 .77 11 .78 12 .74 13 .80 13
.80 15 .78 16
;

proc model data=a list xref listcode;
   parms a 0.45 b 0.5 c -0.0025;

   x0 = -.5*b / c;      /* join point */
   if x < x0 then       /* Quadratic part of model */
      y = a + b*x + c*x*x;
   else                 /* Plateau part of model */
      y = a + b*x0 + c*x0*x0;

   fit y;
run;
```

Program Listing

The LIST option produces a listing of the model program. The statements are printed one per line with the original line number and column position of the statement.

The program listing from the example program is shown in Figure 26.88.

Figure 26.88 LIST Output for Segmented Model

QUADRATIC MODEL WITH PLATEAU

The MODEL Procedure

Listing of Compiled Program Code

Stmt	Line:Col	Statement as Parsed
1	4356:4	x0 = (-0.5 * b) / c;
2	4357:4	if x < x0 then
3	4358:7	PRED.y = a + b * x + c * x * x;
3	4358:7	RESID.y = PRED.y - ACTUAL.y;
3	4358:7	ERROR.y = PRED.y - y;
4	4359:4	else
5	4360:7	PRED.y = a + b * x0 + c * x0 * x0;
5	4360:7	RESID.y = PRED.y - ACTUAL.y;
5	4360:7	ERROR.y = PRED.y - y;

The LIST option also shows the model translations that PROC MODEL performs. LIST output is useful for understanding the code generated by the %AR and the %MA macros.

Cross-Reference

The XREF option produces a cross-reference listing of the variables in the model program. The XREF listing is usually used in conjunction with the LIST option. The XREF listing does not include derivative (@-prefixed) variables. The XREF listing does not include generated assignments to equation variables, PRED., RESID., and ERROR.-prefixed variables, unless the DETAILS option is used.

The cross-reference from the example program is shown in Figure 26.89.

Figure 26.89 XREF Output for Segmented Model

QUADRATIC MODEL WITH PLATEAU

The MODEL Procedure

Cross Reference Listing For Program

Symbol----------	Kind	Type	References (statement)/(line):(col)
a	Var	Num	Used: 3/66357:13 5/66359:13
b	Var	Num	Used: 1/66355:12 3/66357:16 5/66359:16
c	Var	Num	Used: 1/66355:15 3/66357:22 5/66359:23
x0	Var	Num	Assigned: 1/66355:15
			Used: 2/66356:11 5/66359:16 5/66359:23 5/66359:26
x	Var	Num	Used: 2/66356:11 3/66357:16 3/66357:22 3/66357:24
PRED.y	Var	Num	Assigned: 3/66357:19 5/66359:20

Compiler Listing

The LISTCODE option lists the model code and derivatives tables produced by the compiler. This listing is useful only for debugging and should not normally be needed.

LISTCODE prints the operator and operands of each operation generated by the compiler for each model program statement. Many of the operands are temporary variables generated by the compiler and given names such as #temp1. When derivatives are taken, the code listing includes the operations generated for the derivatives calculations. The derivatives tables are also listed.

A LISTCODE option prints the transformed equations from the example shown in Figure 26.90 and Figure 26.91.

Figure 26.90 LISTCODE Output for Segmented Model—Statements as Parsed

Derivatives		
WRT-Variable	**Object-Variable**	**Derivative-Variable**
a	RESID.y	@RESID.y/@a
b	RESID.y	@RESID.y/@b
c	RESID.y	@RESID.y/@c

Listing of Compiled Program Code		
Stmt	**Line:Col**	**Statement as Parsed**
1	4356:4	x0 = (-0.5 * b) / c;
1	4356:4	@x0/@b = -0.5 / c;
1	4356:4	@x0/@c = - x0 / c;
2	4357:4	if x < x0 then
3	4358:7	PRED.y = a + b * x + c * x * x;
3	4358:7	@PRED.y/@a = 1;
3	4358:7	@PRED.y/@b = x;
3	4358:7	@PRED.y/@c = x * x;
3	4358:7	RESID.y = PRED.y - ACTUAL.y;
3	4358:7	@RESID.y/@a = @PRED.y/@a;
3	4358:7	@RESID.y/@b = @PRED.y/@b;
3	4358:7	@RESID.y/@c = @PRED.y/@c;
3	4358:7	ERROR.y = PRED.y - y;
4	4359:4	else
5	4360:7	PRED.y = a + b * x0 + c * x0 * x0;
5	4360:7	@PRED.y/@a = 1;
5	4360:7	@PRED.y/@b = x0 + b * @x0/@b + (c * @x0/@b * x0 + c * x0 * @x0/@b);
5	4360:7	@PRED.y/@c = b * @x0/@c + ((x0 + c * @x0/@c) * x0 + c * x0 * @x0/@c);
5	4360:7	RESID.y = PRED.y - ACTUAL.y;
5	4360:7	@RESID.y/@a = @PRED.y/@a;
5	4360:7	@RESID.y/@b = @PRED.y/@b;
5	4360:7	@RESID.y/@c = @PRED.y/@c;
5	4360:7	ERROR.y = PRED.y - y;

Figure 26.91 LISTCODE Output for Segmented Model—Compiled Code

```
1 Stmt ASSIGN  line 4356 column 4. (1) arg=x0 argsave=x0
               Source Text:                                  x0 = -.5*b / c;
Oper *         at 4356:12 (30,0,2).                          * : _temp1 <- -0.5 b
Oper /         at 4356:15 (31,0,2).                          / : x0 <- _temp1 c
Oper eeocf     at 4356:15 (18,0,1).                          eeocf : _DER_ <- _DER_
Oper /         at 4356:15 (31,0,2).                          / : @x0/@b <- -0.5 c
Oper -         at 4356:15 (24,0,1).                          - : @1dt1_2 <- x0
Oper /         at 4356:15 (31,0,2).                          / : @x0/@c <- @1dt1_2 c

2 Stmt IF      line 4357 column 4. (2) arg=_temp1 argsave=_temp1  ref.st=ASSIGN stmt number 5 at 4360:7
               Source Text:                                  if x < x0 then
Oper <         at 4357:11 (36,0,2).                          < : _temp1 <- x x0

3 Stmt ASSIGN  line 4358 column 7. (1) arg=PRED.y argsave=y
               Source Text:                                  /* Quadratic part of model */ y = a + b*x + c*x*x;
Oper *         at 4358:16 (30,0,2).                          * : _temp1 <- b x
Oper +         at 4358:13 (32,0,2).                          + : _temp2 <- a _temp1
Oper *         at 4358:22 (30,0,2).                          * : _temp3 <- c x
Oper *         at 4358:24 (30,0,2).                          * : _temp4 <- _temp3 x
Oper +         at 4358:19 (32,0,2).                          + : PRED.y <- _temp2 _temp4
Oper eeocf     at 4358:19 (18,0,1).                          eeocf : _DER_ <- _DER_
Oper =         at 4358:19 (1,0,1).                           = : @PRED.y/@a <- 1
Oper =         at 4358:19 (1,0,1).                           = : @PRED.y/@b <- x
Oper *         at 4358:24 (30,0,2).                          * : @1dt1_1 <- x x
Oper =         at 4358:19 (1,0,1).                           = : @PRED.y/@c <- @1dt1_1

3 Stmt Assign  line 4358 column 7. (1) arg=RESID.y argsave=y
Oper -         at 4358:7 (33,0,2).                           - : RESID.y <- PRED.y ACTUAL.y
Oper eeocf     at 4358:7 (18,0,1).                           eeocf : _DER_ <- _DER_
Oper =         at 4358:7 (1,0,1).                            = : @RESID.y/@a <- @PRED.y/@a
Oper =         at 4358:7 (1,0,1).                            = : @RESID.y/@b <- @PRED.y/@b
Oper =         at 4358:7 (1,0,1).                            = : @RESID.y/@c <- @PRED.y/@c

3 Stmt Assign  line 4358 column 7. (1) arg=ERROR.y argsave=y
Oper -         at 4358:7 (33,0,2).                           - : ERROR.y <- PRED.y y

4 Stmt ELSE    line 4359 column 4. (9)                       ref.st=FIT stmt number 5 at 4362:4
               Source Text:                                  else

5 Stmt ASSIGN  line 4360 column 7. (1) arg=PRED.y argsave=y
               Source Text:                                  /* Plateau part of model */ y = a + b*x0 + c*x0*x0;
Oper *         at 4360:16 (30,0,2).                          * : _temp1 <- b x0
Oper +         at 4360:13 (32,0,2).                          + : _temp2 <- a _temp1
Oper *         at 4360:23 (30,0,2).                          * : _temp3 <- c x0
Oper *         at 4360:26 (30,0,2).                          * : _temp4 <- _temp3 x0
Oper +         at 4360:20 (32,0,2).                          + : PRED.y <- _temp2 _temp4
Oper eeocf     at 4360:20 (18,0,1).                          eeocf : _DER_ <- _DER_
Oper =         at 4360:20 (1,0,1).                           = : @PRED.y/@a <- 1
```

Figure 26.91 *continued*

Oper *	at 4360:16 (30,0,2).	* : @1dt1_1 <- b @x0/@b
Oper +	at 4360:16 (32,0,2).	+ : @1dt1_2 <- x0 @1dt1_1
Oper *	at 4360:23 (30,0,2).	* : @1dt1_3 <- c @x0/@b
Oper *	at 4360:26 (30,0,2).	* : @1dt1_4 <- @1dt1_3 x0
Oper *	at 4360:26 (30,0,2).	* : @1dt1_5 <- _temp3 @x0/@b
Oper +	at 4360:26 (32,0,2).	+ : @1dt1_6 <- @1dt1_4 @1dt1_5
Oper +	at 4360:20 (32,0,2).	+ : @PRED.y/@b <- @1dt1_2 @1dt1_6
Oper *	at 4360:16 (30,0,2).	* : @1dt1_8 <- b @x0/@c
Oper *	at 4360:23 (30,0,2).	* : @1dt1_9 <- c @x0/@c
Oper +	at 4360:23 (32,0,2).	+ : @1dt1_10 <- x0 @1dt1_9
Oper *	at 4360:26 (30,0,2).	* : @1dt1_11 <- @1dt1_10 x0
Oper *	at 4360:26 (30,0,2).	* : @1dt1_12 <- _temp3 @x0/@c
Oper +	at 4360:26 (32,0,2).	+ : @1dt1_13 <- @1dt1_11 @1dt1_12
Oper +	at 4360:20 (32,0,2).	+ : @PRED.y/@c <- @1dt1_8 @1dt1_13
5 Stmt Assign	line 4360 column 7. (1) arg=RESID.y argsave=y	
Oper -	at 4360:7 (33,0,2).	- : RESID.y <- PRED.y ACTUAL.y
Oper eeocf	at 4360:7 (18,0,1).	eeocf : _DER_ <- _DER_
Oper =	at 4360:7 (1,0,1).	= : @RESID.y/@a <- @PRED.y/@a
Oper =	at 4360:7 (1,0,1).	= : @RESID.y/@b <- @PRED.y/@b
Oper =	at 4360:7 (1,0,1).	= : @RESID.y/@c <- @PRED.y/@c
5 Stmt Assign	line 4360 column 7. (1) arg=ERROR.y argsave=y	
Oper -	at 4360:7 (33,0,2).	- : ERROR.y <- PRED.y y

Analyzing the Structure of Large Models

PROC MODEL provides several features to aid in analyzing the structure of the model program. These features summarize properties of the model in various forms.

Simulation Dependency Analysis

During the development of model programs for simulation, misspecification of the equations or variables that compose the systems of nonlinear equations is common. These misspecification errors can occur both in the original formulation of the model and in the encoding of the model into PROC MODEL statements. For large systems these errors can be difficult and time consuming to isolate and repair. Similarly, the process of becoming familiar with an existing simulation model that is encoded in PROC MODEL can be laborious when available documentation is insufficient to understand the model's implementation. To address these issues, the ANALYZEDEP= option can be applied to SOLVE steps to produce graphical analyses of a model's structure.

The graphical output that is produced by the ANALYZEDEP= option displays the results of two separate, hierarchical analyses that are both based on the dependence of equations on solve variables in the nonlinear system of equations. First, the system is partitioned to identify which equations overdetermine solve variables, which equations underdetermine solve variables, and which equations consistently determine solve variables. These three partitions of equations and their corresponding three partitions of solve variables are identified in the graphical output and listing produced by the ANALYZEDEP= option. Second, each partition from the first analysis is analyzed to identify subpartitions of equations and solve variables such that all the solve variables within each subpartition depend either directly or indirectly on one another. In the graphical output the subpartitions are represented as blocks in a dependency matrix. The subpartition blocks are ordered so that the matrix of dependencies has a block upper-triangular form.

The first-level partitioning of the system into underdetermined, overdetermined, and consistent systems of equations and variables uses a Dulmage-Mendelsohn (DM) decomposition to define the three partitions, following the work by Dulmage and Mendelsohn (1958); Pothen and Fan (1990). The overdetermining equations in a DM decomposition are the set of all equations that do not have dependent variables on the diagonal of any dependency matrix that contains the maximum possible number of entries on the diagonal. The dependency matrices for a problem consist of the set of pairs of orderings of the problem's equations and solve variables. Correspondingly, the DM decomposition defines underdetermined variables as the set of all variables that do not appear on the diagonal of any dependency matrix that contains the maximum number of entries on the diagonal. Therefore, the DM decomposition is canonical in the sense that its partitioning of the system is invariant to the order equations and variables are specified in the model program. The following PROC MODEL statements illustrate how to partition a simple model with five equations and five unknowns:

```
proc model data=_null_;
   endo a b c d e;

   f(a)     = 0;
   g(a,b)   = 0;
   h(a,b)   = 0;
   i(b,d)   = 0;
   j(c,d,e) = 0;

   solve / analyzedep=(block);
quit;
```

Figure 26.92 Block Dependency Analysis

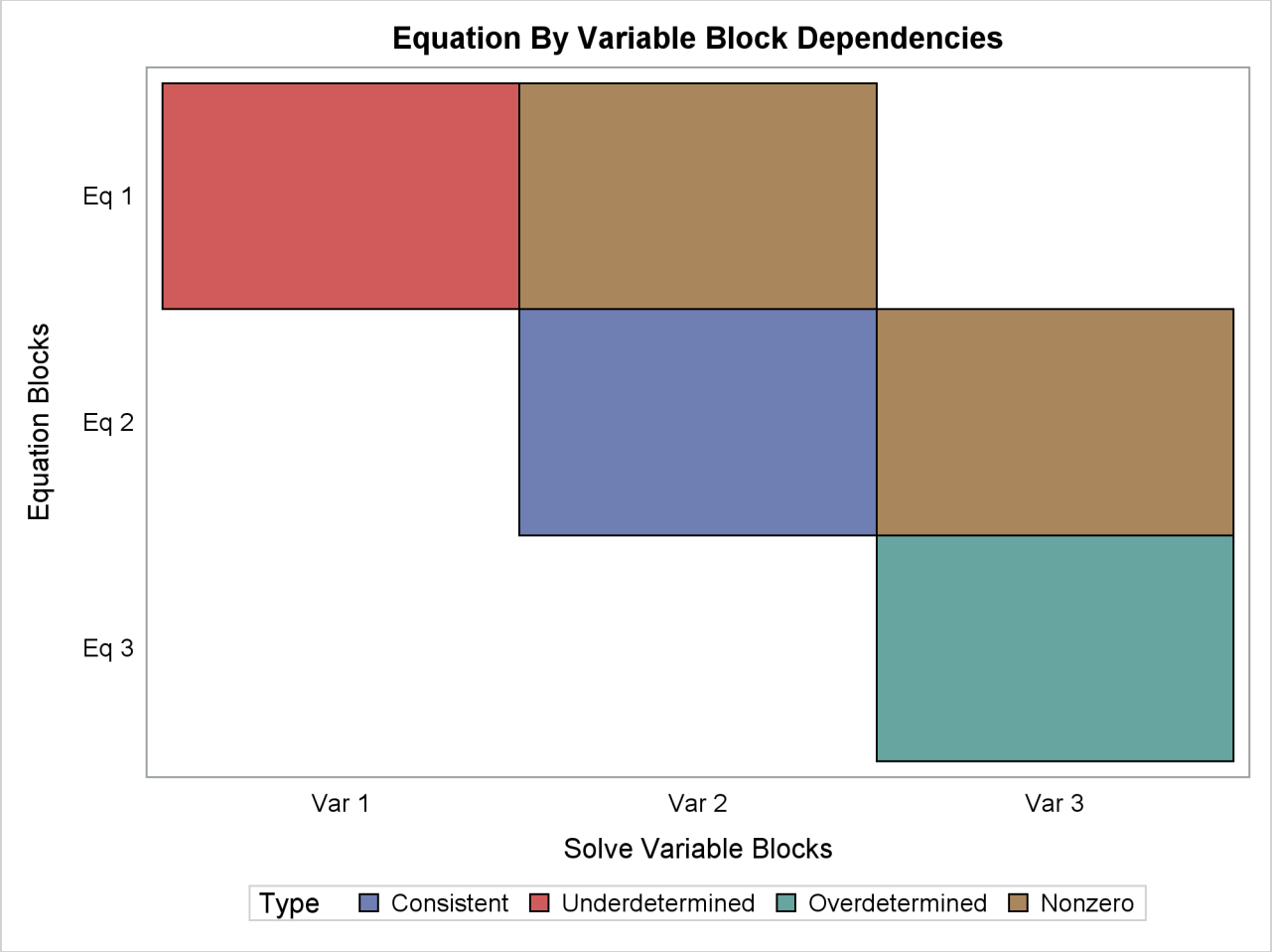


Figure 26.93 Block Partitions

Equation and Variable Blocks

Type	Block	Symbols
Underdetermined	Eq 1	j(c,d,e)
	Var 1	c e
Consistent	Eq 2	i(b,d)
	Var 2	d
Overdetermined	Eq 3	f(a) g(a,b) h(a,b)
	Var 3	a b

Figure 26.92 and Figure 26.93 illustrate which equations and variables belong to each block and which blocks are in each partition. The cells that are marked "Nonzero" in the plot represent a dependency between blocks that are above the diagonal in the dependency matrix. The exact functional forms of the equations in this example are not shown; however, the dependency analysis here reveals that this model is structurally singular because it contains overdetermined and underdetermined components. Some modification of the model specification is necessary before a SOLVE step can be executed.

For large systems of equations, the graphical output that the ANALYZEDEP= option produces can be used as a starting point to explore dependency relationships when the models' programming statement listings and dependency tables are too long to read and comprehend. For example, one econometric model of U.S. agriculture involves thousand of equation and variable dependencies whose structure is difficult to interpret in textual listings of the model. If you examine the block triangular form of its dependency matrix in Figure 26.94, one pattern of dependencies that becomes apparent is the vertical grouping of block dependencies in the middle of the plot. Figure 26.95 shows the dependency matrix for this important subpartition of equations and variables responsible for coupling the vertical grouping of blocks.

Figure 26.94 Block Triangular Form of U.S. Agriculture Model

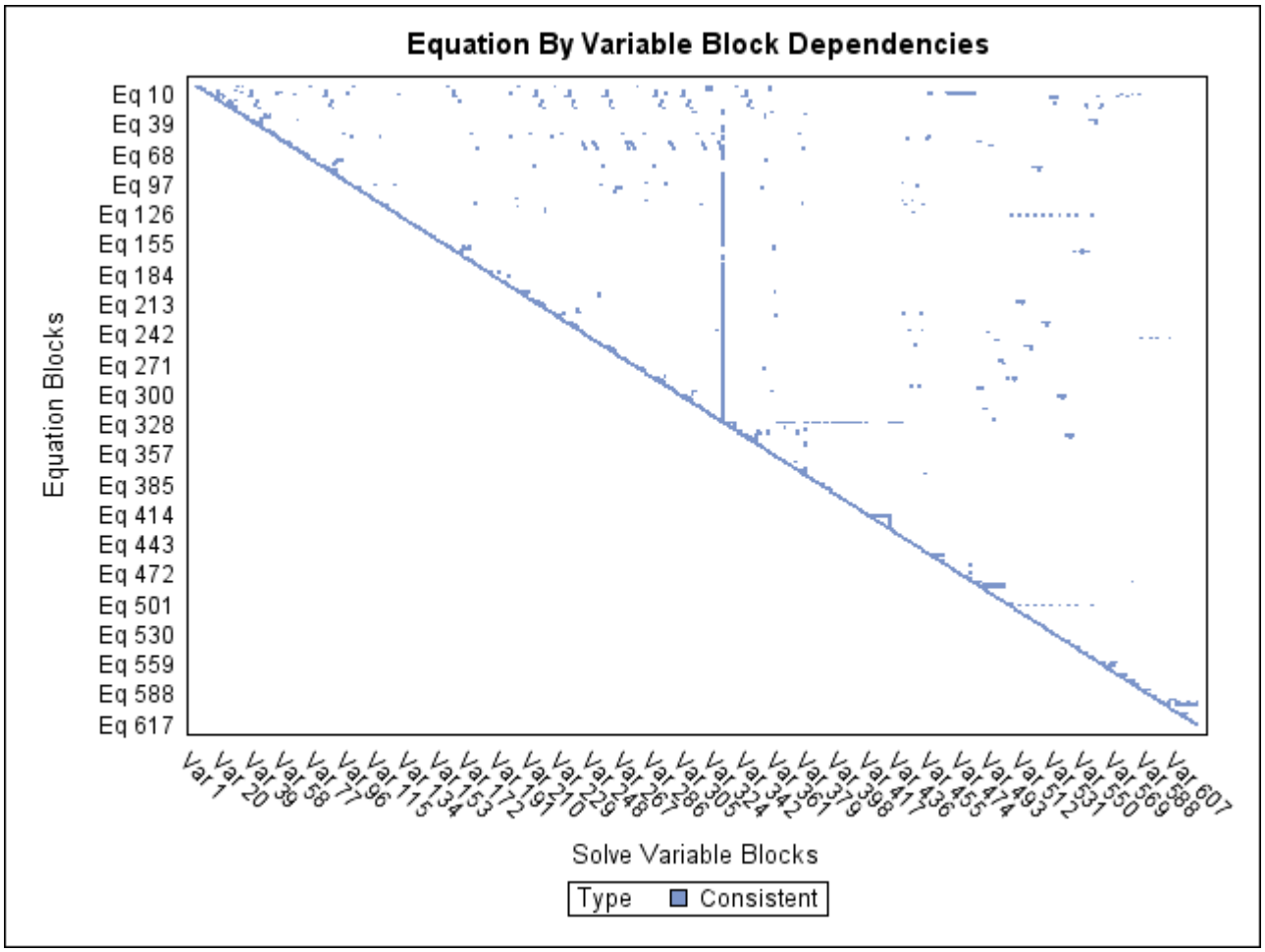
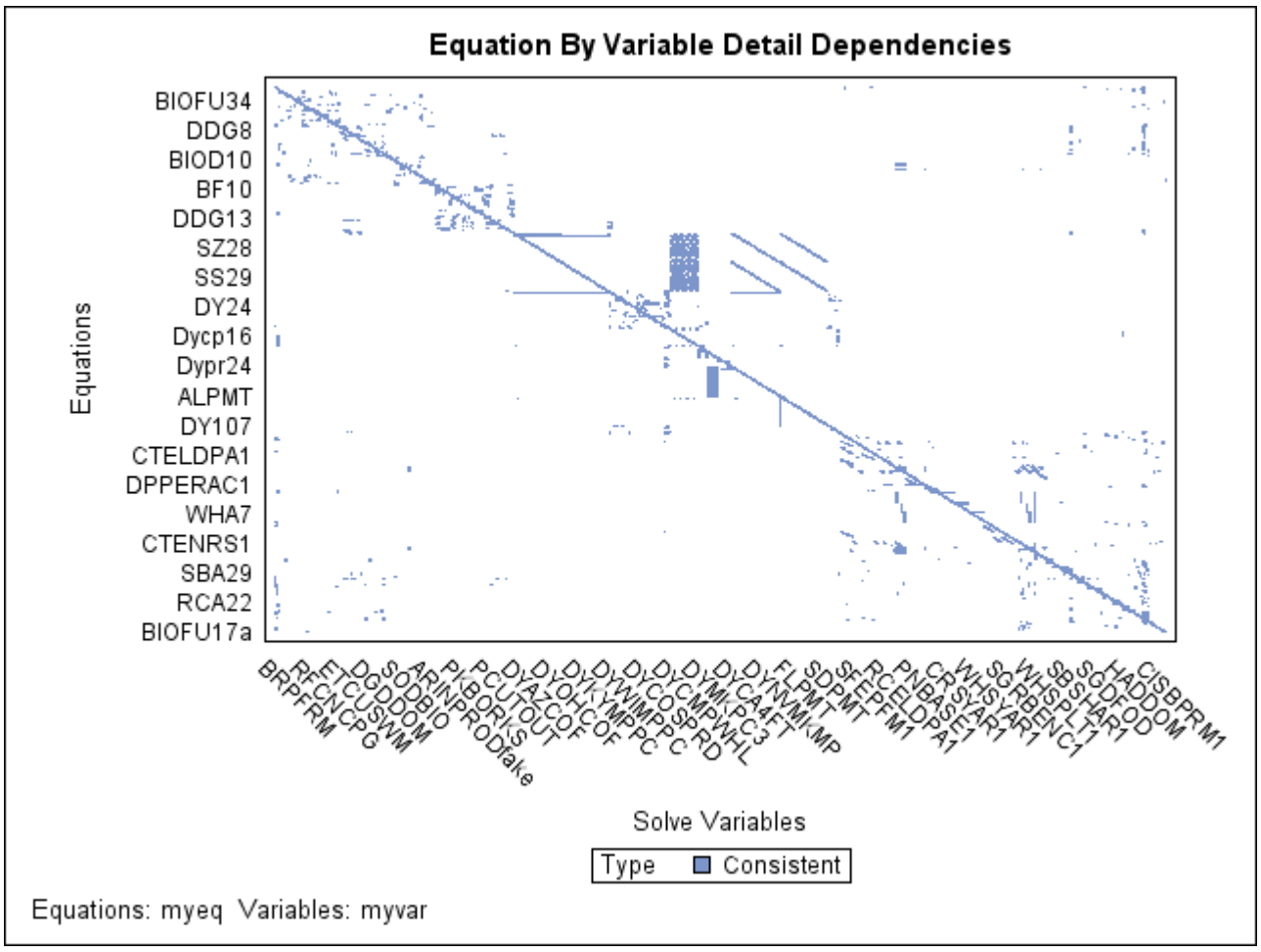


Figure 26.95 Important Component in U.S. Agriculture Model



Compared to the BLOCK and GRAPH options, the ANALYZEDEP= option has the following advantages:

- shows which equations and solve variables are overdetermined, consistent, and underdetermined
- works with any combination of normal form and general form equations
- can display dependency matrices involving many more equations and variables
- can be limited to a subset of the equations and variables in the model

The following Klein's model program is used to introduce the LISTDEP, BLOCK, and GRAPH options:

```
proc model  out=m data=klein listdep graph block;
   endogenous c p w i x wsum k y;
   exogenous  wp g t year;
   parms c0-c3 i0-i3 w0-w3;
   a: c = c0 + c1 * p + c2 * lag(p) + c3 * wsum;
   b: i = i0 + i1 * p + i2 * lag(p) + i3 * lag(k);
   c: w = w0 + w1 * x + w2 * lag(x) + w3 * year;
   x = c + i + g;
   y = c + i + g-t;
   p = x-w-t;
   k = lag(k) + i;
   wsum = w + wp;
   id year;
quit;
```

Dependency List

The LISTDEP option produces a dependency list for each variable in the model program. For each variable, a list of variables that depend on it and a list of variables it depends on is given. The dependency list produced by the example program is shown in Figure 26.96.

Figure 26.96 A Portion of the LISTDEP Output for Klein's Model

The MODEL Procedure

Dependency Listing For Program	
Symbol----------	**Dependencies**
c	Current values affect: RESID.c ERROR.c PRED.x RESID.x ERROR.x PRED.y RESID.y ERROR.y
p	Current values affect: PRED.c RESID.c ERROR.c PRED.i RESID.i ERROR.i RESID.p ERROR.p
	Lagged values affect: PRED.c PRED.i
w	Current values affect: RESID.w ERROR.w PRED.p RESID.p ERROR.p PRED.wsum RESID.wsum ERROR.wsum
i	Current values affect: RESID.i ERROR.i PRED.x RESID.x ERROR.x PRED.y RESID.y ERROR.y PRED.k RESID.k ERROR.k
x	Current values affect: PRED.w RESID.w ERROR.w RESID.x ERROR.x PRED.p RESID.p ERROR.p
	Lagged values affect: PRED.w
wsum	Current values affect: PRED.c RESID.c ERROR.c RESID.wsum ERROR.wsum
k	Current values affect: RESID.k ERROR.k
	Lagged values affect: PRED.i RESID.i ERROR.i PRED.k
y	Current values affect: RESID.y ERROR.y
wp	Current values affect: PRED.wsum RESID.wsum ERROR.wsum
g	Current values affect: PRED.x RESID.x ERROR.x PRED.y RESID.y ERROR.y
t	Current values affect: PRED.y RESID.y ERROR.y PRED.p RESID.p ERROR.p
year	Current values affect: PRED.w RESID.w ERROR.w
c0	Current values affect: PRED.c RESID.c ERROR.c
c1	Current values affect: PRED.c RESID.c ERROR.c
c2	Current values affect: PRED.c RESID.c ERROR.c
c3	Current values affect: PRED.c RESID.c ERROR.c
i0	Current values affect: PRED.i RESID.i ERROR.i
i1	Current values affect: PRED.i RESID.i ERROR.i
i2	Current values affect: PRED.i RESID.i ERROR.i
i3	Current values affect: PRED.i RESID.i ERROR.i
w0	Current values affect: PRED.w RESID.w ERROR.w
w1	Current values affect: PRED.w RESID.w ERROR.w
w2	Current values affect: PRED.w RESID.w ERROR.w
w3	Current values affect: PRED.w RESID.w ERROR.w
PRED.c	Depends on current values of: p wsum c0 c1 c2 c3
	Depends on lagged values of: p
	Current values affect: RESID.c ERROR.c
RESID.c	Depends on current values of: PRED.c c p wsum c0 c1 c2 c3
ERROR.c	Depends on current values of: PRED.c c p wsum c0 c1 c2 c3
ACTUAL.c	Current values affect: RESID.c ERROR.c PRED.x RESID.x ERROR.x PRED.y RESID.y ERROR.y
PRED.i	Depends on current values of: p i0 i1 i2 i3
	Depends on lagged values of: p k
	Current values affect: RESID.i ERROR.i
RESID.i	Depends on current values of: PRED.i p i i0 i1 i2 i3
	Depends on lagged values of: k
ERROR.i	Depends on current values of: PRED.i p i i0 i1 i2 i3
	Depends on lagged values of: k
ACTUAL.i	Current values affect: RESID.i ERROR.i PRED.x RESID.x ERROR.x PRED.y RESID.y ERROR.y PRED.k RESID.k ERROR.k
PRED.w	Depends on current values of: x year w0 w1 w2 w3
	Depends on lagged values of: x
	Current values affect: RESID.w ERROR.w

Figure 26.96 *continued*

The MODEL Procedure

Dependency Listing For Program	
Symbol----------	**Dependencies**
RESID.w	Depends on current values of: PRED.w w x year w0 w1 w2 w3
ERROR.w	Depends on current values of: PRED.w w x year w0 w1 w2 w3
ACTUAL.w	Current values affect: RESID.w ERROR.w PRED.p RESID.p ERROR.p PRED.wsum RESID.wsum ERROR.wsum
PRED.x	Depends on current values of: c i g
	Current values affect: RESID.x ERROR.x
RESID.x	Depends on current values of: PRED.x c i x g
ERROR.x	Depends on current values of: PRED.x c i x g
ACTUAL.x	Current values affect: PRED.w RESID.w ERROR.w RESID.x ERROR.x PRED.p RESID.p ERROR.p
	Lagged values affect: PRED.w
PRED.y	Depends on current values of: c i g t
	Current values affect: RESID.y ERROR.y
RESID.y	Depends on current values of: PRED.y c i y g t
ERROR.y	Depends on current values of: PRED.y c i y g t
ACTUAL.y	Current values affect: RESID.y ERROR.y
PRED.p	Depends on current values of: w x t
	Current values affect: RESID.p ERROR.p
RESID.p	Depends on current values of: PRED.p p w x t
ERROR.p	Depends on current values of: PRED.p p w x t
ACTUAL.p	Current values affect: PRED.c RESID.c ERROR.c PRED.i RESID.i ERROR.i RESID.p ERROR.p
	Lagged values affect: PRED.c PRED.i
PRED.k	Depends on current values of: i
	Depends on lagged values of: k
	Current values affect: RESID.k ERROR.k
RESID.k	Depends on current values of: PRED.k i k
ERROR.k	Depends on current values of: PRED.k i k
ACTUAL.k	Current values affect: RESID.k ERROR.k
	Lagged values affect: PRED.i RESID.i ERROR.i PRED.k
PRED.wsum	Depends on current values of: w wp
	Current values affect: RESID.wsum ERROR.wsum
RESID.wsum	Depends on current values of: PRED.wsum w wsum wp
ERROR.wsum	Depends on current values of: PRED.wsum w wsum wp
ACTUAL.wsum	Current values affect: PRED.c RESID.c ERROR.c RESID.wsum ERROR.wsum

BLOCK Listing

The BLOCK option prints an analysis of the program variables based on the assignments in the model program. The output produced by the example is shown in Figure 26.97.

Figure 26.97 The BLOCK Output for Klein's Model

The MODEL Procedure
Model Structure Analysis
(Based on Assignments to Endogenous Model Variables)

Exogenous Variables	wp g t year
Endogenous Variables	c p w i x wsum k y

Block Structure of the System
Block 1 c p w i x wsum

Dependency Structure of the System	
Block 1	Depends On All_Exogenous
k	Depends On Block 1 All_Exogenous
y	Depends On Block 1 All_Exogenous

One use for the block output is to put a model in recursive form. Simulations of the model can be done with the SEIDEL method, which is efficient if the model is recursive and if the equations are in recursive order. By examining the block output, you can determine how to reorder the model equations for the most efficient simulation.

Adjacency Graph

The GRAPH option displays the same information as the BLOCK option with the addition of an adjacency graph. An X in a column in an adjacency graph indicates that the variable associated with the row depends on the variable associated with the column. The output produced by the example is shown in Figure 26.98.

The first and last graphs are straightforward. The middle graph represents the dependencies of the nonexogenous variables after transitive closure has been performed (that is, A depends on B, and B depends on C, so A depends on C). The preceding transitive closure matrix indicates that K and Y do not directly or indirectly depend on each other.

Figure 26.98 The GRAPH Output for Klein's Model

Adjacency Matrix for Graph of System

Variable	c	p	w	i	x	wsum	k	y	wp	g	t	year
									*	*	*	*
c	X	X	.	.	.	X	.	.	.	.	.	.
p	.	X	X	.	X	.	.	.	.	.	X	.
w	.	.	X	.	X	.	.	.	.	.	.	X
i	.	X	.	X	.	.	.	.	.	.	.	.
x	X	.	.	X	X	.	.	.	.	X	.	.
wsum	.	.	X	.	.	X	.	.	X	.	.	.
k	.	.	.	X	.	.	X	.	.	.	.	.
y	X	.	.	X	.	.	.	X	.	X	X	.
wp	*	.	.	.	.	.	.	.	X	.	.	.
g	*	.	.	.	.	.	.	.	.	X	.	.
t	*	.	.	.	.	.	.	.	.	.	X	.
year	*	.	.	.	.	.	.	.	.	.	.	X

(Note: * = Exogenous Variable.)

Transitive Closure Matrix of Sorted System

Block	Variable	c	p	w	i	x	wsum	k	y
1	c	X	X	X	X	X	X	.	.
1	p	X	X	X	X	X	X	.	.
1	w	X	X	X	X	X	X	.	.
1	i	X	X	X	X	X	X	.	.
1	x	X	X	X	X	X	X	.	.
1	wsum	X	X	X	X	X	X	.	.
	k	X	X	X	X	X	X	X	.
	y	X	X	X	X	X	X	.	X

Adjacency Matrix for Graph of System Including Lagged Impacts

Block	Variable	c	p	w	i	x	wsum	k	y	wp	g	t	year
										*	*	*	*
1	c	X	L	.	.	.	X	.	.	.	.	.	.
1	p	.	X	X	.	X	.	.	.	.	.	X	.
1	w	.	.	X	.	L	.	.	.	.	.	.	X
1	i	.	L	.	X	.	.	L	.	.	.	.	.
1	x	X	.	.	X	X	.	.	.	.	X	.	.
1	wsum	.	.	X	.	.	X	.	.	X	.	.	.
	k	.	.	.	X	.	.	L	.	.	.	.	.
	y	X	.	.	X	.	.	.	X	.	X	X	.
	wp	*	.	.	.	.	.	.	.	X	.	.	.
	g	*	.	.	.	.	.	.	.	.	X	.	.
	t	*	.	.	.	.	.	.	.	.	.	X	.
	year	*	.	.	.	.	.	.	.	.	.	.	X

(Note: * = Exogenous Variable.)

Examples: MODEL Procedure

Example 26.1: OLS Single Nonlinear Equation

This example illustrates the use of the MODEL procedure for nonlinear ordinary least squares (OLS) regression. The model is a logistic growth curve for the population of the United States. The data is the population in millions recorded at ten-year intervals starting in 1790 and ending in 2000. For an explanation of the starting values given by the START= option, see the section "Troubleshooting Convergence Problems" on page 1558. Portions of the output from the following statements are shown in Output 26.1.1 through Output 26.1.3.

```
title 'Logistic Growth Curve Model of U.S. Population';
data uspop;
   input pop :6.3 @@;
   retain year 1780;
   year=year+10;
   label pop='U.S. Population in Millions';
   datalines;
3929  5308  7239   9638  12866  17069  23191  31443  39818 50155
62947 75994 91972 105710 122775 131669 151325 179323 203211
226542 248710
;

proc model data=uspop;
   label a = 'Maximum Population'
         b = 'Location Parameter'
         c = 'Initial Growth Rate';
   pop = a / ( 1 + exp( b - c * (year-1790) ) );
   fit pop start=(a 1000  b 5.5  c .02) / out=resid outresid;
run;
```

Output 26.1.1 Logistic Growth Curve Model Summary

Logistic Growth Curve Model of U.S. Population

The MODEL Procedure

Model Summary	
Model Variables	1
Parameters	3
Equations	1
Number of Statements	1

Model Variables	pop
Parameters(Value)	a(1000) b(5.5) c(0.02)
Equations	pop

The Equation to Estimate is
pop = F(a, b, c)

Output 26.1.2 Logistic Growth Curve Estimation Summary

Logistic Growth Curve Model of U.S. Population

The MODEL Procedure
OLS Estimation Summary

Data Set Options	
DATA=	USPOP
OUT=	RESID

Minimization Summary	
Parameters Estimated	3
Method	Gauss
Iterations	7
Subiterations	6
Average Subiterations	0.857143

Final Convergence Criteria	
R	0.00068
PPC(a)	0.000145
RPC(a)	0.001507
Object	0.000065
Trace(S)	19.20198
Objective Value	16.45884

Observations Processed	
Read	21
Solved	21

Output 26.1.3 Logistic Growth Curve Estimates

Logistic Growth Curve Model of U.S. Population

The MODEL Procedure

Nonlinear OLS Summary of Residual Errors

Equation	DF Model	DF Error	SSE	MSE	Root MSE	R-Square	Adj R-Sq	Label
pop	3	18	345.6	19.2020	4.3820	0.9972	0.9969	U.S. Population in Millions

Nonlinear OLS Parameter Estimates

Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|	Label
a	387.9307	30.0404	12.91	<.0001	Maximum Population
b	3.990385	0.0695	57.44	<.0001	Location Parameter
c	0.022703	0.00107	21.22	<.0001	Initial Growth Rate

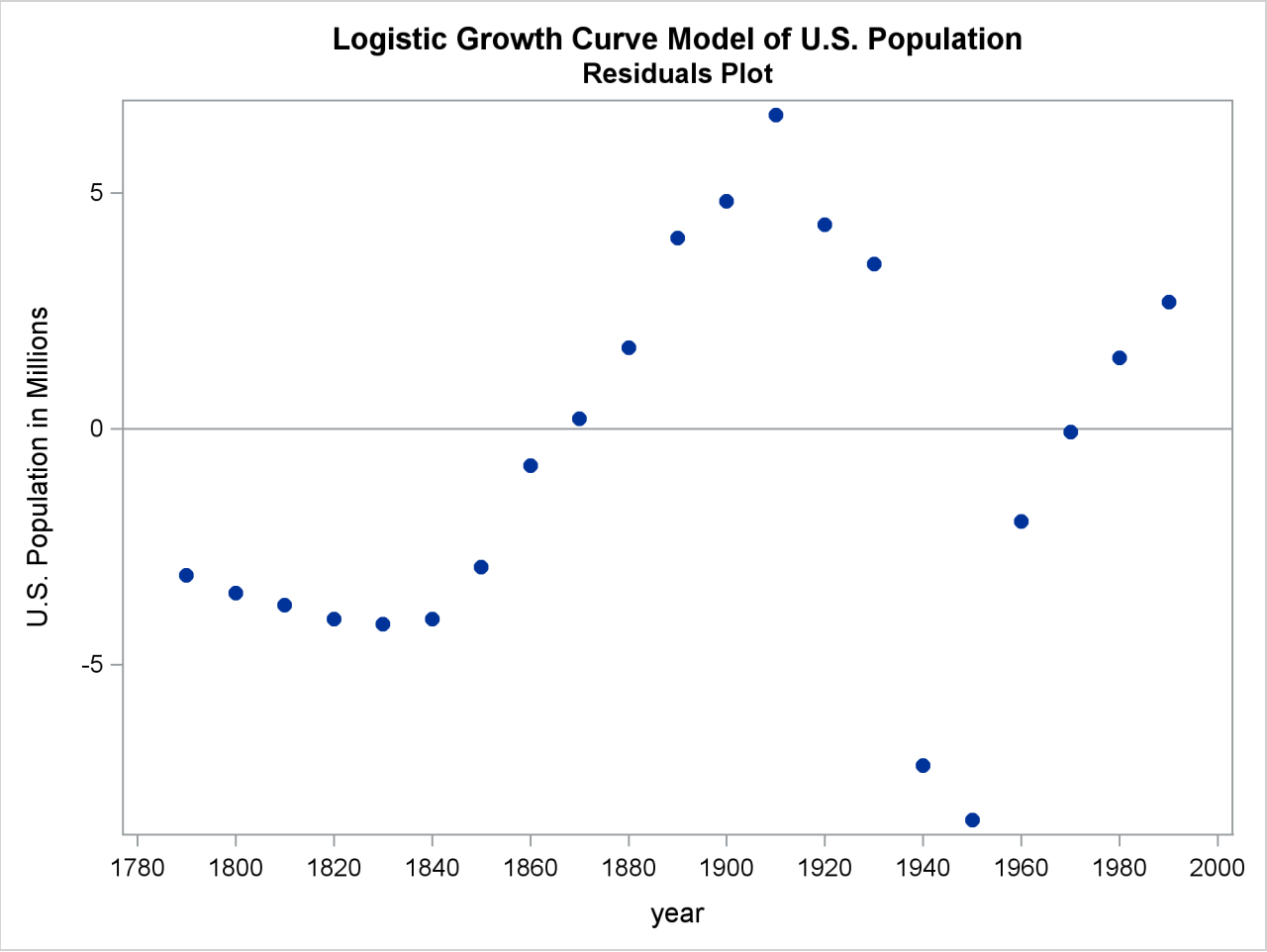
The adjusted R^2 value indicates the model fits the data well. There are only 21 observations and the model is nonlinear, so significance tests on the parameters are only approximate. The significance tests and associated approximate probabilities indicate that all the parameters are significantly different from 0.

The FIT statement included the options OUT=RESID and OUTRESID so that the residuals from the estimation are saved to the data set RESID. The residuals are plotted to check for heteroscedasticity by using PROC SGPLOT as follows.

```
title2 "Residuals Plot";
proc sgplot data=resid;
   refline 0;
   scatter x=year y=pop / markerattrs=(symbol=circlefilled);
   xaxis values=(1780 to 2000 by 20);
run;
```

The plot is shown in Output 26.1.4.

Output 26.1.4 Residual for Population Model (Actual–Predicted)



The residuals do not appear to be independent, and the model could be modified to explain the remaining nonrandom errors.

Example 26.2: A Consumer Demand Model

This example shows the estimation of a system of nonlinear consumer demand equations based on the translog functional form by using seemingly unrelated regression (SUR). Expenditure shares and corresponding normalized prices are given for three goods.

Since the shares add up to one, the system is singular; therefore, one equation is omitted from the estimation process. The choice of which equation to omit is arbitrary. The nonlinear system is first estimated in unrestricted form by the following statements:

```
title1 'Consumer Demand--Translog Functional Form';
title2 'Asymmetric Model';

proc model data=tlog1;
   endogenous share1 share2;
   parms a1 a2 b11 b12 b13 b21 b22 b23 b31 b32 b33;

   bm1 = b11 + b21 + b31;
   bm2 = b12 + b22 + b32;
   bm3 = b13 + b23 + b33;
   lp1 = log(p1);
   lp2 = log(p2);
   lp3 = log(p3);
   share1 = ( a1 + b11 * lp1 + b12 * lp2 + b13 * lp3 ) /
            ( -1 + bm1 * lp1 + bm2 * lp2 + bm3 * lp3 );
   share2 = ( a2 + b21 * lp1 + b22 * lp2 + b23 * lp3 ) /
            ( -1 + bm1 * lp1 + bm2 * lp2 + bm3 * lp3 );

   fit share1 share2
       start=( a1 -.14 a2 -.45 b11 .03 b12 .47 b22 .98 b31 .20
               b32 1.11 b33 .71 ) / outsused=smatrix sur;
run;
```

A portion of the printed output produced by this example is shown in Output 26.2.1 through Output 26.2.3.

Output 26.2.1 Translog Demand Model Summary

Consumer Demand--Translog Functional Form
Asymmetric Model

The MODEL Procedure

Model Summary	
Model Variables	2
Endogenous	2
Parameters	11
Equations	2
Number of Statements	8

Model Variables	share1 share2
Parameters(Value)	a1(-0.14) a2(-0.45) b11(0.03) b12(0.47) b13 b21 b22(0.98) b23 b31(0.2) b32(1.11) b33(0.71)
Equations	share1 share2

Output 26.2.1 *continued*

The 2 Equations to Estimate	
share1 =	F(a1, b11, b12, b13, b21, b22, b23, b31, b32, b33)
share2 =	F(a2, b11, b12, b13, b21, b22, b23, b31, b32, b33)

Output 26.2.2 Estimation Summary for the Unrestricted Model

NOTE: At SUR Iteration 2 CONVERGE=0.001 Criteria Met.

Consumer Demand--Translog Functional Form Asymmetric Model

The MODEL Procedure
SUR Estimation Summary

Data Set Options	
DATA=	TLOG1
OUTSUSED=	SMATRIX

Minimization Summary	
Parameters Estimated	11
Method	Gauss
Iterations	2

Final Convergence Criteria	
R	0.00016
PPC(b11)	0.00116
RPC(b11)	0.012106
Object	2.921E-6
Trace(S)	0.000078
Objective Value	1.749312

Observations Processed	
Read	44
Solved	44

Output 26.2.3 Estimation Results for the Unrestricted Model

Consumer Demand--Translog Functional Form Asymmetric Model

The MODEL Procedure

Nonlinear SUR Summary of Residual Errors							
Equation	**DF Model**	**DF Error**	**SSE**	**MSE**	**Root MSE**	**R-Square**	**Adj R-Sq**
share1	5.5	38.5	0.00166	0.000043	0.00656	0.8067	0.7841
share2	5.5	38.5	0.00135	0.000035	0.00592	0.9445	0.9380

Output 26.2.3 *continued*

Nonlinear SUR Parameter Estimates

Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|
a1	-0.14881	0.00225	-66.08	<.0001
a2	-0.45776	0.00297	-154.29	<.0001
b11	0.048382	0.0498	0.97	0.3379
b12	0.43655	0.0502	8.70	<.0001
b13	0.248588	0.0516	4.82	<.0001
b21	0.586326	0.2089	2.81	0.0079
b22	0.759776	0.2565	2.96	0.0052
b23	1.303821	0.2328	5.60	<.0001
b31	0.297808	0.1504	1.98	0.0550
b32	0.961551	0.1633	5.89	<.0001
b33	0.8291	0.1556	5.33	<.0001

Number of Observations		Statistics for System	
Used	44	**Objective**	1.7493
Missing	0	**Objective*N**	76.9697

The model is then estimated under the restriction of symmetry ($b_{ij} = b_{ji}$), as shown in the following statements:

```
title2 'Symmetric Model';
proc model data=tlog1;
   var share1 share2 p1 p2 p3;
   parms a1 a2 b11 b12 b22 b31 b32 b33;
   bm1 = b11 + b12 + b31;
   bm2 = b12 + b22 + b32;
   bm3 = b31 + b32 + b33;
   lp1 = log(p1);
   lp2 = log(p2);
   lp3 = log(p3);
   share1 = ( a1 + b11 * lp1 + b12 * lp2 + b31 * lp3 ) /
            ( -1 + bm1 * lp1 + bm2 * lp2 + bm3 * lp3 );
   share2 = ( a2 + b12 * lp1 + b22 * lp2 + b32 * lp3 ) /
            ( -1 + bm1 * lp1 + bm2 * lp2 + bm3 * lp3 );
   fit share1 share2
       start=( a1 -.14 a2 -.45 b11 .03 b12 .47 b22 .98 b31 .20
               b32 1.11 b33 .71 ) / sdata=smatrix sur;
run;
```

A portion of the printed output produced for the symmetry restricted model is shown in Output 26.2.4 and Output 26.2.5.

Output 26.2.4 Model Summary from the Restricted Model

Consumer Demand--Translog Functional Form Symmetric Model

The MODEL Procedure

The 2 Equations to Estimate	
share1 =	F(a1, b11, b12, b22, b31, b32, b33)
share2 =	F(a2, b11, b12, b22, b31, b32, b33)

Output 26.2.5 Estimation Results for the Restricted Model

Consumer Demand--Translog Functional Form Symmetric Model

The MODEL Procedure

Nonlinear SUR Summary of Residual Errors

Equation	DF Model	DF Error	SSE	MSE	Root MSE	R-Square	Adj R-Sq
share1	4	40	0.00166	0.000041	0.00644	0.8066	0.7920
share2	4	40	0.00139	0.000035	0.00590	0.9428	0.9385

Nonlinear SUR Parameter Estimates

Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|
a1	-0.14684	0.00135	-108.99	<.0001
a2	-0.4597	0.00167	-275.34	<.0001
b11	0.02886	0.00741	3.89	0.0004
b12	0.467827	0.0115	40.57	<.0001
b22	0.970079	0.0177	54.87	<.0001
b31	0.208143	0.00614	33.88	<.0001
b32	1.102415	0.0127	86.51	<.0001
b33	0.694245	0.0168	41.38	<.0001

Number of Observations		Statistics for System	
Used	44	**Objective**	1.7820
Missing	0	**Objective*N**	78.4097

Hypothesis testing requires that the **S** matrix from the unrestricted model be imposed on the restricted model, as explained in the section "Tests on Parameters" on page 1601. The **S** matrix saved in the data set SMATRIX is requested by the SDATA= option.

A chi-square test is used to see if the hypothesis of symmetry is accepted or rejected. (Oc–Ou) has a chi-square distribution asymptotically, where Oc is the constrained OBJECTIVE*N and Ou is the unconstrained OBJECTIVE*N. The degrees of freedom is equal to the difference in the number of free parameters in the two models.

In this example, Ou is 76.9697 and Oc is 78.4097, resulting in a difference of 1.44 with 3 degrees of freedom. You can obtain the probability value by using the following statements:

```
data _null_;
   /* probchi( reduced-full, n-restrictions )*/
   p = 1-probchi( 1.44, 3 );
   put p=;
run;
```

The output from this DATA step run is $p = 0.6961858724$. With this p-value you cannot reject the hypothesis of symmetry. This test is asymptotically valid.

Example 26.3: Vector AR(1) Estimation

This example shows the estimation of a two-variable vector AR(1) error process for the Grunfeld model (Grunfeld and Griliches 1960) by using the %AR macro. First, the full model is estimated. Second, the model is estimated with the restriction that the errors are univariate AR(1) instead of a vector process. The following statements produce Output 26.3.1 through Output 26.3.5.

```
data grunfeld;
   input year gei gef gec whi whf whc;
   label gei = 'Gross Investment GE'
         gec = 'Capital Stock Lagged GE'
         gef = 'Value of Outstanding Shares GE Lagged'
         whi = 'Gross Investment WH'
         whc = 'Capital Stock Lagged WH'
         whf = 'Value of Outstanding Shares Lagged WH';
datalines;
1935     33.1      1170.6    97.8      12.93     191.5     1.8
1936     45.0      2015.8    104.4     25.90     516.0     .8

   ... more lines ...


title1 'Example of Vector AR(1) Error Process Using Grunfeld''s Model';
/* Note: GE stands for General Electric
         WH stands for Westinghouse     */

proc model outmodel=grunmod;
   var gei whi gef gec whf whc;
   parms ge_int ge_f ge_c wh_int wh_f wh_c;
   label ge_int = 'GE Intercept'
         ge_f   = 'GE Lagged Share Value Coef'
         ge_c   = 'GE Lagged Capital Stock Coef'
         wh_int = 'WH Intercept'
         wh_f   = 'WH Lagged Share Value Coef'
         wh_c   = 'WH Lagged Capital Stock Coef';
   gei = ge_int + ge_f * gef + ge_c * gec;
   whi = wh_int + wh_f * whf + wh_c * whc;
run;
```

The preceding PROC MODEL step defines the structural model and stores it in the model file named GRUNMOD.

The following PROC MODEL step reads in the model, adds the vector autoregressive terms using %AR, and requests SUR estimation by using the FIT statement.

```
title2 'With Unrestricted Vector AR(1) Error Process';

proc model data=grunfeld model=grunmod;
   %ar( ar, 1, gei whi )
   fit gei whi / sur;
run;
```

The final PROC MODEL step estimates the restricted model, as shown in the following statements:

```
title2 'With restricted AR(1) Error Process';

proc model data=grunfeld model=grunmod;
   %ar( gei, 1 )
   %ar( whi, 1)
   fit gei whi / sur;
run;
```

Output 26.3.1 Model Summary for the Unrestricted Model

Example of Vector AR(1) Error Process Using Grunfeld's Model With Unrestricted Vector AR(1) Error Process

The MODEL Procedure

Model Summary	
Model Variables	6
Parameters	10
Equations	2
Number of Statements	7

Model Variables	gei whi gef gec whf whc
Parameters(Value)	ge_int ge_f ge_c wh_int wh_f wh_c ar_l1_1_1(0) ar_l1_1_2(0) ar_l1_2_1(0) ar_l1_2_2(0)
Equations	gei whi

The 2 Equations to Estimate	
gei =	F(ge_int, ge_f, ge_c, wh_int, wh_f, wh_c, ar_l1_1_1, ar_l1_1_2)
whi =	F(ge_int, ge_f, ge_c, wh_int, wh_f, wh_c, ar_l1_2_1, ar_l1_2_2)

NOTE: At SUR Iteration 9 CONVERGE=0.001 Criteria Met.

Output 26.3.2 Estimation Summary for the Unrestricted Model

Example of Vector AR(1) Error Process Using Grunfeld's Model With Unrestricted Vector AR(1) Error Process

The MODEL Procedure
SUR Estimation Summary

Data Set Options
DATA= GRUNFELD

Output 26.3.2 *continued*

Minimization Summary	
Parameters Estimated	10
Method	Gauss
Iterations	9

Final Convergence Criteria	
R	0.000609
PPC(wh_int)	0.002798
RPC(wh_int)	0.005411
Object	6.243E-7
Trace(S)	720.2454
Objective Value	1.374476

Observations Processed	
Read	20
Solved	20

Output 26.3.3 Estimation Results for the Unrestricted Model

Example of Vector AR(1) Error Process Using Grunfeld's Model With Unrestricted Vector AR(1) Error Process

The MODEL Procedure

Nonlinear SUR Summary of Residual Errors

Equation	DF Model	DF Error	SSE	MSE	Root MSE	R-Square	Adj R-Sq	Label
gei	5	15	9374.5	625.0	24.9993	0.7910	0.7352	Gross Investment GE
whi	5	15	1429.2	95.2807	9.7612	0.7940	0.7391	Gross Investment WH

Nonlinear SUR Parameter Estimates

Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|	Label
ge_int	-42.2858	30.5284	-1.39	0.1863	GE Intercept
ge_f	0.049894	0.0153	3.27	0.0051	GE Lagged Share Value Coef
ge_c	0.123946	0.0458	2.70	0.0163	GE Lagged Capital Stock Coef
wh_int	-4.68931	8.9678	-0.52	0.6087	WH Intercept
wh_f	0.068979	0.0182	3.80	0.0018	WH Lagged Share Value Coef
wh_c	0.019308	0.0754	0.26	0.8015	WH Lagged Capital Stock Coef
ar_l1_1_1	0.990902	0.3923	2.53	0.0233	AR(ar) gei: LAG1 parameter for gei
ar_l1_1_2	-1.56252	1.0882	-1.44	0.1716	AR(ar) gei: LAG1 parameter for whi
ar_l1_2_1	0.244161	0.1783	1.37	0.1910	AR(ar) whi: LAG1 parameter for gei
ar_l1_2_2	-0.23864	0.4957	-0.48	0.6372	AR(ar) whi: LAG1 parameter for whi

Output 26.3.4 Model Summary for the Restricted Model

Example of Vector AR(1) Error Process Using Grunfeld's Model With restricted AR(1) Error Process

The MODEL Procedure

Model Summary	
Model Variables	6
Parameters	8
Equations	2
Number of Statements	7

Model Variables	gei whi gef gec whf whc
Parameters(Value)	ge_int ge_f ge_c wh_int wh_f wh_c gei_l1(0) whi_l1(0)
Equations	gei whi

Output 26.3.5 Estimation Results for the Restricted Model

Example of Vector AR(1) Error Process Using Grunfeld's Model With restricted AR(1) Error Process

The MODEL Procedure

Nonlinear SUR Summary of Residual Errors

Equation	DF Model	DF Error	SSE	MSE	Root MSE	R-Square	Adj R-Sq	Label
gei	4	16	10558.8	659.9	25.6890	0.7646	0.7204	Gross Investment GE
whi	4	16	1669.8	104.4	10.2157	0.7594	0.7142	Gross Investment WH

Nonlinear SUR Parameter Estimates

Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|	Label
ge_int	-30.1239	29.7227	-1.01	0.3259	GE Intercept
ge_f	0.043527	0.0149	2.93	0.0099	GE Lagged Share Value Coef
ge_c	0.119206	0.0423	2.82	0.0124	GE Lagged Capital Stock Coef
wh_int	3.112671	9.2765	0.34	0.7416	WH Intercept
wh_f	0.053932	0.0154	3.50	0.0029	WH Lagged Share Value Coef
wh_c	0.038246	0.0805	0.48	0.6410	WH Lagged Capital Stock Coef
gei_l1	0.482397	0.2149	2.24	0.0393	AR(gei) gei lag1 parameter
whi_l1	0.455711	0.2424	1.88	0.0784	AR(whi) whi lag1 parameter

Example 26.4: MA(1) Estimation

This example estimates parameters for an MA(1) error process for the Grunfeld model, using both the unconditional least squares and the maximum likelihood methods. The ARIMA procedure estimates for Westinghouse equation are shown for comparison. The output of the following statements is summarized in Output 26.4.1:

```
proc model outmodel=grunmod;
   var gei whi gef gec whf whc;
   parms ge_int ge_f ge_c wh_int wh_f wh_c;
   label ge_int = 'GE Intercept'
         ge_f   = 'GE Lagged Share Value Coef'
         ge_c   = 'GE Lagged Capital Stock Coef'
         wh_int = 'WH Intercept'
         wh_f   = 'WH Lagged Share Value Coef'
         wh_c   = 'WH Lagged Capital Stock Coef';
   gei = ge_int + ge_f * gef + ge_c * gec;
   whi = wh_int + wh_f * whf + wh_c * whc;
run;

title1 'Example of MA(1) Error Process Using Grunfeld''s Model';
title2 'MA(1) Error Process Using Unconditional Least Squares';

proc model data=grunfeld model=grunmod;
   %ma(gei,1, m=uls);
   %ma(whi,1, m=uls);
   fit whi gei start=( gei_m1 0.8 -0.8) / startiter=2;
run;
```

Output 26.4.1 PROC MODEL Results by Using ULS Estimation

Example of MA(1) Error Process Using Grunfeld's Model
MA(1) Error Process Using Unconditional Least Squares

The MODEL Procedure

Nonlinear OLS Summary of Residual Errors

Equation	DF Model	DF Error	SSE	MSE	Root MSE	R-Square	Adj R-Sq	Label
whi	4	16	1874.0	117.1	10.8224	0.7299	0.6793	Gross Investment WH
resid.whi		16	1295.6	80.9754	8.9986			Gross Investment WH
gei	4	16	13835.0	864.7	29.4055	0.6915	0.6337	Gross Investment GE
resid.gei		16	7646.2	477.9	21.8607			Gross Investment GE

Nonlinear OLS Parameter Estimates

Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|	Label
ge_int	-26.839	32.0908	-0.84	0.4153	GE Intercept
ge_f	0.038226	0.0150	2.54	0.0217	GE Lagged Share Value Coef
ge_c	0.137099	0.0352	3.90	0.0013	GE Lagged Capital Stock Coef
wh_int	3.680835	9.5448	0.39	0.7048	WH Intercept
wh_f	0.049156	0.0172	2.85	0.0115	WH Lagged Share Value Coef
wh_c	0.067271	0.0708	0.95	0.3559	WH Lagged Capital Stock Coef
gei_m1	-0.87615	0.1614	-5.43	<.0001	MA(gei) gei lag1 parameter
whi_m1	-0.75001	0.2368	-3.17	0.0060	MA(whi) whi lag1 parameter

The estimation summary from the following PROC ARIMA statements is shown in Output 26.4.2.

```
title2 'PROC ARIMA Using Unconditional Least Squares';

proc arima data=grunfeld;
   identify var=whi cross=(whf whc ) noprint;
   estimate q=1 input=(whf whc) method=uls maxiter=40;
run;
```

Output 26.4.2 PROC ARIMA Results by Using ULS Estimation

Example of MA(1) Error Process Using Grunfeld's Model
PROC ARIMA Using Unconditional Least Squares

The ARIMA Procedure

Unconditional Least Squares Estimation

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|	Lag	Variable	Shift
MU	3.68608	9.54425	0.39	0.7044	0	whi	0
MA1,1	-0.75005	0.23704	-3.16	0.0060	1	whi	0
NUM1	0.04914	0.01723	2.85	0.0115	0	whf	0
NUM2	0.06731	0.07077	0.95	0.3557	0	whc	0

Constant Estimate	3.686077
Variance Estimate	80.97535
Std Error Estimate	8.998631
AIC	149.0044
SBC	152.9873
Number of Residuals	20

The model stored in Example 26.3 is read in by using the MODEL= option and the moving-average terms are added using the %MA macro.

The MA(1) model using maximum likelihood is estimated by using the following statements:

```
title2 'MA(1) Error Process Using Maximum Likelihood ';

proc model data=grunfeld model=grunmod;
   %ma(gei,1, m=ml);
   %ma(whi,1, m=ml);
   fit whi gei;
run;
```

For comparison, the model is estimated by using PROC ARIMA as follows:

```
title2 'PROC ARIMA Using Maximum Likelihood ';

proc arima data=grunfeld;
   identify var=whi cross=(whf whc) noprint;
   estimate q=1 input=(whf whc) method=ml;
run;
```

PROC ARIMA does not estimate systems, so only one equation is evaluated.

The estimation results are shown in Output 26.4.3 and Output 26.4.4. The small differences in the parameter values between PROC MODEL and PROC ARIMA can be eliminated by tightening the convergence criteria for both procedures.

Output 26.4.3 PROC MODEL Results by Using ML Estimation

Example of MA(1) Error Process Using Grunfeld's Model
MA(1) Error Process Using Maximum Likelihood

The MODEL Procedure

Nonlinear OLS Summary of Residual Errors

Equation	DF Model	DF Error	SSE	MSE	Root MSE	R-Square	Adj R-Sq	Label
whi	4	16	1857.5	116.1	10.7746	0.7323	0.6821	Gross Investment WH
resid.whi		16	1344.0	84.0012	9.1652			Gross Investment WH
gei	4	16	13742.5	858.9	29.3071	0.6936	0.6361	Gross Investment GE
resid.gei		16	8095.3	506.0	22.4935			Gross Investment GE

Nonlinear OLS Parameter Estimates

Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|	Label
ge_int	-25.002	34.2933	-0.73	0.4765	GE Intercept
ge_f	0.03712	0.0161	2.30	0.0351	GE Lagged Share Value Coef
ge_c	0.137788	0.0380	3.63	0.0023	GE Lagged Capital Stock Coef
wh_int	2.946761	9.5638	0.31	0.7620	WH Intercept
wh_f	0.050395	0.0174	2.89	0.0106	WH Lagged Share Value Coef
wh_c	0.066531	0.0729	0.91	0.3749	WH Lagged Capital Stock Coef
gei_m1	-0.78516	0.1942	-4.04	0.0009	MA(gei) gei lag1 parameter
whi_m1	-0.69389	0.2540	-2.73	0.0148	MA(whi) whi lag1 parameter

Output 26.4.4 PROC ARIMA Results by Using ML Estimation

Example of MA(1) Error Process Using Grunfeld's Model
PROC ARIMA Using Maximum Likelihood

The ARIMA Procedure

Maximum Likelihood Estimation

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|	Lag	Variable	Shift
MU	2.95645	9.20752	0.32	0.7481	0	whi	0
MA1,1	-0.69305	0.25307	-2.74	0.0062	1	whi	0
NUM1	0.05036	0.01686	2.99	0.0028	0	whf	0
NUM2	0.06672	0.06939	0.96	0.3363	0	whc	0

Constant Estimate	2.956449
Variance Estimate	81.29645
Std Error Estimate	9.016455
AIC	148.9113
SBC	152.8942
Number of Residuals	20

Example 26.5: Polynomial Distributed Lags by Using %PDL

This example shows the use of the %PDL macro for polynomial distributed lag models. Simulated data is generated so that Y is a linear function of six lags of X, with the lag coefficients following a quadratic polynomial. The model is estimated by using a fourth-degree polynomial, both with and without endpoint constraints. The example uses simulated data generated from the following model:

$$y_t = 10 + \sum_{z=0}^{6} f(z) x_{t-z} + \epsilon$$

$$f(z) = -5z^2 + 1.5z$$

The LIST option prints the model statements added by the %PDL macro. The following statements generate simulated data as shown:

```
/*-------------------------------------------------------------*/
/*  Generate Simulated Data for a Linear Model with a PDL on X  */
/*        y = 10 + x(6,2) + e                                   */
/*        pdl(x) = -5.*(lg)**2 + 1.5*(lg) + 0.                  */
/*-------------------------------------------------------------*/
data pdl;
   pdl2=-5.; pdl1=1.5; pdl0=0;
   array zz(i) z0-z6;
   do i=1 to 7;
      z=i-1;
      zz=pdl2*z**2 + pdl1*z + pdl0;
      end;
   do n=-11 to 30;
      x  =10*ranuni(1234567)-5;
      pdl=z0*x + z1*xl1 + z2*xl2 + z3*xl3 + z4*xl4 + z5*xl5 + z6*xl6;
      e  =10*rannor(1234567);
      y  =10+pdl+e;
      if n>=1 then output;
      xl6=xl5; xl5=xl4; xl4=xl3; xl3=xl2; xl2=xl1; xl1=x;
      end;
run;

title1 'Polynomial Distributed Lag Example';
title3 'Estimation of PDL(6,4) Model-- No Endpoint Restrictions';

proc model data=pdl;
   parms int;                  /* declare the intercept parameter */
   %pdl( xpdl, 6, 4 )          /* declare the lag distribution */
   y = int + %pdl( xpdl, x );  /* define the model equation */
   fit y / list;               /* estimate the parameters */
run;
```

The LIST output for the model without endpoint restrictions is shown in Output 26.5.1. The first seven statements in the generated program are the polynomial expressions for lag parameters XPDL_L0 through XPDL_L6. The estimated parameters are INT, XPDL_0, XPDL_1, XPDL_2, XPDL_3, and XPDL_4.

Output 26.5.1 PROC MODEL Listing of Generated Program

Polynomial Distributed Lag Example

Estimation of PDL(6,4) Model-- No Endpoint Restrictions

The MODEL Procedure

Listing of Compiled Program Code

Stmt	Line:Col	Statement as Parsed
1	4825:14	XPDL_L0 = XPDL_0;
2	4825:14	XPDL_L1 = XPDL_0 + XPDL_1 + XPDL_2 + XPDL_3 + XPDL_4;
3	4825:14	XPDL_L2 = XPDL_0 + XPDL_1 * 2 + XPDL_2 * 2 ** 2 + XPDL_3 * 2 ** 3 + XPDL_4 * 2 ** 4;
4	4825:14	XPDL_L3 = XPDL_0 + XPDL_1 * 3 + XPDL_2 * 3 ** 2 + XPDL_3 * 3 ** 3 + XPDL_4 * 3 ** 4;
5	4825:14	XPDL_L4 = XPDL_0 + XPDL_1 * 4 + XPDL_2 * 4 ** 2 + XPDL_3 * 4 ** 3 + XPDL_4 * 4 ** 4;
6	4825:14	XPDL_L5 = XPDL_0 + XPDL_1 * 5 + XPDL_2 * 5 ** 2 + XPDL_3 * 5 ** 3 + XPDL_4 * 5 ** 4;
7	4825:14	XPDL_L6 = XPDL_0 + XPDL_1 * 6 + XPDL_2 * 6 ** 2 + XPDL_3 * 6 ** 3 + XPDL_4 * 6 ** 4;
8	4826:4	PRED.y = int + XPDL_L0 * x + XPDL_L1 * LAG1(x) + XPDL_L2 * LAG2(x) + XPDL_L3 * LAG3(x) + XPDL_L4 * LAG4(x) + XPDL_L5 * LAG5(x) + XPDL_L6 * LAG6(x);
8	4826:4	RESID.y = PRED.y - ACTUAL.y;
8	4826:4	ERROR.y = PRED.y - y;
9	4825:15	ESTIMATE XPDL_L0, XPDL_L1, XPDL_L2, XPDL_L3, XPDL_L4, XPDL_L5, XPDL_L6;
10	4825:15	_est0 = XPDL_L0;
11	4825:15	_est1 = XPDL_L1;
12	4825:15	_est2 = XPDL_L2;
13	4825:15	_est3 = XPDL_L3;
14	4825:15	_est4 = XPDL_L4;
15	4825:15	_est5 = XPDL_L5;
16	4825:14	_est6 = XPDL_L6;

The FIT results for the model without endpoint restrictions are shown in Output 26.5.2.

Output 26.5.2 PROC MODEL Results That Specify No Endpoint Restrictions

Polynomial Distributed Lag Example

Estimation of PDL(6,4) Model-- No Endpoint Restrictions

The MODEL Procedure

Nonlinear OLS Summary of Residual Errors

Equation	DF Model	DF Error	SSE	MSE	Root MSE	R-Square	Adj R-Sq
y	6	18	2070.8	115.0	10.7259	0.9998	0.9998

Output 26.5.2 *continued*

Nonlinear OLS Parameter Estimates

Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|	Label
int	9.621969	2.3238	4.14	0.0006	
XPDL_0	0.084374	0.7587	0.11	0.9127	PDL(XPDL,6,4) parameter for (L)**0
XPDL_1	0.749956	2.0936	0.36	0.7244	PDL(XPDL,6,4) parameter for (L)**1
XPDL_2	-4.196	1.6215	-2.59	0.0186	PDL(XPDL,6,4) parameter for (L)**2
XPDL_3	-0.21489	0.4253	-0.51	0.6195	PDL(XPDL,6,4) parameter for (L)**3
XPDL_4	0.016133	0.0353	0.46	0.6528	PDL(XPDL,6,4) parameter for (L)**4

Portions of the output produced by the following PDL model with endpoints of the model restricted to zero are presented in Output 26.5.3.

```
title3 'Estimation of PDL(6,4) Model-- Both Endpoint Restrictions';

proc model data=pdl ;
   parms int;                    /* declare the intercept parameter */
   %pdl( xpdl, 6, 4, r=both )    /* declare the lag distribution */
   y = int + %pdl( xpdl, x );    /* define the model equation */
   fit y /list;                  /* estimate the parameters */
run;
```

Output 26.5.3 PROC MODEL Results Specifying Both Endpoint Restrictions

Polynomial Distributed Lag Example

Estimation of PDL(6,4) Model-- Both Endpoint Restrictions

The MODEL Procedure

Nonlinear OLS Summary of Residual Errors

Equation	DF Model	DF Error	SSE	MSE	Root MSE	R-Square	Adj R-Sq
y	4	20	449868	22493.4	150.0	0.9596	0.9535

Nonlinear OLS Parameter Estimates

Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|	Label
int	17.08581	32.4032	0.53	0.6038	
XPDL_2	13.88433	5.4361	2.55	0.0189	PDL(XPDL,6,4) parameter for (L)**2
XPDL_3	-9.3535	1.7602	-5.31	<.0001	PDL(XPDL,6,4) parameter for (L)**3
XPDL_4	1.032421	0.1471	7.02	<.0001	PDL(XPDL,6,4) parameter for (L)**4

Note that XPDL_0 and XPDL_1 are not shown in the estimate summary. They were used to satisfy the endpoint restrictions analytically by the generated %PDL macro code. Their values can be determined by back substitution.

To estimate the PDL model with one or more of the polynomial terms dropped, specify the largest degree of the polynomial desired with the %PDL macro and use the DROP= option in the FIT statement to remove the unwanted terms. The dropped parameters should be set to 0. The following PROC MODEL statements demonstrate estimation with a PDL of degree 2 without the 0th order term.

```
title3 'Estimation of PDL(6,2) Model-- With XPDL_0 Dropped';

proc model data=pdl list;
   parms int;                    /* declare the intercept parameter */
   %pdl( xpdl, 6, 2 )            /* declare the lag distribution */
   y = int + %pdl( xpdl, x );    /* define the model equation */
   xpdl_0 =0;
   fit y drop=xpdl_0;            /* estimate the parameters */
run;
```

The results from this estimation are shown in Output 26.5.4.

Output 26.5.4 PROC MODEL Results That Specify %PDL(XPDL, 6, 2)

Polynomial Distributed Lag Example

Estimation of PDL(6,2) Model-- With XPDL_0 Dropped

The MODEL Procedure

Nonlinear OLS Summary of Residual Errors

Equation	DF Model	DF Error	SSE	MSE	Root MSE	R-Square	Adj R-Sq
y	3	21	2114.1	100.7	10.0335	0.9998	0.9998

Nonlinear OLS Parameter Estimates

Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|	Label
int	9.536382	2.1685	4.40	0.0003	
XPDL_1	1.883315	0.3159	5.96	<.0001	PDL(XPDL,6,2) parameter for (L)**1
XPDL_2	-5.08827	0.0656	-77.56	<.0001	PDL(XPDL,6,2) parameter for (L)**2

Example 26.6: General Form Equations

Data for this example are generated. General form equations are estimated and forecast by using PROC MODEL. The system is a basic supply and demand model.

The following statements specify the form of the model:

```
title1 "General Form Equations for Supply-Demand Model";

proc model outmodel=model;
   var price quantity income unitcost;
   parms d0-d2 s0-s2;
   eq.demand=d0+d1*price+d2*income-quantity;
   eq.supply=s0+s1*price+s2*unitcost-quantity;
run;
```

Three data sets are used in this example. The first data set, HISTORY, is used to estimate the parameters of the model. The ASSUME data set is used to produce a forecast of PRICE and QUANTITY. Notice that the ASSUME data set does not need to contain the variables PRICE and QUANTITY. The HISTORY data set is shown as follows:

```
data history;
   input year income unitcost price quantity;
datalines;
1976    2221.87    3.31220     0.17903    266.714
1977    2254.77    3.61647     0.06757    276.049
1978    2285.16    2.21601     0.82916    285.858

   ... more lines ...
```

The ASSUME data set is shown as follows:

```
data assume;
   input year income unitcost;
datalines;
1986    2571.87    2.31220
1987    2609.12    2.45633
1988    2639.77    2.51647
1989    2667.77    1.65617
1990    2705.16    1.01601
;
```

The third data set, GOAL, used in a forecast of PRICE and UNITCOST as a function of INCOME and QUANTITY is as follows:

```
data goal;
   input year income quantity;
datalines;
1986    2571.87     371.4
1987    2721.08     416.5
1988    3327.05     597.3
1989    3885.85     764.1
1990    3650.98     694.3
;
```

The following statements fit the model to the HISTORY data set and solve the fitted model for the ASSUME data set.

```
proc model model=model outmodel=model;

/* estimate the model parameters */
   fit supply demand / data=history outest=est n2sls;
   instruments income unitcost year;
run;

/* produce forecasts for income and unitcost assumptions */
   solve price quantity / data=assume out=pq;
run;

title2 "Parameter Estimates for the System";
proc print data=est;
run;
```

```
title2 "Price Quantity Solution";
proc print data=pq;
run;
```

The model summary of the supply and demand model is shown as follows:

Output 26.6.1 Model Summary

General Form Equations for Supply-Demand Model

The MODEL Procedure

Model Summary	
Model Variables	4
Parameters	6
Equations	2
Number of Statements	3

Model Variables	price quantity income unitcost
Parameters	d0 d1 d2 s0 s1 s2
Equations	demand supply

The 2 Equations to Estimate	
supply =	F(s0(1), s1(price), s2(unitcost))
demand =	F(d0(1), d1(price), d2(income))
Instruments	1 income unitcost year

The estimation results are shown in Output 26.6.2 and the OUTEST= data set is show in Output 26.6.3. The output data set produced by the SOLVE statement is shown in Output 26.6.4.

Output 26.6.2 Output from the FIT Statement

General Form Equations for Supply-Demand Model

The MODEL Procedure

Nonlinear 2SLS Summary of Residual Errors							
Equation	DF Model	DF Error	SSE	MSE	Root MSE	R-Square	Adj R-Sq
supply	3	7	3.3240	0.4749	0.6891		
demand	3	7	1.0829	0.1547	0.3933		

Nonlinear 2SLS Parameter Estimates				
Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|
d0	-395.887	4.1841	-94.62	<.0001
d1	0.717328	0.5673	1.26	0.2466
d2	0.298061	0.00187	159.65	<.0001
s0	-107.62	4.1780	-25.76	<.0001
s1	201.5711	1.5977	126.16	<.0001
s2	102.2116	1.1217	91.12	<.0001

Output 26.6.3 Listing of OUTEST= Data Set Created in the FIT Statement

General Form Equations for Supply-Demand Model
Parameter Estimates for the System

Obs	_NAME_	_TYPE_	_STATUS_	_NUSED_	d0	d1	d2	s0	s1	s2
1		2SLS	0 Converged	10	-395.887	0.71733	0.29806	-107.620	201.571	102.212

Output 26.6.4 Listing of OUT= Data Set Created in the First SOLVE Statement

General Form Equations for Supply-Demand Model
Price Quantity Solution

Obs	_TYPE_	_MODE_	_ERRORS_	price	quantity	income	unitcost	year
1	PREDICT	SIMULATE	0	1.20473	371.552	2571.87	2.31220	1986
2	PREDICT	SIMULATE	0	1.18666	382.642	2609.12	2.45633	1987
3	PREDICT	SIMULATE	0	1.20154	391.788	2639.77	2.51647	1988
4	PREDICT	SIMULATE	0	1.68089	400.478	2667.77	1.65617	1989
5	PREDICT	SIMULATE	0	2.06214	411.896	2705.16	1.01601	1990

The following statements produce the goal-seeking solutions for PRICE and UNITCOST by using the GOAL dataset.

```
title2 "Price Unitcost Solution";

/* produce goal-seeking solutions for
     income and quantity assumptions*/
proc model model=model;
   solve price unitcost / data=goal out=pc;
run;

proc print data=pc;
run;
```

The output data set produced by the final SOLVE statement is shown in Output 26.6.5.

Output 26.6.5 Listing of OUT= Data Set Created in the Second SOLVE Statement

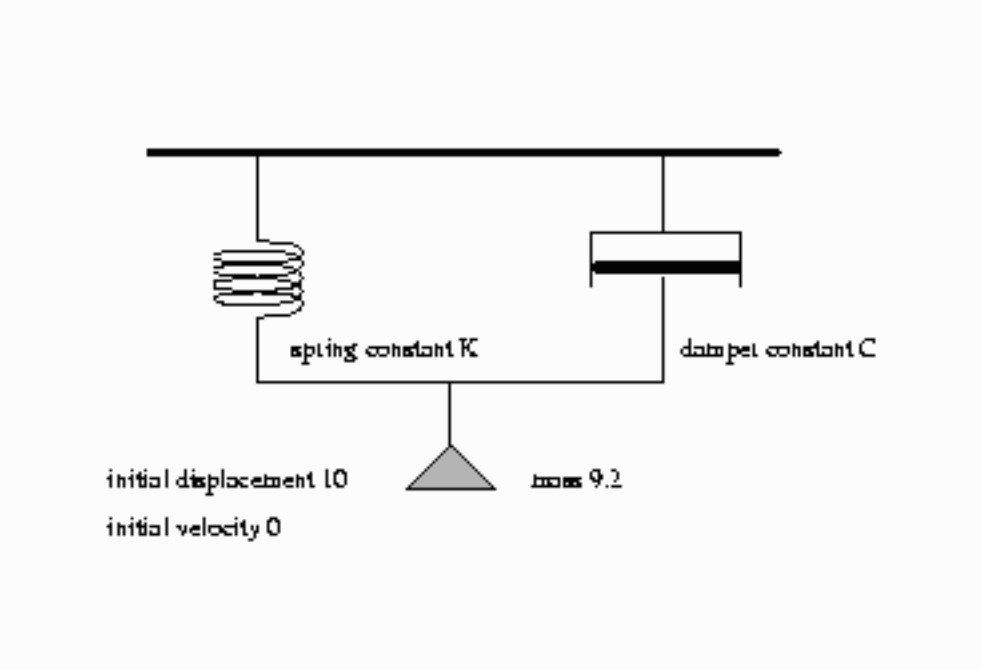
General Form Equations for Supply-Demand Model
Price Unitcost Solution

Obs	_TYPE_	_MODE_	_ERRORS_	price	quantity	income	unitcost	year
1	PREDICT	SIMULATE	0	0.99284	371.4	2571.87	2.72857	1986
2	PREDICT	SIMULATE	0	1.86594	416.5	2721.08	1.44798	1987
3	PREDICT	SIMULATE	0	2.12230	597.3	3327.05	2.71130	1988
4	PREDICT	SIMULATE	0	2.46166	764.1	3885.85	3.67395	1989
5	PREDICT	SIMULATE	0	2.74831	694.3	3650.98	2.42576	1990

Example 26.7: Spring and Damper Continuous System

This model simulates the mechanical behavior of a spring and damper system shown in Figure 26.99.

Figure 26.99 Spring and Damper System Model



A mass is hung from a spring with spring constant K. The motion is slowed by a damper with damper constant C. The damping force is proportional to the velocity, while the spring force is proportional to the displacement.

This is actually a continuous system; however, the behavior can be approximated by a discrete time model. We approximate the differential equation

$$\frac{\partial\, disp}{\partial\, time} = velocity$$

with the difference equation

$$\frac{\Delta\, disp}{\Delta\, time} = velocity$$

This is rewritten as

$$\frac{disp - \mathrm{LAG}(disp)}{dt} = velocity$$

where *dt* is the time step used. In PROC MODEL, this is expressed with the program statement

```
disp = lag(disp) + vel * dt;
```

or

```
dert.disp = vel;
```

The first statement is simply a computing formula for Euler's approximation for the integral

$$disp = \int velocity\ dt$$

If the time step is small enough with respect to the changes in the system, the approximation is good. Although PROC MODEL does not have the variable step-size and error-monitoring features of simulators designed for continuous systems, the procedure is a good tool to use for less challenging continuous models.

The second form instructs the MODEL procedure to do the integration for you.

This model is unusual because there are no exogenous variables, and endogenous data are not needed. Although you still need a SAS data set to count the simulation periods, no actual data are brought in.

Since the variables DISP and VEL are lagged, initial values specified in the VAR statement determine the starting state of the system. The mass, time step, spring constant, and damper constant are declared and initialized by a CONTROL statement as shown in the following statements:

```
title1 'Simulation of Spring-Mass-Damper System';

/*- Data to drive the simulation time periods ---*/
data one;
   do n=1 to 100;
      output;
   end;
run;

proc model data=one outmodel=spring;
   var      force -200  disp  10  vel  0  accel -20  time 0;
   control  mass   9.2  c    1.5  dt  .1  k      20;
   force = -k * disp -c * vel;
   disp  = lag(disp) + vel * dt;
   vel   = lag(vel) + accel * dt;
   accel = force / mass;
   time  = lag(time) + dt;
run;
```

The displacement scale is zeroed at the point where the force of gravity is offset, so the acceleration of the gravity constant is omitted from the force equation. The control variable C and K represent the damper and the spring constants respectively.

The model is simulated three times, and the simulation results are written to output data sets. The first run uses the original initial conditions specified in the VAR statement. In the second run, the initial displacement is doubled; the results show that the period of the motion is unaffected by the amplitude. In the third run, the DERT. syntax is used to do the integration. Notice that the path of the displacement is close to the old path, indicating that the original time step is short enough to yield an accurate solution. These simulations are performed by the following statements:

```
proc model data=one model=spring;
   title2 "Simulation of the model for the base case";
   control run '1';
   solve / out=a;
run;

   title2 "Simulation of the model with twice the initial displacement";
   control run '2';
   var disp 20;
   solve / out=b;
run;

data two;
   do time = 0 to 10 by .2; output;end;
run;

title2 "Simulation of the model using the dert. syntax";
proc model data=two;
   var      force -200  disp  10  vel  0  accel -20  time 0;
   control  mass   9.2  c    1.5  dt  .1  k      20;
   control run '3' ;
   force = -k * disp -c * vel;
   dert.disp  = vel ;
   dert.vel   = accel;
   accel = force / mass;
   solve / out=c;
   id time ;
run;
```

The output SAS data sets that contain the solution results are merged and the displacement time paths for the three simulations are plotted. The three runs are identified on the plot as 1, 2, and 3. The following statements produce Output 26.7.1 through Output 26.7.5.

```
data p;
   set a b c;
run;

title2 'Overlay Plot of All Three Simulations';
proc sgplot data=p;
   series x=time y=disp / group=run lineattrs=(pattern=1);
   xaxis values=(0 to 10 by 1);
   yaxis values=(-20 to 20 by 10);
run;
```

Output 26.7.1 Model Summary

Simulation of Spring-Mass-Damper System
Simulation of the model for the base case

The MODEL Procedure

Model Summary	
Model Variables	5
Control Variables	5
Equations	5
Number of Statements	6
Program Lag Length	1

Model Variables	force(-200) disp(10) vel(0) accel(-20) time(0)
Control Variables	mass(9.2) c(1.5) dt(0.1) k(20) run(1)
Equations	force disp vel accel time

Output 26.7.2 Printed Output Produced by PROC MODEL SOLVE Statements

Simulation of Spring-Mass-Damper System
Simulation of the model for the base case

The MODEL Procedure
Dynamic Simultaneous Simulation

Data Set Options	
DATA=	ONE
OUT=	A

Solution Summary	
Variables Solved	5
Simulation Lag Length	1
Solution Method	NEWTON
CONVERGE=	1E-8
Maximum CC	8.68E-15
Maximum Iterations	1
Total Iterations	99
Average Iterations	1

Observations Processed	
Read	100
Lagged	1
Solved	99
First	2
Last	100

Variables Solved For	force disp vel accel time

Output 26.7.3 Printed Output Produced by PROC MODEL SOLVE Statements

Simulation of Spring-Mass-Damper System
Simulation of the model with twice the initial displacement

The MODEL Procedure
Dynamic Simultaneous Simulation

Data Set Options	
DATA=	ONE
OUT=	B

Solution Summary	
Variables Solved	5
Simulation Lag Length	1
Solution Method	NEWTON
CONVERGE=	1E-8
Maximum CC	2.64E-14
Maximum Iterations	1
Total Iterations	99
Average Iterations	1

Observations Processed	
Read	100
Lagged	1
Solved	99
First	2
Last	100

Variables Solved For	force disp vel accel time

Output 26.7.4 Printed Output Produced by PROC MODEL SOLVE Statements

Simulation of Spring-Mass-Damper System
Simulation of the model using the dert. syntax

The MODEL Procedure
Simultaneous Simulation

Data Set Options	
DATA=	TWO
OUT=	C

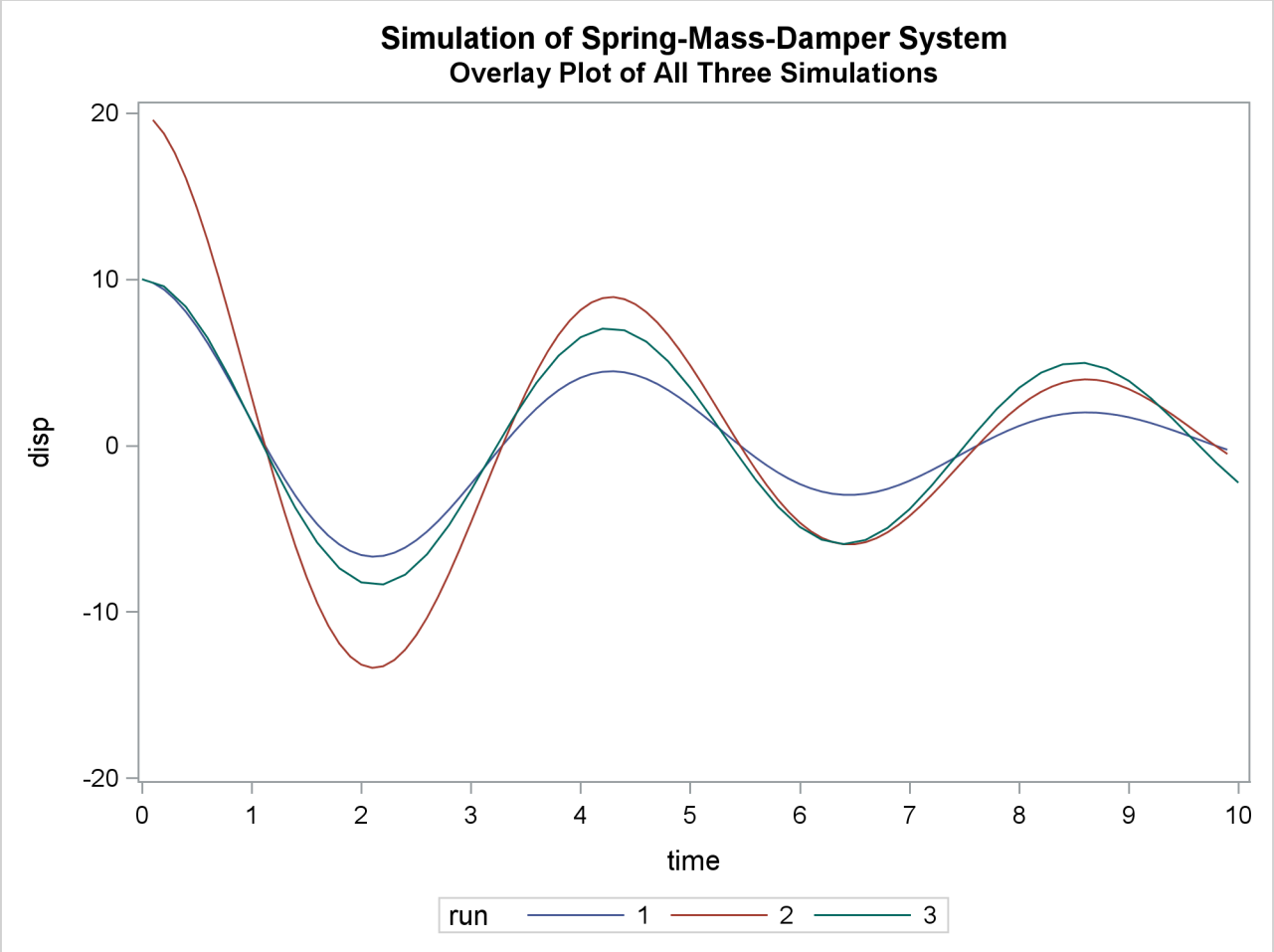
Solution Summary	
Variables Solved	4
Solution Method	NEWTON
Maximum Iterations	0

Output 26.7.4 *continued*

Observations Processed	
Read	51
Solved	51

Variables Solved For	force disp vel accel
ODE's	dert.disp dert.vel
Auxiliary Equations	force accel

Output 26.7.5 Overlay Plot of Three Simulations



Example 26.8: Nonlinear FIML Estimation

The data and model for this example were obtained from Bard (1974, pp. 133–138). The example is a two-equation econometric model used by Bodkin and Klein to fit U.S production data for the years 1909–1949. The model is the following:

$$g_1 = c_1 10^{c_2 z_4}(c_5 z_1^{-c_4} + (1 - c_5) z_2^{-c_4})^{-c_3/c_4} - z_3 = 0$$

$$g_2 = [c_5/(1-c_5)](z_1/z_2)^{(-1-c_4)} - z_5 = 0$$

where z_1 is capital input, z_2 is labor input, z_3 is real output, z_4 is time in years with 1929 as year zero, and z_5 is the ratio of price of capital services to wage scale. The c_i's are the unknown parameters. z_1 and z_2 are considered endogenous variables. A FIML estimation is performed by using the following statements:

```
data bodkin;
   input z1 z2 z3 z4 z5;
datalines;
1.33135 0.64629 0.4026 -20 0.24447
1.39235 0.66302 0.4084 -19 0.23454
1.41640 0.65272 0.4223 -18 0.23206

   ... more lines ...


title1 "Nonlinear FIML Estimation";

proc model data=bodkin;
   parms c1-c5;
   endogenous z1 z2;
   exogenous z3 z4 z5;

   eq.g1 = c1 * 10 **(c2 * z4) * (c5*z1**(-c4)+
           (1-c5)*z2**(-c4))**(-c3/c4) - z3;
   eq.g2 = (c5/(1-c5))*(z1/z2)**(-1-c4) -z5;

   fit g1 g2 / fiml ;
run;
```

When FIML estimation is selected, the log likelihood of the system is output as the objective value. The results of the estimation are shown in Output 26.8.1.

Output 26.8.1 FIML Estimation Results for U.S. Production Data

Nonlinear FIML Estimation

The MODEL Procedure

Nonlinear FIML Summary of Residual Errors

Equation	DF Model	DF Error	SSE	MSE	Root MSE	R-Square	Adj R-Sq
g1	4	37	0.0529	0.00143	0.0378		
g2	1	40	0.0173	0.000431	0.0208		

Nonlinear FIML Parameter Estimates

Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|
c1	0.58395	0.0218	26.76	<.0001
c2	0.005877	0.000673	8.74	<.0001
c3	1.3636	0.1148	11.87	<.0001
c4	0.473688	0.2699	1.75	0.0873
c5	0.446748	0.0596	7.49	<.0001

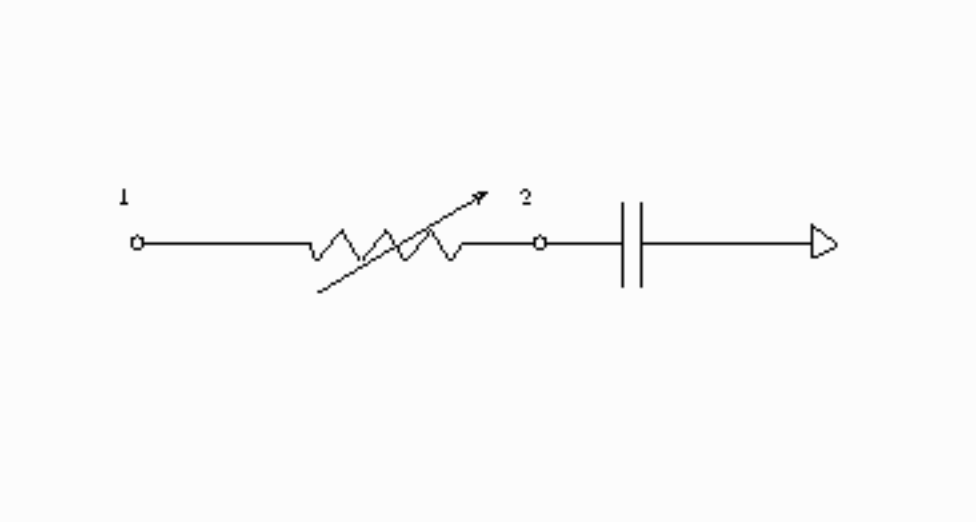
Output 26.8.1 *continued*

Number of Observations		Statistics for System	
Used	41	**Log Likelihood**	110.7773
Missing	0		

Example 26.9: Circuit Estimation

Consider the nonlinear circuit shown in Figure 26.100.

Figure 26.100 Nonlinear Resistor Capacitor Circuit



The theory of electric circuits is governed by Kirchhoff's laws: the sum of the currents flowing to a node is zero, and the net voltage drop around a closed loop is zero. In addition to Kirchhoff's laws, there are relationships between the current I through each element and the voltage drop V across the elements. For the circuit in Figure 26.100, the relationships are

$$C\frac{dV}{dt} = I$$

for the capacitor and

$$V = (R_1 + R_2(1 - \exp(-V)))I$$

for the nonlinear resistor. The following differential equation describes the current at node 2 as a function of time and voltage for this circuit:

$$C\frac{dV_2}{dt} - \frac{V_1 - V_2}{R_1 + R_2(1 - \exp(-V))} = 0$$

This equation can be written in the form

$$\frac{dV_2}{dt} = \frac{V_1 - V_2}{(R_1 + R_2(1 - \exp(-V)))C}$$

Consider the following data.

```
data circ;
   input v2 v1 time@@;
datalines;
-0.00007 0.0 0.0000000001 0.00912 0.5 0.0000000002
 0.03091 1.0 0.0000000003 0.06419 1.5 0.0000000004
 0.11019 2.0 0.0000000005 0.16398 2.5 0.0000000006
 0.23048 3.0 0.0000000007 0.30529 3.5 0.0000000008
 0.39394 4.0 0.0000000009 0.49121 4.5 0.0000000010
 0.59476 5.0 0.0000000011 0.70285 5.0 0.0000000012
 0.81315 5.0 0.0000000013 0.90929 5.0 0.0000000014
 1.01412 5.0 0.0000000015 1.11386 5.0 0.0000000016
 1.21106 5.0 0.0000000017 1.30237 5.0 0.0000000018
 1.40461 5.0 0.0000000019 1.48624 5.0 0.0000000020
 1.57894 5.0 0.0000000021 1.66471 5.0 0.0000000022
;
```

You can estimate the parameters in the preceding equation by using the following SAS statements:

```
title1 'Circuit Model Estimation Example';

proc model data=circ mintimestep=1.0e-23;
   parm R2 2000  R1 4000 C 5.0e-13;
   dert.v2 = (v1-v2)/((r1 + r2*(1-exp( -(v1-v2)))) * C);
   fit v2;
run;
```

The results of the estimation are shown in Output 26.9.1.

Output 26.9.1 Circuit Estimation

Circuit Model Estimation Example

The MODEL Procedure

Nonlinear OLS Parameter Estimates

Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|
R2	3002.471	1517.1	<------	Biased
R1	4984.842	1466.8	<------	Biased
C	5E-13	0	<------	Biased

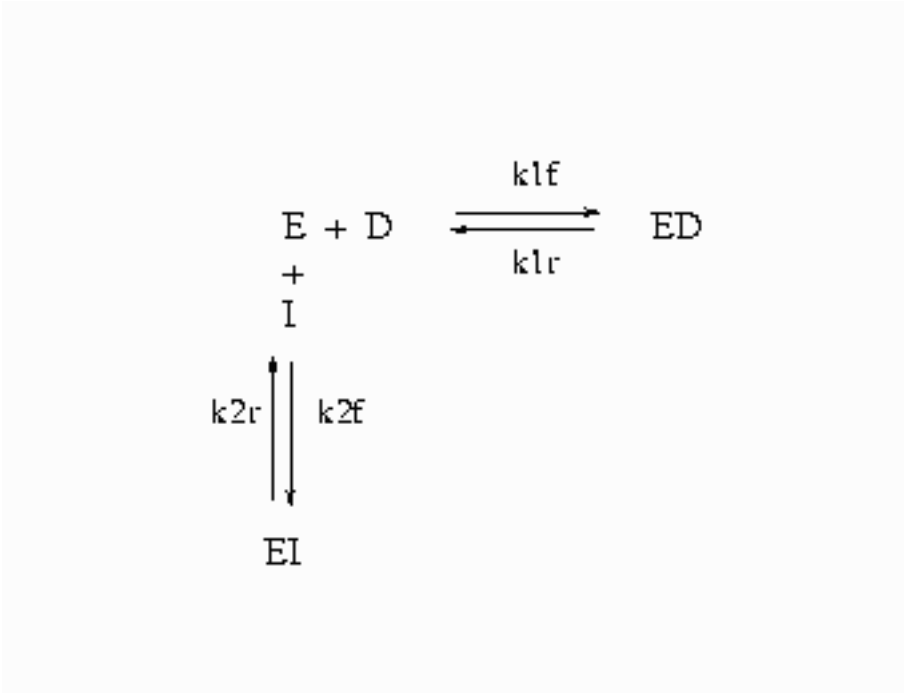
Note: The model was singular. Some estimates are marked 'Biased'.

In this case, the model equation is such that there is linear dependency that causes biased results and inflated variances. The Jacobian matrix is singular or nearly singular, but eliminating one of the parameters is not a solution in this case.

Example 26.10: Systems of Differential Equations

The following is a simplified reaction scheme for the competitive inhibitors with recombinant human renin (Morelock et al. 1995).

Figure 26.101 Competitive Inhibition of Recombinant Human Renin



In Figure 26.101, E=enzyme, D=probe, and I=inhibitor.

The differential equations that describe this reaction scheme are as follows:

$$\frac{dD}{dt} = k1r*ED - k1f*E*D$$

$$\frac{dED}{dt} = k1f*E*D - k1r*ED$$

$$\frac{dE}{dt} = k1r*ED - k1f*E*D + k2r*EI - k2f*E*I$$

$$\frac{dEI}{dt} = k2f*E*I - k2r*EI$$

$$\frac{dI}{dt} = k2r*EI - k2f*E*I$$

For this system, the initial values for the concentrations are derived from equilibrium considerations (as a function of parameters) or are provided as known values.

The experiment used to collect the data was carried out in two ways; preincubation (type='disassoc') and no preincubation (type='assoc'). The data also contain repeated measurements. The data contain values for fluorescence F, which is a function of concentration. Since there are no direct data for the concentrations, all the differential equations are simulated dynamically.

The SAS statements used to fit this model are as follows:

```
title1 'Systems of Differential Equations Example';

proc sort data=fit;
   by type time;
run;

%let k1f = 6.85e6  ;
%let k1r = 3.43e-4 ;
%let k2f = 1.8e7  ;
%let k2r = 2.1e-2 ;

%let qf  = 2.1e8  ;
%let qb  = 4.0e9  ;

%let dt  = 5.0e-7 ;
%let et  = 5.0e-8 ;
%let it  = 8.05e-6 ;

proc model data=fit;

   parameters qf  = 2.1e8
              qb  = 4.0e9
              k2f = 1.8e5
              k2r = 2.1e-3
              l   = 0;

              k1f = 6.85e6;
              k1r = 3.43e-4;

      /* Initial values for concentrations */
   control dt 5.0e-7
           et 5.0e-8
           it 8.05e-6;

      /* Association initial values ---------------*/
   if type = 'assoc' and time=0 then do;
      ed = 0;
      /* solve quadratic equation ----------*/
      a = 1;
      b = -(&it+&et+(k2r/k2f));
      c = &it*&et;
      ei = (-b-(((b**2)-(4*a*c))**.5))/(2*a);
      d = &dt-ed;
      i = &it-ei;
      e = &et-ed-ei;
      end;

      /* Disassociation initial values ----------*/
   if type = 'disassoc' and time=0 then do;
         ei = 0;
         a = 1;
         b = -(&dt+&et+(&k1r/&k1f));
         c = &dt*&et;
```

```
         ed = (-b-(((b**2)-(4*a*c))**.5))/(2*a);
         d = &dt-ed;
         i = &it-ei;
         e = &et-ed-ei;
      end;

   if time ne 0 then do;

      dert.d = k1r* ed  - k1f *e *d;

      dert.ed = k1f* e *d - k1r*ed;

      dert.e = k1r* ed - k1f* e * d  + k2r * ei - k2f * e *i;

      dert.ei = k2f* e *i - k2r * ei;

      dert.i = k2r * ei - k2f* e *i;

      end;

   /* L - offset between curves  */
   if type = 'disassoc' then
      F = (qf*(d-ed)) + (qb*ed) -L;
   else
      F = (qf*(d-ed)) + (qb*ed);

   fit F / method=marquardt;
run;
```

This estimation requires the repeated simulation of a system of 41 differential equations (5 base differential equations and 36 differential equations to compute the partials with respect to the parameters).

The results of the estimation are shown in Output 26.10.1.

Output 26.10.1 Kinetics Estimation

Systems of Differential Equations Example

The MODEL Procedure

Nonlinear OLS Summary of Residual Errors

Equation	DF Model	DF Error	SSE	MSE	Root MSE	R-Square	Adj R-Sq
f	5	797	2525.0	3.1681	1.7799	0.9980	0.9980

Nonlinear OLS Parameter Estimates

Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|
qf	2.0413E8	681443	299.55	<.0001
qb	4.2263E9	9133195	462.74	<.0001
k2f	6451186	866998	7.44	<.0001
k2r	0.007808	0.00103	7.55	<.0001
l	-5.76974	0.4138	-13.94	<.0001

Example 26.11: Monte Carlo Simulation

This example illustrates how the form of the error in a ODE model affects the results from a static and dynamic estimation. The differential equation studied is

$$\frac{dy}{dt} = a - ay$$

The analytical solution to this differential equation is

$$y = 1 - \exp(-at)$$

The first data set contains errors that are strictly additive and independent. The data for this estimation are generated by the following DATA step:

```
data drive1;
   a = 0.5;
   do iter=1 to 100;
      do time = 0 to 50;
         y = 1 - exp(-a*time) + 0.1 *rannor(123);
         output;
      end;
   end;
run;
```

The second data set contains errors that are cumulative in form.

```
data drive2;
   a = 0.5;
   yp = 1.0 + 0.01 *rannor(123);
   do iter=1 to 100;
      do time = 0 to 50;
         y = 1 - exp(-a)*(1 - yp);
         yp = y + 0.01 *rannor(123);
         output;
      end;
   end;
run;
```

The following statements perform the 100 static estimations for each data set:

```
title1 'Monte Carlo Simulation of ODE';

proc model data=drive1 noprint;
   parm a 0.5;
   dert.y = a - a * y;
   fit y / outest=est;
   by iter;
run;
```

Similar statements are used to produce 100 dynamic estimations with a fixed and an unknown initial value. The first value in the data set is used to simulate an error in the initial value. The following PROC UNIVARIATE statements process the estimations:

```
proc univariate data=est noprint;
   var a;
   output out=monte mean=mean p5=p5 p95=p95;
run;

proc print data=monte;
run;
```

The results of these estimations are summarized in Table 26.6.

Table 26.6 Monte Carlo Summary, A=0.5

Estimation Type	Additive Error			Cumulative Error		
	mean	p95	p5	mean	p95	p5
static	0.77885	1.03524	0.54733	0.57863	1.16112	0.31334
dynamic fixed	0.48785	0.63273	0.37644	3.8546E24	8.88E10	-51.9249
dynamic unknown	0.48518	0.62452	0.36754	641704.51	1940.42	-25.6054

For this example model, it is evident that the static estimation is the least sensitive to misspecification.

Example 26.12: Cauchy Distribution Estimation

In this example a nonlinear model is estimated by using the Cauchy distribution. Then a simulation is done for one observation in the data.

The following DATA step creates the data for the model.

```
/* Generate a Cauchy distributed Y */
data c;
   format date monyy.;
   call streaminit(156789);
   do t=0 to 20 by 0.1;
      date=intnx('month','01jun90'd,(t*10)-1);
      x=rand('normal');
      e=rand('cauchy') + 10 ;
      y=exp(4*x)+e;
      output;
   end;
run;
```

The model to be estimated is

$$\begin{aligned} y &= e^{-a\,x} + \epsilon \\ \epsilon &\sim \text{Cauchy}(nc) \end{aligned}$$

That is, the residuals of the model are distributed as a Cauchy distribution with noncentrality parameter nc.

The log likelihood for the Cauchy distribution is

$$\text{ll} = -\log \pi(1 + (x - nc)^2)$$

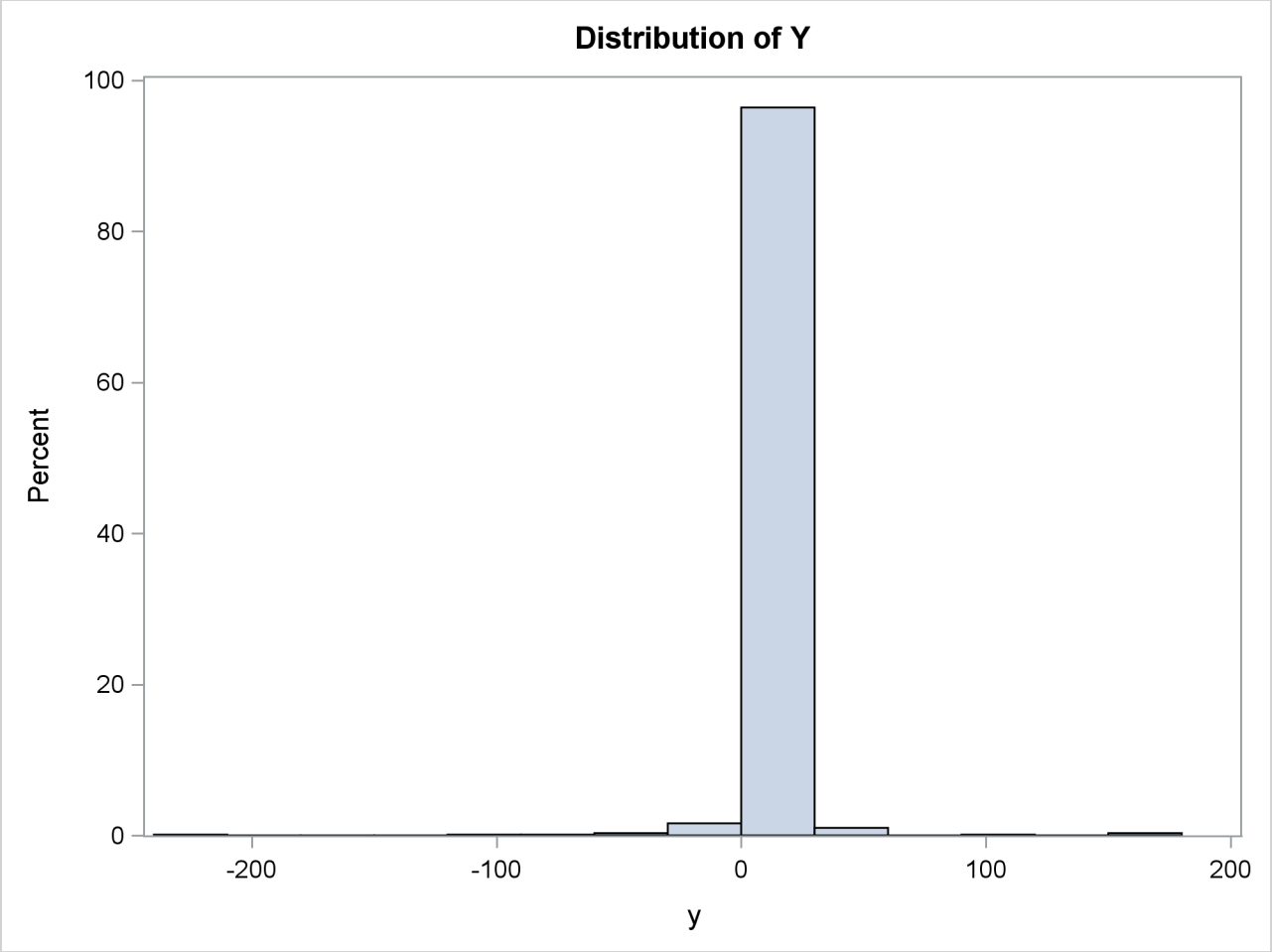
The following SAS statements specify the model and the log-likelihood function.

```
title1 'Cauchy Distribution';

proc model data=c ;
   dependent y;
   parm a -2 nc 4;
   y=exp(-a*x);

       /* Likelihood function for the residuals */
   obj = log(constant('pi')*(1+(-resid.y-nc)**2));

   errormodel y ~ general(obj) cdf=cauchy(nc);

   fit y / outsn=s1 method=marquardt;
   solve y / sdata=s1 data=c(obs=1) random=1000
             seed=256789 out=out1;
run;

title 'Distribution of Y';
proc sgplot data=out1;
   histogram y;
run;
```

The FIT statement uses the OUTSN= option to output the $\mathbf{\Sigma}$ matrix for residuals from the normal distribution. The $\mathbf{\Sigma}$ matrix is 1×1 and has value 1.0 because it is a correlation matrix. The OUTS= matrix is the scalar 2989.0. Because the distribution is univariate (no covariances), the OUTS= option would produce the same simulation results. The simulation is performed by using the SOLVE statement.

The distribution of y is shown in the following output.

Output 26.12.1 Distribution of Y



Example 26.13: Switching Regression Example

Take the usual linear regression problem

$$y = \mathbf{X}\beta + u$$

where Y denotes the n column vector of the dependent variable, $\mathbf{X}$ denotes the $(n \times k)$ matrix of independent variables, β denotes the k column vector of coefficients to be estimated, n denotes the number of observations ($i = 1, 2, \ldots, n$), and k denotes the number of independent variables.

You can take this basic equation and split it into two regimes, where the ith observation on y is generated by one regime or the other:

$$y_i = \sum_{j=1}^{k} \beta_{1j} X_{ji} + u_{1i} = x_i' \beta_1 + u_{1i}$$

$$y_i = \sum_{j=1}^{k} \beta_{2j} X_{ji} + u_{2i} = x_i' \beta_2 + u_{2i}$$

where x_{hi} and x_{hj} are the ith and jth observations, respectively, on x_h. The errors, u_{1i} and u_{2i}, are assumed to be distributed normally and independently with mean zero and constant variance. The variance for the first regime is σ_1^2, and the variance for the second regime is σ_2^2. If $\sigma_1^2 \neq \sigma_2^2$ and $\beta_1 \neq \beta_2$, the regression system given previously is thought to be switching between the two regimes.

The problem is to estimate β_1, β_2, σ_1, and σ_2 without knowing *a priori* which of the n values of the dependent variable, y, was generated by which regime. If it is known *a priori* which observations belong to which regime, a simple Chow test can be used to test $\sigma_1^2 = \sigma_2^2$ and $\beta_1 = \beta_2$.

Using Goldfeld and Quandt's D-method for switching regression, you can solve this problem. Assume that observations exist on some exogenous variables z_{1i}, z_{2i}, ..., z_{pi}, where z determines whether the ith observation is generated from one equation or the other. The equations are given as follows:

$$\begin{aligned} y_i &= x_i'\beta_1 + u_{1i} \quad \text{if} \sum_{j=1}^{p} \pi_j z_{ji} \leq 0 \\ y_i &= x_i'\beta_2 + u_{2i} \quad \text{if} \sum_{j=1}^{p} \pi_j z_{ji} > 0 \end{aligned}$$

where π_j are unknown coefficients to be estimated. Define $d(z_i)$ as a continuous approximation to a step function. Replacing the unit step function with a continuous approximation by using the cumulative normal integral enables a more practical method that produces consistent estimates.

$$d(z_i) = \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{\sum \pi_j z_{ji}} exp\left[-\frac{1}{2}\frac{\xi^2}{\sigma^2}\right] d\xi$$

D is the n dimensional diagonal matrix consisting of $d(z_i)$:

$$\mathbf{D} = \begin{bmatrix} d(z_1) & 0 & 0 & 0 \\ 0 & d(z_2) & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & d(z_n) \end{bmatrix}$$

The parameters to estimate are now the k β_1's, the k β_2's, σ_1^2, σ_2^2, p π's, and the σ introduced in the $d(z_i)$ equation. The σ can be considered as given *a priori*, or it can be estimated, in which case, the estimated magnitude provides an estimate of the success in discriminating between the two regimes (Goldfeld and Quandt 1976). Given the preceding equations, the model can be written as:

$$Y = (\mathbf{I} - \mathbf{D})\,\mathbf{X}\beta_1 + \mathbf{D}\mathbf{X}\beta_2 + W$$

where $W = (\mathbf{I} - \mathbf{D})U_1 + \mathbf{D}U_2$, and W is a vector of unobservable and heteroscedastic error terms. The covariance matrix of W is denoted by $\mathbf{\Omega}$, where $\mathbf{\Omega} = (\mathbf{I} - \mathbf{D})^2\sigma_1^2 + \mathbf{D}^2\sigma_2^2$. The maximum likelihood parameter estimates maximize the following log-likelihood function.

$$\begin{aligned} \log L = &-\frac{n}{2}\log 2\pi - \frac{1}{2}\log |\, \mathbf{\Omega} \,| - \\ &\frac{1}{2} * \left[[Y - (\mathbf{I} - \mathbf{D})\mathbf{X}\beta_1 - \mathbf{D}\mathbf{X}\beta_2]'\, \mathbf{\Omega}^{-1}\, [Y - (\mathbf{I} - \mathbf{D})\mathbf{X}\beta_1 - \mathbf{D}\mathbf{X}\beta_2]\right] \end{aligned}$$

As an example, you now can use this switching regression likelihood to develop a model of housing starts as a function of changes in mortgage interest rates. The data for this example are from the U.S. Census Bureau and cover the period from January 1973 to March 1999. The hypothesis is that there are different coefficients on your model based on whether the interest rates are going up or down.

So the model for z_i is

$$z_i = p * (\mathrm{rate}_i - \mathrm{rate}_{i-1})$$

where rate_i is the mortgage interest rate at time *i* and *p* is a scale parameter to be estimated.

The regression model is

$$\begin{aligned} \mathrm{starts}_i &= \mathrm{intercept}_1 + \mathrm{ar1} * \mathrm{starts}_{i-1} + \mathrm{djf1} * \mathrm{decjanfeb} \qquad z_i < 0 \\ \mathrm{starts}_i &= \mathrm{intercept}_2 + \mathrm{ar2} * \mathrm{starts}_{i-1} + \mathrm{djf2} * \mathrm{decjanfeb} \qquad z_i >= 0 \end{aligned}$$

where starts_i is the number of housing starts at month *i* and $\mathrm{decjanfeb}$ is a dummy variable that indicates that the current month is one of December, January, or February.

This model is written by using the following SAS statements:

```
title1 'Switching Regression Example';

proc model data=switch;
   parms sig1=10 sig2=10 int1 b11 b13 int2 b21 b23 p;
   bounds 0.0001 < sig1 sig2;

   decjanfeb = ( month(date) = 12 | month(date) <= 2 );

   a = p*dif(rate);       /* Upper bound of integral */
   d = probnorm(a);       /* Normal CDF as an approx of switch */

                          /* Regime 1 */
   y1 = int1 + zlag(starts)*b11 + decjanfeb *b13 ;
                          /* Regime 2 */
   y2 = int2 + zlag(starts)*b21 + decjanfeb *b23 ;
                          /* Composite regression equation */
   starts  = (1 - d)*y1 +  d*y2;

                         /* Resulting log-likelihood function */
   logL = (1/2)*( (log(2*3.1415)) +
        log( (sig1**2)*((1-d)**2)+(sig2**2)*(d**2) )
       + (resid.starts*( 1/( (sig1**2)*((1-d)**2)+
        (sig2**2)*(d**2) ) )*resid.starts) ) ;

   errormodel starts ~ general(logL);

   fit starts / method=marquardt converge=1.0e-5;

     /* Test for significant differences in the parms */
   test int1 = int2 ,/ lm;
   test b11 = b21 ,/ lm;
   test b13 = b23 ,/ lm;
```

```
   test sig1 = sig2 ,/ lm;

run;
```

Four TEST statements are added to test the hypothesis that the parameters are the same in both regimes. The parameter estimates and ANOVA table from this run are shown in Output 26.13.1.

Output 26.13.1 Parameter Estimates from the Switching Regression

Switching Regression Example

The MODEL Procedure

Nonlinear Liklhood Summary of Residual Errors

Equation	DF Model	DF Error	SSE	MSE	Root MSE	R-Square	Adj R-Sq	Label
starts	9	304	85878.0	282.5	16.8075	0.7806	0.7748	Housing Starts

Nonlinear Liklhood Parameter Estimates

Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|
sig1	15.47484	0.9476	16.33	<.0001
sig2	19.77808	1.2710	15.56	<.0001
int1	32.82221	5.9083	5.56	<.0001
b11	0.73952	0.0444	16.64	<.0001
b13	-15.4556	3.1912	-4.84	<.0001
int2	42.73348	6.8159	6.27	<.0001
b21	0.734117	0.0478	15.37	<.0001
b23	-22.5184	4.2985	-5.24	<.0001
p	25.94712	8.5205	3.05	0.0025

The test results shown in Output 26.13.2 suggest that the variance of the housing starts, SIG1 and SIG2, are significantly different in the two regimes. The tests also show a significant difference in the AR term on the housing starts.

Output 26.13.2 Test Results for Switching Regression

Test Results

Test	Type	Statistic	Pr > ChiSq	Label
Test0	L.M.	1.00	0.3185	int1 = int2
Test1	L.M.	15636	<.0001	b11 = b21
Test2	L.M.	1.45	0.2280	b13 = b23
Test3	L.M.	4.39	0.0361	sig1 = sig2

Example 26.14: Simulating from a Mixture of Distributions

This example illustrates how to perform a multivariate simulation by using models that have different error distributions. Three models are used. The first model has *t* distributed errors. The second model is a GARCH(1,1) model with normally distributed errors. The third model has a noncentral Cauchy distribution.

The following SAS statements generate the data for this example. The *t* and the CAUCHY data sets use a common seed so that those two series are correlated.

```
/* set distribution parameters */
%let df = 7.5;
%let sig1 = .5;
%let var2 = 2.5;

data t;
   format date monyy.;
   do date='1jun2001'd  to '1nov2002'd;
              /* t-distribution with df,sig1 */
      t = .05 * date + 5000  + &sig1*tinv(ranuni(1234),&df);
      output;
   end;
run;

data normal;
   format date monyy.;
   le = &var2;
   lv = &var2;
   do date='1jun2001'd  to '1nov2002'd;
              /* Normal with GARCH error structure */
      v = 0.0001 + 0.2 * le**2 + .75 * lv;
      e = sqrt( v) * rannor(12345) ;
      normal = 25 + e;
      le = e;
      lv = v;
      output;
   end;
run;

data cauchy;
   format date monyy.;
   PI = 3.1415926;
   do date='1jun2001'd  to '1nov2002'd;
      cauchy = -4 + tan((ranuni(1234) - 0.5) * PI);
      output;
   end;
run;
```

Since the multivariate joint likelihood is unknown, the models must be estimated separately. The residuals for each model are saved by using the OUT= option. Also, each model is saved by using the OUTMODEL= option. The ID statement is used to provide a variable in the residual data set to merge by. The XLAG function is used to model the GARCH(1,1) process. The XLAG function returns the lag of the first argument if it is nonmissing, otherwise it returns the second argument.

```
title1 't-distributed Errors Example';

proc model data=t outmod=tModel;
   parms df 10 vt 4;
   t = a * date + c;
   errormodel t ~ t( vt, df );
   fit t / out=tresid;
   id date;
run;
```

```
title1 'GARCH-distributed Errors Example';

proc model data=normal outmodel=normalModel;
   normal = b0 ;
   h.normal = arch0 + arch1 * xlag(resid.normal **2 , mse.normal)
            + GARCH1 * xlag(h.normal, mse.normal);

   fit normal /fiml out=nresid;
   id date;
run;

title1 'Cauchy-distributed Errors Example';

proc model data=cauchy outmod=cauchyModel;
   parms nc = 1;
   /* nc is noncentrality parm to Cauchy dist */
   cauchy = nc;
   obj = log(1+resid.cauchy**2 * 3.1415926);
   errormodel cauchy ~ general(obj) cdf=cauchy(nc);

   fit cauchy / out=cresid;
   id date;
run;
```

The simulation requires a covariance matrix created from normal residuals. The following DATA step statements use the inverse CDFs of the *t* and Cauchy distributions to convert the residuals to the normal distribution. The CORR procedure is used to create a correlation matrix that uses the converted residuals.

```
/* Merge and normalize the 3 residual data sets */
data c; merge tresid nresid cresid; by date;
   t = probit(cdf("T", t/sqrt(0.2789), 16.58 ));
   cauchy = probit(cdf("CAUCHY", cauchy, -4.0623));
run;

proc corr data=c out=s;
   var t normal cauchy;
run;
```

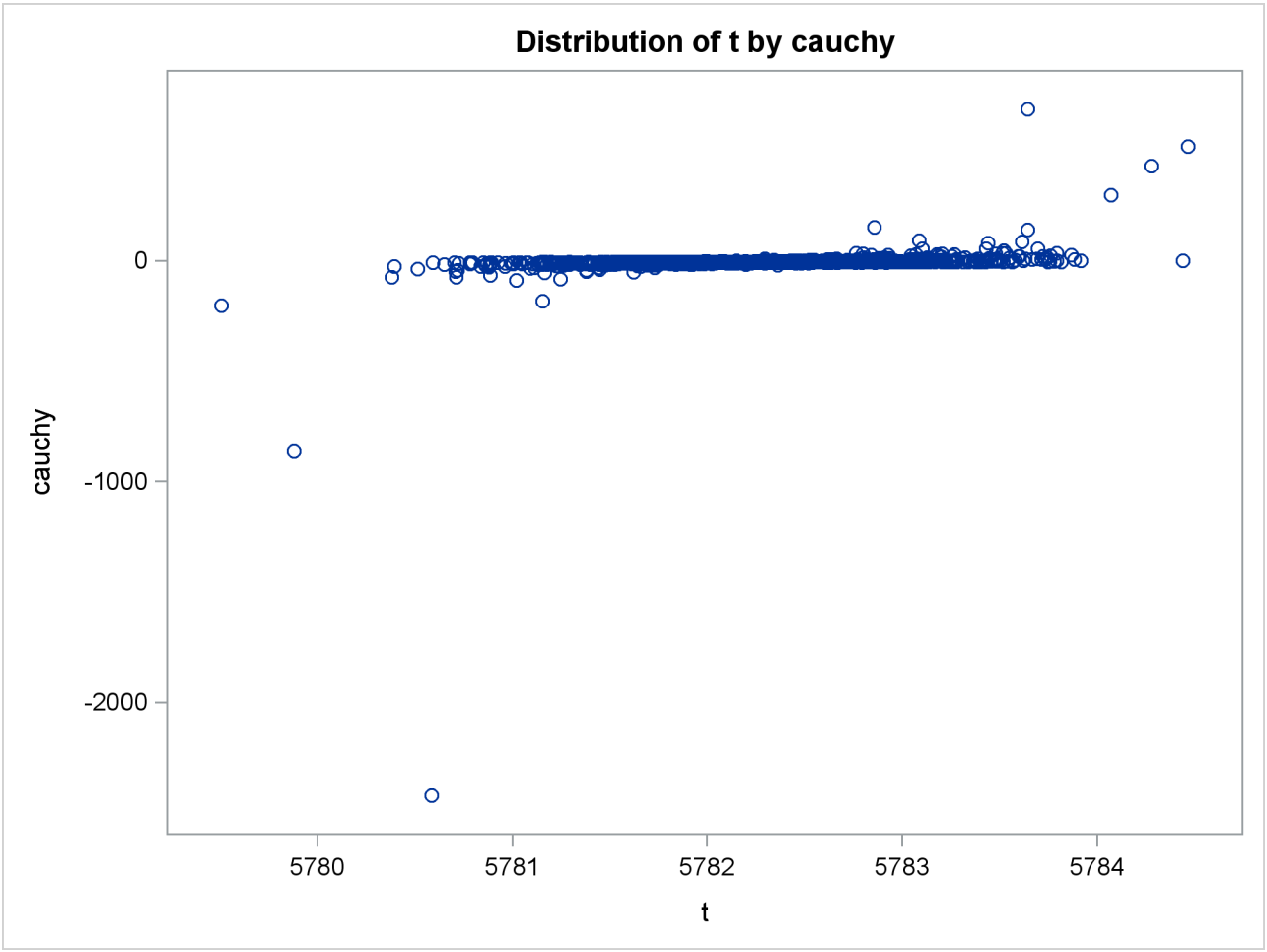
Now the models can be simulated together by using the MODEL procedure SOLVE statement. The data set created by the CORR procedure is used as the correlation matrix.

```
title1 'Simulating Equations with Different Error Distributions';

/* Create one observation driver data set */
data sim; merge t normal cauchy; by date;
data sim; set sim(firstobs = 519 );

proc model data=sim model=( tModel normalModel cauchyModel );
   errormodel t ~ t( vt, df );
   errormodel cauchy ~ cauchy(nc);
   solve t cauchy normal / random=2000 seed=1962 out=monte
                           sdata=s(where=(_type_="CORR"));
run;
```

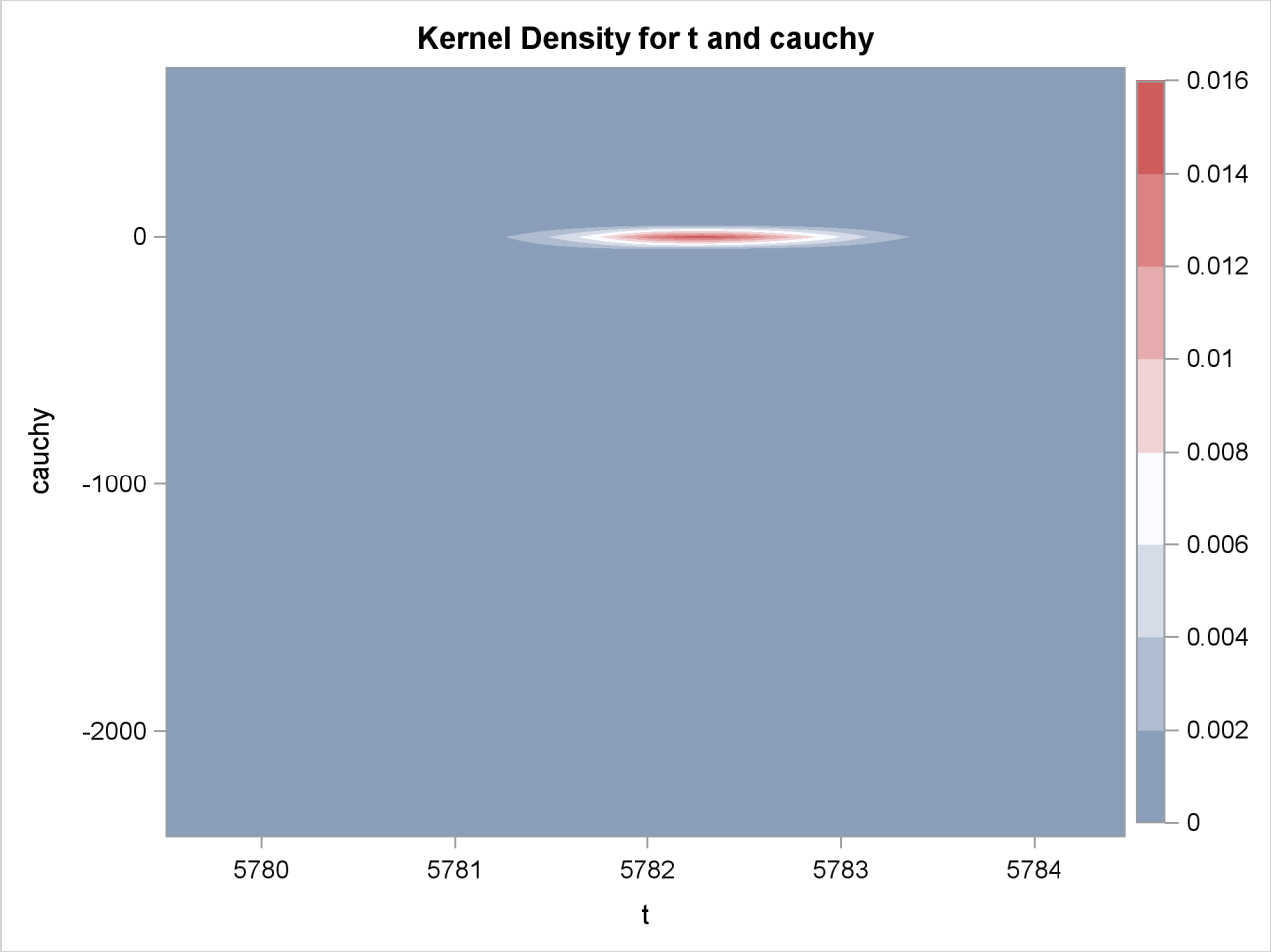
An estimation of the joint density of the *t* and Cauchy distribution is created by using the KDE procedure. Bounds are placed on the Cauchy dimension because of its fat tail behavior. The joint PDF is shown in Output 26.14.1.

```
title "T and Cauchy Distribution";

proc kde data=monte;
   univar t       / out=t_dens;
   univar cauchy  / out=cauchy_dens;
   bivar t cauchy / out=density
                    plots=all;
run;
```

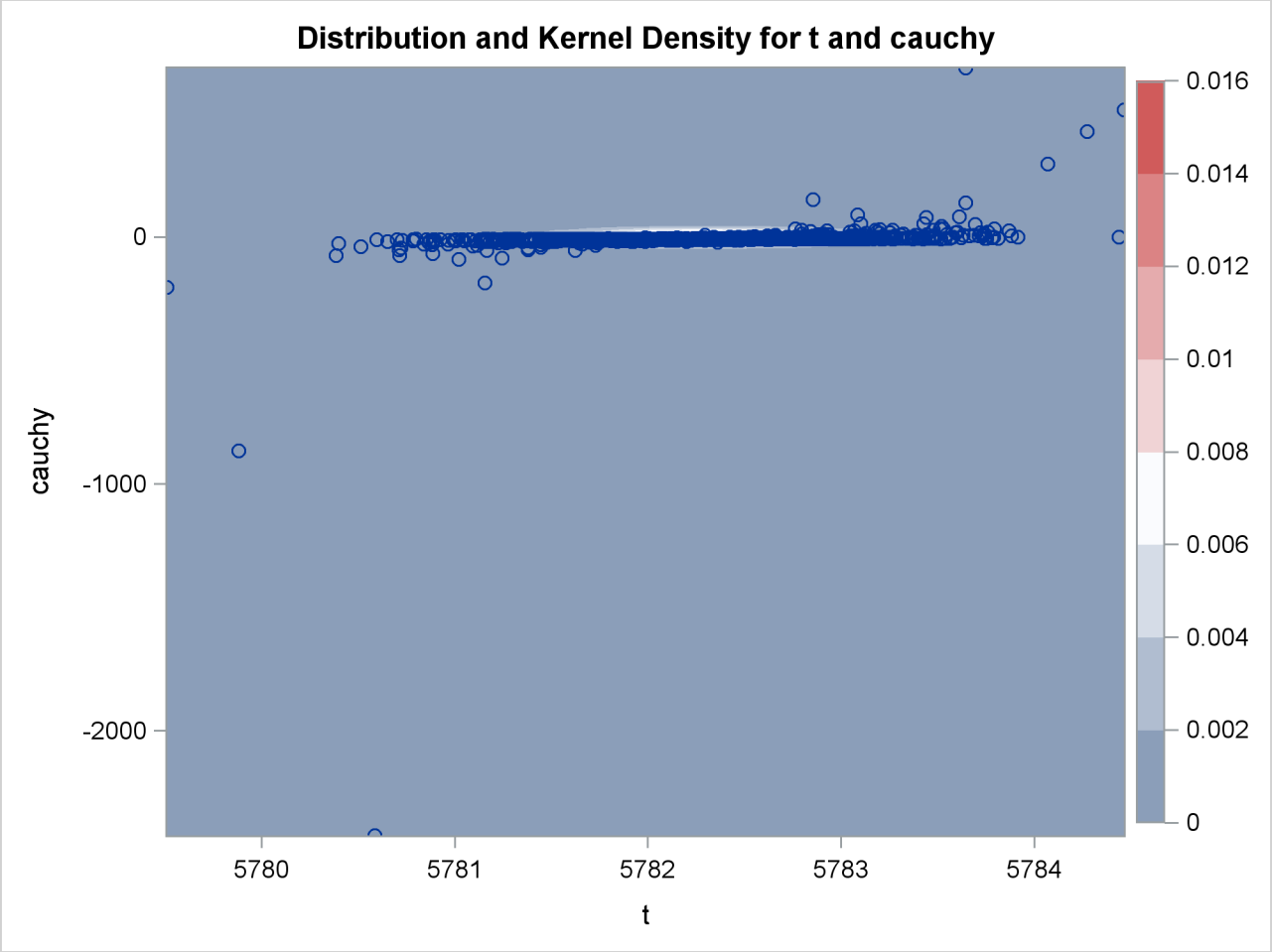
Output 26.14.1 Bivariate Density of *t* and Cauchy, Distribution of *t* by Cauchy



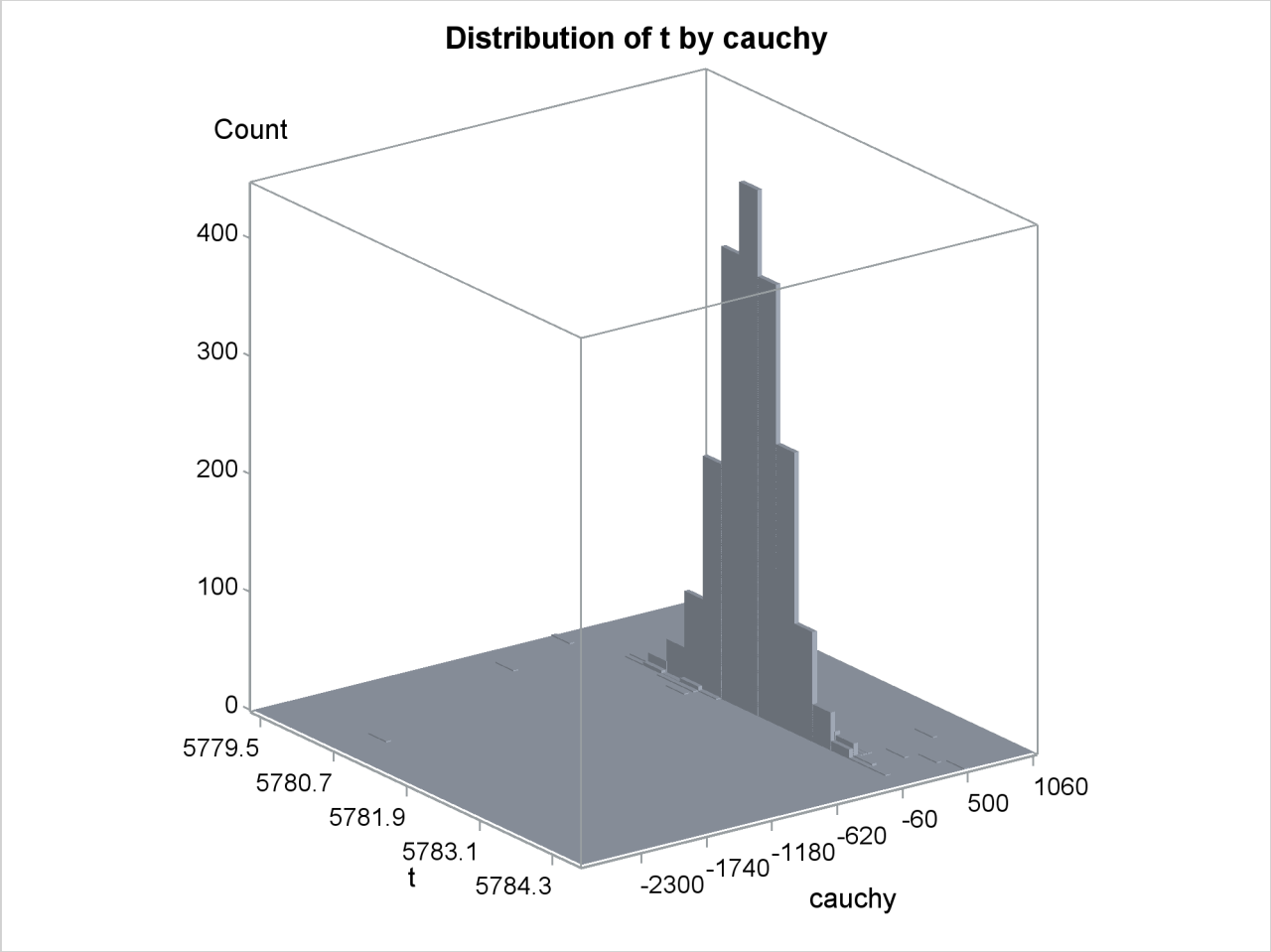
Output 26.14.2 Bivariate Density of *t* and Cauchy, Kernel Density for *t* and Cauchy



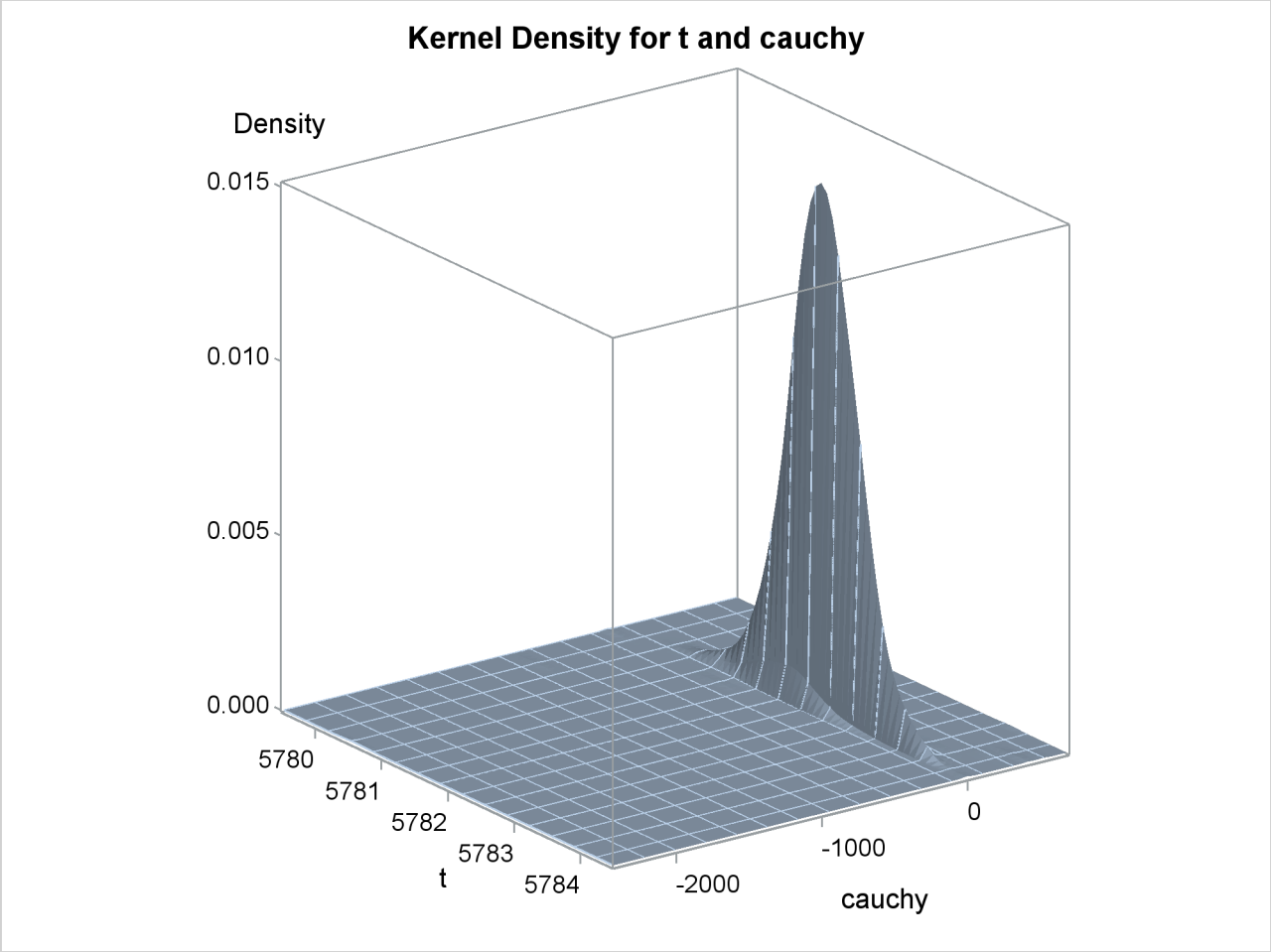
Output 26.14.3 Bivariate Density of *t* and Cauchy, Distribution and Kernel Density for *t* and Cauchy



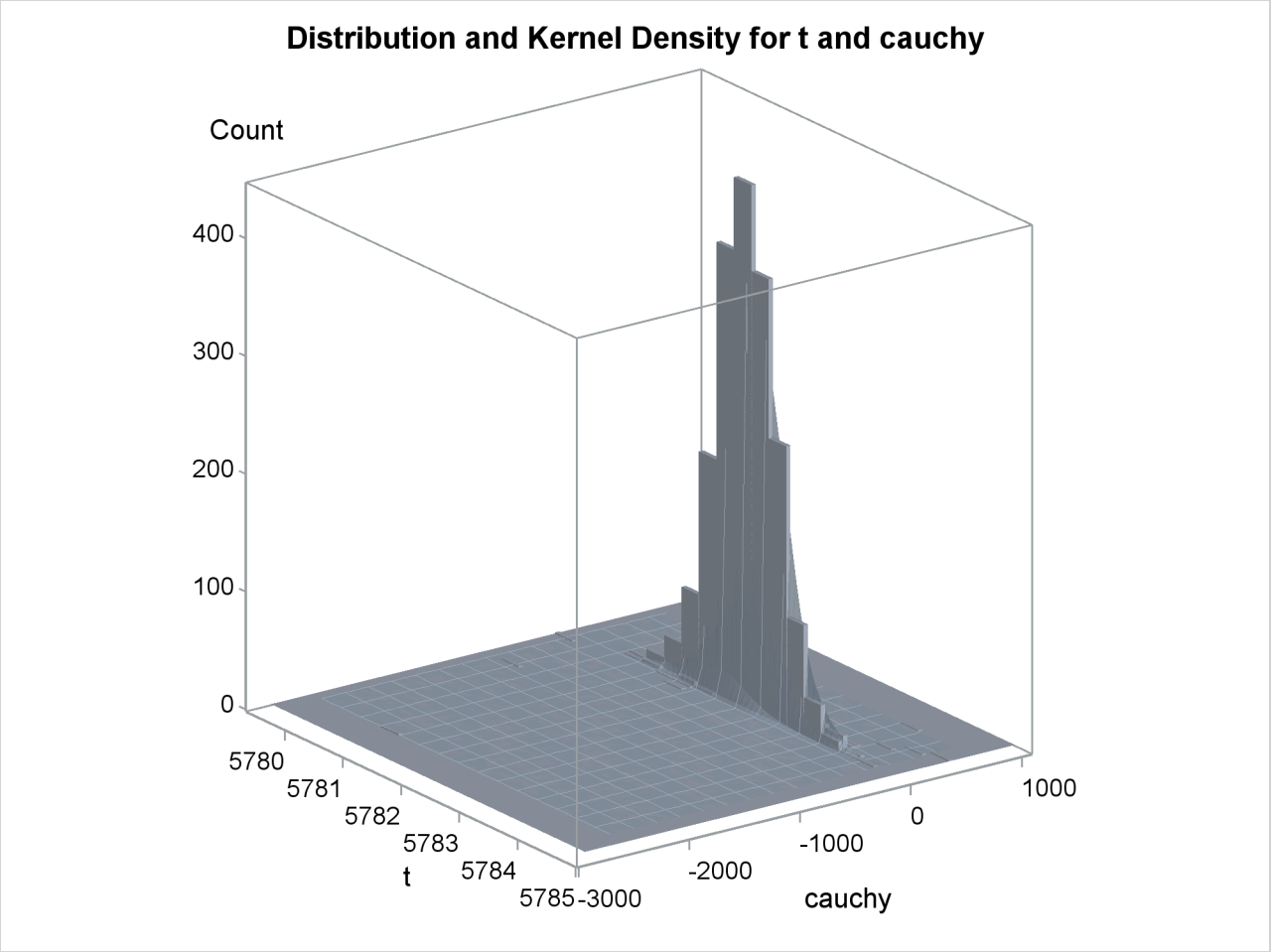
Output 26.14.4 Bivariate Density of *t* and Cauchy, Distribution of *t* by Cauchy



Output 26.14.5 Bivariate Density of *t* and Cauchy, Kernel Density for *t* and Cauchy



Output 26.14.6 Bivariate Density of *t* and Cauchy, Distribution and Kernel Density for *t* and Cauchy



Example 26.15: Simulated Method of Moments—Simple Linear Regression

This example illustrates how to use SMM to estimate a simple linear regression model for the following process:

$$y = a + bx + \epsilon,\ \epsilon \sim iid\ N(0, s^2)$$

In the following SAS statements, $ysim$ is simulated, and the first moment and the second moment of $ysim$ are compared with those of the observed endogenous variable y.

```
title "Simple regression model";

data regdata;
   do i=1 to 500;
      x = rannor( 1013 );
      Y = 2 + 1.5 * x + 1.5 * rannor( 1013 );
      output;
   end;
run;
```

```
proc model data=regdata;
   parms a b s;
   instrument x;

   ysim = (a+b*x) + s * rannor( 8003 );
   y = ysim;
   eq.ysq = y*y - ysim*ysim;

   fit y ysq / gmm ndraw;
   bound s > 0;
run;
```

Alternatively, the MOMENT statement can be used to specify the moments using the following syntax:

```
proc model data=regdata;
   parms a b s;
   instrument x;

   ysim = (a+b*x) + s * rannor( 8003 );
   y = ysim;
   moment y = (2);

   fit y / gmm ndraw;
   bound s > 0;
run;
```

The output of the MODEL procedure is shown in Output 26.15.1:

Output 26.15.1 PROC MODEL Output

Simple regression model

The MODEL Procedure

Model Summary	
Model Variables	1
Parameters	3
Equations	2
Number of Statements	4

Model Variables	Y
Parameters	a b s
Equations	ysq Y

The 2 Equations to Estimate	
Y =	F(a(1), b(x), s)
ysq =	F(a, b, s)
Instruments	1 x

Output 26.15.1 *continued*

Nonlinear GMM Parameter Estimates

Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|
a	2.065983	0.0657	31.45	<.0001
b	1.511075	0.0565	26.73	<.0001
s	1.483358	0.0498	29.78	<.0001

Example 26.16: Simulated Method of Moments—AR(1) Process

This example illustrates how to use SMM to estimate an AR(1) regression model for the following process:

$$\begin{aligned} y_t &= a + bx_t + u_t \\ u_t &= \alpha u_{t-1} + \epsilon_t \\ \epsilon_t &\sim iid\ N(0, s^2) \end{aligned}$$

In the following SAS statements, $ysim$ is simulated by using this model, and the endogenous variable y is set to be equal to $ysim$. The MOMENT statement creates two more moments for the estimation. One is the second moment, and the other is the first-order autocovariance. The NPREOBS=10 option instructs PROC MODEL to run the simulation 10 times before $ysim$ is compared to the first observation of y. Because the initial zlag(u) is zero, the first $ysim$ is $a + b * x + s *$ rannor(8003). Without the NPREOBS option, this $ysim$ is matched with the first observation of y. With NPREOBS, this $ysim$ and the next nine $ysim$ are thrown away, and the moment match starts with the eleventh $ysim$ with the first observation of y. This way, the initial values do not exert a large influence on the simulated endogenous variables.

```
%let nobs=500;
data ardata;
   lu =0;
   do i=-10 to &nobs;
      x = rannor( 1011 );
      e = rannor( 1011 );
      u = .6 * lu + 1.5 * e;
      Y = 2 + 1.5 * x + u;
      lu = u;
      if i > 0 then output;
   end;
run;

title1 'Simulated Method of Moments for AR(1) Process';

proc model data=ardata ;
   parms a b s 1 alpha .5;
   instrument x;

   u = alpha * zlag(u) + s * rannor( 8003 );
   ysim = a + b * x + u;
   y = ysim;
   moment y = (2) lag1(1);
```

```
   fit y / gmm npreobs=10 ndraw=10;
   bound s > 0, 1 > alpha > 0;
run;
```

The output of the MODEL procedure is shown in Output 26.16.1:

Output 26.16.1 PROC MODEL Output

Simulated Method of Moments for AR(1) Process

The MODEL Procedure

Model Summary	
Model Variables	1
Parameters	4
Equations	3
Number of Statements	8
Program Lag Length	1

Model Variables	Y
Parameters(Value)	a b s(1) alpha(0.5)
Equations	_moment_2 _moment_1 Y

The 3 Equations to Estimate	
_moment_2 =	F(a, b, s, alpha)
_moment_1 =	F(a, b, s, alpha)
Y =	F(a(1), b(x), s, alpha)
Instruments	1 x

Nonlinear GMM Parameter Estimates

Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|
a	1.632798	0.1038	15.73	<.0001
b	1.513197	0.0698	21.67	<.0001
s	1.427888	0.0984	14.52	<.0001
alpha	0.543985	0.0809	6.72	<.0001

Example 26.17: Simulated Method of Moments—Stochastic Volatility Model

This example illustrates how to use SMM to estimate a stochastic volatility model as in Andersen and Sorensen (1996):

$$\begin{aligned} y_t &= \sigma_t z_t \\ log(\sigma_t^2) &= a + b\, log(\sigma_{t-1}^2) + s u_t \\ (z_t, u_t) &\sim iid\ N(0, I_2) \end{aligned}$$

This model is widely used in modeling the return process of stock prices and foreign exchange rates. This is called the stochastic volatility model because the volatility is stochastic as the random variable u_t appears in the volatility equation. The following SAS statements use three moments: absolute value, the second-order

moment, and absolute value of the first-order autoregressive moment. Note the ADJSMMV option in the FIT statement to request the SMM covariance adjustment for the parameter estimates. Although these moments have closed form solution as shown by Andersen and Sorensen (1996), the simulation approach significantly simplifies the moment conditions.

```
%let nobs=1000;
data _tmpdata;
   a = -0.736; b=0.9; s=0.363;
   ll=sqrt( exp(a/(1-b)));;
   do i=-10 to &nobs;
      u = rannor( 101 );
      z = rannor( 101 );
      lnssq = a+b*log(ll**2) +s*u;
      st = sqrt(exp(lnssq));
      ll = st;
      y = st * z;
      if i > 0 then output;
   end;
run;

title1 'Simulated Method of Moments for Stochastic Volatility Model';

proc model data=_tmpdata ;
   parms a b .5 s 1;
   instrument / intonly;

   u = rannor( 8801 );
   z = rannor( 9701 );
   lsigmasq = xlag(sigmasq,exp(a));
   lnsigmasq = a + b * log(lsigmasq) + s * u;
   sigmasq = exp( lnsigmasq );

   ysim = sqrt(sigmasq) * z;
   eq.m1 = abs(y) - abs(ysim);
   eq.m2 = y**2 - ysim**2;
   eq.m5 = abs(y*lag(y))-abs(ysim*lag(ysim));

   fit m1 m2 m5 / gmm npreobs=10 ndraw=10 adjsmmv;
   bound s > 0, 1 > b > 0;
run;
```

The output of the MODEL procedure is shown in Output 26.17.1.

Output 26.17.1 PROC MODEL Output

Simulated Method of Moments for Stochastic Volatility Model

The MODEL Procedure

Model Summary	
Parameters	3
Equations	3
Number of Statements	10
Program Lag Length	1

Output 26.17.1 *continued*

Parameters(Value)	a b(0.5) s(1)
Equations	m1 m2 m5

The 3 Equations to Estimate	
m1 =	F(a, b, s)
m2 =	F(a, b, s)
m5 =	F(a, b, s)
Instruments	1

Nonlinear GMM Parameter Estimates

Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|
a	-2.2299	1.1357	-1.96	0.0499
b	0.695469	0.1554	4.47	<.0001
s	0.747779	0.1648	4.54	<.0001

Example 26.18: Duration Data Model with Unobserved Heterogeneity

All of the previous three models actually have closed-form moment conditions, so the simulation approach is not necessarily required for the estimation. This example illustrates how to use SMM to estimate a model for which there is no closed-form solution for the moments and thus the traditional GMM method does not apply. The model is the duration data model with unobserved heterogeneity in Gourieroux and Monfort (1993):

$$\begin{aligned} y_i &= -exp(-bx_i - \sigma u_i) log(v_i) \\ u_i &\sim N(0,1) \quad v_i \sim U_{[0,1]} \end{aligned}$$

The SAS statements are:

```
title1 'SMM for Duration Model with Unobserved Heterogeneity';

%let nobs=1000;
data durationdata;
   b=0.9; s=0.5;
   do i=1 to &nobs;
      u = rannor( 1011 );
      v = ranuni( 1011 );
      x = 2 * ranuni( 1011 );
      y = -exp(-b * x + s * u) * log(v);
      output;
   end;
run;

proc model data=durationdata;
   parms b .5 s 1;
   instrument x;
```

```
   u = rannor( 1011 );
   v = ranuni( 1011 );
   y = -exp(-b * x + s * u) * log(v);

   moment y = (2 3 4);
   fit y / gmm ndraw=10 ;* maxiter=500;
   bound s > 0, b > 0;
run;
```

The output of the MODEL procedure is shown in Output 26.18.1.

Output 26.18.1 PROC MODEL Output

SMM for Duration Model with Unobserved Heterogeneity

The MODEL Procedure

Model Summary	
Model Variables	1
Parameters	2
Equations	4
Number of Statements	9

Model Variables	y
Parameters(Value)	b(0.5) s(1)
Equations	_moment_3 _moment_2 _moment_1 y

The 4 Equations to Estimate	
_moment_3 =	F(b, s)
_moment_2 =	F(b, s)
_moment_1 =	F(b, s)
y =	F(b, s)
Instruments	1 x

Nonlinear GMM Parameter Estimates

Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|
b	0.92983	0.0331	28.08	<.0001
s	0.341825	0.0608	5.62	<.0001

Example 26.19: EMM Estimation of a Stochastic Volatility Model

The efficient method of moments (EMM) (Bansal et al. 1993, 1995; Gallant and Tauchen 2001), can be considered a variant of SMM. The idea is to match the efficiency of the maximum likelihood (ML) estimation with the flexibility of the SMM procedure. ML itself can be interpreted as a method of moments procedure, where the *score vector*, the vector of derivatives of the log-likelihood function with respect to the parameters, provides the exactly identifying moment conditions. EMM employs an auxiliary (or pseudo) model that closely matches the true model. The score vector of the auxiliary model provides the moment conditions in the SMM step.

This example uses the SMM feature of PROC MODEL to estimate the simple stochastic volatility (SV) model of Example 26.17 with the EMM method.

Suppose that your data are the time series $\{y_1, y_2, \ldots, y_n\}$, and the model that you want to estimate, or the structural model, is characterized by the vector of parameters $\boldsymbol{\theta}$. For the SV model, $\boldsymbol{\theta}$ is given by (a, b, s).

The first step of the EMM method is to fit the data with an auxiliary model (or score generator) that has transition density $f(y_t|Y_{t-1}, \boldsymbol{\eta})$, parameterized by the pseudo parameter $\boldsymbol{\eta}$, where $Y_{t-1} = \{y_{t-1}, \ldots, y_1\}$. The auxiliary model must approximate the true data-generating process as closely as possible and be such that ML estimation is feasible.

The only identification requirement is that the dimension of the pseudo parameter $\boldsymbol{\eta}$ be greater than or equal to that of the structural parameter $\boldsymbol{\theta}$.

Andersen, Chung, and Sorensen (1999) showed that the GARCH(1,1) is an appropriate auxiliary model that leads to a good performance of the EMM estimator for the SV model.

The analytical expression for the GARCH(1,1) model with mean zero is

$$\begin{aligned} y_t &= \sigma_t z_t \\ \sigma_t^2 &= \omega + \alpha y_{t-1} + \beta \sigma_{t-1}^2 \end{aligned}$$

The pseudo parameter vector $\boldsymbol{\eta}$ is given by (ω, α, β).

One advantage of such a class of models is that the conditional density of y_t is Gaussian—that is,

$$f(y_t|Y_{t-1}, \boldsymbol{\eta}) \propto \frac{1}{\sigma_t} \exp\left(-\frac{y_t^2}{2\sigma_t^2}\right)$$

Therefore the score vector can easily be computed analytically.

The AUTOREG procedure provides the ML estimates, $\hat{\boldsymbol{\eta}}_n$. The estimates are stored in the garchest data set.

```
title1 'Efficient Method of Moments for Stochastic Volatility Model';

/* estimate GARCH(1,1) model */
proc autoreg data=svdata(keep=y)
             outest=garchest
             noprint covout;
   model y =  / noint garch=(q=1,p=1,type=nonneg);
run;
```

If the pseudo model is close enough to the structural model, in a suitable sense, Gallant and Long (1997) showed that a consistent estimator of the asymptotic covariance matrix of the sample pseudo-score vector can be obtained from the formula

$$\hat{\mathbf{V}}_n = \frac{1}{n} \sum_{t=1}^{n} s_f(Y_t, \hat{\boldsymbol{\eta}}_n) s_f(Y_t, \hat{\boldsymbol{\eta}}_n)'$$

where $s_f(Y_t, \hat{\boldsymbol{\eta}}_n) = (\partial/\partial \boldsymbol{\eta}_n) \log f(y_t|Y_{t-1}, \hat{\boldsymbol{\eta}}_n)$ denotes the score function of the auxiliary model computed at the ML estimates.

The ML estimates of the GARCH(1,1) model are used in the following SAS statements to compute the variance-covariance matrix $\hat{\mathbf{V}}_n$.

```
/* compute the V matrix */
data vvalues;
   set scores;

   array score{*} dlldw dllda dlldb;
   array v_t{*} v_t_1-v_t_6;
   array v{*} v_1-v_6;

   /* compute external product of score vector */
   do i=1 to 3;
      do j=i to 3;
         v_t{j*(j-1)/2 + i} = score{i}*score{j};
      end;
   end;

   /* average them over t */
   do s=1 to 6;
      v{s}+ v_t{s}/&nobs;
   end;
run;
```

The $\hat{\mathbf{V}}$ matrix must be formatted to be used with the VDATA= option of the MODEL procedure. See the section "VDATA= Input data set" on page 1631 for more information about the VDATA= data set.

```
/* Create a VDATA dataset acceptable to PROC MODEL */

/* Transpose the last obs in the dataset */
proc transpose data=vvalues(firstobs=&nobs keep=v_1-v_6)
               out=tempv;
run;

/* Add eq and inst labels */
data vhat;
   set tempv(drop=_name_);
   value = col1;
   drop col1;
   input _type_ $ eq_row $ eq_col $ inst_row $ inst_col $; *$;
   datalines;
      gmm m1 m1 1 1  /* intcpt is the only inst we use */
      gmm m1 m2 1 1
      gmm m2 m2 1 1
      gmm m1 m3 1 1
      gmm m2 m3 1 1
      gmm m3 m3 1 1
    ;
```

The last step of the EMM procedure is to estimate $\boldsymbol{\theta}$ by using SMM, where the moment conditions are given by the scores of the auxiliary model.

Given a fixed value of the parameter vector $\boldsymbol{\theta}$ and an arbitrarily large T, one can simulate a series $\{\hat{y}_1(\boldsymbol{\theta}), \hat{y}_2(\boldsymbol{\theta}), \ldots, \hat{y}_T(\boldsymbol{\theta})\}$ from the structural model. The EMM estimator is the value $\hat{\boldsymbol{\theta}}_n$ that minimizes the quantity

$$m_T(\boldsymbol{\theta}, \hat{\boldsymbol{\eta}}_n)' \hat{\mathbf{V}}_n^{-1} m_T(\boldsymbol{\theta}, \hat{\boldsymbol{\eta}}_n)$$

where

$$m_T(\boldsymbol{\theta}, \hat{\boldsymbol{\eta}}_n) = \frac{1}{T} \sum_{k=1}^{T} s_f(\hat{Y}_k(\boldsymbol{\theta}), \hat{\boldsymbol{\eta}}_n)$$

is the sample moment condition evaluated at the fixed estimated pseudo parameter $\hat{\boldsymbol{\eta}}_n$. Note that the target function depends on the parameter $\boldsymbol{\theta}$ only through the simulated series $\hat{y}_k$.

The following statements generate a data set that contains $T = 20,000$ replicates of the estimated pseudo parameter $\hat{\boldsymbol{\eta}}_n$ and that is then input to the MODEL procedure. The EMM estimates are found by using the SMM option of the FIT statement. The $\hat{\mathbf{V}}_n$ matrix computed above serves as weighting matrix by using the VDATA= option, and the scores of the GARCH(1,1) auxiliary model evaluated at the ML estimates are the moment conditions in the GMM step.

Since the number of structural parameters to estimate (3) is equal to the number of moment equations (3) times the number of instruments (1), the model is exactly identified and the objective function has value zero at the minimum.

For simplicity, the starting values are set to the true values of the parameters.

```
/* USE SMM TO FIND EMM ESTIMATES */

/* Generate dataset of length T */
data emm;
   set garchest(where=(_type_="PARM") rename=(_ah_0=w _ah_1=a _gh_1=b _mse_=mse)
               keep=_type_ _ah_0 _ah_1 _gh_1 _mse_);
   do i=1 to 20000;
      output;
   end;
   drop i;
run;

title2 'EMM estimates';
/* Find the EMM estimates */
proc model data=emm maxiter=1000 plot=none;
   parms aa -0.736 bb 0.9 ss 0.363;
   instruments _exog_ / intonly;

   /* Describe the structural model */
   u = rannor( 8801 );
   z = rannor( 9701 );
   lsigmasq = xlag(sigmasq,exp(aa));
   lnsigmasq = aa + bb * log(lsigmasq) + ss * u;
   sigmasq = exp( lnsigmasq );
   ysim = sqrt(sigmasq) * z;

   /* Compute scores of GARCH(1,1) */
   /* derivative of loglik wrt sigma-sq */
   ysim2 = ysim*ysim;
   lagvar = w + a*xlag(ysim2,mse) + xlag(lagvar,0)*b;
   var = lagvar + mse*b**_n_;
   dlldv = (-1 + ysim2/var)/var/2;

   /* arch 0 */
```

```
    dvdw = b*xlag(dvdw,0) + 1;
    dlldw = dlldv*dvdw;

    /* arch 1 */
    dvda = b*xlag(dvda,0) + xlag(ysim2,mse);
    dllda = dlldv*dvda;

    /* garch 1 */
    currdvdb = w + a*xlag(ysim2,mse);
    dvdb = - b*b*xlag2(dvdb,0) + 2*b*xlag(dvdb,0) + xlag(currdvdb,0);
    dlldb = dlldv*(dvdb + _n_*b**(_n_-1)*mse);

    /* Use scores of the GARCH model as moment conditions */
    eq.m1 = dlldw;
    eq.m2 = dllda;
    eq.m3 = dlldb;

    /* Fit scores using SMM and estimated Vhat */
    fit m1 m2 m3 / gmm npreobs=10 ndraw=1 /* smm options */
                   vdata=vhat /* use estimated Vhat */
                   kernel=(bart,0,) /* turn smoothing off */;
    bounds ss > 0, 0 < bb < 1;
quit;
```

The output of the MODEL procedure is shown in Output 26.19.1.

Output 26.19.1 PROC MODEL Output

Efficient Method of Moments for Stochastic Volatility Model
EMM estimates

The MODEL Procedure

Model Summary	
Parameters	3
Equations	3
Number of Statements	21

Parameters(Value)	aa(-0.736) bb(0.9) ss(0.363)
Equations	m1 m2 m3

The 3 Equations to Estimate	
m1 =	F(aa, bb, ss)
m2 =	F(aa, bb, ss)
m3 =	F(aa, bb, ss)
Instruments	1

Nonlinear GMM Parameter Estimates

Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|
aa	-0.49702	0.0101	-49.40	<.0001
bb	0.930294	0.00137	677.59	<.0001
ss	0.316689	0.00395	80.14	<.0001

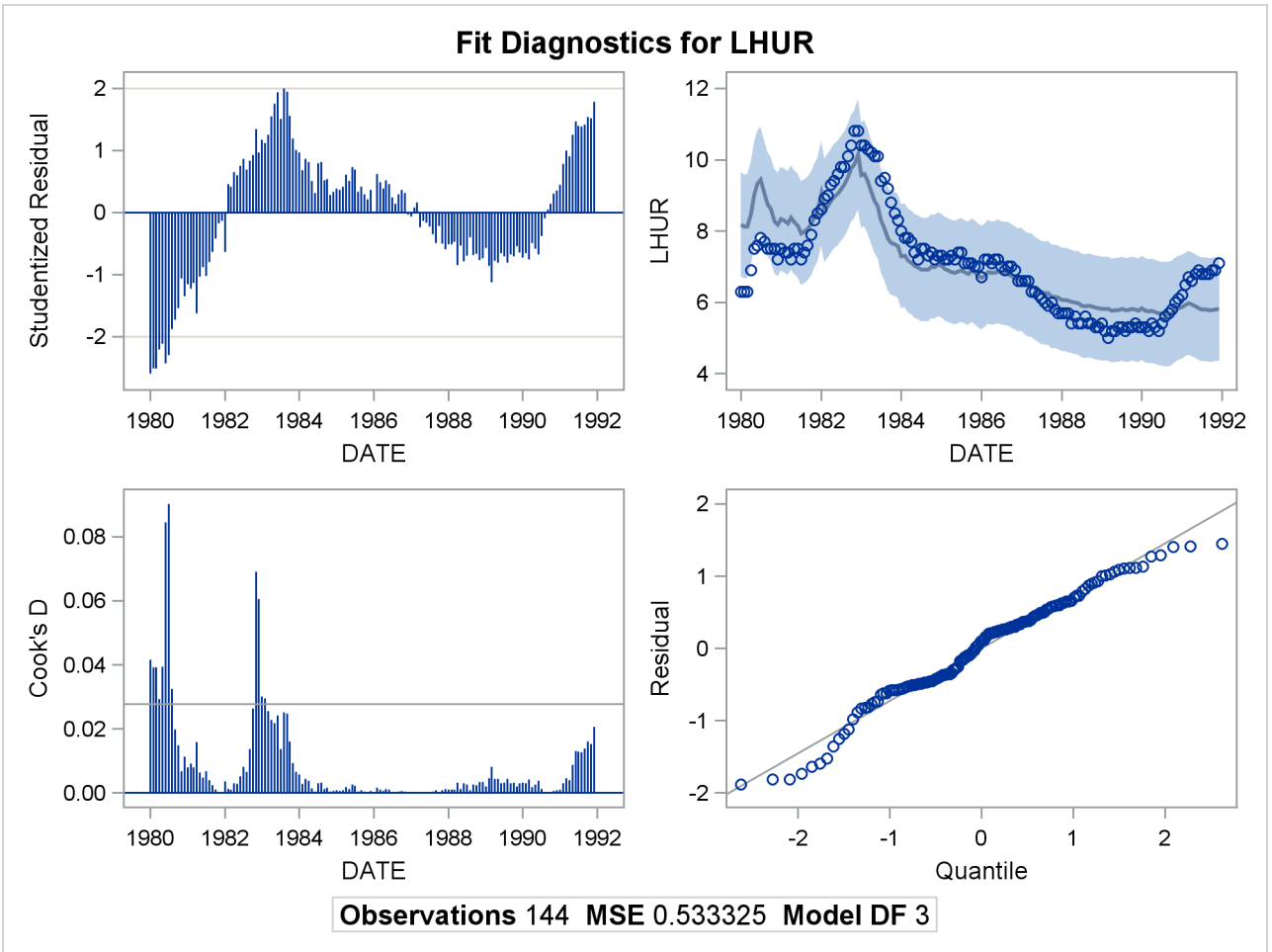
Example 26.20: Illustration of ODS Graphics

This example illustrates graphical output from PROC MODEL. This is a continuation of the section "Nonlinear Regression Analysis" on page 1475. For information about the graphics available in the MODEL procedure, see the section "ODS Graphics" on page 1637.

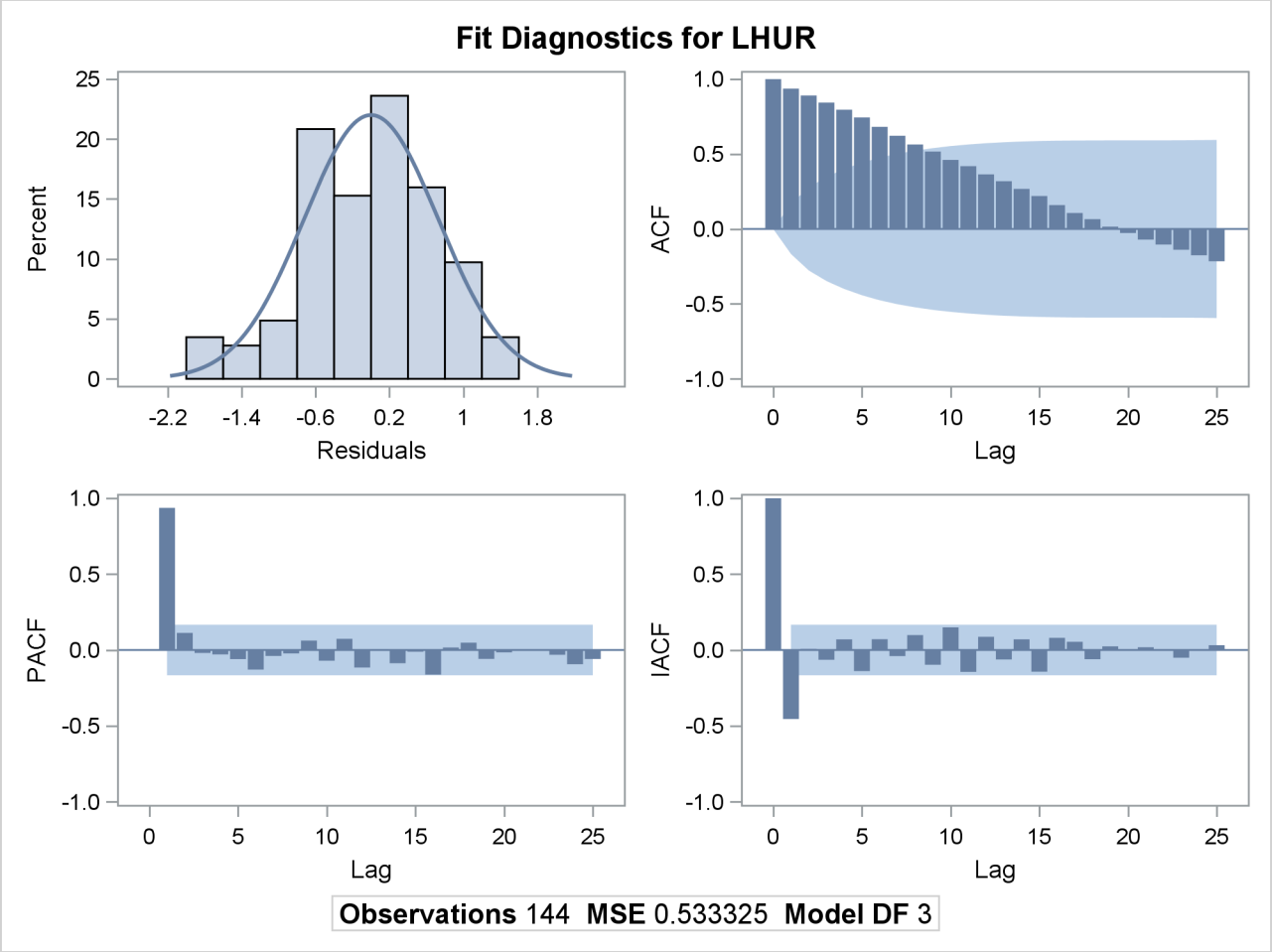
The following statements show how to generate ODS Graphics plots with the MODEL procedure. The plots are displayed in Output 26.20.1 and Output 26.20.2. Note that the variable date in the ID statement is used to define the horizontal tick mark values when appropriate.

```
title1 'Example of Graphical Output from PROC MODEL';

proc model data=sashelp.citimon;
   lhur = 1/(a * ip + b) + c;
   fit lhur;
   id date;
run;
```

Output 26.20.1 Diagnostics Plots



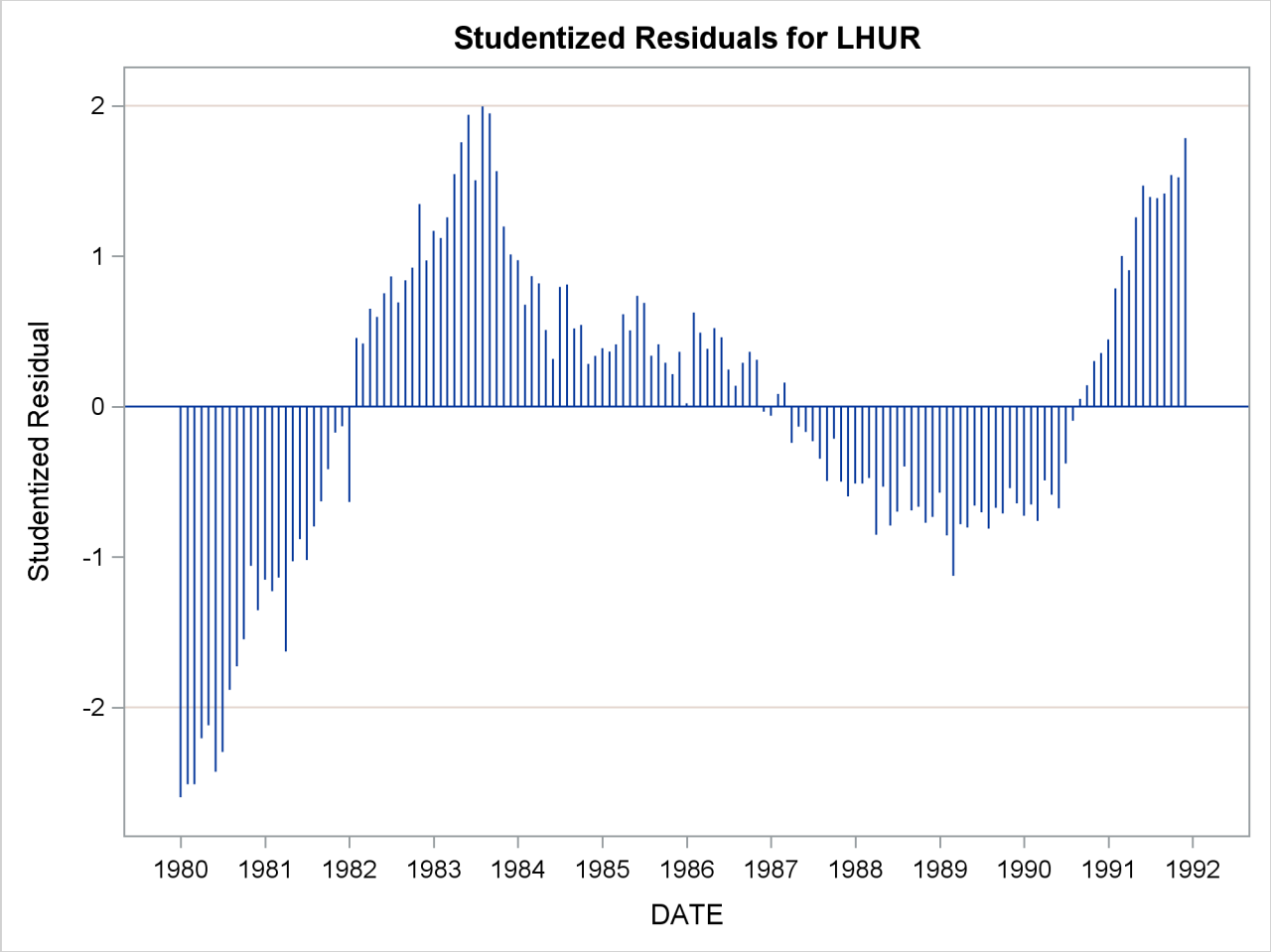
Output 26.20.2 Diagnostics Plots



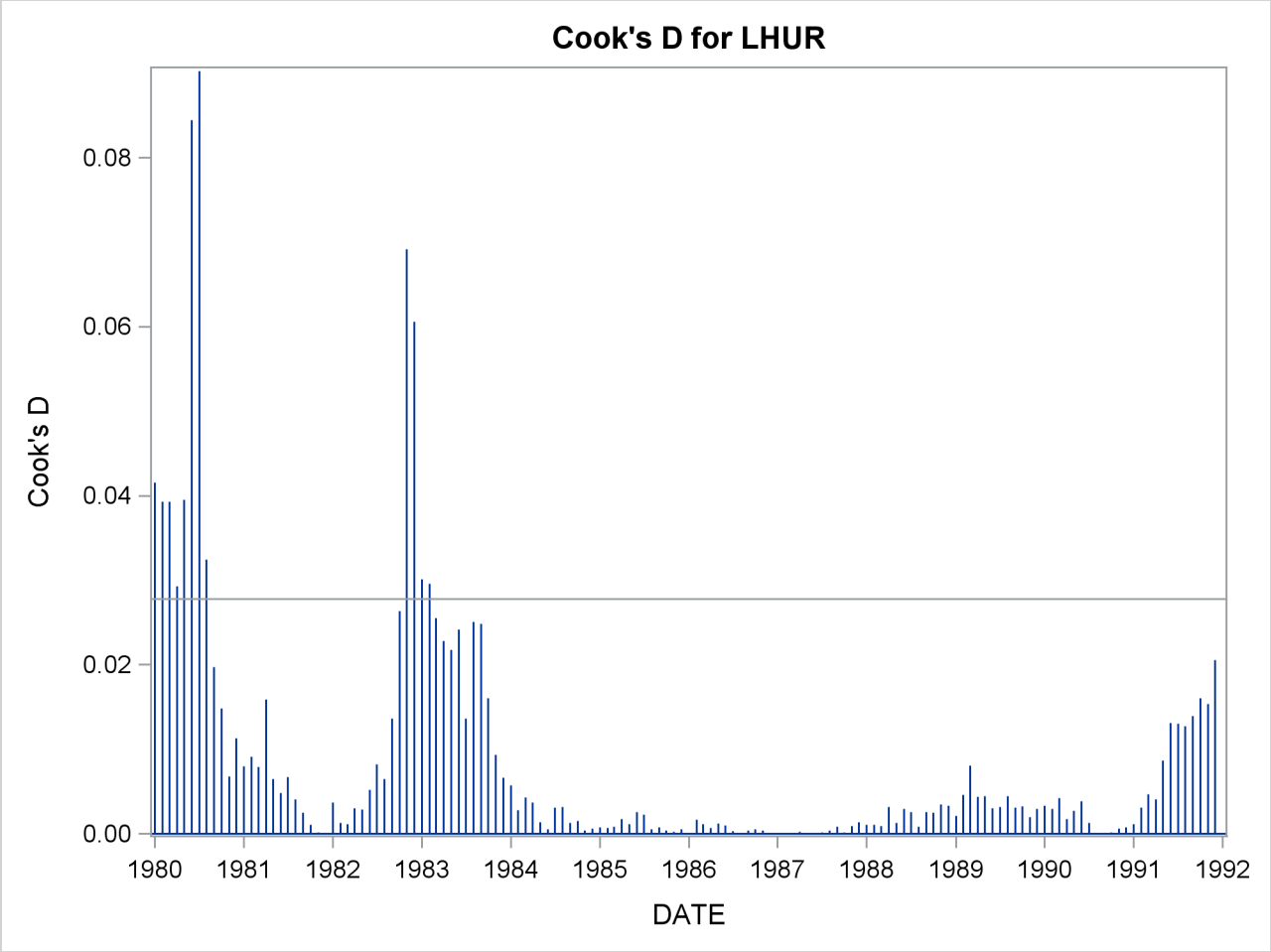
You can also obtain the plots in the diagnostics panel as separate graphs by specifying the PLOTS(UNPACK) option. These plots are displayed in Output 26.20.3 through Output 26.20.10.

```
title1 'Unpacked Graphical Output from PROC MODEL';

proc model data=sashelp.citimon plots(unpack);
   lhur = 1/(a * ip + b) + c;
   fit lhur;
   id date;
run;
```

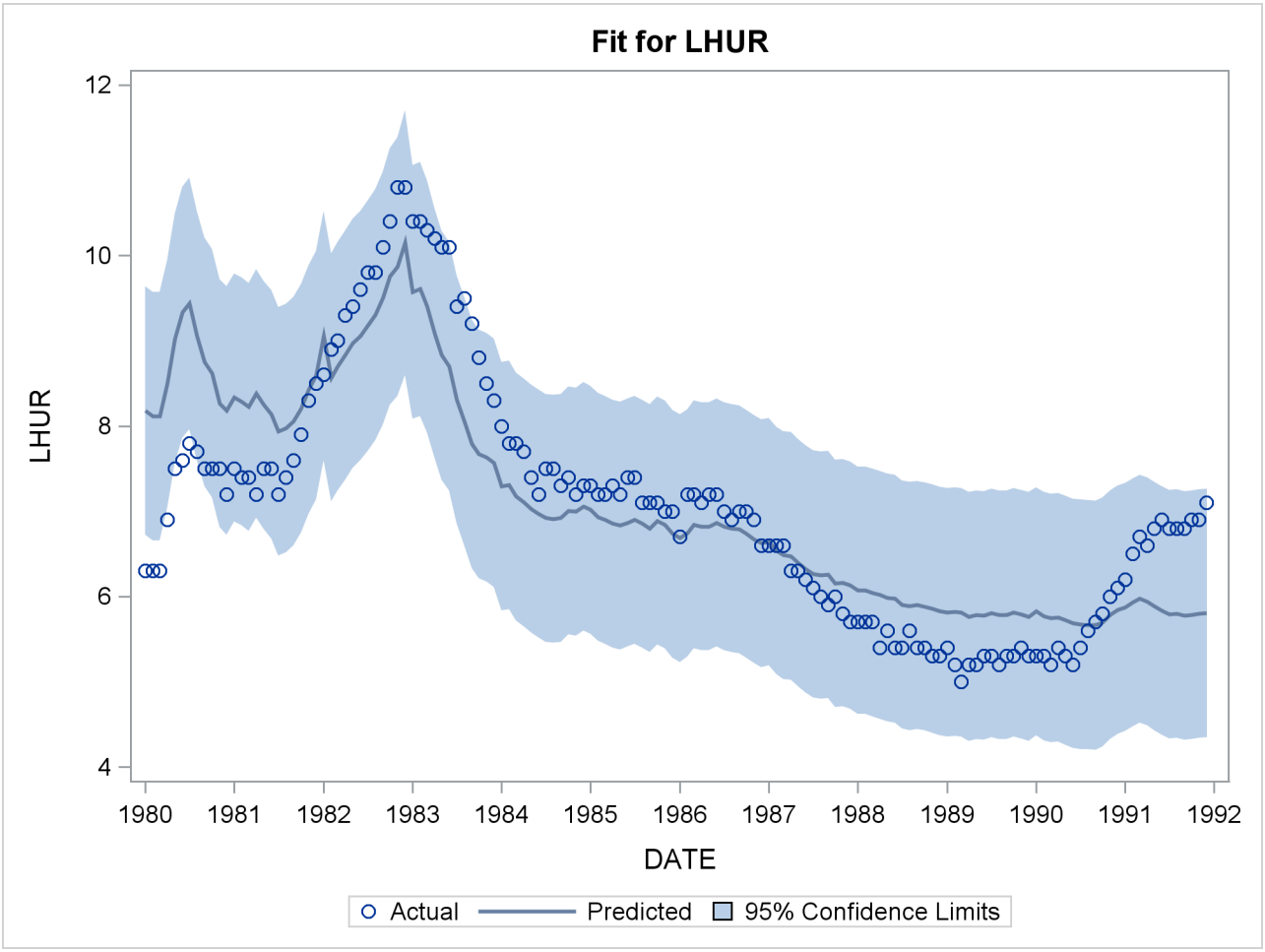
Output 26.20.3 Studentized Residuals Plot



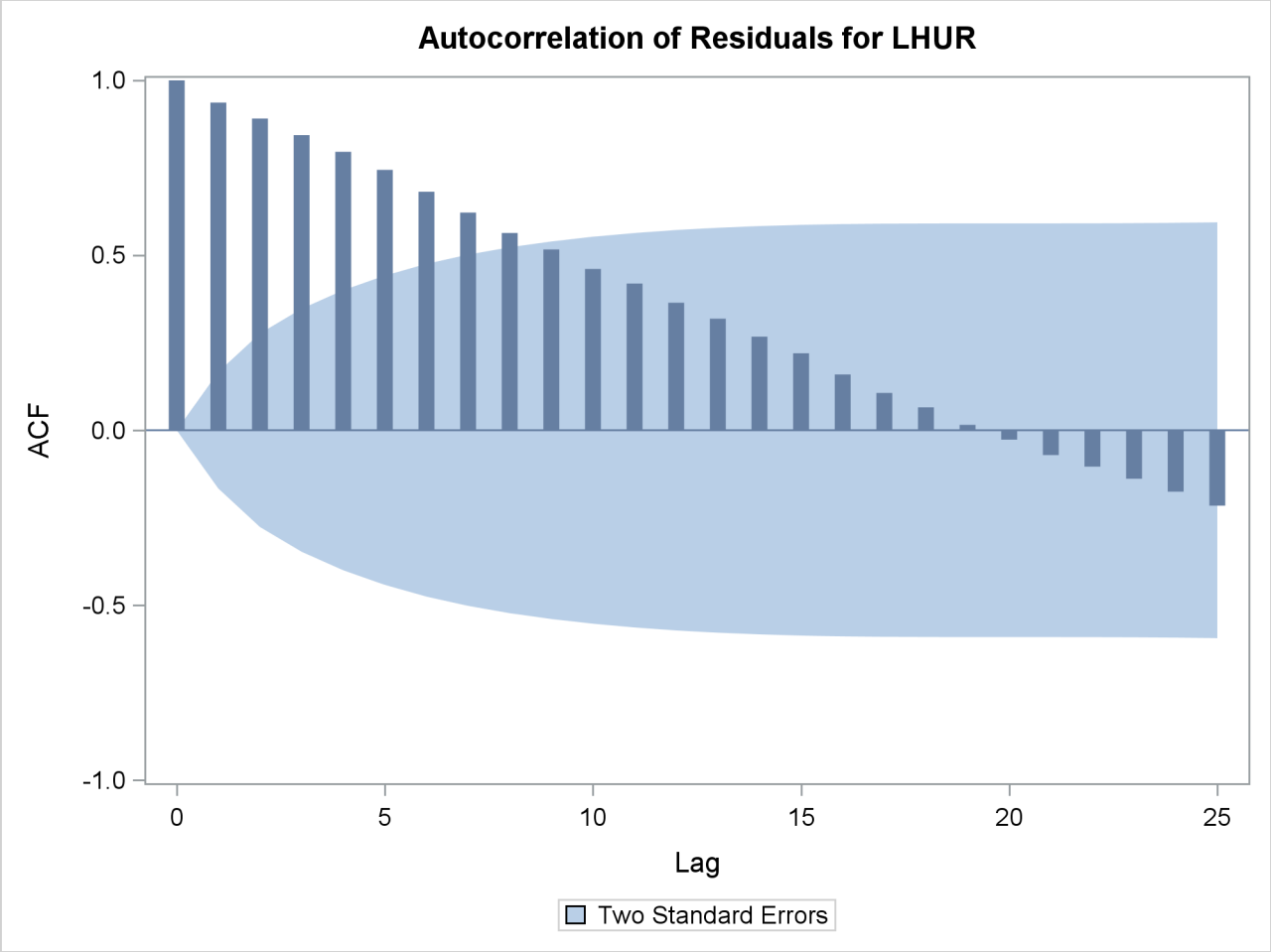
Output 26.20.4 Cook's D Plot



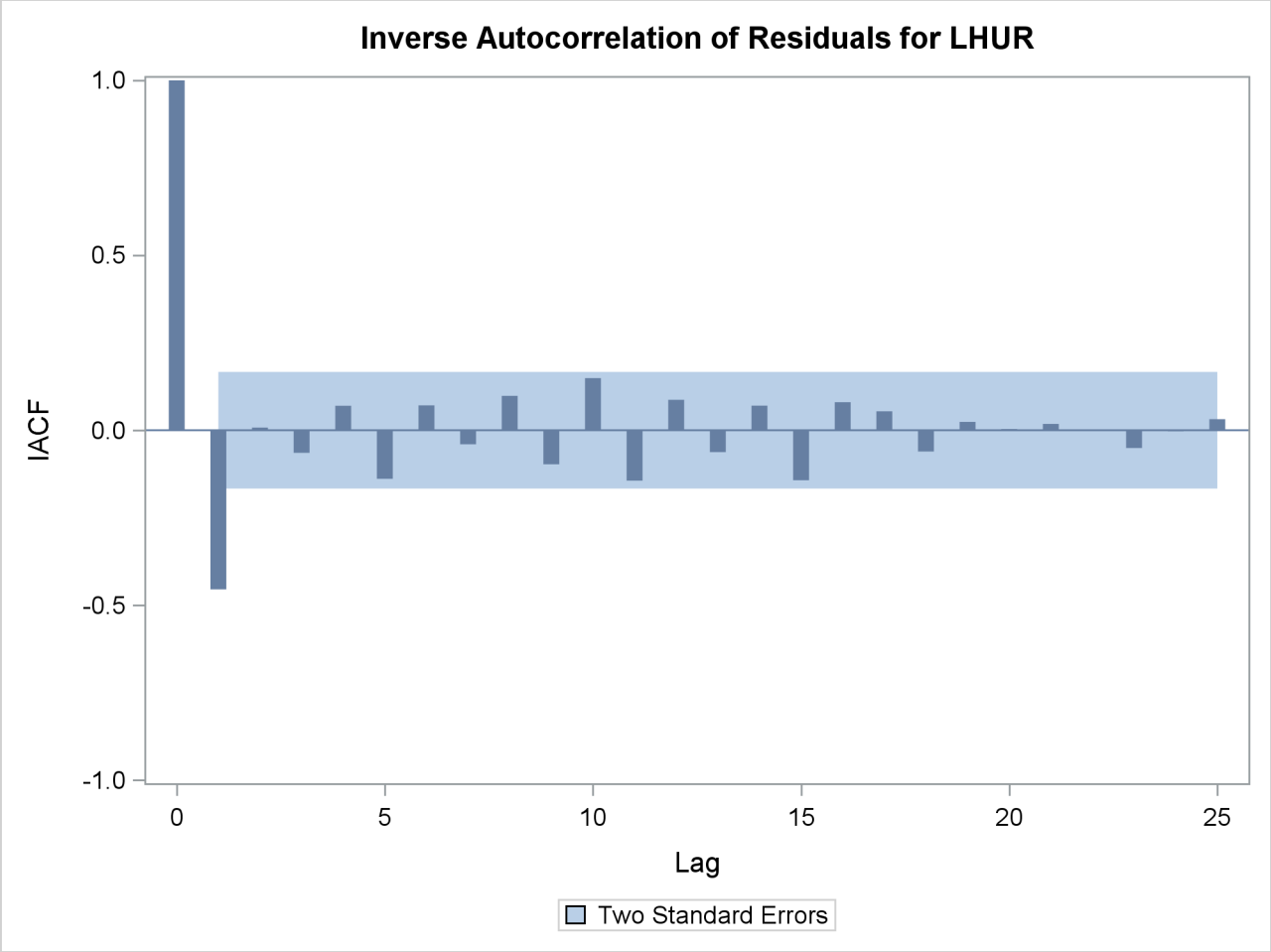
Output 26.20.5 Predicted versus Actual Plot



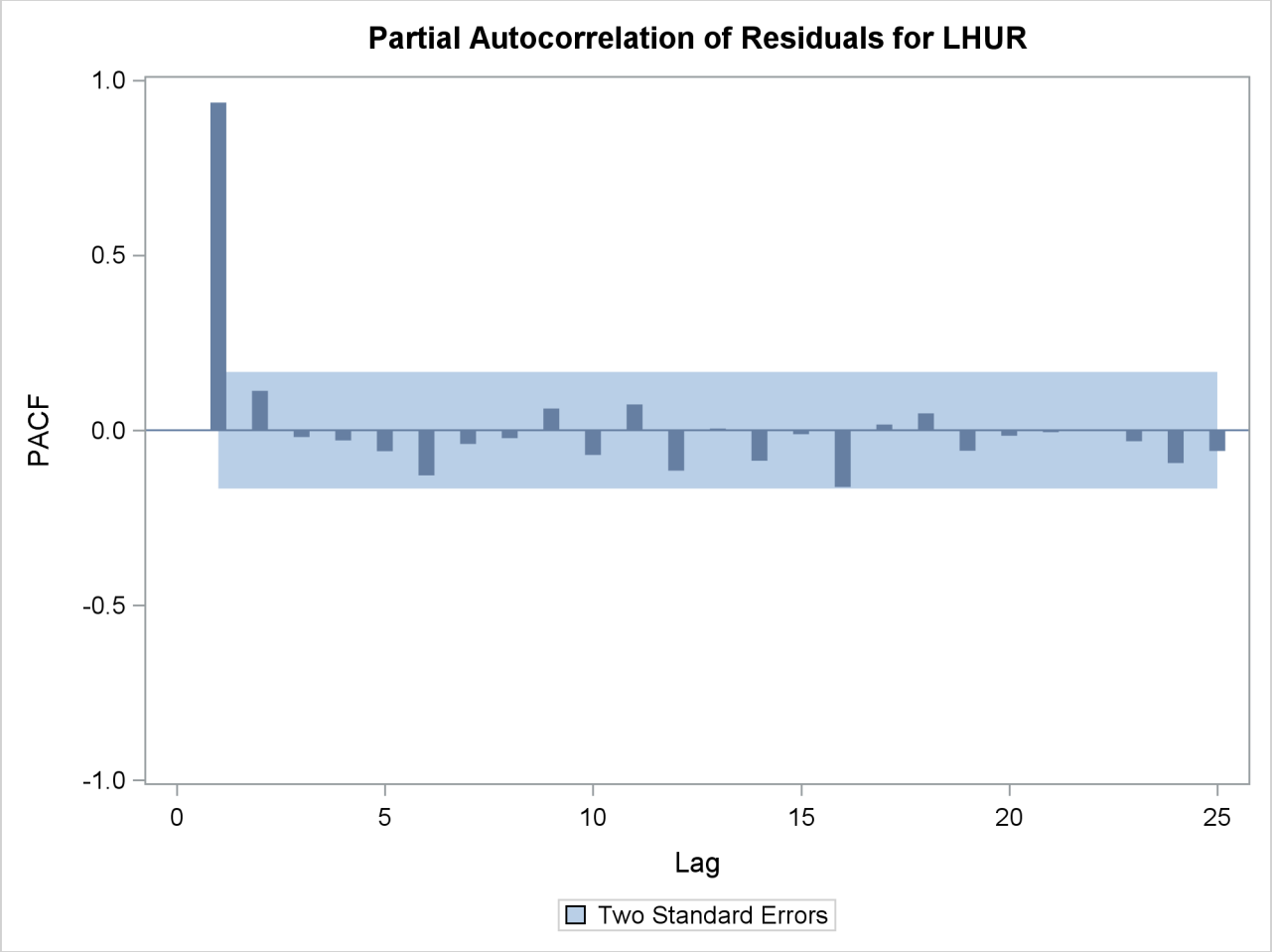
Output 26.20.6 Autocorrelation of Residuals Plot



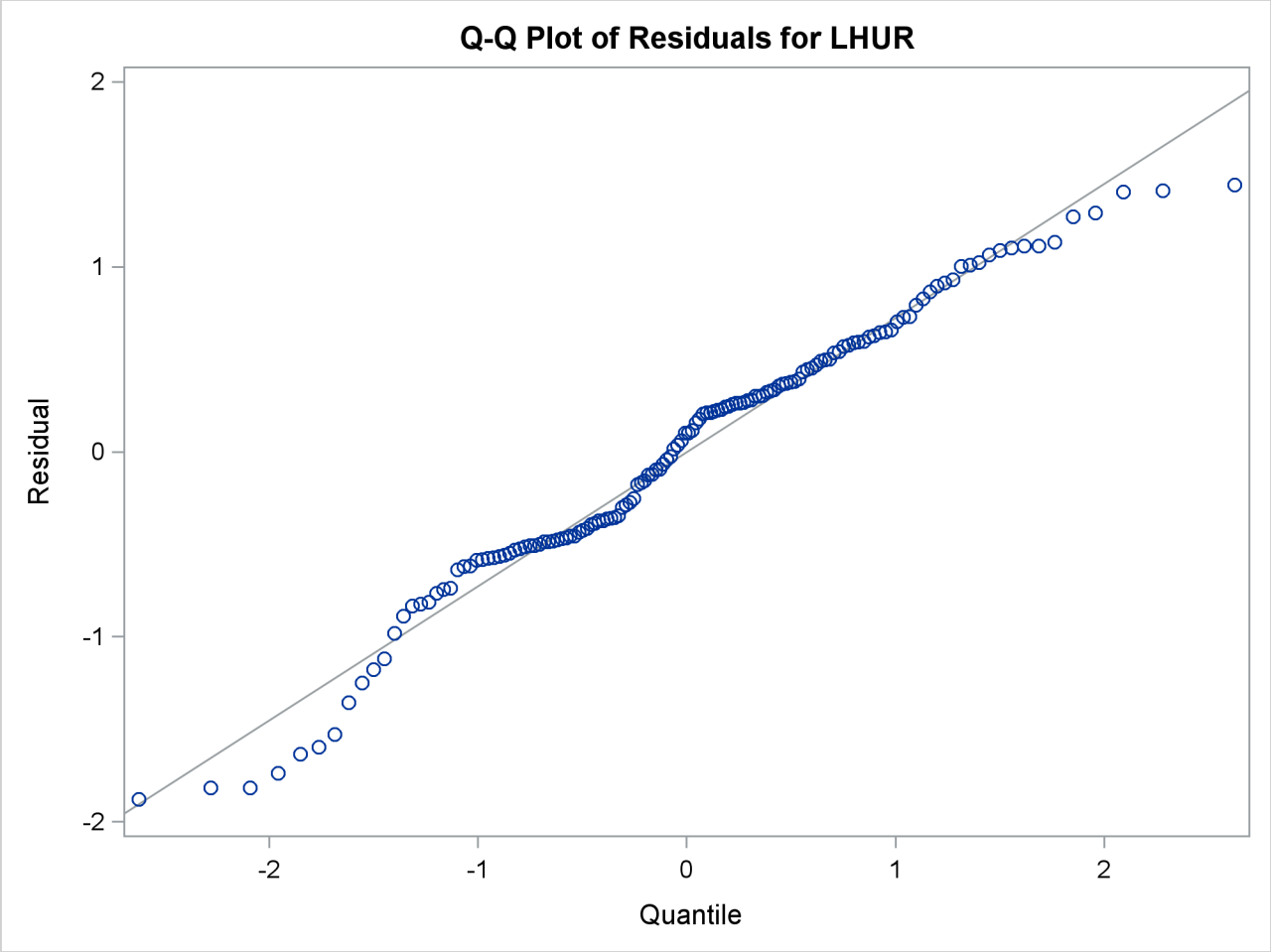
Output 26.20.7 Partial Autocorrelation of Residuals Plot



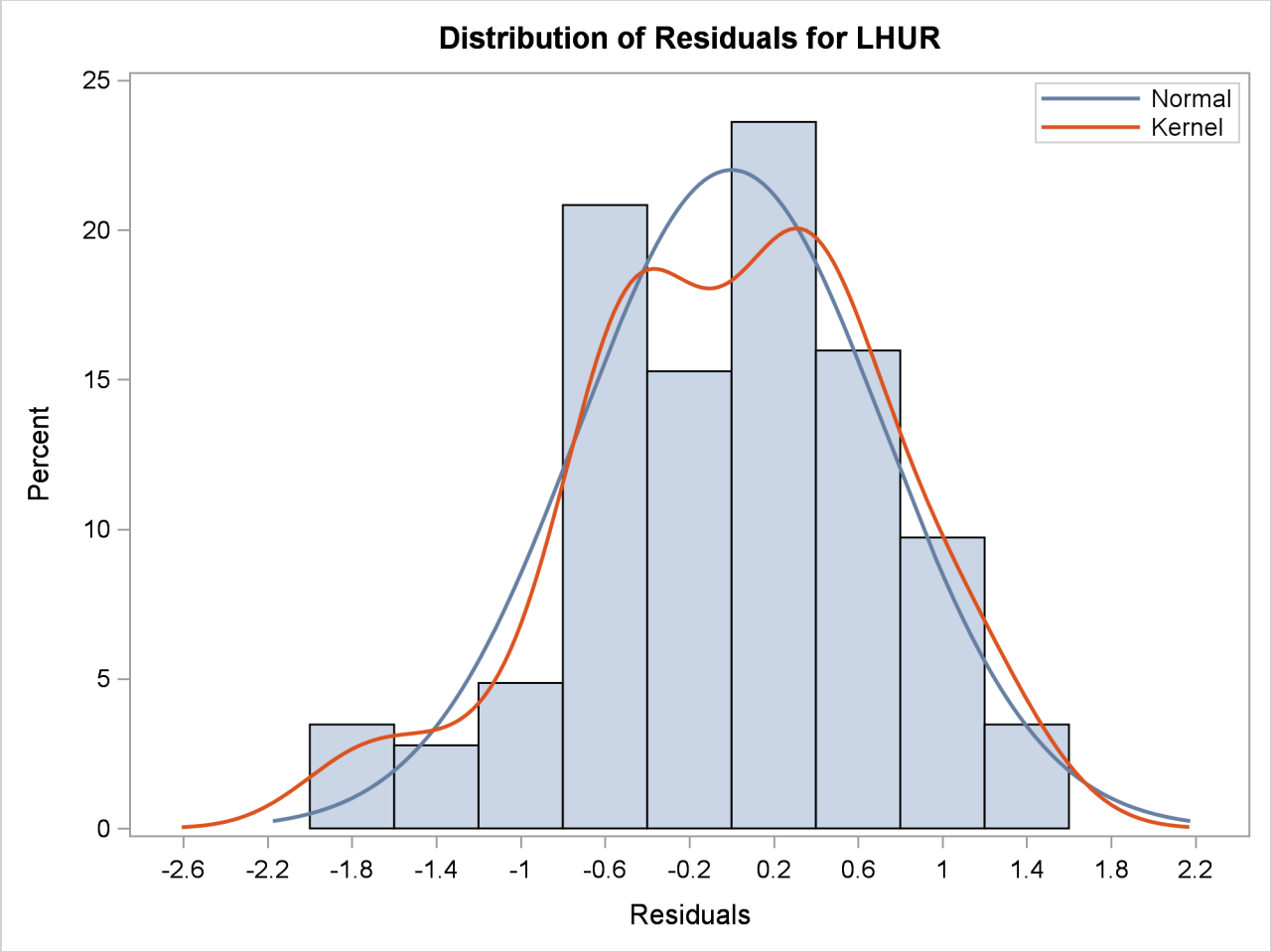
Output 26.20.8 Inverse Autocorrelation of Residuals Plot



Output 26.20.9 Q-Q Plot of Residuals



Output 26.20.10 Histogram of Residuals



Example 26.21: A Translog Cost Function and Derived Demands

This example shows the use of iterated seemingly unrelated regression (ITSUR) to estimate a system of nonlinear derived demand equations that are based on a translog cost function. Data pertain to the United States textile manufacturing sector (standard industrial classification code 22). The series runs from 1949 to 2001 and contains real quantity indices, price indices, and cost measures for a single aggregate industry output and five aggregate inputs: capital (K), labor (L), energy (E), materials (M), and services (S). The original data and information about other industrial sectors can be obtained from the Multifactor Productivity home page of the Bureau of Labor Statistics at http://www.bls.gov/mfp/.

The demand equations that are derived from the translog cost function are expressed by using a cost share as the endogenous variable. Because these data do not contain explicit information about cost shares, the shares must be formed by taking the ratio of the value of each input and the cost measure. The following statements compute the cost share for each input:

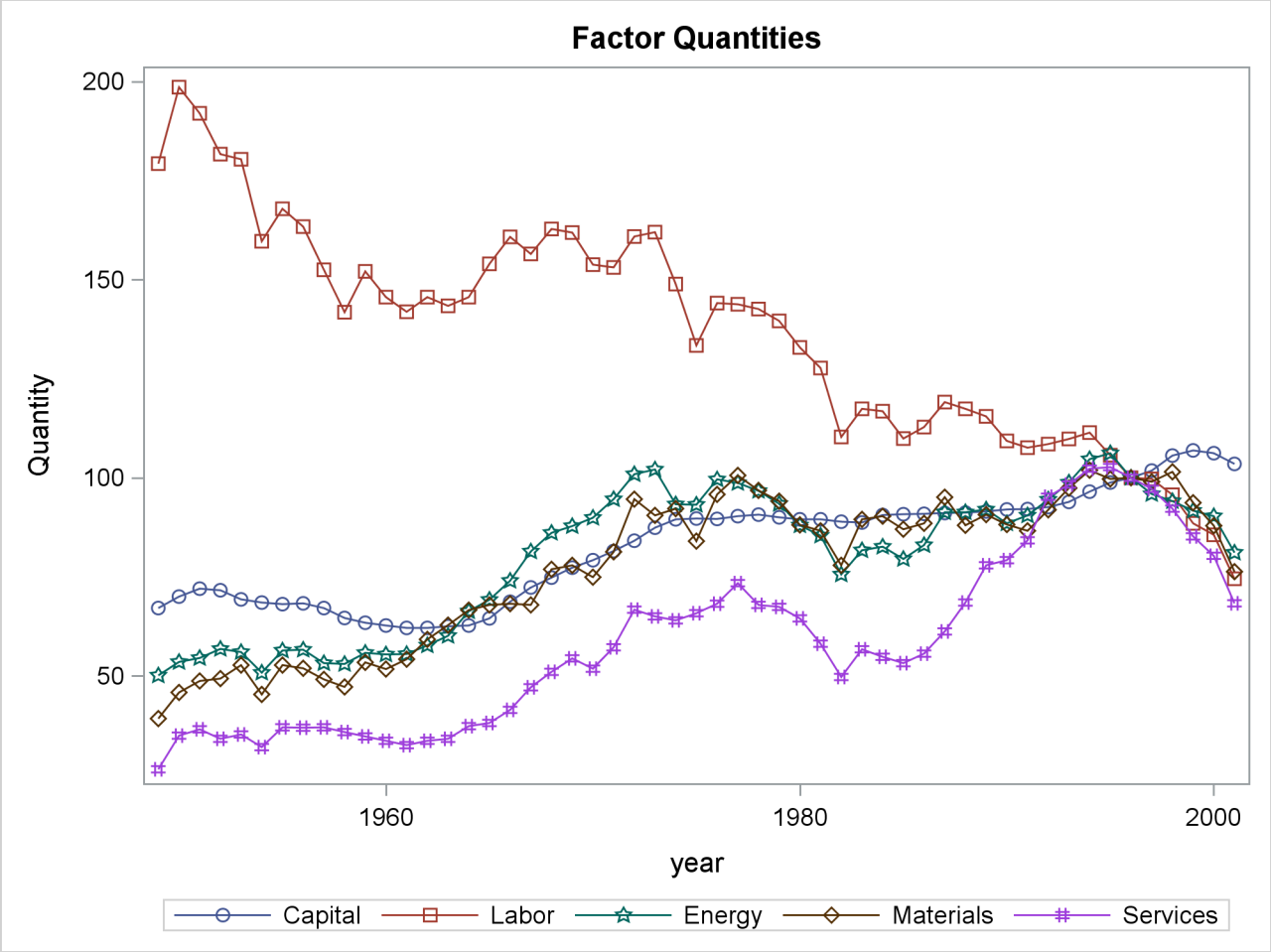
```
data klems;
set klems;
   array values {5} vk vl ve vm vs;
   array costshares {5} sk sl se sm ss;
   cost = sum(vk,vl,ve,vm,vs);
   do i = 1 to 5;
      costshares{i} = values{i}/cost;
   end;
run;
```

The following statements produce a time series plot of quantity indices and generate further plots of price indices and cost shares:

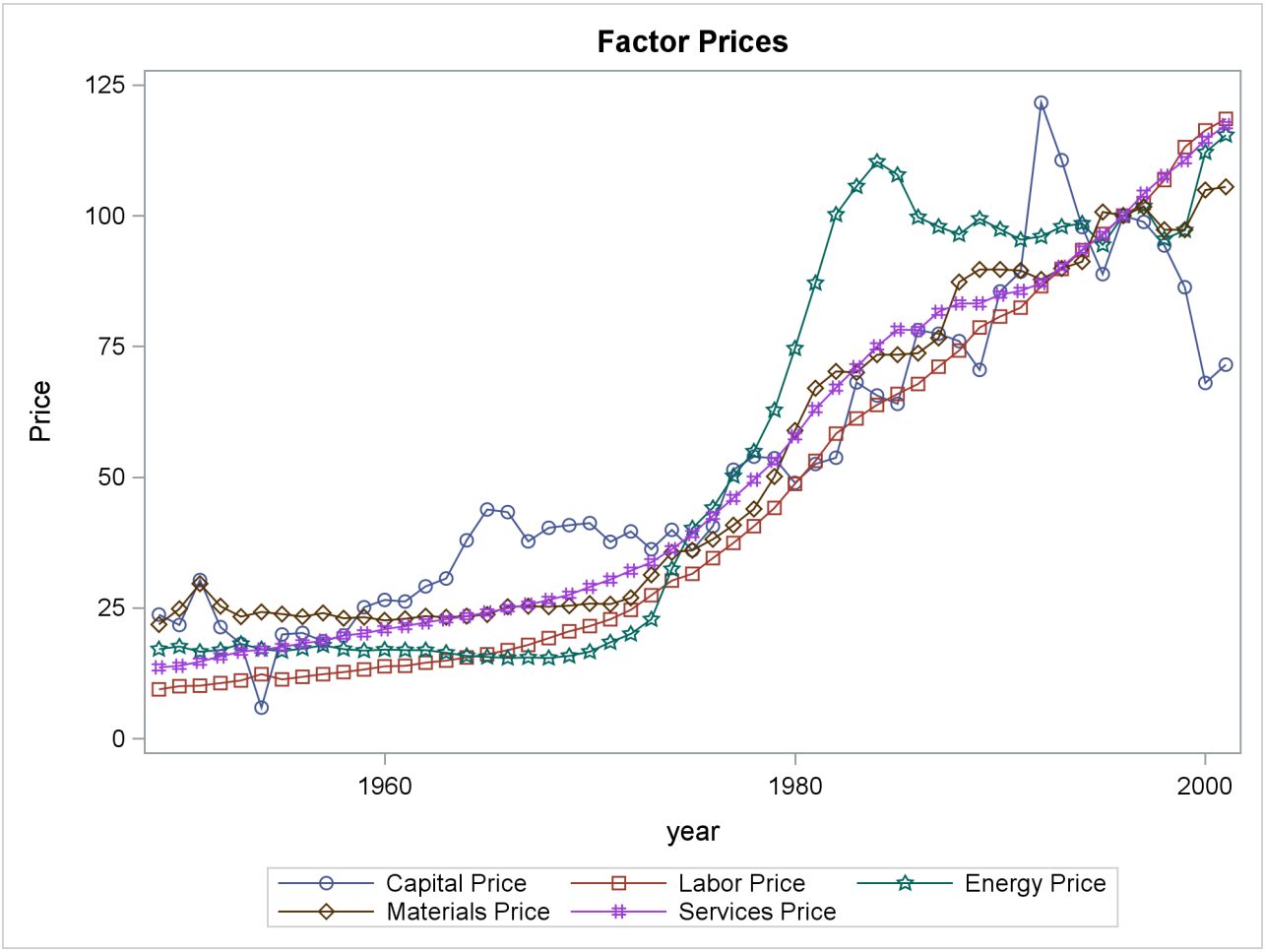
```
proc sgplot data = klems;
   series x = year y = k / markers markerattrs =(symbol=circle);
   series x = year y = l / markers markerattrs =(symbol=square);
   series x = year y = e / markers markerattrs =(symbol=star);
   series x = year y = m / markers markerattrs =(symbol=diamond);
   series x = year y = s / markers markerattrs =(symbol=hash);
   title 'Factor Quantities';
   yaxis label = 'Quantity';
run;
```

Output 26.21.1 shows time series plots of quantity indices, price indices, and cost shares over time, indicating the dynamics of the US textile sector.

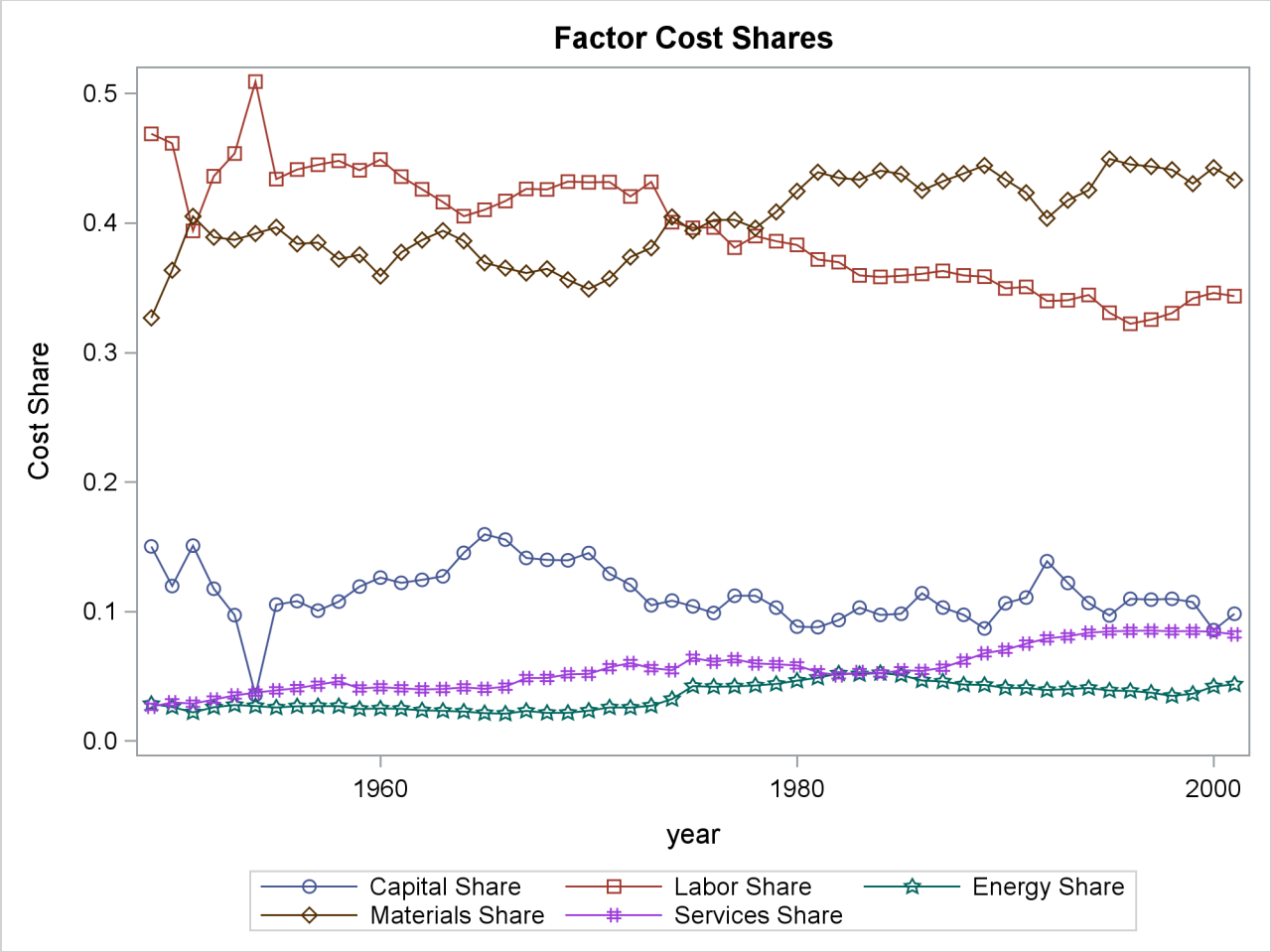
Output 26.21.1 Changes in Variables over Time



Output 26.21.1 *continued*



Output 26.21.1 *continued*



Textile manufacturing was once a significant part of total manufacturing output in the United States. As international trade increased, many textile mills moved overseas, where labor costs are lower than they are in the United States. The first graph in Output 26.21.1 shows that labor use has steadily declined while use of other inputs has grown. Perhaps the textile industry adjusted to foreign competition by increasing use of inputs besides labor. The price of energy increased rapidly in the 1970s, reflecting what has commonly been called the "energy crisis." As energy prices increased, energy use remained flat or declined. The result of these two movements is a higher cost share for energy in general. As labor use and cost shares declined in the sector, there were nearly coincident increases in the quantity indices and cost shares of capital and purchased services. This relationship suggests that capital and services can substitute for labor in the production of textiles.

Output 26.21.1 does not provide quantifiable measures of the relationship between input use and price or of input substitution. Price elasticities and substitution elasticities must be calculated from the parameters of the cost function or from factor demands. One benefit of the translog form is that the system of factor demands produces nearly the same information as the cost function. The only parameter of the cost function that is not captured by the derived demand system is the intercept term, which is not used in calculating the desired elasticities. Often, only the system of derived demands is estimated. In this example, the MODEL procedure is used to fit four derived demand equations. One of the equations (the derived demand equation for services) has been arbitrarily dropped from estimation; only $N - 1$ of the factor demands are linearly independent because the dependent variables are cost shares, which must sum to one. The parameters of the dropped derived demand equation can be recovered after estimation through homogeneity and symmetry restrictions.

The following statements estimate the system of derived demand equations without imposing any restrictions. Likelihood ratio tests are performed to determine whether homogeneity and symmetry restrictions hold both singularly and jointly.

```
proc model data = klems;
   parameters a_k gkk gkl gke gkm gks gky
              a_l glk gll gle glm gls gly
                  a_e gek gel gee gem ges gey
                  a_m gmk gml gme gmm gms gmy;
   endogenous sk sl se sm;
   exogenous pk pl pe pm ps y;

   /*System of Derived Demand Equations*/
   sk = a_k + gkk*log(pk) + gkl*log(pl) + gke*log(pe) + gkm*log(pm) + gks*log(ps)
            + gky*log(y);
   sl = a_l + glk*log(pk) + gll*log(pl) + gle*log(pe) + glm*log(pm) + gls*log(ps)
            + gly*log(y);
   se = a_e + gek*log(pk) + gel*log(pl) + gee*log(pe) + gem*log(pm) + ges*log(ps)
            + gey*log(y);
   sm = a_m + gmk*log(pk) + gml*log(pl) + gme*log(pe) + gmm*log(pm) + gms*log(ps)
            + gmy*log(y);

   fit sk sl se sm / itsur;

   test "Homogeneity"
      gkk+gkl+gke+gkm+gks=0,
      glk+gll+gle+glm+gls=0,
      gek+gel+gee+gem+ges=0,
      gmk+gml+gme+gmm+gms=0, / lr;
```

```
   test "Symmetry"
      gkl=glk,
      gke=gek,
      gkm=gmk,
      glm=gml,
      gle=gel,
      gem=gme, / lr;

   test "Joint Homogeneity and Symmetry"
      gkk+gkl+gke+gkm+gks=0,
      glk+gll+gle+glm+gls=0,
      gek+gel+gee+gem+ges=0,
      gmk+gml+gme+gmm+gms=0,
      gkl=glk,
      gke=gek,
      gkm=gmk,
      glm=gml,
      gle=gel,
      gem=gme, / lr;
run;
```

The summary of residual errors provides fit statistics for each of the estimated demand equations and is shown in Output 26.21.2. In this case, the model fits the data well based on R-squared values.

Output 26.21.2 Residual Summary

The MODEL Procedure

Nonlinear ITSUR Summary of Residual Errors

Equation	DF Model	DF Error	SSE	MSE	Root MSE	R-Square	Adj R-Sq	Label
sk	7	46	0.00118	0.000026	0.00506	0.9519	0.9457	Capital Share
sl	7	46	0.00449	0.000098	0.00988	0.9565	0.9508	Labor Share
se	7	46	0.000079	1.724E-6	0.00131	0.9847	0.9827	Energy Share
sm	7	46	0.00436	0.000095	0.00973	0.9146	0.9035	Materials Share

Because the form of the elasticities is somewhat complicated, it can be difficult to interpret the values and signs of parameter estimates. It is far easier to compute the elasticities directly. The test results in Output 26.21.3 indicate that both symmetry and homogeneity are rejected. A common practice is to assume that such restrictions hold and to impose them in estimation.

Output 26.21.3 Tests of Symmetry and Homogeneity

Test Results

Test	Type	Statistic	Pr > ChiSq	Label
Homogeneity	L.R.	114.22	<.0001	gkk+gkl+gke+gkm+gks=0, glk+gll+gle+glm+gls=0, gek+gel+gee+gem+ges=0, gmk+gml+gme+gmm+gms=0
Symmetry	L.R.	109.72	<.0001	gkl=glk, gke=gek, gkm=gmk, glm=gml, gle=gel, gem=gme
Joint Homogeneity and Symmetry	L.R.	240.80	<.0001	gkk+gkl+gke+gkm+gks=0, glk+gll+gle+glm+gls=0, gek+gel+gee+gem+ges=0, gmk+gml+gme+gmm+gms=0, gkl=glk, gke=gek, gkm=gmk, glm=gml, gle=gel, gem=gme

The following code imposes both symmetry and homogeneity restrictions on the underlying model:

```
proc model data = klems;
    parameters a_k gkk gkl gke gkm gks gky
               a_l glk gll gle glm gls gly
               a_e gek gel gee gem ges gey
               a_m gmk gml gme gmm gms gmy;
    endogenous sk sl se sm;
    exogenous pk pl pe pm ps y;
    restrict /*Homogeneity Restrictions*/
             gks+gkk+gkl+gke+gkm=0,
             gls+gkl+gll+gle+glm=0,
             ges+gke+gle+gee+gem=0,
             gms+gkm+glm+gem+gmm=0,
             /*Symmetry Restrictions*/
             gkl=glk, gke=gek, gkm=gmk, gle=gel, glm=gml, gem=gme;

    /*System of Derived Demand Equations*/
    sk = a_k + gkk*log(pk) + gkl*log(pl) + gke*log(pe) + gkm*log(pm) + gks*log(ps)
             + gky*log(y);
    sl = a_l + glk*log(pk) + gll*log(pl) + gle*log(pe) + glm*log(pm) + gls*log(ps)
             + gly*log(y);
    se = a_e + gek*log(pk) + gel*log(pl) + gee*log(pe) + gem*log(pm) + ges*log(ps)
             + gey*log(y);
    sm = a_m + gmk*log(pk) + gml*log(pl) + gme*log(pe) + gmm*log(pm) + gms*log(ps)
             + gmy*log(y);

    fit sk sl se sm / itsur chow = (24) outest=est;

    test "Constant Returns to Scale"
       gky=0,
       gly=0,
       gey=0,
       gmy=0, / lr;
run;
```

The symmetry restriction shrinks the number of parameters of the model considerably. This shrinkage is particularly useful when the time series is not long and degrees of freedom need to be conserved. Output 26.21.4 shows the parameter estimates of the MODEL procedure.

Output 26.21.4 Restricted Model

The MODEL Procedure

Nonlinear ITSUR Parameter Estimates

Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|	Label
a_k	0.109996	0.0219	5.02	<.0001	SK Intercept
gkk	0.062014	0.00357	17.38	<.0001	SK K Price
gkl	-0.01898	0.00725	-2.62	0.0118	SK L Price
gke	-0.00179	0.000836	-2.15	0.0369	SK E Price
gkm	-0.03872	0.00610	-6.35	<.0001	SK M Price
gks	-0.00252	0.00342	-0.74	0.4648	SK S Price
gky	-0.00104	0.00496	-0.21	0.8354	SK Output
a_l	0.865473	0.0903	9.58	<.0001	SL Intercept
glk	-0.01898	0.00725	-2.62	0.0118	SL K Price
gll	-0.00999	0.0360	-0.28	0.7826	SL L Price
gle	-0.00106	0.00420	-0.25	0.8013	SL E Price
glm	-0.06766	0.0248	-2.73	0.0087	SL M Price
gls	0.097692	0.0214	4.57	<.0001	SL S Price
gly	-0.11488	0.0200	-5.74	<.0001	SL Output
a_e	0.012747	0.00984	1.30	0.2013	SE Intercept
gek	-0.00179	0.000836	-2.15	0.0370	SE K Price
gel	-0.00106	0.00420	-0.25	0.8013	SE L Price
gee	0.029876	0.00112	26.76	<.0001	SE E Price
gem	-0.01954	0.00275	-7.12	<.0001	SE M Price
ges	-0.00748	0.00325	-2.30	0.0257	SE S Price
gey	0.005808	0.00217	2.68	0.0101	SE Output
a_m	-0.08173	0.0701	-1.17	0.2492	SM Intercept
gmk	-0.03872	0.00610	-6.35	<.0001	SM K Price
gml	-0.06766	0.0248	-2.73	0.0087	SM L Price
gme	-0.01954	0.00275	-7.12	<.0001	SM E Price
gmm	0.146849	0.0222	6.62	<.0001	SM M Price
gms	-0.02092	0.0103	-2.03	0.0477	SM S Price
gmy	0.113729	0.0157	7.27	<.0001	SM Output
Restrict0	-563.453	242.8	-2.32	0.0187	gks+gkk+gkl+gke+gkm=0
Restrict1	82.23307	193.0	0.43	0.6747	gls+gkl+gll+gle+glm=0
Restrict2	321.7446	689.2	0.47	0.6455	ges+gke+gle+gee+gem=0
Restrict3	-279.71	211.0	-1.33	0.1879	gms+gkm+glm+gem+gmm=0
Restrict4	-261.228	196.3	-1.33	0.1860	gkl=glk
Restrict5	-1041.84	755.0	-1.38	0.1700	gke=gek
Restrict6	-27.855	219.3	-0.13	0.9005	gkm=gmk
Restrict7	-880.821	742.7	-1.19	0.2396	gle=gel
Restrict8	259.513	215.6	1.20	0.2326	glm=gml
Restrict9	1103.343	320.8	3.44	0.0003	gem=gme

The majority of the parameter estimates are significant, and insignificant parameters are statistically equivalent to 0. When the g_{ij} are all 0, the translog cost function reduces to the Cobb-Douglas cost function. Statistically insignificant parameter estimates imply that corresponding elasticities of substitution are equal to the Cobb-Douglas value of 1.

A TEST statement is used to determine whether this industry exhibits constant returns to scale in the range of the sample. The CHOW option in the FIT statement performs a Chow test for a structural break at the 24th year of the sample (1973). In October of that year the Organization of Petroleum Exporting Countries (OPEC) declared an oil embargo. Markets were affected by significant shocks to oil prices, and gasoline in the United States was rationed. Output 26.21.5 shows the results of the two tests.

Output 26.21.5 CRS and Chow Test Results

Test Results

Test	Type	Statistic	Pr > ChiSq	Label
Constant Returns to Scale	L.R.	74.27	<.0001	gky=0, gly=0, gey=0, gmy=0

Structural Change Test

Test	Break Point	Num DF	Den DF	F Value	Pr > F
Chow	24	23	2	0.28	0.9560

The null hypothesis of constant returns to scale is rejected. The null hypothesis of the Chow test cannot be rejected. Even with the turmoil of the oil embargo, there is no evidence of a structural break in 1973.

Derivations of the Hicks-Allen elasticity of substitution, the Morishima elasticity of substitution, and the price elasticity of demand for the translog cost function can be found in Chambers (1988). The elasticities are evaluated at the sample mean, so the MEANS procedure is used in the following statements to produce data that contain the sample means of the cost shares:

```
proc means data = klems noprint mean;
   variables sk sl se sm ss;
   output out = meanshares mean = sk sl se sm ss;
run;
```

Because some of the parameters are not estimated, their values must be backed out through application of homogeneity and symmetry restrictions. The IML procedure is used in the following statements to read in parameter estimates and then calculate elasticities:

```
proc iml;
   /*Read in parameter estimates*/
   use est;
   read all var {gkk gkl gke gkm gks};
   read all var {gll gle glm gls};
   read all var {gee gem ges};
   read all var {gmm gms};
   close est;

   /*Calculate S parameter based on homogeneity constraint*/
   gss=0-gks-gls-ges-gms;

   /*Read in mean cost shares and construct vector*/
   use meanshares;
   read all var {sk sl se sm ss};
   close meanshares;

   w = sk//sl//se//sm//ss;
```

```
  print w;

  /*Construct matrix of parameter estimates*/
  gij = (gkk||gkl||gke||gkm||gks)//
        (gkl||gll||gle||glm||gls)//
        (gke||gle||gee||gem||ges)//
        (gkm||glm||gem||gmm||gms)//
        (gks||gls||ges||gms||gss);

   print gij;

   nk=ncol(gij);
   mi = -1#I(nk);    /*Initialize negative identity matrix*/
   eos = j(nk,nk,0); /*Initialize Marshallian EOS Matrix*/
   mos = j(nk,nk,0); /*Initialize Morishima EOS Matrix*/
   ep  = j(nk,nk,0); /*Initialize Price EOD Matrix*/

   /*Calculate Marshallian EOS and Price EOD Matrices*/
   i=1;
   do i=1 to nk;
   j=1;
   do j=1 to nk;
       eos[i,j] = (gij[i,j]+w[i]#w[j]+mi[i,j]#w[i])/(w[i]#w[j]);
       ep[i,j] = w[j]#eos[i,j];
   end;
   end;

   /*Calculate Morishima EOS Matrix*/
   i=1;
   do i=1 to nk;
   j=1;
   do j=1 to nk;
      mos[i,j] = ep[i,j]-ep[j,j];
   end;
   end;

run;
```

Output 26.21.6 shows the elasticity matrices that are generated by the IML procedure.

Output 26.21.6 Elasticity Matrices

Price Elasticities of Demand					
	Capital	**Labor**	**Energy**	**Materials**	**Services**
Capital	-0.338	0.227	0.0183	0.0593	0.0335
Labor	0.0650	-0.630	0.0315	0.231	0.303
Energy	0.0606	0.364	-0.0915	-0.170	-0.163
Materials	0.0167	0.227	-0.0145	-0.233	0.00367
Services	0.0679	2.148	-0.1000	0.0265	-2.142

Output 26.21.6 *continued*

Hicks-Allen Elasticities of Substitution					
	Capital	**Labor**	**Energy**	**Materials**	**Services**
Capital	-2.993	0.575	0.536	0.148	0.600
Labor	0.575	-1.594	0.921	0.574	5.435
Energy	0.536	0.921	-2.679	-0.423	-2.925
Materials	0.148	0.574	-0.423	-0.579	0.0658
Services	0.600	5.435	-2.925	0.0658	-38.437

Morishima Elasticities of Substitution					
	Capital	**Labor**	**Energy**	**Materials**	**Services**
Capital	0	0.857	0.110	0.292	2.176
Labor	0.403	0	0.123	0.463	2.445
Energy	0.399	0.994	0	0.0627	1.979
Materials	0.355	0.857	0.0771	0	2.146
Services	0.406	2.778	-0.0084	0.259	0

Own price elasticities are all negative as expected. Based on the price elasticity of demand, all pairs of inputs are substitutes except energy and services and energy and materials. The matrix of Hicks-Allen elasticities is symmetric by design. In general, most of the elasticities are less than 1 in absolute value and the degree of substitution is low. However, the elasticity between labor and services is high, indicating that the textile industry might have responded to increased competition from foreign firms that have low labor cost by shifting away from labor to greater use of services. The Morishima elasticities support this interpretation, but there are subtle differences between the two measures. The relationship between capital and services is more elastic when the Morishima elasticity is used. The estimation of elasticities thoroughly describes production in this industry and produces quantifiable measures of the relationships between inputs.

Example 26.22: Reducing Parameter Variance in a Tree Biomass Model

This example uses various dimensions of willow oak trees to model their biomass. Unlike a tree's biomass, a tree's dimensions can be measured noninvasively. The model and data for this example are taken from Parresol (1999). This biomass model uses four equations to model the bole wood, bole bark, crown, and total mass of the trees,

$$
\begin{aligned}
y_{\text{wood}} &= b_{10} + b_{11} D^2 H \\
y_{\text{bark}} &= b_{20} + b_{21} D^2 H \\
y_{\text{crown}} &= b_{30} + b_{31} \frac{D^2 H L}{1000} + b_{32} H \\
y_{\text{total}} &= b_{40} + b_{41} D^2 H + b_{42} \frac{D^2 H L}{1000} + b_{43} H
\end{aligned}
$$

where y_{wood}, y_{bark}, and y_{crown} are the three components of a tree's total biomass, y_{total}; D is the tree's diameter at breast height; H is the tree's height; and L is the tree's live crown length. The efficiency of parameter estimates in this model is improved by taking into account the correlations between errors in the equations

for the four components of the trees' biomass. Also, the efficiency of estimates is improved by taking into account heteroscedasticity in the sample of 39 trees that is used to develop the model.

In an initial OLS estimation of this model's parameters, some general properties of the model and data can be quantified using the following statements:

```
proc model data=trees;
   endo wood bark crown total;

   vol = dbh**2*height;
   cvol = vol*lcl/1000;

   wood  = b10 + b11*vol;
   bark  = b20 + b21*vol;
   crown = b30 + b31*cvol + b32*height;
   total = b40 + b41*vol + b42*cvol + b43*height;

   fit / ols outs=stree out=otree outresid;
quit;
```

Here the covariance of equation errors is saved to the data set STREE, and the residuals are saved to the data set OTREE for later analysis.

The covariance matrix of equation errors that is computed in the OLS estimation can be used in a seemingly unrelated regression (SUR) estimation of the model to improve the efficiency of parameter estimates. The total biomass of trees is restricted to equal the other three biomass components in the following SUR estimation:

```
proc model data=trees;
   endo wood bark crown total;

   vol = dbh**2*height;
   cvol = vol*lcl/1000;

   wood  = b10 + b11*vol;
   bark  = b20 + b21*vol;
   crown = b30 + b31*cvol + b32*height;
   total = b40 + b41*vol + b42*cvol + b43*height;

   fit / nools sur sdata=stree;

   restrict b10 + b20 + b30 = b40,
            b11 + b21 = b41,
            b42 = b31,
            b43 = b32;
quit;
```

Output 26.22.1 shows the parameters and standard errors for the restricted SUR tree biomass model.

Output 26.22.1 SUR Parameter Estimates for Willow Oak Model

The MODEL Procedure

Nonlinear SUR Parameter Estimates

Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|	Label
b10	59.18653	48.2435	1.23	0.2274	
b11	0.026598	0.000558	47.67	<.0001	
b20	29.65101	10.3839	2.86	0.0069	
b21	0.003225	0.000120	26.79	<.0001	
b30	133.1066	26.6246	5.00	<.0001	
b31	0.061543	0.00508	12.11	<.0001	
b32	-5.33604	1.1141	-4.79	<.0001	
b40	221.9441	62.1834	3.57	0.0010	
b41	0.029824	0.000623	47.91	<.0001	
b42	0.061543	0.00508	12.11	<.0001	
b43	-5.33604	1.1141	-4.79	<.0001	
Restrict0	-656E-14	0.1316	-0.00	1.0000	b10 + b20 + b30 = b40
Restrict1	148.5607	11355.0	0.01	0.9898	b11 + b21 = b41
Restrict2	68.66564	178.8	0.38	0.7067	b42 = b31
Restrict3	0.388714	3.5449	0.11	0.9145	b43 = b32

An analysis of heteroscedasticity in the unrestricted OLS model suggests that the efficiency of the estimation could be improved further by weighting the observations using the following variance model:

$$\sigma^2_{\text{wood}} = \exp(w_1 + w_2 \ln D^2 H)$$
$$\sigma^2_{\text{bark}} = \exp(k_1 + k_2 \ln D^2 H)$$
$$\sigma^2_{\text{crown}} = \exp(c_1 + c_2 \ln \frac{D^2 H L}{1000} - c_3 H^2)$$
$$\sigma^2_{\text{total}} = \exp(t_1 + t_2 \ln D^2 H)$$

where σ_i^2 is the variance of the ith component of the biomass and the parameters w_1, w_2, k_1, k_2, c_1, c_2, c_3, t_1, and t_2 are determined by regressing the square of the residuals from the unrestricted OLS estimation against the tree dimensions. Estimates of the parameters in the variance model can be determined using the following statements:

```
proc model data=otree;
   vol = dbh**2*height;
   cvol = vol*lcl/1000;

   eq.varwood  = log(wood*wood) - (w1 + w2*log(vol));
   eq.varbark  = log(bark*bark) - (k1 + k2*log(vol));
   eq.varcrown = log(crown*crown) - (c1 + c2*log(cvol) - c3*height**2);
   eq.vartotal = log(total*total) - (t1 + t2*log(vol));

   fit varwood varbark varcrown vartotal;
quit;
```

The biomass component variance model can be used to account for heteroscedasticity and improve the efficiency of parameter estimates by weighting observations in the biomass model. The following PROC MODEL

statements accommodate both the covariance among the biomass equations' errors and the heteroscedasticity that is observed in each component of the trees' biomass:

```
proc model data=trees;
   endo wood bark crown total;

   vol = dbh**2*height;
   cvol = vol*lcl/1000;

   wood  = b10 + b11*vol;
   bark  = b20 + b21*vol;
   crown = b30 + b31*cvol + b32*height;
   total = b40 + b41*vol + b42*cvol + b43*height;

   h.wood = exp(&w1)*vol**&w2;
   h.bark = exp(&k1)*vol**&k2;
   h.crown = exp(&c1)*cvol**&c2 * exp (-&c3*height*height);
   h.total = exp(&t1)*vol**&t2;

   fit / sur;

   restrict b10 + b20 + b30 = b40,
            b11 + b21 = b41,
            b42 = b31,
            b43 = b32;
quit;
```

Output 26.22.2 shows the parameters and standard errors for the heteroscedastic tree biomass model.

Output 26.22.2 SUR Parameter Estimates for Heteroscedastic Willow Oak Model

The MODEL Procedure

Nonlinear SUR Parameter Estimates

Parameter	Estimate	Approx Std Err	t Value	Approx Pr > |t|	Label
b10	10.22253	14.2208	0.72	0.4766	
b11	0.027515	0.000534	51.48	<.0001	
b20	15.66358	4.8173	3.25	0.0024	
b21	0.003476	0.000138	25.13	<.0001	
b30	103.8782	10.6122	9.79	<.0001	
b31	0.055226	0.00306	18.02	<.0001	
b32	-4.05163	0.4603	-8.80	<.0001	
b40	129.7643	21.5621	6.02	<.0001	
b41	0.030991	0.000614	50.47	<.0001	
b42	0.055226	0.00306	18.02	<.0001	
b43	-4.05163	0.4603	-8.80	<.0001	
Restrict0	-0.28369	1.1969	-0.24	0.8164	b10 + b20 + b30 = b40
Restrict1	-56046.2	49194.0	-1.14	0.2603	b11 + b21 = b41
Restrict2	539.3854	697.4	0.77	0.4471	b42 = b31
Restrict3	4.374001	29.6959	0.15	0.8853	b43 = b32

Output 26.22.3 shows the improved efficiency of estimates in the heteroscedastic model as compared to the homoscedastic model. For most of the parameters, the standard errors are reduced significantly in the heteroscedastic estimation.

Output 26.22.3 Standard Error Reduction

Parameter	StdErr Rel. Change
b10	(71%)
b11	(4%)
b20	(54%)
b21	15%
b30	(60%)
b31	(40%)
b32	(59%)
b40	(65%)
b41	(1%)
b42	(40%)
b43	(59%)

References

Aiken, R. C. (1985). *Stiff Computation*. New York: Oxford University Press.

Amemiya, T. (1974). "The Nonlinear Two-Stage Least-Squares Estimator." *Journal of Econometrics* 2:105–110.

Amemiya, T. (1977). "The Maximum Likelihood Estimator and the Nonlinear Three-Stage Least Squares Estimator in the General Nonlinear Simultaneous Equation Model." *Econometrica* 45:955–968.

Amemiya, T. (1985). *Advanced Econometrics*. Cambridge, MA: Harvard University Press.

Andersen, T. G., Chung, H.-J., and Sorensen, B. E. (1999). "Efficient Method of Moments Estimation of a Stochastic Volatility Model: A Monte Carlo Study." *Journal of Econometrics* 91:61–87.

Andersen, T. G., and Sorensen, B. E. (1996). "GMM Estimation of a Stochastic Volatility Model: A Monte Carlo Study." *Journal of Business and Economic Statistics* 14:328–352.

Andrews, D. W. K. (1991). "Heteroscedasticity and Autocorrelation Consistent Covariance Matrix Estimation." *Econometrica* 59:817–858.

Andrews, D. W. K., and Monahan, J. C. (1992). "Improved Heteroscedasticity and Autocorrelation Consistent Covariance Matrix Estimator." *Econometrica* 60:953–966.

Bansal, R., Gallant, A. R., Hussey, R., and Tauchen, G. E. (1993). "Computational Aspects of Nonparametric Simulation Estimation." In *Computational Techniques for Econometrics and Economic Analysis*, edited by D. A. Belsey, 3–22. Boston: Kluwer Academic.

Bansal, R., Gallant, A. R., Hussey, R., and Tauchen, G. E. (1995). "Nonparametric Estimation of Structural Models for High-Frequency Currency Market Data." *Journal of Econometrics* 66:251–287.

Bard, Y. (1974). *Nonlinear Parameter Estimation*. New York: Academic Press.

Bates, D. M., and Watts, D. G. (1981). "A Relative Offset Orthogonality Convergence Criterion for Nonlinear Least Squares." *Technometrics* 23:179–183.

Belsley, D. A., Kuh, E., and Welsch, R. E. (1980). *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity*. New York: John Wiley & Sons.

Binkley, J. K., and Nelson, G. (1984). "Impact of Alternative Degrees of Freedom Corrections in Two and Three Stage Least Squares." *Journal of Econometrics* 24:223–233.

Bowden, R. J., and Turkington, D. A. (1984). *Instrumental Variables*. Cambridge: Cambridge University Press.

Bratley, P., Fox, B. L., and Niederreiter, H. (1992). "Implementation and Tests of Low-Discrepancy Sequences." *ACM Transactions on Modeling and Computer Simulation* 2:195–213.

Breusch, T. S., and Pagan, A. R. (1979). "A Simple Test for Heteroscedasticity and Random Coefficient Variation." *Econometrica* 47:1287–1294.

Breusch, T. S., and Pagan, A. R. (1980). "The Lagrange Multiplier Test and Its Applications to Model Specification in Econometrics." *Review of Econometric Studies* 47:239–253.

Byrne, G. D., and Hindmarsh, A. C. (1975). "A Polyalgorithm for the Numerical Solution of ODEs." *ACM Transactions on Mathematical Software* 1:71–96.

Calzolari, G., and Panattoni, L. (1988). "Alternative Estimators of FIML Covariance Matrix: A Monte Carlo Study." *Econometrica* 56:701–714.

Chambers, R. G. (1988). *Applied Production Analysis: A Dual Approach*. New York: Cambridge University Press.

Chan, K. C., Karolyi, G. A., Longstaff, F. A., and Sanders, A. B. (1992). "An Empirical Comparison of Alternate Models of the Short-Term Interest Rate." *Journal of Finance* 47:1209–1227.

Christensen, L. R., Jorgenson, D. W., and Lau, L. J. (1975). "Transcendental Logarithmic Utility Functions." *American Economic Review* 65:367–383.

Dagenais, M. G. (1978). "The Computation of FIML Estimates as Iterative Generalized Least Squares Estimates in Linear and Nonlinear Simultaneous Equation Models." *Econometrica* 46:1351–1362.

Davidian, M., and Giltinan, D. M. (1995). *Nonlinear Models for Repeated Measurement Data*. New York: Chapman & Hall.

Davidson, R., and MacKinnon, J. G. (1993). *Estimation and Inference in Econometrics*. New York: Oxford University Press.

Duffie, D., and Singleton, K. J. (1993). "Simulated Moments Estimation of Markov Models of Asset Prices." *Econometrica* 61:929–952.

Dulmage, A. L., and Mendelsohn, N. F. (1958). "Coverings of Bipartite Graphs." *Canadian Journal of Mathematics* 10:517–534.

Fair, R. C. (1984). *Specification, Estimation, and Analysis of Macroeconometric Models*. Cambridge, MA: Harvard University Press.

Ferson, W. E., and Foerster, S. R. (1994). "Finite Sample Properties of the Generalized Method of Moments in Tests of Conditional Asset Pricing Models." *Journal of Financial Economics* 36:29–56. Previously released as Working Paper No. 77, University of Washington.

Fox, B. L. (1986). "Algorithm 647: Implementation and Relative Efficiency of Quasirandom Sequence Generators." *ACM Transactions on Mathematical Software* 12:362–376.

Gallant, A. R. (1987). *Nonlinear Statistical Models*. New York: John Wiley & Sons.

Gallant, A. R., and Holly, A. (1980). "Statistical Inference in an Implicit, Nonlinear, Simultaneous Equation Model in the Context of Maximum Likelihood Estimation." *Econometrica* 48:697–720.

Gallant, A. R., and Jorgenson, D. W. (1979). "Statistical Inference for a System of Simultaneous, Nonlinear, Implicit Equations in the Context of Instrumental Variables Estimation." *Journal of Econometrics* 11:275–302.

Gallant, A. R., and Long, J. R. (1997). "Estimating Stochastic Differential Equations Efficiently by Minimum Chi-Squared." *Biometrika* 84:125–141.

Gallant, A. R., and Tauchen, G. E. (2001). "Efficient Method of Moments." Working paper. http://www.econ.duke.edu/~get/wpapers/ee.pdf.

Gill, P. E., Murray, W., and Wright, M. H. (1981). *Practical Optimization*. New York: Academic Press.

Godfrey, L. G. (1978a). "Testing against General Autoregressive and Moving Average Error Models When the Regressors Include Lagged Dependent Variables." *Econometrica* 46:1293–1301.

Godfrey, L. G. (1978b). "Testing for Higher Order Serial Correlation in Regression Equations When the Regressors Include Lagged Dependent Variables." *Econometrica* 46:1303–1310.

Goldfeld, S. M., and Quandt, R. E. (1972). *Nonlinear Methods in Econometrics*. Amsterdam: North-Holland.

Goldfeld, S. M., and Quandt, R. E. (1973a). "The Estimation of Structural Shifts by Switching Regressions." *Annals of Economic and Social Measurement* 2:475–486.

Goldfeld, S. M., and Quandt, R. E. (1973b). "A Markov Model for Switching Regressions." *Journal of Econometrics* 3–16.

Goldfeld, S. M., and Quandt, R. E. (1976). *Studies in Nonlinear Estimation*. Cambridge, MA: Ballinger.

Goodnight, J. H. (1979). "A Tutorial on the Sweep Operator." *American Statistician* 33:149–158.

Gourieroux, C., and Monfort, A. (1993). "Simulation Based Inference: A Survey with Special Reference to Panel Data Models." *Journal of Econometrics* 59:5–33.

Greene, W. H. (1993). *Econometric Analysis*. 2nd ed. New York: Macmillan.

Greene, W. H. (2004). *Econometric Analysis*. New York: Macmillan.

Gregory, A. W., and Veall, M. R. (1985). "On Formulating Wald Tests for Nonlinear Restrictions." *Econometrica* 53:1465–1468.

Grunfeld, Y., and Griliches, Z. (1960). "Is Aggregation Necessarily Bad?" *Review of Economics and Statistics* 42:1–13.

Hansen, L. P. (1982). "Large Sample Properties of Generalized Method of Moments Estimators." *Econometrica* 50:1029–1054.

Hansen, L. P. (1985). "A Method for Calculating Bounds on the Asymptotic Covariance Matrices of Generalized Method of Moments Estimators." *Journal of Econometrics* 30:203–238.

Hatanaka, M. (1978). "On the Efficient Estimation Methods for the Macro-economic Models Nonlinear in Variables." *Journal of Econometrics* 8:323–356.

Hausman, J. A. (1978). "Specification Tests in Econometrics." *Econometrica* 46:1251–1271.

Hausman, J. A., and Taylor, W. E. (1982). "A Generalized Specification Test." *Economics Letters* 8:239–245.

Henze, N., and Zirkler, B. (1990). "A Class of Invariant Consistent Tests for Multivariate Normality." *Communications in Statistics—Theory and Methods* 19:3595–3617.

Johnston, J. (1984). *Econometric Methods*. 3rd ed. New York: McGraw-Hill.

Jorgenson, D. W., and Laffont, J. (1974). "Efficient Estimation of Nonlinear Simultaneous Equations with Additive Disturbances." *Annals of Social and Economic Measurement* 3:615–640.

Joy, C., Boyle, P. P., and Tan, K. S. (1996). "Quasi-Monte Carlo Methods in Numerical Finance." *Management Science* 42:926–938.

LaMotte, L. R. (1994). "A Note on the Role of Independence in *t* Statistics Constructed from Linear Statistics in Regression Models." *American Statistician* 48:238–240.

Lee, B., and Ingram, B. (1991). "Simulation Estimation of Time Series Models." *Journal of Econometrics* 47:197–205.

MacKinnon, J. G., and White, H. (1985). "Some Heteroskedasticity-Consistent Covariance Matrix Estimators with Improved Finite Sample Properties." *Journal of Econometrics* 29:305–325.

Maddala, G. S. (1977). *Econometrics*. New York: McGraw-Hill.

Mardia, K. V. (1970). "Measures of Multivariate Skewness and Kurtosis with Applications." *Biometrika* 57:519–530.

Mardia, K. V. (1974). "Applications of Some Measures of Multivariate Skewness and Kurtosis in Testing Normality and Robustness Studies." *Indian Journal of Statistics* 36:115–128.

Matis, J. H., Miller, T. H., and Allen, D. M. (1991). *Metal Ecotoxicology: Concepts and Applications.* Edited by M. C. Newman and A. W. McIntosh. Chelsea, MI: Lewis Publishers.

Matsumoto, M., and Nishimura, T. (1998). "Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-random Number Generator." *ACM Transactions on Modeling and Computer Simulation* 8:3–30.

McFadden, D. (1989). "A Method of Simulated Moments for Estimation of Discrete Response Models without Numerical Integration." *Econometrica* 57:995–1026.

McNeil, A., Frey, R., and Embrechts, P. (2005). *Quantitative Risk Management: Concepts, Techniques, and Tools*. Princeton, NJ: Princeton University Press.

Messer, K., and White, H. (1994). "A Note on Computing the Heteroskedasticity Consistent Covariance Matrix Using Instrumental Variable Techniques." *Oxford Bulletin of Economics and Statistics* 46:181–184.

Mikhail, W. M. (1975). "A Comparative Monte Carlo Study of the Properties of Economic Estimators." *Journal of the American Statistical Association* 70:94–104.

Miller, D. M. (1984). "Reducing Transformation Bias in Curve Fitting." *American Statistician* 38:124–126.

Morelock, M. M., Pargellis, C. A., Graham, E. T., Lamarre, D., and Jung, G. (1995). "Time-Resolved Ligand Exchange Reactions: Kinetic Models for Competitive Inhibitors with Recombinant Human Renin." *Journal of Medical Chemistry* 38:1751–1761.

Nelsen, R. B. (1999). *An Introduction to Copulas*. New York: Springer-Verlag.

Newey, W. K., and West, D. W. (1987). "A Simple, Positive Semi-definite, Heteroskedasticity and Autocorrelation Consistent Covariance Matrix." *Econometrica* 55:703–708.

Noble, B., and Daniel, J. W. (1977). *Applied Linear Algebra*. Englewood Cliffs, NJ: Prentice-Hall.

Ortega, J. M., and Rheinbolt, W. C. (1970). *Iterative Solution of Nonlinear Equations in Several Variables*. Burlington, MA: Academic Press.

Pakes, A., and Pollard, D. (1989). "Simulation and the Asymptotics of Optimization Estimators." *Econometrica* 57:1027–1057.

Parresol, B. R. (1999). "Assessing Tree and Stand Biomass: A Review with Examples and Critical Comparisons." *Forest Science* 45:573–593.

Parzen, E. (1957). "On Consistent Estimates of the Spectrum of a Stationary Time Series." *Annals of Mathematical Statistics* 28:329–348.

Pearlman, J. G. (1980). "An Algorithm for the Exact Likelihood of a High-Order Autoregressive–Moving Average Process." *Biometrika* 67:232–233.

Petzold, L. R. (1982). "Differential/Algebraic Equations Are Not ODEs." *SIAM Journal on Scientific and Statistical Computing* 3:367–384.

Phillips, C. B., and Park, J. Y. (1988). "On Formulating Wald Tests of Nonlinear Restrictions." *Econometrica* 56:1065–1083.

Pindyck, R. S., and Rubinfeld, D. L. (1981). *Econometric Models and Econometric Forecasts*. 2nd ed. New York: McGraw-Hill.

Pothen, A., and Fan, C.-J. (1990). "Computing the Block Triangular Form of a Sparse Matrix." *ACM Transactions on Mathematical Software* 16:303–324.

Rebonato, R., and Jäckel, P. (1999). "The Most General Methodology for Creating Valid Correlation Matrix for Risk Management and Option Pricing Purposes." *Journal of Risk* 2:17–27.

Savin, N. E., and White, K. J. (1978). "Testing for Autocorrelation with Missing Observations." *Econometrica* 46:59–67.

Sobol, I. M. (1994). *A Primer for the Monte Carlo Method*. Boca Raton, FL: CRC Press.

Srivastava, V., and Giles, D. E. A. (1987). *Seemingly Unrelated Regression Equation Models*. New York: Marcel Dekker.

Theil, H. (1971). *Principles of Econometrics*. New York: John Wiley & Sons.

Thursby, J. (1982). "Misspecification, Heteroscedasticity, and the Chow and Goldfield-Quandt Test." *Review of Economics and Statistics* 64:314–321.

Venzon, D. J., and Moolgavkar, S. H. (1988). "A Method for Computing Profile-Likelihood-Based Confidence Intervals." *Journal of the Royal Statistical Society, Series C* 37:87–94.

White, H. (1980). "A Heteroskedasticity-Consistent Covariance Matrix Estimator and a Direct Test for Heteroskedasticity." *Econometrica* 48:817–838.

Wu, D. M. (1973). "Alternative Tests of Independence between Stochastic Regressors and Disturbances." *Econometrica* 41:733–750.

Chapter 27
The PANEL Procedure

Contents

Overview: PANEL Procedure **1794**
Getting Started: PANEL Procedure **1796**
Syntax: PANEL Procedure **1798**
- Functional Summary 1798
- PROC PANEL Statement 1801
- BY Statement 1803
- CLASS Statement 1803
- COMPARE Statement 1803
- FLATDATA Statement 1806
- ID Statement 1807
- INSTRUMENTS Statement 1807
- LAG, CLAG, SLAG, XLAG, and ZLAG Statements 1809
- MODEL Statement 1810
- OUTPUT Statement 1827
- RESTRICT Statement 1827
- TEST Statement 1828

Details: PANEL Procedure **1829**
- Specifying the Input Data 1829
- Specifying the Regression Model 1830
- Unbalanced Data 1831
- Missing Values 1831
- Computational Resources 1831
- Restricted Estimates 1832
- Notation 1833
- One-Way Fixed-Effects Model 1833
- Two-Way Fixed-Effects Model 1835
- Balanced Panels 1835
- Unbalanced Panels 1837
- First-Differenced Methods for One-Way and Two-Way Models 1840
- Between Estimators 1840
- Pooled Estimator 1841
- One-Way Random-Effects Model 1841
- Two-Way Random-Effects Model 1844
- Hausman-Taylor Estimation 1849
- Amemiya-MaCurdy Estimation 1850
- Parks Method (Autoregressive Model) 1851

Da Silva Method (Variance-Component Moving Average Model) 1853
Dynamic Panel Estimators 1855
Linear Hypothesis Testing 1865
Heteroscedasticity-Corrected Covariance Matrices 1866
Heteroscedasticity- and Autocorrelation-Consistent Covariance Matrices 1870
R-Square 1872
Specification Tests 1873
Panel Data Poolability Test 1874
Panel Data Cross-Sectional Dependence Test 1875
Panel Data Unit Root Tests 1876
Lagrange Multiplier (LM) Tests for Cross-Sectional and Time Effects 1887
Tests for Serial Correlation and Cross-Sectional Effects 1889
Troubleshooting 1892
Creating ODS Graphics 1893
OUTPUT OUT= Data Set 1894
OUTEST= Data Set 1894
OUTTRANS= Data Set 1895
Printed Output 1896
ODS Table Names 1897
Examples: PANEL Procedure **1898**
Example 27.1: Analyzing Demand for Liquid Assets 1898
Example 27.2: The Airline Cost Data: Fixtwo Model 1903
Example 27.3: The Airline Cost Data: Further Analysis 1908
Example 27.4: The Airline Cost Data: Random-Effects Models 1909
Example 27.5: Panel Study of Income Dynamics (PSID): Hausman-Taylor Models . . 1912
Example 27.6: The Cigarette Sales Data: Dynamic Panel Estimation with GMM . . . 1916
Example 27.7: Using the FLATDATA Statement 1919
References **1920**

Overview: PANEL Procedure

The PANEL procedure analyzes a class of linear econometric models that commonly arise when time series and cross-sectional data are combined. This type of pooled data on time series cross-sectional bases is often referred to as panel data. Typical examples of panel data include observations over time on households, countries, firms, trade, and so on. For example, in the case of survey data on household income, the panel is created by repeatedly surveying the same households in different time periods (years).

The panel data models can be grouped into several categories depending on the structure of the error term. The PANEL procedure uses the following error structures and the corresponding methods to analyze data:

- one-way and two-way models
- fixed-effects, random-effects, and hybrid models

- autoregressive models
- moving average models

A one-way model depends only on the cross section to which the observation belongs. A two-way model depends on both the cross section and the time period to which the observation belongs.

Apart from the possible one-way or two-way nature of the effect, the other dimension of difference between the possible specifications is the nature of the cross-sectional or time-series effect. The models are referred to as fixed-effects models if the effects are nonrandom and as random-effects models otherwise.

If the effects are fixed, the models are essentially regression models with dummy variables that correspond to the specified effects. For fixed-effects models, ordinary least squares (OLS) estimation is the best linear unbiased estimator. Random-effects models use a two-stage approach. In the first stage, variance components are calculated by using methods described by: Fuller and Battese (1974); Wansbeek and Kapteyn (1989); Wallace and Hussain (1969); Nerlove (1971). In the second stage, variance components are used to standardize the data, and ordinary least squares (OLS) regression is performed.

Random effects models are more efficient than fixed effects models, and they have the ability to estimate effects for variables that do not vary within cross sections. The cost of these added features is that random effects models carry much more stringent assumptions than their fixed-effects counterparts. The PANEL procedure supports models that blend the desirable features of both random and fixed effects. These hybrid models are those by Hausman and Taylor (1981) and Amemiya and MaCurdy (1986).

Two types of models in the PANEL procedure accommodate an autoregressive structure: The Parks method estimates a first-order autoregressive model with contemporaneous correlation, and the dynamic panel estimator estimates an autoregressive model with lagged dependent variable.

The Da Silva method estimates a mixed variance-component moving-average error process. The regression parameters are estimated by two-step generalized least squares (GLS).

The PANEL procedure enhances the features that were previously implemented in the TSCSREG procedure. The following list shows the most important additions.

- You can fit models for dynamic panel data using the generalized method of moments (GMM).
- The Hausman-Taylor and Amemiya-MaCurdy estimators offer a compromise between fixed and random effects estimation in models where some variables are correlated with individual effects.
- The MODEL statement supports between and pooled estimation.
- The variance components for random-effects models can be calculated for both balanced and unbalanced panels by using the methods described by: Fuller and Battese (1974); Wansbeek and Kapteyn (1989); Wallace and Hussain (1969); Nerlove (1971).
- The CLASS statement allows classification variables (and their interactions) directly into the analysis.
- The TEST statement includes new options for Wald, LaGrange multiplier, and the likelihood ratio tests.
- The new RESTRICT statement specifies linear restrictions on the parameters.
- The FLATDATA statement enables the data to be in a compressed (or wide) form.

- Several methods that produce heteroscedasticity-consistent (HCCME) and heteroscedasticity- and Autocorrelation-Consistent (HAC) covariance matrices are added because the presence of heteroscedasticity and autocorrelation can result in inefficient and biased estimates of the covariance matrix in an OLS framework.
- Tests are added for poolability, panel stationarity, the existence of cross sectional and time effects, autocorrelation, and cross sectional dependence.
- The LAG and related statements provide functionality for creating lagged variables from within the PANEL procedure. Using these statements is preferable to using the DATA step because creating lagged variables in a panel setting can prove difficult, often requiring multiple loops and careful consideration of missing values.

  Working within the PANEL procedure makes the creation of lagged values easy. The missing values can be replaced with zeros, overall mean, time mean, or cross section mean by using the LAG, ZLAG, XLAG, SLAG, and CLAG statements, respectively.
- The OUTPUT statement enables you to output data and estimates that can be used in other analyses.

Getting Started: PANEL Procedure

The following statements use the cost function data from Greene (1990) to estimate a variance components model. The variable PRODUCTION is the log of output in millions of kilowatt-hours, and COST is the log of cost in millions of dollars. See Greene (1990) for details.

```
data greene;
   input firm year production cost @@;
datalines;
1 1955   5.36598   1.14867  1 1960   6.03787   1.45185
1 1965   6.37673   1.52257  1 1970   6.93245   1.76627
2 1955   6.54535   1.35041  2 1960   6.69827   1.71109
2 1965   7.40245   2.09519  2 1970   7.82644   2.39480
3 1955   8.07153   2.94628  3 1960   8.47679   3.25967

   ... more lines ...
```

You decide to fit the following model to the data:

$$C_{it} = \text{Intercept} + \boldsymbol{\beta} P_{it} + v_i + e_t + \epsilon_{it} \quad i = 1, \ldots, N;\ t = 1, \ldots, T$$

where C_{it} and P_{it} represent the cost and production, and v_i, e_t and ϵ_{it} are the cross-sectional, time series, and error variance components.

If you assume that the time and cross-sectional effects are random, you are left with four possible estimators for the variance components. You choose Fuller-Battese.

The following statements fit this model:

```
proc sort data=greene;
   by firm year;
run;

proc panel data=greene;
   model cost = production / rantwo vcomp = fb;
   id firm year;
run;
```

The PANEL procedure output is shown in Figure 27.1. A model description is printed first, which reports the estimation method used and the number of cross sections and time periods. Fit statistics and variance components estimates are printed next. A Hausman specification test compares this model to its fixed-effects analog. Finally, the table of regression parameter estimates shows the estimates, standard errors, and *t* tests.

Figure 27.1 The Variance Components Estimates

The PANEL Procedure
Fuller and Battese Variance Components (RanTwo)

Dependent Variable: cost

Model Description	
Estimation Method	RanTwo
Number of Cross Sections	6
Time Series Length	4

Fit Statistics			
SSE	0.3481	DFE	22
MSE	0.0158	Root MSE	0.1258
R-Square	0.8136		

Variance Component Estimates	
Variance Component for Cross Sections	0.046907
Variance Component for Time Series	0.00906
Variance Component for Error	0.008749

Hausman Test for Random Effects			
Coefficients	DF	m Value	Pr > m
1	1	26.46	<.0001

Parameter Estimates					
Variable	DF	Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	-2.99992	0.6478	-4.63	0.0001
production	1	0.746596	0.0762	9.80	<.0001

Syntax: PANEL Procedure

The following statements are used with the PANEL procedure:

PROC PANEL *options* ;
BY *variables* ;
CLASS *variables* < / *options* > ;
COMPARE < *model-list* > < / *options* > ;
FLATDATA *options* < / **OUT**=*SAS-data-set* > ;
ID *cross-section-id time-series-id* ;
INSTRUMENTS *options* ;
LAG *lag-specifications* / **OUT**=*SAS-data-set* ;
MODEL *response* = *regressors* < / *options* > ;
OUTPUT < *options* > ;
RESTRICT *equation1* < ,*equation2*. . . > ;
TEST *equation1* < ,*equation2*. . . > ;

Functional Summary

The statements and options used with the PANEL procedure are summarized in the following table.

Description	Statement	Option
Data Set Options		
Includes correlations in the OUTEST= data set	PANEL	CORROUT
Includes covariances in the OUTEST= data set	PANEL	COVOUT
Specifies the input data set	PANEL	DATA=
Specifies variables to keep but not transform	FLATDATA	KEEP=
Specifies the output data set for CLASS STATEMENT	CLASS	OUT =
Specifies the output data set	FLATDATA	OUT =
Specifies the name of an output SAS data set	OUTPUT	OUT=
Writes parameter estimates to an output data set	PANEL	OUTEST=
Writes the transformed series to an output data set	PANEL	OUTTRANS=
Requests that the procedure produce graphics via the Output Delivery System	PANEL	PLOTS=
Declaring the Role of Variables		
Specifies BY-group processing	BY	
Specifies the classification variables	CLASS	
Transfers the data into uncompressed form	FLATDATA	

Description	Statement	Option
Specifies the cross section and time ID variables	ID	
Declares instrumental variables	INSTRUMENTS	
Lag Generation		
Specifies output data set for lags where missing values are replaced with the cross section mean	CLAG	OUT=
Specifies output data set for lags with missing values included	LAG	OUT=
Specifies output data set for lags where missing values are replaced with the time period mean	SLAG	OUT=
Specifies output data set for lags where missing values are replaced with overall mean	XLAG	OUT=
Specifies output data set for lags where missing values are replaced with zero	ZLAG	OUT=
Printing Control Options		
Prints correlations of the estimates	MODEL	CORRB
Prints covariances of the estimates	MODEL	COVB
Suppresses printed output	MODEL	NOPRINT
Requests that the procedure produce graphics via the Output Delivery System	MODEL	PLOTS=
Prints fixed effects	MODEL	PRINTFIXED
Performs tests of linear hypotheses	TEST	
Model Estimation Options		
Specifies the Amemiya-MaCurdy model	MODEL	AMACURDY
Requests the R_ρ statistic for serial correlation under fixed effects	MODEL	BFN
Requests the Baltagi and Li joint Lagrange multiplier (LM) test for serial correlation and random cross-sectional effects	MODEL	BL91
Requests the Baltagi and Li LM test for first-order correlation under fixed effects	MODEL	BL95
Requests the Breusch-Pagan test for one-way random effects	MODEL	BP
Requests the Breusch-Pagan test for two-way random effects	MODEL	BP2
Requests the Bera, Sosa Escudero, and Yoon modified Rao's score test	MODEL	BSY
Specifies the between-groups model	MODEL	BTWNG
Specifies the between-time-periods model	MODEL	BTWNT
Requests the Berenblut-Webb statistic for serial correlation under fixed effects	MODEL	BW
Requests cross-sectional dependence tests	MODEL	CDTEST
Requests the clustered HCCME estimator for the covariance matrix	MODEL	CLUSTER

Description	Statement	Option
Specifies the Da Silva method	MODEL	DASILVA
Requests the Durbin-Watson statistic for serial correlation under fixed effects	MODEL	DW
Specifies the one-way fixed-effects model	MODEL	FIXONE
Specifies the one-way fixed-effects model with respect to time	MODEL	FIXONETIME
Specifies the two-way fixed-effects model	MODEL	FIXTWO
Specifies the first-differenced methods for one-way models	MODEL	FDONE
Specifies the first-differenced methods for one-way models with respect to time	MODEL	FDONETIME
Specifies the first-differenced methods for two-way models	MODEL	FDTWO
Specifies the Moore-Penrose generalized inverse	MODEL	GINV = G4
Requests the Gourieroux, Holly, and Monfort test for two-way random effects	MODEL	GHM
Specifies the dynamic panel model (one-step GMM estimation)	MODEL	GMM1
Specifies the dynamic panel model (two-step GMM estimation)	MODEL	GMM2
Requests the HAC estimator for the variance-covariance matrix	MODEL	HAC=
Requests the HCCME estimator for the covariance matrix	MODEL	HCCME=
Requests the Honda test for one-way random effects	MODEL	HONDA
Requests the Honda test for two-way random effects	MODEL	HONDA2
Specifies the Hausman-Taylor model	MODEL	HTAYLOR
Specifies the dynamic panel estimator model (iterated GMM)	MODEL	ITGMM
Requests the King and Wu test for two-way random effects	MODEL	KW
Specifies the order of the moving average error process for Da Silva method	MODEL	M=
Suppresses the intercept term	MODEL	NOINT
Specifies the Parks method	MODEL	PARKS
Prints the Φ matrix for Parks method	MODEL	PHI
Specifies the pooled model	MODEL	POOLED
Requests poolability tests for one-way fixed effects and pooled model	MODEL	POOLTEST
Specifies the one-way random-effects model	MODEL	RANONE
Specifies the two-way random-effects model	MODEL	RANTWO

Description	Statement	Option
Prints autocorrelation coefficients for Parks method	MODEL	RHO
Controls the check for singularity	MODEL	SINGULAR=
Specifies the method for panel unit root/stationarity test	MODEL	UROOTTEST=
Specifies the method for the variance components estimator	MODEL	VCOMP=
Specifies linear equality restrictions on the parameters	RESTRICT	
Specifies the TEST statement	TEST	WALD, LM, LR
Requests the Wooldridge (2002) test for the presence of unobserved effects	MODEL	WOOLDRIDGE02
Comparing Models		
Create tables with side-by-side model comparisons	COMPARE	

PROC PANEL Statement

PROC PANEL *options* **;**

The following options can be specified on the PROC PANEL statement.

DATA=*SAS-data-set*
names the input data set. The input data set must be sorted by cross section and by time period within cross section. If you omit the DATA= option, the most recently created SAS data set is used.

OUTEST=*SAS-data-set*
names an output data set to contain the parameter estimates. When the OUTEST= option is not specified, the OUTEST= data set is not created. See the section "OUTEST= Data Set" on page 1894 for details about the structure of the OUTEST= data set.

OUTTRANS=*SAS-data-set*
names an output data set to contain the transformed data. Several models supported by the PANEL procedure are estimated by first transforming the data and then applying standard regression techniques to the transformed data. This option allows you access to the transformed data. See the section "OUTTRANS= Data Set" on page 1895 for details about the structure of the OUTTRANS= data set.

OUTCOV
COVOUT
writes the standard errors and covariance matrix of the parameter estimates to the OUTEST= data set. See the section "OUTEST= Data Set" on page 1894 for details.

OUTCORR

CORROUT

writes the correlation matrix of the parameter estimates to the OUTEST= data set. See the section "OUTEST= Data Set" on page 1894 for details.

PLOTS < (*global-plot-options* **< (NCROSS=***value***) >) > < = (***specific-plot-options***) >**

selects plots to be produced via the Output Delivery System. For general information about ODS Graphics, see Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*). The *global-plot-options* apply to all relevant plots generated by the PANEL procedure.

Global Plot Options

The following *global-plot-options* are supported:

ONLY

suppresses the default plots. Only the plots specifically requested are produced.

UNPACKPANEL

UNPACK

displays each graph separately. By default, some graphs can appear together in a single panel.

NCROSS=*value*

specifies the number of cross sections to be combined into one time series plot.

Specific Plot Options

The following *specific-plot-options* are supported:

ACTSURFACE	produces a surface plot of actual values.
ALL	produces all appropriate plots.
FITPLOT	plots the predicted and actual values.
NONE	suppresses all plots.
PREDSURFACE	produces a surface plot of predicted values.
QQ	produces a QQ plot of residuals.
RESIDSTACK | RESSTACK	produces a stacked plot of residuals.
RESIDSURFACE	produces a surface plot of residual values.
RESIDUAL | RES	plots the residuals.
RESIDUALHISTOGRAM | RESIDHISTOGRAM	plots the histogram of residuals.

For more details, see the section "Creating ODS Graphics" on page 1893.

In addition, any of the following MODEL statement options can be specified in the PROC PANEL statement: CORRB, COVB, FIXONE, FIXONETIME, FIXTWO, FDONE, FDONETIME, FDTWO, BTWNG, BTWNT, POOLED, RANONE, RANTWO, HTAYLOR, AMACURDY, PARKS, DASILVA, NOINT, NOPRINT, PRINTFIXED, M=, PHI, RHO, VCOMP=, and SINGULAR=. When specified in the PROC PANEL statement, these options apply globally to every MODEL statement. See the section "MODEL Statement" on page 1810 for a complete description of each of these options.

BY Statement

BY *variables* **;**

A BY statement obtains separate analyses on observations in groups that are defined by the BY variables. When a BY statement appears, the input data set must be sorted both by the BY variables and by cross section and time period within the BY groups.

The following statements show an example:

```
proc sort data=a;
   by byvar1 byvar2 csid tsid;
run;

proc panel data=a;
   by byvar1 byvar2;
   id csid tsid;
   ...
run;
```

CLASS Statement

CLASS *variables* < /**OUT=***SAS-data-set*> **;**

The CLASS statement names the classification variables to be used in the analysis. Classification variables can be either character or numeric.

The **OUT=***SAS-data=set* option enables you to output the regression dummy variables used to represent the classification variables, augmented by a copy of the original data.

COMPARE Statement

COMPARE < *model-list*> < / *options*> **;**

A COMPARE statement creates tables of side-by-side comparisons of parameter estimates and other model statistics. You can fit multiple models simultaneously by specifying multiple MODEL statements, and you can specify a COMPARE statement to create tables that compare the models.

The COMPARE statement creates two tables: the first table compares model fit statistics such as R-square and mean square error; the second table compares regression coefficients, their standard errors, and (optionally) *t* tests.

By default, comparison tables are created for all fitted models, but you can use the optional *model-list* to limit the comparison to a subset of the fitted models. The *model-list* consists of a set of model labels, as specified in the MODEL statement; for more information see the section "MODEL Statement" on page 1810. If a model does not have a label, you refer to it generically as "Model *i*," where the corresponding model is the *i*th MODEL statement specified. If model labels are longer than 16 characters, then only the first 16 characters of the labels in *model-list* are used to determine a match.

You can specify one or more COMPARE statements. The following code illustrates the use of the COMPARE statement:

```
proc panel data=a;
   id csid tsid;
   mod_one: model y = x1 x2 x3    / fixone;
   model "Second Model" y = x1 x2 / fixone;
   model y = x1 x2 x3 x4          / fixone;
   compare;
   compare "Second Model" "Model 3";
run;
```

The first COMPARE statement compares all three fitted models. The second COMPARE statement compares the second and third models and uses the generic "Model 3" to identify the third model.

You can specify the following *options* in the COMPARE statement after a slash (/):

MSTAT(*mstat-list***)**
specifies a list of model fit statistics to be displayed. A set of statistics is displayed by default, but you can use this option to specify a custom set of model statistics.

mstat-list can contain one or more of the following keywords:

ALL
displays all model fit statistics. Not all statistics are appropriate for all models, and thus not always calculated. A blank cell in the table indicates that the statistic is not appropriate for that model.

DFE
displays the error degrees of freedom. This statistic is displayed by default.

F
displays the *F* statistic of the overall test for no fixed effects.

FNUMDF
displays the numerator degrees of freedom of the overall test for no fixed effects.

FDENDF
displays the denominator degrees of freedom of the overall test for no fixed effects.

M
displays the Hausman test *m* statistic.

MDF
displays the Hausman test degrees of freedom.

MSE
displays the model mean square error. This statistic is displayed by default.

NCS
displays the number of cross sections. This statistic is displayed by default.

NONE
suppresses the table of model fit statistics.

NTS
displays the maximum time-series length. This statistic is displayed by default.

PROBF
displays the significance level of the overall test for no fixed effects.

PROBM
displays the significance level of the Hausman test.

RMSE
displays the model root mean square error.

RSQUARE
displays the model R-square fit statistic. This statistic is displayed by default.

SSE
displays the model sum of squares.

VARCS
displays the variance component due to cross sections in random-effects models.

VARERR
displays the variance component due to error in random-effects models.

VARTS
displays the variance component due to time series in random-effects models.

OUTPARM=*SAS-data-set*
names an output data set to contain the data from the comparison table for parameter estimates, standard errors, and *t* tests.

OUTSTAT=*SAS-data-set*
names an output data set to contain the data from the comparison table for model fit statistics, such as R-square and mean square error.

PSTAT(*pstat-list***)**
specifies a list of parameter statistics to be displayed. By default, estimated regression coefficients and their standard errors are displayed. Use this option to specify a custom set of parameter statistics.

pstat-list can contain one or more of the following keywords:

ALL
displays all parameter statistics.

ESTIMATE
displays the estimated regression coefficient. This statistic is displayed by default.

NONE
suppresses the table of parameter statistics.

STDERR
displays the standard error. This statistic is displayed by default.

PROBT
displays the significance level of the *t* test.

T
displays the *t* statistic.

See Example 27.4 for a demonstration of the COMPARE statement.

FLATDATA Statement

FLATDATA *options* < / **OUT=***SAS-data-set* > **;**

The FLATDATA statement allows you to use PROC PANEL when you have data in flat (or wide) format, where all measurements for a given cross section are contained within one observation. See Example 27.7 for a demonstration. If you have flat data, you should issue the FLATDATA statement first in PROC PANEL, before you reference any variables you create with this statement.

The following options must be specified in the FLATDATA statement:

BASE=(*basename basename . . . basename***)**
specifies the variables that are to be transformed into a proper PROC PANEL format. All variables to be transformed must be named according to the convention: *basename*_timeperiod. You supply just the basenames, and the procedure extracts the appropriate variables to transform. If some year's data are missing for a variable, then PROC PANEL detects this and fills in with missing values.

INDID=*variable*
names the variable in the input data set that uniquely identifies each individual. The INDID variable can be a character or numeric variable.

KEEP=(*variable variable . . . variable***)**
specifies the variables that are to be copied without any transformation. These variables remain constant with respect to time when the data are converted to PROC PANEL format. This is an optional item.

TSNAME=*name*
specifies a name for the generated time identifier. The name must satisfy the requirements for the name of a SAS variable. The name can be quoted, but it must not be the name of a variable in the input data set.

The following options can be specified on the FLATDATA statement after the slash (/):

OUT =*SAS-data-set*
saves the converted flat data set to a PROC PANEL formatted data set.

ID Statement

ID *cross-section-id time-series-id* **;**

The ID statement is used to specify variables in the input data set that identify the cross section and time period for each observation.

When an ID statement is used, the PANEL procedure verifies that the input data set is sorted by the cross section ID variable and by the time series ID variable within each cross section. The PANEL procedure also verifies that the time series ID values are the same for all cross sections.

To make sure the input data set is correctly sorted, use PROC SORT to sort the input data set with a BY statement with the variables listed exactly as they are listed in the ID statement, as shown in the following:

```
proc sort data=a;
   by csid tsid;
run;

proc panel data=a;
   id csid tsid;
   ...
run;
```

INSTRUMENTS Statement

INSTRUMENTS *options* **;**

The INSTRUMENTS statement selects variables to be used in the moment condition equations of the dynamic panel estimator. It is also used to specify variables that are correlated with individual effects during Hausman-Taylor (HTAYLOR) or Amemiya-MaCurdy (AMACURDY) estimation.

You can specify the following *options*:

CONSTANT
includes an intercept (column of ones) as an uncorrelated exogenous instrument.

CORRELATED=(*variable variable . . . variable***)**
specifies a list of variables correlated with the unobserved individual effects. These variables are correlated with the error terms in the level equations, so they are not used in forming moment conditions from those equations.

DEPVAR<(LEVEL | DIFF | DIFFERENCE | BOTH)>
specifies instruments related to the dependent variable. With LEVEL, the lagged dependent variables are included as instruments for differenced equations. With DIFFERENCE, the differenced dependent variable is included as instruments for equations. With BOTH or nothing specified, both level and differenced dependent variables are included in the instrument matrix.

DIFFEQ=(*variable variable . . . variable***)**

DIFFERENCEDEQ=(*variable variable . . . variable***)**
specifies a list of variables that can be used as standard instruments for the differenced equations.

EXOGENOUS=(*variable variable . . . variable***)**
specifies a list of variables that are not correlated with the disturbances given the unobserved individual effects.

LEVELEQ=(*variable variable . . . variable***)**

LEVELSEQ=(*variable variable . . . variable***)**
specifies a list of variables that can be used as standard instruments for the level equations.

PREDETERMINED=(*variable variable . . . variable***)**
specifies a list of variables whose future realizations can be correlated with the disturbances but whose present and past realizations are not conditional on the individual effects.

For estimation with dynamic panels, because a variable can be used as an instrument only if it is either exogenous or predetermined, the variables listed in the CORRELATED= option must be included in either the EXOGENOUS= list or the PREDETERMINED= list. If a variable listed in the EXOGENOUS= list is not included in the CORRELATED= list, then it is considered to be uncorrelated to the error term in the level equations, which consist only of the individual effects and the disturbances. Moreover, it is uncorrelated with the error term in the differenced equations, which consist only of the disturbances. For example, in the following statements, the exogenous instruments are Z1, Z2, and X1. Because Z1 is an instrument that is correlated with the individual fixed effects, it is included in the differenced equations but not in the level equations. Because Z2 is not correlated with either the individual effects or the disturbances, it is included in both the level equations and the differenced equations.

```
proc panel data=a;
   instruments exogenous = (Z1 Z2 X1) correlated = (Z1) constant depvar;
   model Y = X1 X2 X3 / gmm1;
run;
```

For a detailed discussion of the model set up and the use of the INSTRUMENTS statement for dynamic panels, see "Dynamic Panel Estimators" on page 1855.

For Hausman-Taylor or Amemiya-MaCurdy estimation, you specify which variables are correlated with the individual effects by using the CORRELATED= option. All other options are ignored. For these estimators, the specified variables are not instruments—they are merely designated as correlated. The instruments are determined by the method; for more information, see the section "Hausman-Taylor Estimation" on page 1849.

```
proc panel data=a;
   instruments correlated = (X2 Z2);
   model Y = X1 X2 Z1 Z2 / htaylor;
run;
```

One INSTRUMENT statement is required for each MODEL statement. For example, if there are two models to be estimated by using GMM1 within one PANEL procedure, then there should be two INSTRUMENT statements, as follows:

```
proc panel data=test;
  id cs ts;
  instruments depvar predetermined = (x1 x2)
                     exogenous     = (x3 x4 x5)
                     correlated    = (x3 x4 x5);
  model y = y_1 x1 x2 / gmm1;
  instruments        predetermined = (x2 x4)
                     exogenous     = (x3 x5)
                     correlated    = (x3 x4);
  model y = y_1 x2 / gmm1;
run;
```

LAG, CLAG, SLAG, XLAG, and ZLAG Statements

LAG $var_1 (lag_1\ lag_2 \ldots lag_T) \ldots var_N (lag_1\ lag_2 \ldots lag_T)$ / **OUT**= *SAS-data-set* **;**

Generally, creating lags of variables in a panel setting is a tedious process requiring many DATA step statements. The PANEL procedure enables you to generate lags of any series without stepping through individual time series. The LAG statement is a data set generation tool. You can specify more than one LAG statement. Analyzing the generated lagged data requires a subsequent call to PROC PANEL.

The OUT= option is required. The output data set includes all variables in the input set, plus the genereated lags, named using the convention *varname_lag*. The LAG statement tends to generate many missing values in the data. This can be problematic because the number of usable observations diminishes with the lag length. Therefore, PROC PANEL offers the following alternatives to the LAG statement. The following statements can be used in place of LAG with otherwise identical syntax:

CLAG replaces missing values with the cross section mean for that variable.

SLAG replaces missing values with the time mean for that variable.

XLAG replaces missing values with the overall mean for that variable.

ZLAG replaces missing values with 0 for that variable.

For all of the above, missing values are replaced only if they are in the generated (lagged) series. Missing variables in the original variables remain unaltered.

Assume that data set A has been sorted by cross section and by time period within cross section and that the variables are Y, X1, X2, and X3. The following PROC PANEL statements generate a series with lags 1 and 3 of the X1 variable; lags 3, 6, and 9 of the X2 variable; and lag 2 of the X3 variable.

```
proc panel data=A;
   id i t;
   lag X1(1 3) X2(3 6 9) X3(2) / out=A_lag;
run;
```

If you want zeroing instead of missing values, then use ZLAG in place of LAG.

```
proc panel data=A;
   id i t;
   zlag X1(1 3) X2(3 6 9) X3(2) / out=A_zlag;
run;
```

Similarly, you can specify XLAG to replace with overall means, SLAG to replace with time means, and CLAG to replace with cross section means.

MODEL Statement

MODEL < *"string"*> *response* = *regressors* < / *options*> **;**

The MODEL statement specifies the regression model, the error structure that is assumed for the regression residuals, and the estimation technique to be used. The response variable (*response*) on the left side of the equal sign is regressed on the independent variables (*regressors*), which are listed after the equal sign. You can specify any number of MODEL statements. For each MODEL statement, you can specify only one *response*.

You can label models. Model labels are used in the printed output to identify the results for different models. If you do not specify a label, the model is referred to by numerical order wherever necessary. You can label the models in two ways:

First, you can prefix the MODEL statement by a label followed by a colon. For example:

label: **MODEL** . . . ;

Second, you can add a quoted string after the MODEL keyword. For example:

MODEL "*label*" . . . ;

Quoted-string labels are preferable because they allow spaces and special characters and because these labels are case-sensitive. If you specify both types of label, PROC PANEL uses the quoted string.

The MODEL statement supports a multitude of options, some more specific than others. Table 27.2 summarizes the *options* available in the MODEL statement. These are subsequently discussed in detail in the order in which they are presented in the table.

Table 27.2 Summary of MODEL Statement Options

Option	Description
Estimation Technique Options	
AMACURDY	fits a one-way model by using the Amemiya-MaCurdy estimator
BTWNG	fits the between-groups model
BTWNT	fits the between-time-periods model
DASILVA	fits a moving average model by using the Da Silva method
FDONE	fits a one-way model by using first differences
FDONETIME	fits a one-way model for time effects by using first differences
FDTWO	fits a two-way model by using first differences

Table 27.2 *continued*

Option	Description
FIXONE	fits a one-way fixed-effects model
FIXONETIME	fits a one-way fixed-effects model for time effects
FIXTWO	fits a two-way fixed effects model
GMM1	fits a dynamic-panel model by using the one-step generalized method of moments (GMM)
GMM2	fits a dynamic-panel model by using two-step GMM
HTAYLOR	fits a one-way model by using the Hausman-Taylor estimator
ITGMM	fits a dynamic-panel model by using iterated GMM
PARKS	fits an autoregressive model by using the Parks method
POOLED	fits the pooled regression model
RANONE	fits a one-way random-effects model
RANTWO	fits a two-way random-effects model
Estimation Control Options	
M=	specifies the moving average order
NOESTIM	limits estimation to only transforming the data
NOINT	suppresses the intercept
SINGULAR=	specifies a matrix inverse singularity criterion
VCOMP=	specifies the type of variance component estimation for random-effects estimation
Dynamic Panel Estimation Control Options	
ARTEST=	specifies the maximum order of the auto regression (AR) test
ATOL=	specifies the convergence criterion of iterated GMM, with respect to the weighting matrix
BANDOPT=	specifies which neighboring observations to use as instruments, whether TRAILING, CENTERED, or LEADING
BIASCORRECTED	requests bias-corrected variances for two-step GMM
BTOL=	specifies the convergence criterion of iterated GMM, with respect to the parameter matrix
GINV=	specifies the type of generalized matrix inverse
MAXBAND=	specifies the moment condition bandwidth
MAXITER=	specifies the maximum iterations for iterative GMM
NODIFFS	estimates without moment conditions from the difference equations
NOLEVELS	estimates without moment conditions from the level equations
ROBUST	specifies the robust covariance matrix
TIME	includes time dummy variables in the model

Table 27.2 *continued*

Option	Description
Alternative Variances Options	
CLUSTER	corrects covariance for intracluster correlation
HAC(*options*)	specifies a heteroscedasticity- and autocorrelation-consistent (HAC) covariance
HCCME=	specifies a heteroscedasticity-corrected covariance matrix estimator (HCCME)
NEWEYWEST(*options*)	specifies the Newey-West covariance, a special case of the HAC covariance
Unit Root Test Options	
UROOTTEST(*test-options*)	requests one or more panel data unit root and stationarity tests; specify *test-options* **ALL** through **LLC** within this option
STATIONARITY(*test-options*)	synonym for UROOTTEST
ALL	requests that all unit root tests be performed
BREITUNG(*options*)	specifies Breitung's tests that are robust to cross-sectional dependence
COMBINATION(*options*)	specifies one or more unit root tests that combine over all cross sections
FISHER(*options*)	synonym for COMBINATION
HADRI(*options*)	specifies Hadri's (2000) stationarity test
HT	specifies the Harris and Tzavalis (1999) panel unit root test
IPS(*options*)	specifies the Im, Pesaran, and Shin (2003) panel unit root test
LLC(*options*)	specifies the Levin, Lin, and Chu (2002) panel unit root test
Model Specification Test Options	
BFN	requests the R_ρ statistic for serial correlation under fixed effects
BL91	requests the Baltagi and Li (1991) Lagrange multiplier (LM) test for serial correlation and random effects
BL95	requests the Baltagi and Li (1995) LM test for first-order correlation under fixed effects
BP	requests the Breusch-Pagan one-way test for random effects
BP2	requests the Breusch-Pagan two-way test for random effects
BSY	requests the Bera, Sosa Escudero, and Yoon modified Rao's score test
BW	requests the Berenblut-Webb statistic for serial correlation under fixed effects
CDTEST(*options*)	requests a battery of cross-sectional dependence tests.
DW	requests the Durbin-Watson statistic for serial correlation under fixed effects

Table 27.2 *continued*

Option	Description
GHM	requests the Gourieroux, Holly, and Monfort test for two-way random effects
HONDA	requests the Honda one-way test for random effects
HONDA2	requests the Honda two-way test for random effects
KW	requests the King and Wu two-way test for random effects
POOLTEST	requests poolability tests for one-way fixed effects and pooled models
WOOLDRIDGE02	requests the Wooldridge (2002) test for unobserved effects.
Printed Output Options	
CORR	prints the parameter correlation matrix
CORRB	synonym for CORR
COVB	prints the parameter covariance matrix
ITPRINT	prints the iteration history
NOPRINT	suppress normally printed output
PHI	prints the Φ covariance matrix for the Parks method
PRINTFIXED	estimates and prints the fixed effects
RHO	prints the autocorrelation coefficients for the Parks method
VAR	synonym for COVB

You can specify the following *options* in the MODEL statement after a slash (/).

Estimation Technique Options

These options specify the assumed error structure and estimation method. You can specify more than one option, in which case the analysis is repeated for each. The default is RANTWO (two-way random effects).

All estimation methods are detailed in the section "Details: PANEL Procedure" and its subsections.

AMACURDY
requests Amemiya-MaCurdy estimation for a model that has correlated individual (cross-sectional) effects. This option requires that you specify the CORRELATED= option in the INSTRUMENTS statement.

BTWNG
estimates a between-groups model.

BTWNT
estimates a between-time-periods model.

DASILVA
estimates the model by using the Da Silva method, which assumes a mixed variance-component moving average model for the error structure.

FDONE
estimates a one-way model by using first-differenced methods.

FDONETIME
estimates a one-way model that corresponds to time effects by using first-differenced methods.

FDTWO
estimates a two-way model by using first-differenced methods.

FIXONE
estimates a one-way fixed-effects model that corresponds to cross-sectional effects only.

FIXONETIME
estimates a one-way fixed-effects model that corresponds to time effects only.

FIXTWO
estimates a two-way fixed-effects model.

GMM1
estimates the model in a single step by using the dynamic panel estimator method, which allows for autoregressive processes. This option requires you to specify the INSTRUMENTS statement.

GMM2
estimates the model in two steps by using the dynamic panel estimator method. The first step forms an estimator for the weighting matrix that is used in the second step. This option requires you to specify the INSTRUMENTS statement.

HTAYLOR
requests Hausman-Taylor estimation for a model that has correlated individual (cross-sectional) effects. This option requires you to specify the CORRELATED= option in the INSTRUMENTS statement.

ITGMM
estimates the model by using the dynamic panel estimator method, but requests that PROC PANEL keep updating the weighting matrix until either the parameter vector converges or the weighting matrix converges. This option requires you to specify the INSTRUMENTS statement.

PARKS
estimates the model by using the Parks method, which assumes a first-order autoregressive model for the error structure.

POOLED
estimates a pooled (OLS) model.

RANONE
estimates a one-way random-effects model.

RANTWO
estimates a two-way random-effects model.

Estimation Control Options

These options define parameters that control the estimation and can be specific to the chosen technique (for example, how to estimate variance components in a random-effects model).

M=*number*
specifies the order of the moving average process in the Da Silva method. The value of *number* must be less than T-1, where T is the number of time periods. By default, M=1.

NOESTIM
limits the estimation of a FIXONE, FIXONETIME, or RANONE model to the generation of the transformed series. This option is intended for use with an OUTTRANS= data set.

NOINT
suppresses the intercept parameter from the model.

SINGULAR=*number*
specifies a singularity criterion for the inversion of the matrix. The default depends on the precision of the computer system.

VCOMP=FB | NL | WH | WK
specifies the type of variance component estimate to use. You can specify the following values:

FB	uses the Fuller and Battese method.
NL	uses the Nerlove method.
WH	uses the Wallace and Hussain method.
WK	uses the Wansbeek and Kapteyn method.

By default, VCOMP=FB for balanced data and VCOMP=WK for unbalanced data. For more information, see the sections "One-Way Random-Effects Model" on page 1841 and "Two-Way Random-Effects Model" on page 1844.

Dynamic Panel Estimation Control Options

These control options are specific to dynamic panels, where the estimation technique is specified as GMM1, GMM2, or ITGMM. For more information, see the section "Dynamic Panel Estimators" on page 1855.

ARTEST=*integer*
specifies the maximum order of the test for the presence of auto regression (AR) effects in the residual in the dynamic panel model. The value of *integer* must be between 1 and the T-3 inclusive, where T is the number of time periods.

ATOL=*number*
specifies the convergence criterion for the iterated generalized method of moments (GMM) when convergence of the method is determined by convergence in the weighting matrix. The convergence criterion (*number*) must be positive. If you do not specify this option, then the BTOL= option (or its default) is used.

BANDOPT=CENTERED | LEADING | TRAILING
specifies which observations are included in the instrument list when the MAXBAND= option is specified. You can specify the following values:

CENTERED uses both leading and trailing observations.

LEADING uses only leading observations.

TRAILING uses only trailing observations.

This option should be used only for exogenous instruments. By default, BANDOPT=TRAILING.

BIASCORRECTED
requests that the bias-corrected covariance matrix of the two-step dynamic panel estimator be computed. When you specify this option, the ROBUST option is disabled for the two-step GMM estimator.

BTOL=*number*
specifies the convergence criterion for iterated GMM when convergence of the method is determined by convergence in the parameter matrix. The convergence criterion (*number*) must be positive. The default is BTOL=1E–8.

GINV= G2 | G4
specifies what type of generalized inverse to use. You can specify the following values:

G2 uses the G2 generalized inverse.

G4 uses the G4 generalized inverse.

The G4 inverse is generally more stable, but numerically intensive. By default, GINV=G2.

MAXBAND=*integer*
specifies the maximum number of time periods (per instrumental variable) that are allowed into the moment condition. The acceptable range for *integer* is 1 to $T - 1$, where T is the number of time periods. If BANDOPT=LEADING or CENTERED, then the default value of MAXBAND is 2. If BANDOPT=TRAILING, then the default value of MAXBAND is 1. If no BANDOPT option is specified (such as when no exogenous instruments are used), then the default value of MAXBAND is 1.

MAXITER=*integer*
specifies the maximum number of iterations for the ITGMM option. By default, MAXITER=200.

NODIFFS
estimates the dynamic panel model without moment conditions from the difference equations.

NOLEVELS
estimates the dynamic panel model without moment conditions from the level equations.

ROBUST
uses the robust weighting matrix in the calculation of the covariance matrix of the single-step, two-step, and iterated GMM dynamic panel estimators.

TIME
estimates the model by using the dynamic panel estimator method but includes time dummy variables to model any time effects in the data.

Alternative Variances Options

These options specify variance estimation other than conventional model-based variance estimation. They include the robust, cluster robust, HAC, HCCME, and Newey-West techniques.

CLUSTER
specifies the cluster correction for the covariance matrix. You can specify this option when you specify HCCME=0, 1, 2, or 3.

HAC <(*options*) >
specifies the heteroscedasticity- and autocorrelation-consistent (HAC) covariance matrix estimator. This option is not available for between models and cannot be combined with the HCCME option.

For more information, see the section "Heteroscedasticity- and Autocorrelation-Consistent Covariance Matrices" on page 1870.

You can specify the following *options* within parentheses and separated by spaces:

BANDWIDTH=*number* | *method*
specifies the fixed bandwidth value or bandwidth selection method to be used in the kernel function. You can specify either a fixed value (*number*) or one of the *methods* shown after *number*.

number
specifies a fixed value of the bandwidth parameter.

ANDREWS91 | ANDREWS
specifies the Andrews (1991) bandwidth selection method.

NEWEYWEST94<(C=*number***)>**
NW94 <(C=*number***)>**
specifies the Newey and West (1994) bandwidth selection method. You can also specify C=*number* for the calculation of lag selection parameter; the default is C=12.

SAMPLESIZE<(*options***)>**
SS<(*options***)>**
calculates the bandwidth according to the following equation based on the sample size

$$b = \gamma T^r + c$$

where b is the bandwidth parameter; T is the sample size; and γ, r, and c are values specified by the following *options* within parentheses and separated by commas.

GAMMA=*number*
specifies the coefficient γ in the equation. By default, GAMMA=0.75.

RATE=*number*
specifies the growth rate r in the equation. By default, RATE=0.3333.

CONSTANT=*number*
specifies the constant c in the equation. By default, CONSTANT=0.5.

INT
specifies that the bandwidth parameter must be integer; that is, $b = \lfloor \gamma T^r + c \rfloor$, where $\lfloor x \rfloor$ denotes the largest integer less than or equal to x.

By default, BANDWIDTH=ANDREWS91.

KERNEL=BARTLETT | PARZEN | QS | TH | TRUNCATED
specifies the type of kernel function. You can specify the following values:

BARTLETT	specifies the Bartlett kernel function.
PARZEN	specifies the Parzen kernel function.
QS	specifies the quadratic spectral kernel function.
TH	specifies the Tukey-Hanning kernel function.
TRUNCATED	specifies the truncated kernel function.

By default, KERNEL=TRUNCATED.

KERNELLB=*number*
specifies the lower bound of the kernel weight value. Any kernel weight less than *number* is regarded as 0, which accelerates the calculation in large samples, especially for the quadratic spectral kernel function. By default, KERNELLB=0.

PREWHITENING
requires prewhitening in the covariance calculation.

ADJUSTDF
requires adjustment of degrees of freedom in the covariance calculation.

HCCME= NO | *number*
specifies the type of HCCME covariance matrix. You can specify one of the following:

NO	does not correct the covariance matrix.
number	specifies the type of covariance adjustment. The value of *number* can be any integer from 0 to 4, inclusive.

For more information, see the section "Heteroscedasticity-Corrected Covariance Matrices" on page 1866. By default, HCCME=NO.

NEWEYWEST<(*options***)>**
specifies the well-known Newey-West estimator, a special HAC estimator that uses (1) the Bartlett kernel, (2) a bandwidth that is determined by the equation based on the sample size, $b = \lfloor \gamma T^r + c \rfloor$, and (3) no adjustment for degrees of freedom and no prewhitening. By default, the bandwidth parameter for Newey-West estimator is $\lfloor 0.75T^{0.3333} + 0.5 \rfloor$, as shown in equation (15.17) in Stock and Watson (2002). You can specify the following *options* in parentheses and separated by commas:

GAMMA= *number*
specifies the coefficient γ in the equation. By default, GAMMA=0.75.

RATE= *number*
specifies the growth rate r in the equation. By default, RATE=0.3333.

CONSTANT= *number*
specifies the constant c in the equation. By default, CONSTANT=0.5.

To specify a Newey-West bandwidth directly (and not as a function of time-series length), set GAMMA=0 and CONSTANT=b, where b is the bandwidth you want. For example, the two variance specifications in the following statements are equivalent:

```
proc panel data=A;
   id i t;
   model y = x1 x2 x3 / ranone hac(kernel = bartlett bandwidth = 3);
   model y = x1 x2 x3 / ranone neweywest(gamma = 0, constant = 3);
run;
```

Unit Root Test Options

These options request unit root tests on the dependent variable. You begin with the UROOTTEST (or its synonym STATIONARITY) option and specify everything else within parentheses after the UROOTTEST (or SINGULARITY) keyword. The BREITUNG, COMBINATION, HADRI, HT, IPS, and LLC tests are available, and you can request them all by specifying the ALL option.

UROOTTEST(*test1***<(***test-options***),** *test2***<(***test-options***)>... >** **<** *options* **>)**
STATIONARITY(*test1***<(***test-options***),** *test2***<(***test-options***)>... >** **<** *options* **>)**
specifies tests of stationarity or unit root for panel data, and specifies options for each test. These tests apply only to the dependent variable. Six tests are available: BREITUNG, COMBINATION (or FISHER), HADRI, HT, IPS, and LLC. You can specify one or more of these tests, separated by commas. You can also request all tests by specifying UROOTEST(ALL) or STATIONARITY(ALL). If you specify one or more *test-options* (separated by spaces) inside the parentheses after a particular test, they apply only to that test. If you specify one or more *options* separated by spaces after you specify the tests, they apply to all the tests. If you specify both *test-options* and *options*, the *test-options* override the *options*.

You can specify the following *tests* and *test-options*:

BREITUNG<(*test-options***) >**
performs Breitung's unbiased test, t test, and generalized least squares (GLS) t test that are robust to cross-sectional dependence. The tests are described in Breitung and Meyer (1994); Breitung (2000); Breitung and Das (2005). You can specify one or more of the following *test-options* within parentheses and separated by spaces:

DETAIL
requests that intermediate results (lag order) be printed.

LAG=*type* | *value*
specifies the method to choose the lag order for the augmented Dickey-Fuller (ADF) regressions. You can specify a *value* or one of the *types* that are shown after *value*.

value
specifies the lag order. If the lag order is too big to run linear regression (*value* > $T-k$, where T is the number of time periods and k is the number of parameters), then the lag order is set to $\left\lfloor 12(T/100)^{1/4} \right\rfloor$ or $T-k-1$, whichever is smaller.

GS
selects the order of lags by Hall's (1994) sequential testing method, from the most general model (maximum lags) to lower order of lag terms.

SG
selects the order of lags by Hall's (1994) sequential testing method, from no lag term to maximum allowed lags.

AIC
selects the order of lags by Akaike's information criterion (AIC).

SBC
SIC
SBIC
selects the order of lags by the Bayesian information criterion (Schwarz criterion).

HQIC
selects the order of lags by the Hannan-Quinn information criterion.

MAIC
selects the order of lags by the modified AIC that is proposed by Ng and Perron (2001).

By default, LAG=MAIC.

MAXLAG=*value*
specifies the maximum lag order that the model allows. The default value is $\left\lfloor 12(T/100)^{1/4} \right\rfloor$. If *value* is larger than 0 and larger than $T-k$, then the maximum lag order is set to be the default value of $\left\lfloor 12(T/100)^{1/4} \right\rfloor$ or $T-k-1$, whichever is smaller. This option is ignored if you specify LAG=*value*.

COMBINATION < (*test-options***) >**
FISHER < (*test-options***) >**
specifies combination tests that are proposed by Choi (2001); Maddala and Wu (1999). Fisher's test, as proposed by Maddala and Wu (1999), is a special case of combination tests. You can specify one or more of the following *test-options* within parentheses and separated by spaces:

TEST=ADF | PP
selects the time series unit root test for combination tests (Fisher's test). You can specify the following values:

ADF specifies the augmented Dickey-Fuller (ADF) test. The BANDWIDTH and KERNEL options are ignored because they do not pertain to ADF tests.

PP specifies the Phillips and Perron (1988) unit root test. The LAG and MAXLAG options are ignored because they do not pertain to PP tests.

By default, TEST=PP.

KERNEL=BARTLETT | PARZEN | QS | TH | TRUNCATED
specifies the type of kernel function. You can specify the following values:

BARTLETT specifies the Bartlett kernel function.

PARZEN specifies the Parzen kernel function.

QS specifies the quadratic spectral kernel function.

TH specifies the Tukey-Hanning kernel function.

TRUNCATED specifies the truncated kernel function.

By default, KERNEL=QS.

BANDWIDTH=ANDREWS | *number*
specifies the bandwidth for the kernel. You can specify one of the following:

ANDREWS selects the bandwidth by the Andrews method.

number sets the bandwidth to *number*, which must be nonnegative.

By default, BANDWIDTH=ANDREWS.

DETAIL
requests that intermediate results (lag order and long-run variance for each cross section) be printed.

LAG=*type* | *value*
specifies the method to choose the lag order for the augmented Dickey-Fuller (ADF) regressions. You can specify a *value* or one of the *types* that are shown after *value*.

value
specifies the lag order. If the lag order is too big to run linear regression (*value* $> T - k$, where T is the number of time periods and k is the number of parameters), then the lag order is set to $\left\lfloor 12(T/100)^{1/4} \right\rfloor$ or $T - k - 1$, whichever is smaller.

GS
selects the order of lags by Hall's (1994) sequential testing method, from the most general model (maximum lags) to lower order of lag terms.

SG
selects the order of lags by Hall's (1994) sequential testing method, from no lag term to maximum allowed lags.

AIC
selects the order of lags by Akaike's information criterion (AIC).

SBC
SIC
SBIC
selects the order of lags by the Bayesian information criterion (Schwarz criterion).

HQIC
selects the order of lags by the Hannan-Quinn information criterion.

MAIC
selects the order of lags by the modified AIC that is proposed by Ng and Perron (2001).

By default, LAG=MAIC.

MAXLAG=*value*
specifies the maximum lag order that the model allows. The default value is $\left\lfloor 12(T/100)^{1/4} \right\rfloor$. If *value* is larger than 0 and larger than $T-k$, then the maximum lag order is set to be the default value of $\left\lfloor 12(T/100)^{1/4} \right\rfloor$ or $T-k-1$, whichever is smaller. This option is ignored if you specify LAG=*value*.

HADRI < (*test-options***) >**
specifies Hadri's (2000) panel stationarity test. You can specify the following *test-options*:

DETAIL
requests that intermediate results (lag order and long-run variance for each cross section) be printed.

KERNEL=BARTLETT | PARZEN | QS | TH | TRUNCATED
specifies the type of kernel function. You can specify the following values:

BARTLETT	specifies the Bartlett kernel function.
PARZEN	specifies the Parzen kernel function.
QS	specifies the quadratic spectral kernel function.
TH	specifies the Tukey-Hanning kernel function.
TRUNCATED	specifies the truncated kernel function.

By default, KERNEL=QS.

BANDWIDTH=ANDREWS | *number*
specifies the bandwidth for the kernel. You can specify one of the following:

ANDREWS	selects the bandwidth by the Andrews method.
number	sets the bandwidth to *number*, which must be nonnegative.

By default, BANDWIDTH=ANDREWS.

HT

specifies the Harris and Tzavalis (1999) panel unit root test. No options are available for this test.

IPS < (*test-options***) >**

specifies the Im, Pesaran, and Shin (2003) panel unit root test. You can specify one or more of the following *test-options* within parentheses and separated by spaces:

DETAIL

requests that intermediate results (lag order) be printed.

LAG=*type* | *value*

specifies the method to choose the lag order for the augmented Dickey-Fuller (ADF) regressions. You can specify a *value* or one of the *types* that are shown after *value*.

value

specifies the lag order. If the lag order is too big to run linear regression (*value* > $T - k$, where T is the number of time periods and k is the number of parameters), then the lag order is set to $\left\lfloor 12(T/100)^{1/4} \right\rfloor$ or $T - k - 1$, whichever is smaller.

GS

selects the order of lags by Hall's (1994) sequential testing method, from the most general model (maximum lags) to lower order of lag terms.

SG

selects the order of lags by Hall's (1994) sequential testing method, from no lag term to maximum allowed lags.

AIC

selects the order of lags by Akaike's information criterion (AIC).

SBC

SIC

SBIC

selects the order of lags by the Bayesian information criterion (Schwarz criterion).

HQIC

selects the order of lags by the Hannan-Quinn information criterion.

MAIC

selects the order of lags by the modified AIC that is proposed by Ng and Perron (2001).

By default, LAG=MAIC.

MAXLAG=*value*

specifies the maximum lag order that the model allows. The default value is $\left\lfloor 12(T/100)^{1/4} \right\rfloor$. If *value* is larger than 0 and larger than $T - k$, then the maximum lag order is set to be the default value of $\left\lfloor 12(T/100)^{1/4} \right\rfloor$ or $T - k - 1$, whichever is smaller. This option is ignored if you specify LAG=*value*.

LLC < (*test-options*) >
specifies the Levin, Lin, and Chu (2002) panel unit root test. You can specify one or more of the following *test-options* within parentheses and separated by spaces:

DETAIL
requests that intermediate results (lag order and long-run variance for each cross section) be printed.

KERNEL=BARTLETT | PARZEN | QS | TH | TRUNCATED
specifies the type of kernel function. You can specify the following values:

BARTLETT specifies the Bartlett kernel function.

PARZEN specifies the Parzen kernel function.

QS specifies the quadratic spectral kernel function.

TH specifies the Tukey-Hanning kernel function.

TRUNCATED specifies the truncated kernel function.

By default, KERNEL=QS.

BANDWIDTH=ANDREWS | *number*
specifies the bandwidth for the kernel. You can specify one of the following:

ANDREWS selects the bandwidth by the Andrews method.

number sets the bandwidth to *number*, which must be nonnegative. By default, BANDWIDTH=ANDREWS.

LAG=*type* | *value*
specifies the method to choose the lag order for the augmented Dickey-Fuller (ADF) regressions. You can specify a *value* or one of the *types* that are shown after *value*.

value
specifies the lag order. If the lag order is too big to run linear regression (*value* > $T - k$, where T is the number of time periods and k is the number of parameters), then the lag order is set to $\left\lfloor 12(T/100)^{1/4} \right\rfloor$ or $T - k - 1$, whichever is smaller.

GS
selects the order of lags by Hall's (1994) sequential testing method, from the most general model (maximum lags) to lower order of lag terms.

SG
selects the order of lags by Hall's (1994) sequential testing method, from no lag term to maximum allowed lags.

AIC
selects the order of lags by Akaike's information criterion (AIC).

SBC

SIC

SBIC

selects the order of lags by the Bayesian information criterion (Schwarz criterion).

HQIC

selects the order of lags by the Hannan-Quinn information criterion.

MAIC

selects the order of lags by the modified AIC that is proposed by Ng and Perron (2001).

By default, LAG=MAIC.

MAXLAG=*value*

specifies the maximum lag order that the model allows. The default value is $\left\lfloor 12(T/100)^{1/4} \right\rfloor$. If *value* is larger than 0 and larger than $T-k$, then the maximum lag order is set to be the default value of $\left\lfloor 12(T/100)^{1/4} \right\rfloor$ or $T-k-1$, whichever is smaller. This option is ignored if you specify LAG=*value*.

Consider the following example, which requests two tests (LLC and BREITUNG) on the dependent variable:

```
proc panel data=A;
   id i t;
   model y = x1 x2 x3 / unitroot(llc(kernel = parzen lag = aic),
                                 breitung(lag = gs)
                                 maxlag = 2
                                 kernel = bartlett);
run;
```

For the LLC test, the lag order is selected by AIC with maximum lag order 2 and the kernel is specified as Parzen (overriding Bartlett). For the BREITUNG test, the lag order is GS with a maximum lag order 2. The KERNEL option is ignored by BREITUNG because it is not relevant to that test.

Model Specification Test Options

These options request model specification tests, such as a test for poolability in one-way models. These tests depend on the model specifications of dependent and independent variables, but not on the estimation technique that is used to fit the model. For example, a one-way test for random effects does not require you to fit a random effects model, or even a one-way model for that matter. The model fits that are required for the selected tests are performed internally.

BFN (Experimental)

requests the R_ρ statistic for serial correlation under cross-sectional fixed effects.

BL91

requests the Baltagi and Li (1991) joint Lagrange multiplier (LM) test for serial correlation and random cross-sectional effects.

BL95
requests the Baltagi and Li (1995) LM test for first-order correlation under fixed effects.

BP
requests the Breusch-Pagan one-way test for random effects.

BP2
requests the Breusch-Pagan two-way test for random effects.

BSY
requests the Bera, Sosa Escudero, and Yoon modified Rao's score test for random cross-sectional effects or serial correlation or both.

BW (Experimental)
requests the Berenblut-Webb statistic for serial correlation under cross-sectional fixed effects.

CDTEST < (P=*value*) >
requests cross-sectional dependence tests. These include the Breusch and Pagan (1980) LM test, the scaled version of the Breusch and Pagan (1980) test, and the Pesaran (2004) CD test. When you specify P=*value*, the CD test for local cross-sectional dependence is performed with the order *value*, where *value* is an integer greater than 0.

DW (Experimental)
requests the Durbin-Watson statistic for serial correlation under cross-sectional fixed effects.

GHM (Experimental)
requests the Gourieroux, Holly, and Monfort two-way test for random effects.

HONDA
requests the Honda one-way test for random effects.

HONDA2
requests the Honda two-way test for random effects.

KW
requests the King and Wu two-way test for random effects.

POOLTEST
requests poolability tests for one-way fixed effects and pooled models.

WOOLDRIDGE02
requests the Wooldridge (2002) test for the presence of unobserved effects.

Printed Output Options

These options alter how results are presented.

CORRB

CORR
prints the matrix of estimated correlations between the parameter estimates.

COVB

VAR

prints the matrix of estimated covariances between the parameter estimates.

ITPRINT

prints out the iteration history of the parameter and transformed sum of squared errors.

NOPRINT

suppresses the normal printed output.

PHI

prints the Φ matrix of estimated covariances of the observations for the Parks method. The PHI option is relevant only when the PARKS option is specified. For more information, see the section "Parks Method (Autoregressive Model)" on page 1851.

PRINTFIXED

estimates and prints the fixed effects in models where they would normally be absorbed within the estimation.

RHO

prints the estimated autocorrelation coefficients for the Parks method.

OUTPUT Statement

OUTPUT < *options* > **;**

The OUTPUT statement creates an output SAS data set as specified by the following options:

OUT=*SAS-data-set*

names the output SAS data set to contain the predicted and transformed values. If the OUT= option is not specified, the new data set is named according to the DATA*n* convention.

PREDICTED=*name*

P=*name*

writes the predicted values to the output data set.

RESIDUAL=*name*

R=*name*

writes the residuals from the predicted values based on both the structural and time series parts of the model to the output data set.

RESTRICT Statement

RESTRICT < *"string"* > *equation* < *,equation2. . .* > **;**

The RESTRICT statement specifies linear equality restrictions on the parameters in the previous model statement. There can be as many unique restrictions as the number of parameters in the preceding model

statement. Multiple RESTRICT statements are understood as joint restrictions on a model's parameters. Restrictions on the intercept are obtained by the use of the keyword INTERCEPT.

Currently, only linear equality restrictions are permitted in PROC PANEL. Tests and restriction expressions can only be composed of algebraic operations that involve the addition symbol (+), subtraction symbol (–), and multiplication symbol (*).

The RESTRICT statement accepts labels that are produced in the printed output. A RESTRICT statement can be labeled in two ways. A RESTRICT statement can be preceded by a label followed by a colon. This is illustrated in **rest1** in the example below. Alternatively, the keyword RESTRICT can be followed by a quoted string.

The following statements illustrate the use of the RESTRICT statement:

```
proc panel;
   model y = x1 x2 x3;
   restrict x1 = 0, x2 * .5 + 2 * x3= 0;
   rest1: restrict  x2 = 0, x3 = 0;
   restrict "rest2" intercept=1;
run;
```

Note that a restrict statement cannot include a division sign in its formulation.

TEST Statement

TEST < *"string"* > *equation* < *,equation2. . .* >< */ options* > **;**

The TEST statement performs Wald, LaGrange multiplier and likelihood ratio tests of linear hypotheses about the regression parameters in the preceding MODEL statement. TEST and RESTRICT statements before the first MODEL statement are automatically associated with the first MODEL statement, in addition to any TEST and RESTRICT statements that follow it but precede subsequent MODEL statements. Each equation specifies a linear hypothesis to be tested. All hypotheses in one TEST statement are tested jointly. Variable names in the equations must correspond to regressors in the preceding MODEL statement, and each name represents the coefficient of the corresponding regressor. The keyword INTERCEPT refers to the coefficient of the intercept.

The following options can be specified on the TEST statement after the slash (/):

ALL
specifies Wald, LaGrange multiplier and likelihood ratio tests.

WALD
specifies the Wald test.

LM
specifies the LaGrange multiplier test.

LR
specifies the likelihood ratio test.

The Wald test is performed by default.

The following statements illustrate the use of the TEST statement:

```
proc panel;
   id csid tsid;
   model y = x1 x2 x3;
   test x1 = 0, x2 * .5 + 2 * x3 = 0;
   test_int: test intercept = 0, x3 = 0;
run;
```

The first test investigates the joint hypothesis that

$$\beta_1 = 0$$

and

$$.5\beta_2 + 2\beta_3 = 0$$

Currently, only linear equality restrictions and tests are permitted in PROC PANEL. Tests and restriction expressions can be composed only of algebraic operations that involve the addition symbol (+), subtraction symbol (–), and multiplication symbol (*).

The TEST statement accepts labels that are produced in the printed output. A TEST statement can be labeled in two ways. A TEST statement can be preceded by a label followed by a colon. Alternatively, the keyword TEST can be followed by a quoted string. If both are present, PROC PANEL uses the quoted string. If you do not supply a label, PROC PANEL automatically labels the test. If both a TEST and a RESTRICT statement are specified, the test is run with the restrictions applied.

For the DaSilva, Hausman-Taylor, and Amemiya-MaCurdy methods, only the Wald test is available.

Details: PANEL Procedure

Specifying the Input Data

The PANEL procedure is similar to other regression procedures in SAS. Suppose you want to regress the variable Y on regressors X1 and X2. Cross sections are identified by the variable STATE, and time periods are identified by the variable DATE. The input data set used by PROC PANEL must be sorted by cross section and by time within each cross section. Therefore, the first step in PROC PANEL is to make sure that the input data set is sorted. The following statements sort the data set A appropriately:

```
proc sort data=a;
   by state date;
run;
```

The next step is to invoke the PANEL procedure and specify the cross section and time series variables in an ID statement. The following statements show the correct syntax:

```
proc panel data=a;
   id state date;
   model y = x1 x2;
run;
```

Alternatively, PROC PANEL has the capability to read "flat" data. Say that you are using the data set A, which has observations on states. Specifically, the data are composed of observations on Y, X1, and X2. Unlike the previous case, the data is not recorded with a PROC PANEL structure. Instead, you have all of a state's information on a single row. You have variables to denote the name of the state (say state). The time observations for the Y variable are recorded horizontally. So the variable Y_1 is the first period's time observation, Y_10 is the tenth period's observation for some state. The same holds for the other variables. You have variables X1_1 to X1_10, X2_1 to X2_10, and X3_1 to X3_10 for others. With such data, PROC PANEL could be called by using the following syntax:

```
proc panel data=a;
   flatdata indid = state base = (Y X1 X2) tsname = t;
   id state t;
   model Y = X1 X2;
run;
```

See "FLATDATA Statement" on page 1806 and Example 27.7 for more information about the use of the FLATDATA statement.

Specifying the Regression Model

The MODEL statement in PROC PANEL is specified like the MODEL statement in other SAS regression procedures: the dependent variable is listed first, followed by an equal sign, followed by the list of regressor variables, as shown in the following statements:

```
proc panel data=a;
   id state date;
   model y = x1 x2;
run;
```

The major advantage of using PROC PANEL is that you can incorporate a model for the structure of the random errors. It is important to consider what kind of error structure model is appropriate for your data and to specify the corresponding option in the MODEL statement.

The error structure options supported by the PANEL procedure are FIXONE, FIXONETIME, FIXTWO, FDONE, FDONETIME, FDTWO, RANONE, RANTWO, PARKS, DASILVA, GMM1, GMM2, and ITGMM (iterated GMM). See the following sections for more information about these methods and the error structures they assume. The following statements fit a Fuller-Battese one-way random-effects model:

```
proc panel data=a;
   id state date;
   model y = x1 x2 / ranone vcomp=fb;
run;
```

You can specify more than one error structure option in the MODEL statement; the analysis is repeated using each specified method. You can use any number of MODEL statements to estimate different regression models or estimate the same model by using different options. See Example 27.1 for more information.

To aid in model specification within this class of models, PROC PANEL provides two specification test statistics. The first is an F statistic that tests the null hypothesis that the fixed-effects parameters are all 0. The second is a Hausman m statistic that provides information about the appropriateness of the random-effects specification. The m statistic is based on the idea that, under the null hypothesis of no correlation between the effects variables and the regressors, OLS and GLS are consistent. However, OLS is inefficient. Hence, a test can be based on the result that the covariance of an efficient estimator with its difference from an inefficient estimator is 0. Rejection of the null hypothesis might suggest that the fixed-effects model is more appropriate.

The PANEL procedure also provides the Buse R-square measure. This number is interpreted as a measure of the proportion of the transformed sum of squares of the dependent variable that is attributable to the influence of the independent variables. In the case of OLS estimation, the Buse R-square measure is equivalent to the usual R-square measure.

Unbalanced Data

For fixed-effects models, random-effects models, between estimators, and dynamic panel estimators, the PANEL procedure can process data with different numbers of time series observations across different cross sections. The Parks and Da Silva methods cannot be used with unbalanced data. The missing time series observations are recognized by the absence of time series ID variable values in some of the cross sections in the input data set. Moreover, if an observation with a particular time series ID value and cross-sectional ID value is present in the input data set, but one or more of the model variables are missing, that time series point is treated as missing for that cross section.

Missing Values

Any observation in the input data set with a missing value for one or more of the regressors is ignored by PROC PANEL and is not used in the model fit.

If there are observations in the input data set with missing dependent variable values but with nonmissing regressors, PROC PANEL can compute predicted values and store them in an output data set by using the OUTPUT statement. Note that the presence of such observations with missing dependent variable values does not affect the model fit because these observations are excluded from the calculation.

If either some regressors or the dependent variable values are missing, the model is estimated as unbalanced where the number of time series observations across different cross sections does not have to be equal. The Parks and Da Silva methods cannot be used with unbalanced data.

Computational Resources

The more parameters there are to be estimated, the more memory and time are required to estimate the model. Also affecting these resources are the estimation method chosen and the method to calculate variance

components. If the model has p parameters including the intercept, there are at least $p + p(p+1)/2$ numbers being held in the memory.

If the Arellano and Bond GMM approach is used, the amount of memory grows proportionately to the number of instruments in the INSTRUMENT statement. If the ITGMM (iterated GMM) option is selected, the computation time also depends on the convergence criteria selected and the maximum number of iterations allowed.

Restricted Estimates

A consequence of estimating a linear model with a restriction is that the error degrees of freedom increase by the number of restrictions. PROC PANEL produces the LaGrange multiplier associated with each restriction.

Say that you are interested in linear regression in which there are r restrictions. A linear restriction implies the following set of equations that relate the regression coefficients:

$$\begin{aligned} R_{1,1}\beta_1 + R_{1,2}\beta_2 + \cdots + R_{1,p}\beta_p &= q_1 \\ R_{2,1}\beta_1 + R_{2,2}\beta_2 + \cdots + R_{2,p}\beta_p &= q_2 \\ &\vdots \\ R_{r,1}\beta_1 + R_{r,2}\beta_2 + \cdots + R_{r,p}\beta_p &= q_r \end{aligned}$$

To economize on notation, you can represent the restriction structure in the following matrix notation $\mathbf{R}\boldsymbol{\beta} = \mathbf{q}$. Let $\hat{\boldsymbol{\beta}}$ be the unrestricted estimator of $\boldsymbol{\beta}$, and $\mathbf{X}$ be the corresponding set of regressors. The restricted $\boldsymbol{\beta}$ estimator is given by:

$$\boldsymbol{\beta}^* = \hat{\boldsymbol{\beta}} - (\mathbf{X}'\mathbf{X})^{-1}\mathbf{R}' \left[\mathbf{R}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{R}'\right]^{-1} (\mathbf{R}\hat{\boldsymbol{\beta}} - \mathbf{q})$$

The LaGrange multipliers are given as:

$$\boldsymbol{\lambda}_* = \left[\mathbf{R}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{R}'\right]^{-1} (\mathbf{R}\hat{\boldsymbol{\beta}} - \mathbf{q})$$

The standard errors of the LaGrange multipliers are calculated as:

$$\mathrm{Var}(\boldsymbol{\lambda}_*) = \left[\mathbf{R}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{R}'\right]^{-1} \mathbf{R}\mathrm{Var}(\hat{\boldsymbol{\beta}})\mathbf{R}' \left[\mathbf{R}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{R}'\right]^{-1}$$

A significant LaGrange multiplier implies that you can reject the null hypothesis that the restriction is not binding.

Note that in the special case of the fixed-effects models, the NOINT option and RESTRICT INTERCEPT=0 option give different estimates. This is not an error; it reflects two perspectives on the same issue. In the FIXONE case, the intercept is the last cross section's fixed effect (or the last time affecting the case of FIXONETIME). Specifying the NOINT option removes the intercept, but allows the last effect in. The NOINT command simply reclassifies the effects. The dummy variables become true cross section effects. If you specify the NOINT option with the FIXTWO option, the restriction is imposed that the last time effect is zero. A RESTRICT INTERCEPT=0 statement suppresses the estimation of the last effect in the FIXONE and FIXONETIME case. A RESTRICT INTERCEPT=0 has similar effects on the FIXTWO estimator. In general, restricting the intercept to zero is not recommended because OLS loses its unbiased nature.

Notation

The following notation represents the usual panel structure, with the specification of u_{it} dependent on the particular model:

$$y_{it} = \sum_{k=1}^{K} x_{itk}\beta_k + u_{it} \quad i = 1,\ldots N; t = 1,\ldots T_i$$

The total number of observations $M = \sum_{i=1}^{N} T_i$. For the balanced data case, $T_i = T$ for all i. The $M \times M$ covariance matrix of u_{it} is denoted by $\mathbf{V}$. Let $\mathbf{X}$ and $\mathbf{y}$ be the independent and dependent variables arranged by cross section and by time within each cross section. Let $\mathbf{X}_s$ be the X matrix without the intercept. All other notation is specific to each section.

One-Way Fixed-Effects Model

The specification for the one-way fixed-effects model is

$$u_{it} = \gamma_i + \epsilon_{it}$$

where the γ_is are nonrandom parameters to be estimated.

Let $\mathbf{Q_0} = \mathrm{diag}(\mathbf{E}_{T_i})$, with $\bar{\mathbf{J}}_{T_i} = \mathbf{J}_{T_i}/T_i$ and $\mathbf{E}_{T_i} = \mathbf{I}_{T_i} - \bar{\mathbf{J}}_{T_i}$, where $\mathbf{J}_{T_i}$ is a matrix of T_i ones.

The matrix $\mathbf{Q_0}$ represents the within transformation. In the one-way model, the within transformation is the conversion of the raw data to deviations from a cross section's mean. The vector $\tilde{\mathbf{x}}_{it}$ is a row of the general matrix $\mathbf{X}_s$, where the subscripted s implies the constant (column of ones) is missing.

Let $\tilde{\mathbf{X}}_s = \mathbf{Q_0}\mathbf{X}_s$ and $\tilde{\mathbf{y}} = \mathbf{Q_0}\mathbf{y}$. The estimator of the slope coefficients is given by

$$\tilde{\beta}_s = (\tilde{\mathbf{X}}_s'\tilde{\mathbf{X}}_s)^{-1}\tilde{\mathbf{X}}_s'\tilde{\mathbf{y}}$$

Once the slope estimates are in hand, the estimation of an intercept or the cross-sectional fixed effects is handled as follows. First, you obtain the cross-sectional effects:

$$\gamma_i = \bar{y}_{i\cdot} - \tilde{\beta}_s\bar{x}_{i\cdot} \quad \text{for} \quad i = 1\ldots N$$

If the NOINT option is specified, then the dummy variables' coefficients are set equal to the fixed effects. If an intercept is desired, then the *i*th dummy variable is obtained from the following expression:

$$D_i = \gamma_i - \gamma_N \quad \text{for} \quad i = 1\ldots N-1$$

The intercept is the *N*th fixed effect γ_N.

The within model sum of squared errors is:

$$\mathrm{SSE} = \sum_{i=1}^{N}\sum_{t=1}^{T_i}(y_{it} - \gamma_i - \mathbf{X}_{sit}\tilde{\beta}_s)^2$$

The estimated error variance can be written:

$$\hat{\sigma}_\epsilon^2 = \mathrm{SSE}/(M - N - (K-1))$$

Alternatively, an equivalent way to express the error variance is

$$\hat{\sigma}_{\epsilon}^2 = \tilde{\mathbf{u}}^{'}\mathbf{Q_0}\tilde{\mathbf{u}}/(M - N - (K-1))$$

where the residuals $\tilde{\mathbf{u}}$ are given by $\tilde{\mathbf{u}} = (\mathbf{I_M} - \mathbf{j_M}\mathbf{j}^{'}_{M}/M)(\mathbf{y} - \mathbf{X}_s\tilde{\beta}_s)$ if there is an intercept and by $\tilde{\mathbf{u}} = (\mathbf{y} - \mathbf{X}_s\tilde{\beta}_s)$ if there is not. The drawback is that the formula changes (but the results do not) with the inclusion of a constant.

The variance covariance matrix of $\tilde{\beta}_s$ is given by:

$$\mathrm{Var}\left[\tilde{\beta}_s\right] = \hat{\sigma}_{\epsilon}^2(\tilde{\mathbf{X}}_s^{'}\tilde{\mathbf{X}}_s)^{-1}$$

The covariance of the dummy variables and the dummy variables with the $\tilde{\beta}_s$ is dependent on whether the intercept is included in the model.

- *no intercept:*

$$\mathrm{Var}\left[\gamma_i\right] = \mathrm{Var}\left[D_i\right] = \frac{\hat{\sigma}_{\epsilon}^2}{T_i} + \bar{\mathbf{x}}_{i.}^{'}\mathrm{Var}\left[\tilde{\beta}_s\right]\bar{\mathbf{x}}_{i.}$$

$$\mathrm{Cov}\left[\gamma_i, \gamma_j\right] = \mathrm{Cov}\left[D_i D_j\right] = \bar{\mathbf{x}}_{i.}^{'}\mathrm{Var}\left[\tilde{\beta}_s\right]\bar{\mathbf{x}}_{j.}$$

$$\mathrm{Cov}\left[\gamma_i, \tilde{\beta}_s\right] = \mathrm{Cov}\left[D_i\tilde{\beta}_s\right] = -\bar{\mathbf{x}}_{i.}^{'}\mathrm{Var}\left[\tilde{\beta}_s\right]$$

- *intercept:*

$$\mathrm{Var}\left[D_i\right] = \frac{\hat{\sigma}_{\epsilon}^2}{T_i} + \frac{\hat{\sigma}_{\epsilon}^2}{T_N} + (\bar{\mathbf{x}}_{i.} - \bar{\mathbf{x}}_{N.})^{'}\mathrm{Var}\left[\tilde{\beta}_s\right](\bar{\mathbf{x}}_{i.} - \bar{\mathbf{x}}_{N.})$$

$$\mathrm{Cov}\left[D_i, D_j\right] = \frac{\hat{\sigma}_{\epsilon}^2}{T_N} + (\bar{\mathbf{x}}_{i.} - \bar{\mathbf{x}}_{N.})^{'}\mathrm{Var}\left[\tilde{\beta}_s\right](\bar{\mathbf{x}}_{j.} - \bar{\mathbf{x}}_{N.})$$

$$\mathrm{Var}\left[\mathrm{Intercept}\right] = \mathrm{Var}\left[\gamma_N\right] = \frac{\hat{\sigma}_{\epsilon}^2}{T_N} + \bar{\mathbf{x}}_{N.}^{'}\mathrm{Var}\left[\tilde{\beta}_s\right]\bar{\mathbf{x}}_{N.}$$

$$\mathrm{Cov}\left[D_i, \tilde{\beta}_s\right] = -(\bar{\mathbf{x}}_{i.} - \bar{\mathbf{x}}_{N.})^{'}\mathrm{Var}\left[\tilde{\beta}_s\right]$$

$$\mathrm{Cov}\left[\mathrm{Intercept}, D_i\right] = -\frac{\hat{\sigma}_{\epsilon}^2}{T_i} + \bar{\mathbf{x}}_{N.}^{'}\mathrm{Var}\left[\tilde{\beta}_s\right](\bar{\mathbf{x}}_{i.} - \bar{\mathbf{x}}_{N.})$$

$$\mathrm{Cov}\left[\mathrm{Intercept}\tilde{\beta}_s\right] = -\bar{\mathbf{x}}_{N.}^{'}\mathrm{Var}\left[\tilde{\beta}_s\right]$$

Alternatively, the model option FIXONETIME estimates a one-way model where the heterogeneity comes from time effects. This option is analogous to re-sorting the data by time and then by cross section and running a FIXONE model. The advantage of using the FIXONETIME option is that sorting is avoided and the model remains labeled correctly.

Two-Way Fixed-Effects Model

The specification for the two-way fixed-effects model is

$$u_{it} = \gamma_i + \alpha_t + \epsilon_{it}$$

where the γ_is and α_ts are nonrandom parameters to be estimated.

If you do not specify the NOINT option, which suppresses the intercept, the estimates for the fixed effects are reported under the restriction that $\gamma_N = 0$ and $\alpha_T = 0$. If you specify the NOINT option to suppress the intercept, only the restriction $\alpha_T = 0$ is imposed.

Balanced Panels

Assume that the data are balanced (for example, all cross sections have T observations). Then you can write the following:

$$\tilde{y}_{it} = y_{it} - \bar{y}_{i\cdot} - \bar{y}_{\cdot t} + \bar{\bar{y}}$$

$$\tilde{\mathbf{x}}_{it} = \mathbf{x}_{it} - \bar{\mathbf{x}}_{i\cdot} - \bar{\mathbf{x}}_{\cdot t} + \bar{\bar{\mathbf{x}}}$$

where the symbols:

y_{it} and $\mathbf{x}_{it}$ are the dependent variable (a scalar) and the explanatory variables (a vector whose columns are the explanatory variables not including a constant), respectively

$\bar{y}_{i\cdot}$ and $\bar{\mathbf{x}}_{i\cdot}$ are cross section means

$\bar{y}_{\cdot t}$ and $\bar{\mathbf{x}}_{\cdot t}$ are time means

$\bar{\bar{y}}$ and $\bar{\bar{\mathbf{x}}}$ are the overall means

The two-way fixed-effects model is simply a regression of $\tilde{y}_{it}$ on $\tilde{\mathbf{x}}_{it}$. Therefore, the two-way β is given by:

$$\tilde{\beta}_s = \left(\tilde{\mathbf{X}}'\tilde{\mathbf{X}}\right)^{-1}\tilde{\mathbf{X}}'\tilde{\mathbf{y}}$$

The calculations of cross section dummy variables, time dummy variables, and intercepts follow in a fashion similar to that used in the one-way model.

First, you obtain the net cross-sectional and time effects. Denote the cross-sectional effects by γ and the time effects by α. These effects are calculated from the following relations:

$$\hat{\gamma}_i = \left(\bar{y}_{i\cdot} - \bar{\bar{y}}\right) - \tilde{\beta}_s\left(\bar{x}_{i\cdot} - \bar{\bar{x}}\right)$$

$$\hat{\alpha}_t = \left(\bar{y}_{\cdot t} - \bar{\bar{y}}\right) - \tilde{\beta}_s\left(\bar{x}_{\cdot t} - \bar{\bar{x}}\right)$$

Denote the cross-sectional dummy variables and time dummy variables with the superscript C and T. Under the NOINT option the following equations give the dummy variables:

$$D_i^C = \hat{\gamma}_i + \hat{\alpha}_T$$

$$D_t^T = \hat{\alpha}_t - \hat{\alpha}_T$$

When an intercept is specified, the equations for dummy variables and intercept are:

$$D_i^C = \hat{\gamma}_i - \hat{\gamma}_N$$

$$D_t^T = \hat{\alpha}_t - \hat{\alpha}_T$$

$$\text{Intercept} = \hat{\gamma}_N + \hat{\alpha}_T$$

The sum of squared errors is:

$$\text{SSE} = \sum_{i=1}^{N} \sum_{t=1}^{T_i} (y_{it} - \gamma_i - \alpha_t - \mathbf{X}_s \tilde{\beta}_s)^2$$

The estimated error variance is:

$$\hat{\sigma}_\epsilon^2 = \text{SSE}/(M - N - T - (K - 1))$$

With or without a constant, the variance covariance matrix of $\tilde{\beta}_s$ is given by:

$$\text{Var}\left[\tilde{\beta}_s\right] = \hat{\sigma}_\epsilon^2 (\tilde{\mathbf{X}}_s' \tilde{\mathbf{X}}_s)^{-1}$$

Variance Covariance of Dummy Variables with No Intercept

The variances and covariances of the dummy variables are given with the NOINT specification as follows:

$$\begin{aligned} \text{Var}\left(D_i^C\right) &= \hat{\sigma}_\epsilon^2 \left(\frac{1}{T} + \frac{1}{N} - \frac{1}{NT}\right) \\ &+ \left(\bar{\mathbf{x}}_{i\cdot} + \bar{\mathbf{x}}_{\cdot T} - \bar{\bar{\mathbf{x}}}\right)' \text{Var}\left[\tilde{\beta}_s\right] \left(\bar{\mathbf{x}}_{i\cdot} + \bar{\mathbf{x}}_{\cdot T} - \bar{\bar{\mathbf{x}}}\right) \end{aligned}$$

$$\text{Var}\left(D_t^T\right) = \frac{2\hat{\sigma}_\epsilon^2}{N} + \left(\bar{\mathbf{x}}_{\cdot t} - \bar{\mathbf{x}}_{\cdot T}\right)' \text{Var}\left[\tilde{\beta}_s\right] \left(\bar{\mathbf{x}}_{\cdot t} - \bar{\mathbf{x}}_{\cdot T}\right)$$

$$\begin{aligned} \text{Cov}\left(D_i^C, D_j^C\right) &= \hat{\sigma}_\epsilon^2 \left(\frac{1}{N} - \frac{1}{NT}\right) \\ &+ \left(\bar{\mathbf{x}}_{i\cdot} + \bar{\mathbf{x}}_{\cdot t} - \bar{\bar{\mathbf{x}}}\right)' \text{Var}\left[\tilde{\beta}_s\right] \left(\bar{\mathbf{x}}_{j\cdot} + \bar{\mathbf{x}}_{\cdot t} - \bar{\bar{\mathbf{x}}}\right) \end{aligned}$$

$$\text{Cov}\left(D_t^T, D_u^T\right) = \frac{\hat{\sigma}_\epsilon^2}{N} + \left(\bar{\mathbf{x}}_{\cdot t} - \bar{\mathbf{x}}_{\cdot T}\right)' \text{Var}\left[\tilde{\beta}_s\right] \left(\bar{\mathbf{x}}_{\cdot u} - \bar{\mathbf{x}}_{\cdot T}\right)$$

$$\text{Cov}\left(D_i^C, D_t^T\right) = -\frac{\hat{\sigma}_\epsilon^2}{N} + \left(\bar{\mathbf{x}}_{i\cdot} + \bar{\mathbf{x}}_{\cdot t} - \bar{\bar{\mathbf{x}}}\right)' \text{Var}\left[\tilde{\beta}_s\right] \left(\bar{\mathbf{x}}_{\cdot t} - \bar{\mathbf{x}}_{\cdot T}\right)$$

$$\text{Cov}\left(D_i^C, \beta\right) = -\left(\bar{\mathbf{x}}_{i\cdot} + \bar{\mathbf{x}}_{\cdot t} - \bar{\bar{\mathbf{x}}}\right)' \text{Var}\left[\tilde{\beta}_s\right]$$

$$\text{Cov}\left(D_i^T, \beta\right) = -\left(\bar{\mathbf{x}}_{\cdot t} - \bar{\mathbf{x}}_{\cdot T}\right)' \text{Var}\left[\tilde{\beta}_s\right]$$

Variance Covariance of Dummy Variables with Intercept

The variances and covariances of the dummy variables are given when the intercept is included as follows:

$$\begin{aligned}
\text{Var}\left(D_i^C\right) &= \frac{2\hat{\sigma}_\epsilon^2}{T} + \left(\bar{\mathbf{x}}_{i\cdot} - \bar{\mathbf{x}}_{N\cdot}\right)' \text{Var}\left[\tilde{\beta}_s\right] \left(\bar{\mathbf{x}}_{i\cdot} - \bar{\mathbf{x}}_{N\cdot}\right) \\
\text{Var}\left(D_t^T\right) &= \frac{2\hat{\sigma}_\epsilon^2}{N} + \left(\bar{\mathbf{x}}_{\cdot t} - \bar{\mathbf{x}}_{\cdot T}\right)' \text{Var}\left[\tilde{\beta}_s\right] \left(\bar{\mathbf{x}}_{\cdot t} - \bar{\mathbf{x}}_{\cdot T}\right) \\
\text{Var}\left(\text{Intercept}\right) &= \hat{\sigma}_\epsilon^2 \left(\frac{1}{T} + \frac{1}{N} - \frac{1}{NT}\right) + \left(\bar{\mathbf{x}}_{N\cdot} + \bar{\mathbf{x}}_{\cdot T} - \bar{\bar{\mathbf{x}}}\right)' \text{Var}\left[\tilde{\beta}_s\right] \left(\bar{\mathbf{x}}_{N\cdot} + \bar{\mathbf{x}}_{\cdot T} - \bar{\bar{\mathbf{x}}}\right) \\
\text{Cov}\left(D_i^C, D_j^C\right) &= \frac{\hat{\sigma}_\epsilon^2}{T} + \left(\bar{\mathbf{x}}_{i\cdot} - \bar{\mathbf{x}}_{N\cdot}\right)' \text{Var}\left[\tilde{\beta}_s\right] \left(\bar{\mathbf{x}}_{j\cdot} - \bar{\mathbf{x}}_{N\cdot}\right) \\
\text{Cov}\left(D_t^T, D_u^T\right) &= \frac{\hat{\sigma}_\epsilon^2}{N} + \left(\bar{\mathbf{x}}_{\cdot t} - \bar{\mathbf{x}}_{\cdot T}\right)' \text{Var}\left[\tilde{\beta}_s\right] \left(\bar{\mathbf{x}}_{\cdot u} - \bar{\mathbf{x}}_{\cdot T}\right) \\
\text{Cov}\left(D_i^C, D_u^T\right) &= \left(\bar{\mathbf{x}}_{i\cdot} - \bar{\mathbf{x}}_{N\cdot}\right)' \text{Var}\left[\tilde{\beta}_s\right] \left(\bar{\mathbf{x}}_{\cdot u} - \bar{\mathbf{x}}_{\cdot T}\right) \\
\text{Cov}\left(D_i^C, \text{Intercept}\right) &= -\left(\frac{\hat{\sigma}_\epsilon^2}{T}\right) + \left(\bar{\mathbf{x}}_{i\cdot} - \bar{\mathbf{x}}_{N\cdot}\right)' \text{Var}\left(\tilde{\beta}_s\right) \left(\bar{\mathbf{x}}_{N\cdot} + \bar{\mathbf{x}}_{\cdot T} - \bar{\bar{\mathbf{x}}}\right) \\
\text{Cov}\left(D_t^T, \text{Intercept}\right) &= -\left(\frac{\hat{\sigma}_\epsilon^2}{N}\right) + \left(\bar{\mathbf{x}}_{\cdot t} - \bar{\mathbf{x}}_{\cdot T}\right)' \text{Var}\left[\tilde{\beta}_s\right] \left(\bar{\mathbf{x}}_{N\cdot} + \bar{\mathbf{x}}_{\cdot T} - \bar{\bar{\mathbf{x}}}\right) \\
\text{Cov}\left(D_i^C, \tilde{\beta}\right) &= -\left(\bar{\mathbf{x}}_{i\cdot} - \bar{\mathbf{x}}_{N\cdot}\right)' \text{Var}\left[\tilde{\beta}_s\right] \\
\text{Cov}\left(D_t^T, \tilde{\beta}\right) &= -\left(\bar{\mathbf{x}}_{\cdot t} - \bar{\mathbf{x}}_{\cdot T}\right)' \text{Var}\left[\tilde{\beta}_s\right] \\
\text{Cov}\left(\text{Intercept}, \tilde{\beta}\right) &= -\left(\bar{\mathbf{x}}_{N\cdot} + \bar{\mathbf{x}}_{\cdot T} - \bar{\bar{\mathbf{x}}}\right)' \text{Var}\left[\tilde{\beta}_s\right]
\end{aligned}$$

Unbalanced Panels

Let $\mathbf{X}_*$ and $\mathbf{y}_*$ be the independent and dependent variables arranged by time and by cross section within each time period. (Note that the input data set used by the PANEL procedure must be sorted by cross section and then by time within each cross section.) Let M_t be the number of cross sections observed in year t and let $\sum_t M_t = M$. Let $\mathbf{D}_t$ be the $M_t \times N$ matrix obtained from the $N \times N$ identity matrix from which rows that correspond to cross sections not observed at time t have been omitted. Consider

$$\mathbf{Z} = (\mathbf{Z}_1, \mathbf{Z}_2)$$

where $\mathbf{Z}_1 = (\mathbf{D}_1', \mathbf{D}_2', \ldots . \mathbf{D}_T')'$ and $\mathbf{Z}_2 = \text{diag}(\mathbf{D}_1\mathbf{j}_N, \mathbf{D}_2\mathbf{j}_N, \ldots . . \mathbf{D}_T\mathbf{j}_N)$. The matrix $\mathbf{Z}$ gives the dummy variable structure for the two-way model.

Let

$$\begin{aligned}
\mathbf{\Delta}_N &= \mathbf{Z}_1'\mathbf{Z}_1 \\
\mathbf{\Delta}_T &= \mathbf{Z}_2'\mathbf{Z}_2 \\
\mathbf{A} &= \mathbf{Z}_2'\mathbf{Z}_1 \\
\bar{\mathbf{Z}} &= \mathbf{Z}_2 - \mathbf{Z}_1\Delta_N^{-1}\mathbf{A}' \\
\mathbf{Q} &= \Delta_T - \mathbf{A}\Delta_N^{-1}\mathbf{A}' \\
\mathbf{P} &= (\mathbf{I}_M - \mathbf{Z}_1\Delta_N^{-1}\mathbf{Z}_1') - \bar{\mathbf{Z}}\mathbf{Q}^{-1}\bar{\mathbf{Z}}'
\end{aligned}$$

The estimate of the regression slope coefficients is given by

$$\tilde{\beta}_s = (\mathbf{X}_{*s}^{'}\mathbf{P}\mathbf{X}_{*s})^{-1}\mathbf{X}_{*s}^{'}\mathbf{P}\mathbf{y}_{*}$$

where $\mathbf{X}_{*s}$ is the $\mathbf{X}_{*}$ matrix without the vector of 1s.

The estimator of the error variance is

$$\hat{\sigma}_{\epsilon}^{2} = \tilde{\mathbf{u}}^{'}\mathbf{P}\tilde{\mathbf{u}}/(M - T - N + 1 - (K - 1))$$

where the residuals are given by $\tilde{\mathbf{u}} = (\mathbf{I}_{M} - \mathbf{j}_{M}\mathbf{j}_{M}^{'}/M)(\mathbf{y}_{*} - \mathbf{X}_{*s}\tilde{\beta}_s)$ if there is an intercept in the model and by $\tilde{\mathbf{u}} = \mathbf{y}_{*} - \mathbf{X}_{*s}\tilde{\beta}_s$ if there is no intercept.

The actual implementation is quite different from the theory. The PANEL procedure transforms all series using the $\mathbf{P}$ matrix.

$$\bar{\mathbf{v}} = \mathbf{P}\mathbf{v}$$

The variable being transformed is v, which could be $\mathbf{y}$ or any column of $\mathbf{X}$. After the data are properly transformed, OLS is run on the resulting series.

Given $\tilde{\beta}_s$, the next step is estimating the cross-sectional and time effects. Given that $\boldsymbol{\gamma}$ is the column vector of cross-sectional effects and $\boldsymbol{\alpha}$ is the column vector of time effects,

$$\tilde{\boldsymbol{\alpha}} = \mathbf{Q}^{-1}\bar{\mathbf{Z}}^{'}y - \mathbf{Q}^{-1}\bar{\mathbf{Z}}^{'}\mathbf{X}_{s}\tilde{\beta}_s$$

$$\tilde{\boldsymbol{\gamma}} = (\Theta_1 + \Theta_2 - \Theta_3)y - (\Theta_1 + \Theta_2 - \Theta_3)\mathbf{X}_{s}\tilde{\beta}_s$$

$$\Theta_1 = \Delta_{N}^{-1}\mathbf{Z}_{1}^{'}$$

$$\Theta_2 = \Delta_{N}^{-1}\mathbf{A}^{'}Q^{-1}\mathbf{Z}_{2}^{'}$$

$$\Theta_3 = \Delta_{N}^{-1}\mathbf{A}^{'}Q^{-1}\mathbf{A}\Delta_{N}^{-1}\mathbf{Z}_{1}^{'}$$

Given the cross-sectional and time effects, the next step is to derive the associated dummy variables. Using the NOINT option, the following equations give the dummy variables:

$$D_{i}^{C} = \hat{\gamma}_i + \hat{\alpha}_T$$

$$D_{t}^{T} = \hat{\alpha}_t - \hat{\alpha}_T$$

When an intercept is desired, the equations for dummy variables and intercept are:

$$D_{i}^{C} = \hat{\gamma}_i - \hat{\gamma}_N$$

$$D_{t}^{T} = \hat{\alpha}_t - \hat{\alpha}_T$$

$$\text{Intercept} = \hat{\gamma}_N + \hat{\alpha}_T$$

The calculation of the covariance matrix is as follows:

$$\begin{aligned}\text{Var}\left[\hat{\boldsymbol{\gamma}}\right] &= \hat{\sigma}_{\epsilon}^{2}\left(\Delta_{N}^{-1} - \Sigma_1 + \Sigma_2\right) \\ &+ (\Theta_1 + \Theta_2 - \Theta_3)\text{Var}\left[\tilde{\beta}_s\right](\Theta_1 + \Theta_2 - \Theta_3)^{'}\end{aligned}$$

where

$$\Sigma_1 = \Delta_N^{-1}\mathbf{A}'\mathbf{Q}^{-1}\mathbf{A}\Delta_N^{-1}\mathbf{A}'\mathbf{Q}^{-1}\mathbf{A}\Delta_N^{-1}$$
$$\Sigma_2 = \Delta_N^{-1}\mathbf{A}'\mathbf{Q}^{-1}\Delta_T\mathbf{Q}^{-1}\mathbf{A}\Delta_N$$
$$\text{Var}\left[\hat{\boldsymbol{\alpha}}\right] = \hat{\sigma}_\epsilon^2\left(\mathbf{Q}^{-1}\bar{\mathbf{Z}}'\bar{\mathbf{Z}}\mathbf{Q}^{-1}\right) + \left(\mathbf{Q}^{-1}\bar{\mathbf{Z}}'\mathbf{X}_s\right)\text{Var}\left[\tilde{\beta}_s\right]\left(\mathbf{X}_s'\bar{\mathbf{Z}}\mathbf{Q}^{-1}\right)$$

$$\begin{aligned}\text{Cov}\left[\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\gamma}}'\right] &= \hat{\sigma}_\epsilon^2\Delta_N^{-1}\left[\mathbf{A}'\mathbf{Q}^{-1}\Delta_T - \mathbf{A}'\mathbf{Q}^{-1}\mathbf{A}\Delta_N^{-1}\mathbf{A}'\right]\mathbf{Q}^{-1} \\ &+ (\Theta_1 + \Theta_2 - \Theta_3)\text{Var}\left[\tilde{\beta}_s\right]\left(\mathbf{X}_s'\bar{\mathbf{Z}}\mathbf{Q}^{-1}\right)\end{aligned}$$

$$\text{Cov}\left[\hat{\boldsymbol{\gamma}}, \tilde{\beta}\right] = (\Theta_1 + \Theta_2 - \Theta_3)\text{Var}\left[\tilde{\beta}_s\right]$$
$$\text{Cov}\left[\hat{\boldsymbol{\alpha}}, \tilde{\beta}\right] = \left(\mathbf{Q}^{-1}\bar{\mathbf{Z}}'\mathbf{X}_s\right)\text{Var}\left[\tilde{\beta}_s\right]$$

Now you work out the variance covariance estimates for the dummy variables.

Variance Covariance of Dummy Variables with No Intercept

The variances and covariances of the dummy variables are given under the NOINT selection as follows:

$$\text{Cov}\left(D_i^C, D_j^C\right) = \text{Cov}\left(\hat{\gamma}_i, \hat{\gamma}_j\right) + \text{Cov}\left(\hat{\gamma}_i, \hat{\alpha}_T\right) + \text{Cov}\left(\hat{\gamma}_j, \hat{\alpha}_T\right) + \text{Var}\left(\hat{\alpha}_T\right)$$
$$\text{Cov}\left(D_t^T, D_u^T\right) = \text{Cov}\left(\hat{\alpha}_t, \hat{\alpha}_u\right) - \text{Cov}\left(\hat{\alpha}_t, \hat{\alpha}_T\right) - \text{Cov}\left(\hat{\alpha}_u, \hat{\alpha}_T\right) + \text{Var}\left(\hat{\alpha}_T\right)$$
$$\text{Cov}\left(D_i^C, D_t^T\right) = \text{Cov}\left(\hat{\gamma}_i\,\hat{\alpha}_t\right) + \text{Cov}\left(\hat{\gamma}_i, \hat{\alpha}_T\right) - \text{Cov}\left(\hat{\gamma}_i, \hat{\alpha}_T\right) - \text{Var}\left(\hat{\alpha}_T\right)$$
$$\text{Cov}\left(D_i^C, \tilde{\beta}\right) = -\text{Cov}\left(\hat{\gamma}_i, \tilde{\beta}\right) - \text{Cov}\left(\hat{\alpha}_T, \tilde{\beta}\right)$$
$$\text{Cov}\left(D_t^T, \tilde{\beta}\right) = -\text{Cov}\left(\hat{\alpha}_t, \tilde{\beta}\right) + \text{Cov}\left(\hat{\alpha}_T, \tilde{\beta}\right)$$

Variance Covariance of Dummy Variables with Intercept

The variances and covariances of the dummy variables are given as follows when the intercept is included:

$$\text{Cov}\left(D_i^C, D_j^C\right) = \text{Cov}\left(\hat{\gamma}_i, \hat{\gamma}_j\right) - \text{Cov}\left(\hat{\gamma}_i, \hat{\gamma}_N\right) - \text{Cov}\left(\hat{\gamma}_j, \hat{\gamma}_N\right) + \text{Var}\left(\hat{\gamma}_N\right)$$
$$\text{Cov}\left(D_t^T, D_u^T\right) = \text{Cov}\left(\hat{\alpha}_t, \hat{\alpha}_u\right) - \text{Cov}\left(\hat{\alpha}_t, \hat{\alpha}_T\right) - \text{Cov}\left(\hat{\alpha}_u, \hat{\alpha}_T\right) + \text{Var}\left(\hat{\alpha}_T\right)$$
$$\text{Cov}\left(D_i^C, D_t^T\right) = \text{Cov}\left(\hat{\gamma}_i, \hat{\alpha}_t\right) - \text{Cov}\left(\hat{\gamma}_i, \hat{\alpha}_T\right) - \text{Cov}\left(\hat{\gamma}_N, \hat{\alpha}_t\right) + \text{Cov}\left(\hat{\gamma}_N, \hat{\alpha}_T\right)$$
$$\text{Cov}\left(D_i^C, \text{Intercept}\right) = \text{Cov}\left(\hat{\gamma}_i, \hat{\gamma}_N\right) + \text{Cov}\left(\hat{\gamma}_i, \hat{\alpha}_T\right) - \text{Cov}\left(\hat{\gamma}_j, \hat{\alpha}_T\right) - \text{Var}\left(\hat{\gamma}_N\right)$$
$$\text{Cov}\left(D_t^T, \text{Intercept}\right) = \text{Cov}\left(\hat{\alpha}_t, \hat{\alpha}_T\right) + \text{Cov}\left(\hat{\alpha}_t, \hat{\gamma}_N\right) - \text{Cov}\left(\hat{\alpha}_T, \hat{\alpha}_N\right) - \text{Var}\left(\hat{\alpha}_T\right)$$
$$\text{Cov}\left(D_i^C, \tilde{\beta}\right) = -\text{Cov}\left(\hat{\gamma}_i, \tilde{\beta}\right) - \text{Cov}\left(\hat{\gamma}_N, \tilde{\beta}\right)$$
$$\text{Cov}\left(D_t^T, \tilde{\beta}\right) = -\text{Cov}\left(\hat{\alpha}_t, \tilde{\beta}\right) + \text{Cov}\left(\hat{\alpha}_T, \tilde{\beta}\right)$$
$$\text{Cov}\left(\text{Intercept}, \tilde{\beta}_f\right) = -\text{Cov}\left(\hat{\alpha}_T, \tilde{\beta}\right) - \text{Cov}\left(\hat{\gamma}_N, \tilde{\beta}\right)$$

First-Differenced Methods for One-Way and Two-Way Models

The first-differenced (FD) estimator is an approach that is used to address the problem of omitted variables in econometrics and statistics by using panel data. The estimator is obtained by running a pooled OLS estimation for a regression of the differenced variables. The specification of the models, along with the estimation of the fixed effects, is the same as that described in the sections "One-Way Fixed-Effects Model" on page 1833 and "Two-Way Fixed-Effects Model" on page 1835. To eliminate the fixed effects, you use first-differenced methods to difference them out instead of using the within transformation. Because the intercept is differenced out, the intercept cannot be estimated by first-differenced methods.

Let i be the cross sections and t be the time periods. The regressors and dependent variables are denoted as $\mathbf{X}_{i,t}$ and $\mathbf{y}_{i,t}$, respectively. For the models that have only cross-sectional effects, the data are transformed by first-differencing within each cross section. Therefore, the transformed variables are $\tilde{\mathbf{X}}_{i,t} = \mathbf{X}_{i,t} - \mathbf{X}_{i,t-1}$ for regressors and $\tilde{\mathbf{y}}_{i,t} = \mathbf{y}_{i,t} - \mathbf{y}_{i,t-1}$ for the dependent variable.

For models that have only time effects, the transformation is $\tilde{\mathbf{X}}_{i,t} = \mathbf{X}_{i,t} - \mathbf{X}_{i-1,t}$ for regressors and $\tilde{\mathbf{y}}_{i,t} = \mathbf{y}_{i,t} - \mathbf{y}_{i-1,t}$ for the dependent variable.

For models that have both cross-sectional effects and time effects, the transformation is $\tilde{\mathbf{X}}_{s,t} = \mathbf{X}_{s,t} - \mathbf{X}_{i-1,t} - \mathbf{X}_{i,t-1} + \mathbf{X}_{i-1,t-1}$ for regressors and $\tilde{\mathbf{y}}_{i,t} = \mathbf{y}_{i,t} - \mathbf{y}_{i-1,t} - \mathbf{y}_{i,t-1} + \mathbf{y}_{i-1,t-1}$ for the dependent variable.

The first-differenced estimator is

$$\tilde{\beta}_{fd} = (\tilde{\mathbf{X}}'\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}'\tilde{\mathbf{y}}$$

The resulting residual can be denoted as $\tilde{u} = \tilde{\mathbf{y}} - \tilde{\beta}_{fd} * \tilde{\mathbf{X}}$. The degree of freedom is the same as in a one-way fixed-effects model or a two-way fixed-effects model when the within transformation is used.

To calculate the predicted value, you can use the previous time period or last individual's information or both. If the model has only cross-sectional effects, the predicted value is $\hat{y}_{it} = y_{i,t-1} + \tilde{u}_{i,t}$. If the model has only time effects, the predicted value is $\hat{y}_{it} = y_{i-1,t} + \tilde{u}_{i,t}$. If the model has both cross-sectional and time effects, the predicted value is $\hat{y}_{it} = y_{i,t-1} + y_{i-1,t} - y_{i-1,t-1} + \tilde{u}_{i,t}$.

Between Estimators

The between-groups estimator is the regression of the cross section means of $\mathbf{y}$ on the cross section means of $\tilde{\mathbf{X}}_s$. In other words, you fit the following regression:

$$\bar{y}_{i\cdot} = \bar{\mathbf{x}}_{i\cdot}\beta^{BG} + \eta_i$$

The between-time-periods estimator is the regression of the time means of $\mathbf{y}$ on the time means of $\tilde{\mathbf{X}}_s$. In other words, you fit the following regression:

$$\bar{y}_{\cdot t} = \bar{\mathbf{x}}_{\cdot t}\beta^{BT} + \zeta_t$$

In either case, the error is assumed to be normally distributed with mean zero and a constant variance.

Pooled Estimator

PROC PANEL allows you to pool time series cross-sectional data and run regressions on the data. Pooling is admissible if there are no fixed effects or random effects present in the data. This feature is included to aid in analysis and comparison across model types and to give you access to HCCME standard errors and other panel diagnostics. In general, this model type should not be used with time series cross-sectional data.

One-Way Random-Effects Model

The specification for the one-way random-effects model is

$$u_{it} = v_i + \epsilon_{it}$$

Let $\mathbf{j}_{T_i}$ (lowercase j) be a vector of ones of dimension T_i, and $\mathbf{J}_{T_i}$ (uppercase J) be a square matrix of ones of dimension T_i. Define $\mathbf{Z_0} = \mathrm{diag}(\mathbf{j}_{T_i})$, $\mathbf{P_0} = \mathrm{diag}(\bar{\mathbf{J}}_{T_i})$, and $\mathbf{Q_0} = \mathrm{diag}(\mathbf{E}_{T_i})$, with $\bar{\mathbf{J}}_{T_i} = \mathbf{J}_{T_i}/T_i$ and $\mathbf{E}_{T_i} = \mathbf{I}_{T_i} - \bar{\mathbf{J}}_{T_i}$. Define the transformations $\tilde{\mathbf{X}}_s = \mathbf{Q_0}\mathbf{X}_s$ and $\tilde{\mathbf{y}} = \mathbf{Q_0}\mathbf{y}$.

In the one-way model, estimation proceeds in a two-step fashion. First, you obtain estimates of the variance of the σ_ϵ^2 and σ_v^2. There are multiple ways to derive these estimates; PROC PANEL provides four options. All four options are valid for balanced or unbalanced panels. Once these estimates are in hand, they are used to form a weighting factor θ, and estimation proceeds via OLS on partial deviations from group means.

PROC PANEL provides four options for estimating variance components, as described in what follows.

Fuller and Battese Method

The Fuller and Battese method for estimating variance components can be obtained with the option VCOMP = FB and the option RANONE. The variance components are given by the following equations (For the approach in the two-way model, see Baltagi and Chang (1994); Fuller and Battese (1974)). Let

$$R(\nu) = \mathbf{y}'\mathbf{Z_0}(\mathbf{Z_0}'\mathbf{Z_0})^{-1}\mathbf{Z_0}'\mathbf{y}$$

$$R(\beta|\nu) = \tilde{\mathbf{y}}'\tilde{\mathbf{X}}_s'(\tilde{\mathbf{X}}_s'\tilde{\mathbf{X}}_s)^{-1}\tilde{\mathbf{X}}_s'\tilde{\mathbf{y}}$$

$$R(\beta) = \mathbf{y}'\mathbf{X}'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$$

$$R(\nu|\beta) = R(\beta|\nu) + R(\nu) - R(\beta)$$

The estimator of the error variance is given by

$$\hat{\sigma}_\epsilon^2 = \left\{\mathbf{y}'\mathbf{y} - R(\beta|\nu) - R(\nu)\right\} / \{M - N - (K-1)\}$$

If the NOINT option is specified, the estimator is

$$\hat{\sigma}_\epsilon^2 = \left\{\mathbf{y}'\mathbf{y} - R(\beta|\nu) - R(\nu)\right\} / (M - N - K)$$

The estimator of the cross-sectional variance component is given by

$$\hat{\sigma}_v^2 = \left\{R(\nu|\beta) - (N-1)\hat{\sigma}_\epsilon^2\right\} / \left\{M - \mathrm{tr}(\mathbf{Z_0}'\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Z_0})\right\}$$

Note that the error variance is the variance of the residual of the within estimator.

According to Baltagi and Chang (1994), the Fuller and Battese method is appropriate to apply to both balanced and unbalanced data. The Fuller and Battese method is the default for estimation of one-way random-effects models with balanced panels. However, the Fuller and Battese method does not always obtain nonnegative estimates for the cross section (or group) variance. In the case of a negative estimate, a warning is printed and the estimate is set to zero.

Wansbeek and Kapteyn Method

The Wansbeek and Kapteyn method for estimating variance components can be obtained by setting VCOMP = WK (together with the option RANONE). The estimation of the one-way unbalanced data model is performed by using a specialization (Baltagi and Chang 1994) of the approach used by Wansbeek and Kapteyn (1989) for unbalanced two-way models. The Wansbeek and Kapteyn method is the default for unbalanced data. If just RANONE is specified, without the VCOMP= option, PROC PANEL estimates the variance component under Wansbeek and Kapteyn's method.

The estimation of the variance components is performed by using a quadratic unbiased estimation (QUE) method. This involves focusing on quadratic forms of the centered residuals, equating their expected values to the realized quadratic forms, and solving for the variance components.

Let

$$q_1 = \tilde{\mathbf{u}}^{'}\mathbf{Q}_0\tilde{\mathbf{u}}$$

$$q_2 = \tilde{\mathbf{u}}^{'}\mathbf{P}_0\tilde{\mathbf{u}}$$

where the residuals $\tilde{\mathbf{u}}$ are given by $\tilde{\mathbf{u}} = (\mathbf{I}_M - \mathbf{J}_M)\{\mathbf{y} - \mathbf{X}_s(\tilde{\mathbf{X}}'_s\tilde{\mathbf{X}}_s)^{-1}\tilde{\mathbf{X}}_s'\tilde{\mathbf{y}}\}$ if there is an intercept and by $\tilde{\mathbf{u}} = \mathbf{y} - \mathbf{X}_s(\tilde{\mathbf{X}}'_s\tilde{\mathbf{X}}_s)^{-1}\tilde{\mathbf{X}}'_s\tilde{\mathbf{y}}$ if there is not.

Consider the expected values

$$E(q_1) = (M - N - (K-1))\sigma_\epsilon^2$$

$$E(q_2) = (N - 1 + \mathrm{tr}[(\mathbf{X}_s^{'}\mathbf{Q}_0\mathbf{X}_s)^{-1}\mathbf{X}_s^{'}\mathbf{P}_0\mathbf{X}_s] - \mathrm{tr}[(\mathbf{X}_s^{'}\mathbf{Q}_0\mathbf{X}_s)^{-1}\mathbf{X}_s^{'}\bar{\mathbf{J}}_M\mathbf{X}_s])\sigma_\epsilon^2$$
$$[M - (\sum_i T_i^2/M)]\sigma_\nu^2$$

where $\hat{\sigma}_\epsilon^2$ and $\hat{\sigma}_\nu^2$ are obtained by equating the quadratic forms to their expected values.

The estimator of the error variance is the residual variance of the within estimate. The Wansbeek and Kapteyn method can also generate negative variance components estimates.

Wallace and Hussain Method

The Wallace and Hussain method for estimating variance components can be obtained by setting VCOMP = WH (together with the option RANONE). Wallace-Hussain estimates start from OLS residuals on a data that are assumed to exhibit groupwise heteroscedasticity. As in the Wansbeek and Kapteyn method, you start with

$$q_1 = \tilde{\mathbf{u}}_{OLS}^{'}\mathbf{Q}_0\tilde{\mathbf{u}}_{OLS}$$

$$q_2 = \tilde{\mathbf{u}}_{OLS}^{'}\mathbf{P}_0\tilde{\mathbf{u}}_{OLS}$$

However, instead of using the 'true' errors, you substitute the OLS residuals. You solve the system

$$E(\hat{q}_1) = E(\hat{u}'_{OLS}\mathbf{Q}_0\hat{u}_{OLS}) = \delta_{11}\hat{\sigma}_\nu^2 + \delta_{12}\hat{\sigma}_\epsilon^2$$

$$E(\hat{q}_2) = E(\hat{u}'_{OLS}\mathbf{P}_0\hat{u}_{OLS}) = \delta_{21}\hat{\sigma}_\nu^2 + \delta_{22}\hat{\sigma}_\epsilon^2$$

The constants $\delta_{11}, \delta_{12}, \delta_{21}, \delta_{22}$ are given by

$$\delta_{11} = \mathrm{tr}\left(\left(\mathbf{X}'\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{Z}_0\mathbf{Z}_0'\mathbf{X}\right) - \mathrm{tr}\left(\left(\mathbf{X}'\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{P}_0\mathbf{X}\left(\mathbf{X}'\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{Z}_0\mathbf{Z}_0'\mathbf{X}\right)$$

$$\delta_{12} = M - N - K + \mathrm{tr}\left(\left(\mathbf{X}'\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{P}_0\mathbf{X}\right)$$

$$\delta_{21} = M - 2tr\left(\left(\mathbf{X}'\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{Z}_0\mathbf{Z}_0'\mathbf{X}\right) + \mathrm{tr}\left(\left(\mathbf{X}'\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{P}_0\mathbf{X}\right)$$

$$\delta_{22} = N - \mathrm{tr}\left(\left(\mathbf{X}'\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{P}_0\mathbf{X}\right)$$

where tr() is the trace operator on a square matrix.

Solving this system produces the variance components. This method is applicable to balanced and unbalanced panels. However, there is no guarantee of positive variance components. Any negative values are fixed equal to zero.

Nerlove Method

The Nerlove method for estimating variance components can be obtained by setting VCOMP = NL. The Nerlove method (see Baltagi 2008, p. 20) is assured to give estimates of the variance components that are always positive. Furthermore, it is simple in contrast to the previous estimators.

If γ_i is the *i*th fixed effect, Nerlove's method uses the variance of the fixed effects as the estimate of $\hat{\sigma}_\nu^2$. You have $\hat{\sigma}_\nu^2 = \sum_{i=1}^{N} \frac{(\gamma_i - \bar{\gamma})^2}{N-1}$, where $\bar{\gamma}$ is the mean fixed effect. The estimate of σ_ϵ^2 is simply the residual sum of squares of the one-way fixed-effects regression divided by the number of observations.

Transformation and Estimation

After you calculate the variance components from any method, the next task is to estimate the regression model of interest. For each individual, you form a weight (θ_i) as

$$\theta_i = 1 - \sigma_\epsilon / w_i$$

$$w_i^2 = T_i\sigma_\nu^2 + \sigma_\epsilon^2$$

where T_i is the *i*th cross section's time observations.

Taking the θ_i, you form the partial deviations,

$$\tilde{y}_{it} = y_{it} - \theta_i\bar{y}_{i\cdot}$$

$$\tilde{x}_{it} = x_{it} - \theta_i\bar{x}_{i\cdot}$$

where $\bar{y}_{i\cdot}$ and $\bar{x}_{i\cdot}$ are cross-sectional means of the dependent variable and independent variables (including the constant if any), respectively.

The random effects β is then the result of simple OLS on the transformed data.

Two-Way Random-Effects Model

The specification for the two-way random-effects model is

$$u_{it} = v_i + e_t + \epsilon_{it}$$

As in the one-way random-effects model, the PANEL procedure provides four options for variance component estimators. Unlike the one-way random-effects model, unbalanced panels present some special concerns.

Let $\mathbf{X}_*$ and $\mathbf{y}_*$ be the independent and dependent variables arranged by time and by cross section within each time period. (Note that the input data set used by the PANEL procedure must be sorted by cross section and then by time within each cross section.) Let M_t be the number of cross sections observed in time t and $\sum_t M_t = M$. Let $\mathbf{D}_t$ be the $M_t \times N$ matrix obtained from the $N \times N$ identity matrix from which rows that correspond to cross sections not observed at time t have been omitted. Consider

$$\mathbf{Z} = (\mathbf{Z}_1, \mathbf{Z}_2)$$

where $\mathbf{Z}_1 = (\mathbf{D}_1', \mathbf{D}_2', \ldots..\mathbf{D}_T')'$ and $\mathbf{Z}_2 = \mathrm{diag}(\mathbf{D}_1\mathbf{j}_N, \mathbf{D}_2\mathbf{j}_N, \ldots\ldots \mathbf{D}_T\mathbf{j}_N)$.

The matrix $\mathbf{Z}$ gives the dummy variable structure for the two-way model.

For notational ease, let

$$\Delta_N = \mathbf{Z}_1'\mathbf{Z}_1; \quad \Delta_T = \mathbf{Z}_2'\mathbf{Z}_2; \quad \mathbf{A} = \mathbf{Z}_2'\mathbf{Z}_1$$

$$\bar{\mathbf{Z}} = \mathbf{Z}_2 - \mathbf{Z}_1\Delta_N^{-1}\mathbf{A}'$$

$$\bar{\Delta}_1 = \mathbf{I}_M - \mathbf{Z}_1\Delta_N^{-1}\mathbf{Z}_1'$$

$$\bar{\Delta}_2 = \mathbf{I}_M - \mathbf{Z}_2\Delta_T^{-1}\mathbf{Z}_2'$$

$$\mathbf{Q} = \Delta_T - \mathbf{A}\Delta_N^{-1}\mathbf{A}'$$

$$\mathbf{P} = (\mathbf{I}_M - \mathbf{Z}_1\Delta_N^{-1}\mathbf{Z}_1') - \bar{\mathbf{Z}}\mathbf{Q}^{-1}\bar{\mathbf{Z}}'$$

Fuller and Battese Method

The Fuller and Battese method for estimating variance components can be obtained by setting VCOMP = FB (with the option RANTWO). FB is the default method for a RANTWO model with balanced panel. If RANTWO is requested without specifying the VCOMP= option, PROC PANEL proceeds under the Fuller and Battese method.

Following the discussion in Baltagi, Song, and Jung (2002), the Fuller and Battese method forms the estimates as follows.

The estimator of the error variance is

$$\hat{\sigma}_\epsilon^2 = \tilde{\mathbf{u}}'\mathbf{P}\tilde{\mathbf{u}}/(M - T - N + 1 - (K - 1))$$

where $\mathbf{P}$ is the Wansbeek and Kapteyn within estimator for unbalanced (or balanced) panel in a two-way setting.

The estimator of the error variance is the same as that in the Wansbeek and Kapteyn method.

Consider the expected values

$$\begin{aligned} E(q_N) &= \sigma_\epsilon^2 [M - T - K + 1] \\ &+ \sigma_\nu^2 \left[M - T - \text{tr} \left(\mathbf{X}_s' \bar{\Delta}_2 \mathbf{Z}_1 \mathbf{Z}_1' \bar{\Delta}_2 \mathbf{X}_s \left(\mathbf{X}_s' \bar{\Delta}_2 \mathbf{X}_s \right)^{-1} \right) \right] \\ E(q_T) &= \sigma_\epsilon^2 [M - N - K + 1] \\ &+ \sigma_e^2 \left[M - N - \text{tr} \left(\mathbf{X}_s' \bar{\Delta}_1 \mathbf{Z}_2 \mathbf{Z}_2' \bar{\Delta}_1 \mathbf{X}_s \left(\mathbf{X}_s' \bar{\Delta}_1 \mathbf{X}_s \right)^{-1} \right) \right] \end{aligned}$$

Just as in the one-way case, there is always the possibility that the (estimated) variance components will be negative. In such a case, the negative components are fixed to equal zero. After substituting the group sum of the within residuals for (q_N), the time sums of the within residuals for (q_T), and $\hat{\sigma}_\epsilon^2$, the two equations are solved for $\hat{\sigma}_\nu^2$ and $\hat{\sigma}_e^2$.

Wansbeek and Kapteyn Method

The Wansbeek and Kapteyn method for estimating variance components can be obtained by setting VCOMP = WK. The following methodology, outlined in Wansbeek and Kapteyn (1989) is used to handle both balanced and unbalanced data. The Wansbeek and Kapteyn method is the default for a RANTWO model with unbalanced panel. If RANTWO is requested without specifying the VCOMP= option, PROC PANEL proceeds under the Wansbeek and Kapteyn method if the panel is unbalanced.

The estimator of the error variance is

$$\hat{\sigma}_\epsilon^2 = \tilde{\mathbf{u}}' \mathbf{P} \tilde{\mathbf{u}} / (M - T - N + 1 - (K - 1))$$

where the $\tilde{\mathbf{u}}$ are given by $\tilde{\mathbf{u}} = (\mathbf{I}_M - \mathbf{j}_M \mathbf{j}'_M / M)(\mathbf{y}_* - \mathbf{X}_{*s}(\mathbf{X}'_{*s}\mathbf{P}\mathbf{X}_{*s})^{-1}\mathbf{X}_{*s}{}'\mathbf{P}\mathbf{y}_*)$ if there is an intercept and by $\tilde{\mathbf{u}} = (\mathbf{y}_* - \mathbf{X}_{*s}(\mathbf{X}'_{*s}\mathbf{P}\mathbf{X}_{*s})^{-1}\mathbf{X}'_{*s}\mathbf{P}\mathbf{y}_*$ if there is not.

The estimation of the variance components is performed by using a quadratic unbiased estimation (QUE) method that involves computing on quadratic forms of the residuals $\tilde{\mathbf{u}}$, equating their expected values to the realized quadratic forms, and solving for the variance components.

Let

$$q_N = \tilde{\mathbf{u}}' \mathbf{Z}_2 \Delta_T^{-1} \mathbf{Z}_2' \tilde{\mathbf{u}}$$

$$q_T = \tilde{\mathbf{u}}' \mathbf{Z}_1 \Delta_N^{-1} \mathbf{Z}_1' \tilde{\mathbf{u}}$$

The expected values are

$$E(q_N) = (T + k_N - (1 + k_0))\sigma^2 + (T - \frac{\lambda_1}{M})\sigma_\nu^2 + (M - \frac{\lambda_2}{M})\sigma_e^2$$

$$\begin{aligned} E(q_T) &= (N + k_T - (1 + k_0))\sigma^2 \\ &+ (M - \frac{\lambda_1}{M})\sigma_\nu^2 + (N - \frac{\lambda_2}{M})\sigma_e^2 \end{aligned}$$

where

$$k_0 = \mathbf{j}_M' \mathbf{X}_{*s} (\mathbf{X}_{*s}' \mathbf{P} \mathbf{X}_{*s})^{-1} \mathbf{X}_{*s}' \mathbf{j}_M / M$$

$$k_N = \mathrm{tr}((\mathbf{X}_{*s}^{'}\mathbf{P}\mathbf{X}_{*s})^{-1}\mathbf{X}_{*s}^{'}\mathbf{Z}_2\Delta_T^{-1}\mathbf{Z}_2^{'}\mathbf{X}_{*s})$$

$$k_T = \mathrm{tr}((\mathbf{X}_{*s}^{'}\mathbf{P}\mathbf{X}_{*s})^{-1}\mathbf{X}_{*s}^{'}\mathbf{Z}_1\Delta_N^{-1}\mathbf{Z}_1^{'}\mathbf{X}_{*s})$$

$$\lambda_1 = \mathbf{j}_M^{'}\mathbf{Z}_1\mathbf{Z}_1^{'}\mathbf{j}_M$$

$$\lambda_2 = \mathbf{j}_M^{'}\mathbf{Z}_2\mathbf{Z}_2^{'}\mathbf{j}_M$$

The quadratic unbiased estimators for σ_v^2 and σ_e^2 are obtained by equating the expected values to the quadratic forms and solving for the two unknowns.

When the NOINT option is specified, the variance component equations change slightly. In particular, the following is true (Wansbeek and Kapteyn 1989):

$$E(q_N) = (T + k_N)\sigma^2 + T\sigma_v^2 + M\sigma_e^2$$

$$E(q_T) = (N + k_T)\sigma^2 + M\sigma_v^2 + N\sigma_e^2$$

Wallace and Hussain Method

The Wallace and Hussain method for estimating variance components can be obtained by setting VCOMP = WH. Wallace and Hussain's method is by far the most computationally intensive. It uses the OLS residuals to estimate the variance components. In other words, the Wallace and Hussain method assumes that the following holds:

$$q_\epsilon = \tilde{\mathbf{u}}_{OLS}^{'}\mathbf{P}\tilde{\mathbf{u}}_{OLS}$$

$$q_N = \tilde{\mathbf{u}}_{OLS}^{'}\mathbf{Z}_2\Delta_T^{-1}\mathbf{Z}_2^{'}\tilde{\mathbf{u}}_{OLS}$$

$$q_T = \tilde{\mathbf{u}}_{OLS}^{'}\mathbf{Z}_1\Delta_N^{-1}\mathbf{Z}_1^{'}\tilde{\mathbf{u}}_{OLS}$$

Taking expectations yields

$$E(q_\epsilon) = E\left(\tilde{\mathbf{u}}_{OLS}^{'}\mathbf{P}\tilde{\mathbf{u}}_{OLS}\right) = \delta_{11}\sigma_\epsilon^2 + \delta_{12}\sigma_v^2 + \delta_{13}\sigma_e^2$$

$$E(q_N) = E\left(\tilde{\mathbf{u}}_{OLS}^{'}\mathbf{Z}_2\Delta_T^{-1}\mathbf{Z}_2^{'}\tilde{\mathbf{u}}_{OLS}\right) = \delta_{21}\sigma_\epsilon^2 + \delta_{22}\sigma_v^2 + \delta_{23}\sigma_e^2$$

$$E(q_T) = E\left(\tilde{\mathbf{u}}_{OLS}^{'}\mathbf{Z}_1\Delta_N^{-1}\mathbf{Z}_1^{'}\tilde{\mathbf{u}}_{OLS}\right) = \delta_{31}\sigma_\epsilon^2 + \delta_{32}\sigma_v^2 + \delta_{33}\sigma_e^2$$

where the δ_{js} constants are defined by

$$\delta_{11} = M - N - T + 1 - \mathrm{tr}\left(\mathbf{X}^{'}\mathbf{P}\mathbf{X}\left(\mathbf{X}^{'}\mathbf{X}\right)^{-1}\right)$$

$$\delta_{12} = \mathrm{tr}\left(\mathbf{X}^{'}\mathbf{Z}_1\mathbf{Z}_1^{'}\mathbf{X}\left(\mathbf{X}^{'}\mathbf{X}\right)^{-1}\left(\mathbf{X}^{'}\mathbf{P}\mathbf{X}\left(\mathbf{X}^{'}\mathbf{X}\right)^{-1}\right)\right)$$

$$\delta_{13} = \mathrm{tr}\left(\mathbf{X}^{'}\mathbf{Z}_2\mathbf{Z}_2^{'}\mathbf{X}\left(\mathbf{X}^{'}\mathbf{X}\right)^{-1}\left(\mathbf{X}^{'}\mathbf{P}\mathbf{X}\left(\mathbf{X}^{'}\mathbf{X}\right)^{-1}\right)\right)$$

$$\delta_{21} = T - \mathrm{tr}\left(\mathbf{X}^{'}\mathbf{Z}_2\Delta_T^{-1}\mathbf{Z}_2^{'}\mathbf{X}\left(\mathbf{X}^{'}\mathbf{X}\right)^{-1}\right)$$

$$\begin{aligned}\delta_{22} &= T - 2\mathrm{tr}\left(\mathbf{X}'\mathbf{Z}_2\Delta_T^{-1}\mathbf{Z}_2'\mathbf{Z}_1\mathbf{Z}_1'\mathbf{X}\left(\mathbf{X}'\mathbf{X}\right)^{-1}\right)\\ &+ \mathrm{tr}\left(\mathbf{X}'\mathbf{Z}_2\Delta_T^{-1}\mathbf{Z}_2'\mathbf{X}\left(\mathbf{X}'\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{Z}_1\mathbf{Z}_1'\mathbf{X}\left(\mathbf{X}'\mathbf{X}\right)^{-1}\right)\end{aligned}$$

$$\begin{aligned}\delta_{23} &= T - 2\mathrm{tr}\left(\mathbf{X}'\mathbf{Z}_2\mathbf{Z}_2'\mathbf{X}\left(\mathbf{X}'\mathbf{X}\right)^{-1}\right)\\ &+ \mathrm{tr}\left(\mathbf{X}'\mathbf{Z}_2\Delta_T^{-1}\mathbf{Z}_2'\mathbf{X}\left(\mathbf{X}'\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{Z}_2\mathbf{Z}_2'\mathbf{X}\left(\mathbf{X}'\mathbf{X}\right)^{-1}\right)\end{aligned}$$

$$\delta_{31} = N - \mathrm{tr}\left(\mathbf{X}'\mathbf{Z}_1\Delta_N^{-1}\mathbf{Z}_1'\mathbf{X}\left(\mathbf{X}'\mathbf{X}\right)^{-1}\right)$$

$$\begin{aligned}\delta_{32} &= M - 2\mathrm{tr}\left(\mathbf{X}'\mathbf{Z}_1\mathbf{Z}_1'\mathbf{X}\left(\mathbf{X}'\mathbf{X}\right)^{-1}\right)\\ &+ \mathrm{tr}\left(\mathbf{X}'\mathbf{Z}_1\Delta_N^{-1}\mathbf{Z}_1'\mathbf{X}\left(\mathbf{X}'\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{Z}_1\mathbf{Z}_1'\mathbf{X}\left(\mathbf{X}'\mathbf{X}\right)^{-1}\right)\end{aligned}$$

$$\begin{aligned}\delta_{33} &= N - 2\mathrm{tr}\left(\mathbf{X}'\mathbf{Z}_1\Delta_N^{-1}\mathbf{Z}_1'\mathbf{Z}_2\mathbf{Z}_2'\mathbf{X}\left(\mathbf{X}'\mathbf{X}\right)^{-1}\right)\\ &+ \mathrm{tr}\left(\mathbf{X}'\mathbf{Z}_1\Delta_N^{-1}\mathbf{Z}_1'\mathbf{X}\left(\mathbf{X}'\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{Z}_2\mathbf{Z}_2'\mathbf{X}\left(\mathbf{X}'\mathbf{X}\right)^{-1}\right)\end{aligned}$$

The PANEL procedure solves this system for the estimates $\hat{\sigma}_\epsilon$, $\hat{\sigma}_v$, and $\hat{\sigma}_e$. Some of the estimated variance components can be negative. Negative components are set to zero and estimation proceeds.

Nerlove Method

The Nerlove method for estimating variance components can be obtained with by setting VCOMP = NL.

The estimator of the error variance is

$$\hat{\sigma}_\epsilon^2 = \tilde{\mathbf{u}}'\mathbf{P}\tilde{\mathbf{u}}/M$$

The variance components for cross section and time effects are:

$$\hat{\sigma}_v^2 = \sum_{i=1}^{N}\frac{(\gamma_i - \bar{\gamma})^2}{N-1} \text{ where } \gamma_i \text{ is the } i\text{th cross section effect}$$

and

$$\hat{\sigma}_e^2 = \sum_{i=1}^{T}\frac{(\alpha_t - \bar{\alpha})^2}{T-1} \text{ where } \alpha_i \text{ is the } t\text{th time effect}$$

Transformation and Estimation

After you calculate the estimates of the variance components, you can proceed to the final estimation. If the panel is balanced, partial mean deviations are used:

$$\tilde{y}_{it} = y_{it} - \theta_1 \bar{y}_{i\cdot} - \theta_2 \bar{y}_{\cdot t} + \theta_3 \bar{y}_{\cdot\cdot}$$

$$\tilde{x}_{it} = x_{it} - \theta_1 \bar{x}_{i\cdot} - \theta_2 \bar{x}_{\cdot t} + \theta_3 \bar{x}_{\cdot\cdot}$$

The θ estimates are obtained from:

$$\theta_1 = 1 - \frac{\sigma_\epsilon}{\sqrt{T\sigma_\nu^2 + \sigma_\epsilon^2}}$$

$$\theta_2 = 1 - \frac{\sigma_\epsilon}{\sqrt{N\sigma_e^2 + \sigma_\epsilon^2}}$$

$$\theta_3 = \theta_1 + \theta_2 + \frac{\sigma_\epsilon}{\sqrt{T\sigma_\nu^2 + N\sigma_e^2 + \sigma_\epsilon^2}} - 1;$$

With these partial deviations, PROC PANEL uses OLS on the transformed series (including an intercept if you want).

The case of an unbalanced panel is somewhat more complicated. You could naively substitute the variance components in the equation below:

$$\Omega = \sigma_\epsilon^2 \mathbf{I}_M + \sigma_\nu^2 \mathbf{Z}_1 \mathbf{Z}_1' + \sigma_e^2 \mathbf{Z}_2 \mathbf{Z}_2'$$

After inverting the expression for Ω, it is possible to do GLS on the data (even if the panel is unbalanced). However, the inversion of Ω is no small matter because the dimension is at least $\frac{M(M+1)}{2}$.

Wansbeek and Kapteyn show that the inverse of Ω can be written as

$$\sigma_\epsilon^2 \Omega^{-1} = \mathbf{V} - \mathbf{V}\mathbf{Z}_2 \tilde{\mathbf{P}}^{-1} \mathbf{Z}_2' \mathbf{V}$$

with the following:

$$\begin{aligned}
\mathbf{V} &= \mathbf{I}_M - \mathbf{Z}_1 \tilde{\Delta}_N^{-1} \mathbf{Z}_1' \\
\tilde{\mathbf{P}} &= \tilde{\Delta}_T - \mathbf{A} \tilde{\Delta}_N^{-1} \mathbf{A}' \\
\tilde{\Delta}_N &= \Delta_N + \left(\frac{\sigma_\epsilon^2}{\sigma_\nu^2} \right) \mathbf{I}_N \\
\tilde{\Delta}_T &= \Delta_T + \left(\frac{\sigma_\epsilon^2}{\sigma_e^2} \right) \mathbf{I}_T
\end{aligned}$$

Computationally, this is a much less intensive approach.

By using the inverse of the covariance matrix of the error, it becomes possible to complete GLS on the unbalanced panel, using the transform $\tilde{\mathbf{y}_*} = \sigma_\epsilon \Omega^{-1/2} \mathbf{y}_*$, and similarly for the regressors.

Hausman-Taylor Estimation

The Hausman and Taylor (1981) model is a hybrid that combines the consistency of a fixed-effects model with the efficiency and applicability of a random-effects model. One-way random-effects models assume exogeneity of the regressors, namely that they be independent of both the cross-sectional and observation-level errors. In cases where some regressors are correlated with the cross-sectional errors, the random effects model can be adjusted to deal with the endogeneity.

Consider the one-way model:

$$y_{it} = \mathbf{X}_{1it}\boldsymbol{\beta}_1 + \mathbf{X}_{2it}\boldsymbol{\beta}_2 + \mathbf{Z}_{1i}\boldsymbol{\gamma}_1 + \mathbf{Z}_{2i}\boldsymbol{\gamma}_2 + \nu_i + \epsilon_{it}$$

The regressors are subdivided so that the $\mathbf{X}$ variables vary within cross sections whereas the $\mathbf{Z}$ variables do not and would otherwise be dropped from a fixed-effects model. The subscript 1 denotes variables that are independent of both error terms (exogenous variables), and the subscript 2 denotes variables that are independent of the observation-level errors ϵ_{it} but correlated with cross-sectional errors ν_i (endogenous variables). The intercept term (if your model has one) is included as part of $\mathbf{Z}_1$ in what follows.

The Hausman-Taylor estimator is an instrumental variables regression on data that are weighted similarly to data for random-effects estimation. In both cases, the weights are functions of the estimated variance components.

Begin with $\mathbf{P}_0 = \mathrm{diag}(\bar{\mathbf{J}}_{T_i})$ and $\mathbf{Q}_0 = \mathrm{diag}(\mathbf{E}_{T_i})$. The mean transformation vector is $\bar{\mathbf{J}}_{T_i} = \mathbf{J}_{T_i}/T_i$ and the deviations from the mean transform is $\mathbf{E}_{T_i} = \mathbf{I}_{T_i} - \bar{\mathbf{J}}_{T_i}$, where $\mathbf{J}_{T_i}$ is a square matrix of ones of dimension T_i.

The observation-level variance is estimated from a standard fixed-effects model fit. For $\mathbf{X}_s = \{\mathbf{X}_1, \mathbf{X}_2\}$, $\tilde{\mathbf{X}}_s = \mathbf{Q}_0\mathbf{X}_s$, and $\tilde{\mathbf{y}} = \mathbf{Q}_0\mathbf{y}$, let

$$\begin{aligned}
\tilde{\boldsymbol{\beta}}_s &= \left(\tilde{\mathbf{X}}_s'\tilde{\mathbf{X}}_s\right)^{-1}\tilde{\mathbf{X}}_s'\tilde{\mathbf{y}} \\
\mathrm{SSE} &= \left(\tilde{\mathbf{y}} - \tilde{\mathbf{X}}_s\tilde{\boldsymbol{\beta}}_s\right)'\left(\tilde{\mathbf{y}} - \tilde{\mathbf{X}}_s\tilde{\boldsymbol{\beta}}_s\right) \\
\hat{\sigma}_\epsilon^2 &= \mathrm{SSE}/(M-N)
\end{aligned}$$

To estimate the cross-sectional error variance, form the mean residuals $\mathbf{r} = \mathbf{P}_0'(\mathbf{y} - \mathbf{X}_s\tilde{\boldsymbol{\beta}}_s)$. You can use the mean residuals to obtain intermediate estimates of the coefficients for $\mathbf{Z}_1$ and $\mathbf{Z}_2$ via two-stage least squares (2SLS) estimation. At the first stage, use $\mathbf{X}_1$ and $\mathbf{Z}_1$ as instrumental variables to predict $\mathbf{Z}_2$. At the second stage, regress $\mathbf{r}$ on both $\mathbf{Z}_1$ and the predicted $\mathbf{Z}_2$ to obtain $\hat{\boldsymbol{\gamma}}_1^m$ and $\hat{\boldsymbol{\gamma}}_2^m$.

To estimate the cross-sectional variance, form $\hat{\sigma}_\nu^2 = \{R(\nu)/N - \hat{\sigma}_\epsilon^2\}/\bar{T}$, with $\bar{T} = N/(\sum_{i=1}^N T_i^{-1})$ and

$$R(\nu) = \left(\mathbf{r} - \mathbf{Z}_1\hat{\boldsymbol{\gamma}}_1^m - \mathbf{Z}_2\hat{\boldsymbol{\gamma}}_2^m\right)'\left(\mathbf{r} - \mathbf{Z}_1\hat{\boldsymbol{\gamma}}_1^m - \mathbf{Z}_2\hat{\boldsymbol{\gamma}}_2^m\right)$$

After variance-component estimation, transform the dependent variable into partial deviations: $y_{it}^* = y_{it} - \hat{\theta}_i\bar{y}_{i\cdot}$. Likewise, transform the regressors to form $\mathbf{X}_{1it}^*$, $\mathbf{X}_{2it}^*$, $\mathbf{Z}_{1i}^*$, and $\mathbf{Z}_{2i}^*$. The partial weights $\hat{\theta}_i$ are determined by $\hat{\theta}_i = 1 - \hat{\sigma}_\epsilon/\hat{w}_i$, with $\hat{w}_i^2 = T_i\hat{\sigma}_\nu^2 + \hat{\sigma}_\epsilon^2$.

Finally, you obtain the Hausman-Taylor estimates by performing 2SLS regression of y_{it}^* on $\mathbf{X}_{1it}^*$, $\mathbf{X}_{2it}^*$, $\mathbf{Z}_{1i}^*$, and $\mathbf{Z}_{2i}^*$. For the first-stage regression, use the following instruments:

- $\tilde{\mathbf{X}}_{it}$, the deviations from cross-sectional means for all time-varying variables $\mathbf{X}$, for the ith cross section during time period t
- $(1-\hat{\theta}_i)\bar{\mathbf{X}}_{1i.}$, where $\bar{\mathbf{X}}_{1i.}$ are the means of the time-varying exogenous variables for the ith cross section
- $(1-\hat{\theta}_i)\mathbf{Z}_{1i}$

Multiplication by the factor $(1-\hat{\theta}_i)$ is redundant in balanced data, but necessary in the unbalanced case to produce accurate instrumentation; see Gardner (1998).

Let k_1 equal the number of regressors in $\mathbf{X}_1$, and g_2 equal the number of regressors in $\mathbf{Z}_2$. Then the Hausman-Taylor model is identified only if $k_1 \geq g_2$; otherwise, no estimation will take place.

Hausman and Taylor (1981) describe a specification test that compares their model to fixed effects. For a null hypothesis of fixed effects, Hausman's m statistic is calculated by comparing the parameter estimates and variance matrices for both models, identically to how it is calculated for one-way random effects models; for more information, see the section "Specification Tests" on page 1873. The degrees of freedom of the test, however, are not based on matrix rank but instead are equal to $k_1 - g_2$.

Amemiya-MaCurdy Estimation

The Amemiya and MaCurdy (1986) model is similar to the Hausman-Taylor model. Following the development in the section "Hausman-Taylor Estimation" on page 1849, estimation is identical up to the final 2SLS instrumental variables regression. In addition to the set of instruments used by the Hausman-Taylor estimator, use the following:

- $\mathbf{X}_{1i1}, \mathbf{X}_{1i2}, \ldots, \mathbf{X}_{1iT}$

For each observation in the ith cross section, you use the data on the time-varying exogenous regressors for the entire cross section. Because of the structure of the added instruments, the Amemiya-MaCurdy estimator can be applied only to balanced data.

The Amemiya-MaCurdy model attempts to gain efficiency over Hausman-Taylor by adding instruments. This comes at a price of a more stringent assumption on the exogeneity of the $\mathbf{X}_1$ variables. Although the Hausman-Taylor model requires only that the cross-sectional means of $\mathbf{X}_1$ be orthogonal to v_i, the Amemiya-MaCurdy estimation requires orthogonality at every point in time; see Baltagi (2008, sec. 7.4).

A Hausman specification test is provided to test the validity of the added assumption. Define $\boldsymbol{\alpha}' = (\boldsymbol{\beta}_1', \boldsymbol{\beta}_2', \boldsymbol{\gamma}_1', \boldsymbol{\gamma}_2')$, its Hausman-Taylor estimate as $\hat{\boldsymbol{\alpha}}_{\mathrm{HT}}$, and its Amemiya-MaCurdy estimate as $\hat{\boldsymbol{\alpha}}_{\mathrm{AM}}$. Under the null hypothesis, both estimators are consistent and $\hat{\boldsymbol{\alpha}}_{\mathrm{AM}}$ is efficient. The Hausman test statistic is then

$$m = (\hat{\boldsymbol{\alpha}}_{\mathrm{HT}} - \hat{\boldsymbol{\alpha}}_{\mathrm{AM}})' \left(\hat{\mathbf{S}}_{\mathrm{HT}} - \hat{\mathbf{S}}_{\mathrm{AM}}\right)^{-1} (\hat{\boldsymbol{\alpha}}_{\mathrm{HT}} - \hat{\boldsymbol{\alpha}}_{\mathrm{AM}})$$

where $\hat{\mathbf{S}}_{\mathrm{HT}}$ and $\hat{\mathbf{S}}_{\mathrm{AM}}$ are variance-covariance estimates of $\hat{\boldsymbol{\alpha}}_{\mathrm{HT}}$ and $\hat{\boldsymbol{\alpha}}_{\mathrm{AM}}$, respectively. Under the null hypothesis, m is distributed as χ^2 with degrees of freedom equal to the rank of $(\hat{\mathbf{S}}_{\mathrm{HT}} - \hat{\mathbf{S}}_{\mathrm{AM}})^{-1}$.

Parks Method (Autoregressive Model)

Parks (1967) considered the first-order autoregressive model in which the random errors u_{it}, $i = 1, 2, \ldots, N$, and $t = 1, 2, \ldots, T$ have the structure

$$\begin{aligned} E(u_{it}^2) &= \sigma_{ii}(\text{heteroscedasticity}) \\ E(u_{it}u_{jt}) &= \sigma_{ij}(\text{contemporaneously correlated}) \\ u_{it} &= \rho_i u_{i,t-1} + \epsilon_{it}(\text{autoregression}) \end{aligned}$$

where

$$\begin{aligned} E(\epsilon_{it}) &= 0 \\ E(u_{i,t-1}\epsilon_{jt}) &= 0 \\ E(\epsilon_{it}\epsilon_{jt}) &= \phi_{ij} \\ E(\epsilon_{it}\epsilon_{js}) &= 0(s \neq t) \\ E(u_{i0}) &= 0 \\ E(u_{i0}u_{j0}) &= \sigma_{ij} = \phi_{ij}/(1 - \rho_i\rho_j) \end{aligned}$$

The model assumed is first-order autoregressive with contemporaneous correlation between cross sections. In this model, the covariance matrix for the vector of random errors $\mathbf{u}$ can be expressed as

$$E(\mathbf{uu}') = \mathbf{V} = \begin{bmatrix} \sigma_{11}P_{11} & \sigma_{12}P_{12} & \ldots & \sigma_{1N}P_{1N} \\ \sigma_{21}P_{21} & \sigma_{22}P_{22} & \ldots & \sigma_{2N}P_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ \sigma_{N1}P_{N1} & \sigma_{N2}P_{N2} & \ldots & \sigma_{NN}P_{NN} \end{bmatrix}$$

where

$$P_{ij} = \begin{bmatrix} 1 & \rho_j & \rho_j^2 & \ldots & \rho_j^{T-1} \\ \rho_i & 1 & \rho_j & \ldots & \rho_j^{T-2} \\ \rho_i^2 & \rho_i & 1 & \ldots & \rho_j^{T-3} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \rho_i^{T-1} & \rho_i^{T-2} & \rho_i^{T-3} & \ldots & 1 \end{bmatrix}$$

The matrix $\mathbf{V}$ is estimated by a two-stage procedure, and $\boldsymbol{\beta}$ is then estimated by generalized least squares. The first step in estimating $\mathbf{V}$ involves the use of ordinary least squares to estimate $\boldsymbol{\beta}$ and obtain the fitted residuals, as follows:

$$\hat{\mathbf{u}} = \mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}_{OLS}$$

A consistent estimator of the first-order autoregressive parameter is then obtained in the usual manner, as follows:

$$\hat{\rho}_i = \left(\sum_{t=2}^{T} \hat{u}_{it}\hat{u}_{i,t-1}\right) \Big/ \left(\sum_{t=2}^{T} \hat{u}_{i,t-1}^2\right) i = 1, 2, \ldots, N$$

Finally, the autoregressive characteristic of the data is removed (asymptotically) by the usual transformation of taking weighted differences. That is, for $i = 1, 2, \ldots, N$,

$$y_{i1}\sqrt{1-\hat{\rho}_i^2} = \sum_{k=1}^{p} X_{i1k\bullet k}\sqrt{1-\hat{\rho}_i^2} + u_{i1}\sqrt{1-\hat{\rho}_i^2}$$

$$y_{it} - \hat{\rho}_i y_{i,t-1} = \sum_{k=1}^{p} (X_{itk} - \hat{\rho}_i \mathbf{X}_{i,t-1,k})\boldsymbol{\beta}_k + u_{it} - \hat{\rho}_i u_{i,t-1} t = 2, \ldots, T$$

which is written

$$y_{it}^* = \sum_{k=1}^{p} X_{itk}^* \boldsymbol{\beta}_k + u_{it}^* \quad i = 1, 2, \ldots, N; \quad t = 1, 2, \ldots, T$$

Notice that the transformed model has not lost any observations (Seely and Zyskind 1971).

The second step in estimating the covariance matrix $\mathbf{V}$ is applying ordinary least squares to the preceding transformed model, obtaining

$$\hat{\mathbf{u}}^* = \mathbf{y}^* - \mathbf{X}^* \boldsymbol{\beta}_{OLS}^*$$

from which the consistent estimator of σ_{ij} is calculated as follows:

$$s_{ij} = \frac{\hat{\phi}_{ij}}{(1 - \hat{\rho}_i \hat{\rho}_j)}$$

where

$$\hat{\phi}_{ij} = \frac{1}{(T-p)} \sum_{t=1}^{T} \hat{u}_{it}^* \hat{u}_{jt}^*$$

Estimated generalized least squares (EGLS) then proceeds in the usual manner,

$$\hat{\boldsymbol{\beta}}_P = (\mathbf{X}'\hat{\mathbf{V}}^{-1}\mathbf{X})^{-1}\mathbf{X}'\hat{\mathbf{V}}^{-1}\mathbf{y}$$

where $\hat{\mathbf{V}}$ is the derived consistent estimator of $\mathbf{V}$. For computational purposes, $\hat{\boldsymbol{\beta}}_P$ is obtained directly from the transformed model,

$$\hat{\boldsymbol{\beta}}_P = (\mathbf{X}^{*'}(\hat{\Phi}^{-1} \otimes I_T)\mathbf{X}^*)^{-1}\mathbf{X}^{*'}(\hat{\Phi}^{-1} \otimes I_T)\mathbf{y}^*$$

where $\hat{\Phi} = [\hat{\phi}_{ij}]_{i,j=1,\ldots,N}$.

The preceding procedure is equivalent to Zellner's two-stage methodology applied to the transformed model (Zellner 1962).

Parks demonstrates that this estimator is consistent and asymptotically, normally distributed with

$$\mathrm{Var}(\hat{\boldsymbol{\beta}}_P) = (\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1}$$

Standard Corrections

For the PARKS option, the first-order autocorrelation coefficient must be estimated for each cross section. Let ρ be the $N \times 1$ vector of true parameters and $R = (r_1, \ldots, r_N)'$ be the corresponding vector of estimates. Then, to ensure that only range-preserving estimates are used in PROC PANEL, the following modification for R is made:

$$r_i = \begin{cases} r_i & \text{if } |r_i| < 1 \\ \max(.95, \text{rmax}) & \text{if } r_i \geq 1 \\ \min(-.95, \text{rmin}) & \text{if } r_i \leq -1 \end{cases}$$

where

$$\text{rmax} = \begin{cases} 0 & \text{if } r_i < 0 \text{ or } r_i \geq 1 \ \ \forall i \\ \max\limits_j [r_j : 0 \leq r_j < 1] & \text{otherwise} \end{cases}$$

and

$$\text{rmin} = \begin{cases} 0 & \text{if } r_i > 0 \text{ or } r_i \leq -1 \ \ \forall i \\ \max\limits_j [r_j : -1 < r_j \leq 0] & \text{otherwise} \end{cases}$$

Whenever this correction is made, a warning message is printed.

Da Silva Method (Variance-Component Moving Average Model)

The Da Silva method assumes that the observed value of the dependent variable at the *t*th time point on the *i*th cross-sectional unit can be expressed as

$$y_{it} = \mathbf{x}_{it}^{'}\beta + a_i + b_t + e_{it} \qquad i = 1, \ldots, N; t = 1, \ldots, T$$

where

$\mathbf{x}_{it}^{'} = (x_{it1}, \ldots, x_{itp})$ is a vector of explanatory variables for the *t*th time point and *i*th cross-sectional unit

$\beta = (\beta_1, \ldots, \beta_p)'$ is the vector of parameters

a_i is a time-invariant, cross-sectional unit effect

b_t is a cross-sectionally invariant time effect

e_{it} is a residual effect unaccounted for by the explanatory variables and the specific time and cross-sectional unit effects

Since the observations are arranged first by cross sections, then by time periods within cross sections, these equations can be written in matrix notation as

$$\mathbf{y} = \mathbf{X}\beta + \mathbf{u}$$

where

$$\mathbf{u} = (\mathbf{a} \otimes \mathbf{1}_T) + (\mathbf{1}_N \otimes \mathbf{b}) + \mathbf{e}$$

$$\mathbf{y} = (y_{11}, \ldots, y_{1T}, y_{21}, \ldots, y_{NT})'$$

$$\mathbf{X} = (\mathbf{x}_{11}, \ldots, \mathbf{x}_{1T}, \mathbf{x}_{21}, \ldots, \mathbf{x}_{NT})'$$

$$\mathbf{a} = (a_1 \ldots a_N)'$$

$$\mathbf{b} = (b_1 \ldots b_T)'$$

$$\mathbf{e} = (e_{11}, \ldots, e_{1T}, e_{21}, \ldots, e_{NT})'$$

Here $\mathbf{1}_N$ is an $N \times 1$ vector with all elements equal to 1, and $\otimes$ denotes the Kronecker product.

The following conditions are assumed:

1. $\mathbf{x}_{it}$ is a sequence of nonstochastic, known $p \times 1$ vectors in $\Re^p$ whose elements are uniformly bounded in $\Re^p$. The matrix $\mathbf{X}$ has a full column rank p.
2. $\boldsymbol{\beta}$ is a $p \times 1$ constant vector of unknown parameters.
3. $\mathbf{a}$ is a vector of uncorrelated random variables such that $E(a_i) = 0$ and $\mathrm{var}(a_i) = \sigma_a^2$, $\sigma_a^2 > 0, i = 1, \ldots, N$.
4. $\mathbf{b}$ is a vector of uncorrelated random variables such that $E(b_t) = 0$ and $\mathrm{var}(b_t) = \sigma_b^2$ where $\sigma_b^2 > 0$ and $t = 1, \ldots, T$.
5. $\mathbf{e}_i = (e_{i1}, \ldots, e_{iT})'$ is a sample of a realization of a finite moving-average time series of order $m < T - 1$ for each i ; hence,

   $$e_{it} = \alpha_0 \epsilon_{it} + \alpha_1 \epsilon_{it-1} + \ldots + \alpha_m \epsilon_{it-m} \quad t = 1, \ldots, T; i = 1, \ldots, N$$

   where $\alpha_0, \alpha_1, \ldots, \alpha_m$ are unknown constants such that $\alpha_0 \neq 0$ and $\alpha_m \neq 0$, and $\{\epsilon_{ij}\}_{j=-\infty}^{j=\infty}$ is a white noise process for each i—that is, a sequence of uncorrelated random variables with $E(\epsilon_t) = 0$, $E(\epsilon_t^2) = \sigma_\epsilon^2$, and $\sigma_\epsilon^2 > 0$. $\{\epsilon_{ij}\}_{j=-\infty}^{j=\infty}$ for $i = 1, \ldots, N$ are mutually uncorrelated.
6. The sets of random variables $\{a_i\}_{i=1}^N$, $\{b_t\}_{t=1}^T$, and $\{e_{it}\}_{t=1}^T$ for $i = 1, \ldots, N$ are mutually uncorrelated.
7. The random terms have normal distributions $a_i \sim N(0, \sigma_a^2)$, $b_t \sim N(0, \sigma_b^2)$, and $\epsilon_{t-k} \sim N(0, \sigma_\epsilon^2)$, for $i = 1, \ldots, N; t = 1, \ldots T$; and $k = 1, \ldots, m$.

If assumptions 1–6 are satisfied, then

$$E(\mathbf{y}) = \mathbf{X}\beta$$

and

$$\mathrm{var}(\mathbf{y}) = \sigma_a^2(I_N \otimes J_T) + \sigma_b^2(J_N \otimes I_T) + (I_N \otimes \Psi_T)$$

where Ψ_T is a $T \times T$ matrix with elements ψ_{ts} as follows:

$$\mathrm{Cov}(e_{it}e_{is}) = \begin{cases} \psi(|t-s|) & \text{if } |t-s| \leq m \\ 0 & \text{if } |t-s| > m \end{cases}$$

where $\psi(k) = \sigma_\epsilon^2 \sum_{j=0}^{m-k} \alpha_j \alpha_{j+k}$ for $k = |t-s|$. For the definition of I_N, I_T, J_N, and J_T, see the section "Fuller and Battese Method" on page 1841.

The covariance matrix, denoted by $\mathbf{V}$, can be written in the form

$$\mathbf{V} = \sigma_a^2(I_N \otimes J_T) + \sigma_b^2(J_N \otimes I_T) + \sum_{k=0}^{m} \psi(k)(I_N \otimes \Psi_T^{(k)})$$

where $\Psi_T^{(0)} = I_T$, and, for k =1,. . ., m, $\Psi_T^{(k)}$ is a band matrix whose kth off-diagonal elements are 1's and all other elements are 0's.

Thus, the covariance matrix of the vector of observations $\mathbf{y}$ has the form

$$\mathrm{Var}(\mathbf{y}) = \sum_{k=1}^{m+3} \nu_k V_k$$

where

$$\begin{aligned}
\nu_1 &= \sigma_a^2 \\
\nu_2 &= \sigma_b^2 \\
\nu_k &= \psi(k-3) k = 3, \ldots, m+3 \\
V_1 &= I_N \otimes J_T \\
V_2 &= J_N \otimes I_T \\
V_k &= I_N \otimes \Psi_T^{(k-3)} k = 3, \ldots, m+3
\end{aligned}$$

The estimator of $\boldsymbol{\beta}$ is a two-step GLS-type estimator—that is, GLS with the unknown covariance matrix replaced by a suitable estimator of $\mathbf{V}$. It is obtained by substituting Seely estimates for the scalar multiples $\nu_k, k = 1, 2, \ldots, m+3$.

Seely (1969) presents a general theory of unbiased estimation when the choice of estimators is restricted to finite dimensional vector spaces, with a special emphasis on quadratic estimation of functions of the form $\sum_{i=1}^{n} \delta_i \nu_i$.

The parameters ν_i (i =1,. . ., n) are associated with a linear model E($\mathbf{y}$)=$\mathbf{X}$ β with covariance matrix $\sum_{i=1}^{n} \nu_i V_i$ where V_i (i =1, . . ., n) are real symmetric matrices. The method is also discussed by Seely (1970b, a); Seely and Zyskind (1971). Seely and Soong (1971) consider the MINQUE principle, using an approach along the lines of Seely (1969).

Dynamic Panel Estimators

For an example on dynamic panel estimation using GMM option, see "Example 27.6: The Cigarette Sales Data: Dynamic Panel Estimation with GMM" on page 1916.

Consider the case of the following general model:

$$y_{it} = \Sigma_{l=1}^{maxlag} \phi_l y_{i(t-l)} + \Sigma_{k=1}^{K} \beta_k x_{itk} + \gamma_i + \alpha_t + \epsilon_{it}$$

The x variables can include ones that are correlated or uncorrelated to the individual effects, predetermined, or strictly exogenous. The variable x_{it}^p is defined as predetermined in the sense that $E\left(x_{it}^p \epsilon_{is}\right) \neq 0$ for $s < t$

and zero otherwise. The variable x^e_{it} is defined as strictly exogenous if $E\left(x^e_{it}\epsilon_{is}\right) = 0$ for all s and t. The γ_i and α_t are cross-sectional and time series fixed effects, respectively. Arellano and Bond (1991) show that it is possible to define conditions that should result in a consistent estimator.

Consider the simple case of an autoregression in a panel setting (with only individual effects):

$$y_{it} = \phi y_{i(t-1)} + \gamma_i + \epsilon_{it}$$

Differencing the preceding relationship results in:

$$\Delta y_{it} = \phi \Delta y_{i(t-1)} + \nu_{it}$$

where $\nu_{it} = \epsilon_{it} - \epsilon_{it-1}$.

Obviously, y_{it} is not exogenous. However, Arellano and Bond (1991) show that it is still useful as an instrument, if properly lagged. This instrument is required with the option DEPVAR(LEVEL).

For $t = 2$ (assuming the first observation corresponds to time period 1) you have,

$$\Delta y_{i2} = \phi \Delta y_{i1} + \nu_{i2}$$

Using y_{i1} as an instrument is not a good idea since $\mathrm{Cov}\left(\epsilon_{i1}, \nu_{i2}\right) \neq 0$. Therefore, since it is not possible to form a moment restriction, you discard this observation.

For $t = 3$ you have,

$$\Delta y_{i3} = \phi \Delta y_{i2} + \nu_{i3}$$

Clearly, you have every reason to suspect that $\mathrm{Cov}\left(\epsilon_{i1}, \nu_{i3}\right) = 0$. This condition forms one restriction.

For $t = 4$, both $\mathrm{Cov}\left(\epsilon_{i1}, \nu_{i4}\right) = 0$ and $\mathrm{Cov}\left(\epsilon_{i2}, \nu_{i4}\right) = 0$ must hold.

Proceeding in that fashion, you have the following matrix of instruments,

$$\mathbf{Z}_i = \begin{pmatrix} y_{i1} & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & y_{i1} & y_{i2} & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & y_{i1} & y_{i2} & y_{i3} & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & & & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & y_{i1} & \cdots & y_{i(T-2)} \end{pmatrix}$$

Using the instrument matrix, you form the weighting matrix $\mathbf{A_N}$ as

$$\mathbf{A_N} = \left(\frac{1}{N}\sum_i^N \mathbf{Z}_i' \mathbf{H}_i \mathbf{Z}_i\right)^{-1}$$

The initial weighting matrix is

$$\mathbf{H}_i = \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & & & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 \end{pmatrix}$$

Note that the maximum size of the $\mathbf{H}_i$ matrix is T–2. The origins of the initial weighting matrix are the expected error covariances. Notice that on the diagonals,

$$E\left(v_{it}v_{it}\right) = E\left(\epsilon_{it}^2 - 2\epsilon_{it}\epsilon_{i(t-1)} + \epsilon_{i(t-1)}^2\right) = 2\sigma_\epsilon^2$$

and off diagonals,

$$E\left(v_{it}v_{i(t-1)}\right) = E\left(\epsilon_{it}\epsilon_{i(t-1)} - \epsilon_{it}\epsilon_{i(t-2)} - \epsilon_{i(t-1)}\epsilon_{i(t-1)} + \epsilon_{i(t-1)}\epsilon_{i(t-2)}\right) = -\sigma_\epsilon^2$$

If you let the vector of lagged differences (in the series y_{it}) be denoted as Δy_{i-} and the dependent variable as Δy_i, then the optimal GMM estimator is

$$\phi = \left[\left(\sum_i \Delta y_{i-}^{'}\mathbf{Z}_i\right)\mathbf{A_N}\left(\sum_i \mathbf{Z}_i^{'}\Delta y_{i-}\right)\right]^{-1}\left(\sum_i \Delta y_{i-}^{'}\mathbf{Z}_i\right)\mathbf{A_N}\left(\sum_i \mathbf{Z}_i^{'}\Delta y_i\right)$$

Using the estimate, $\hat{\phi}$, you can obtain estimates of the errors, $\hat{\epsilon}$, or the differences, $\hat{v}$. From the errors, the variance is calculated as,

$$\sigma^2 = \frac{\hat{\epsilon}^{'}\hat{\epsilon}}{M-1}$$

where $M = \sum_{i=1}^{N} T_i$ is the total number of observations. With differenced equations, since we lose the first two observations, $M = \sum_{i=1}^{N}\left(T_i - 2\right)$.

Furthermore, you can calculate the variance of the parameter as,

$$\sigma^2\left[\left(\Sigma_i \Delta y_{i-}^{'}\mathbf{Z}_i\right)\mathbf{A_N}\left(\sum_i \mathbf{Z}_i^{'}\Delta y_{i-}\right)\right]^{-1}$$

Alternatively, you can view the initial estimate of the ϕ as a first step. That is, by using $\hat{\phi}$, you can improve the estimate of the weight matrix, $\mathbf{A_N}$.

Instead of imposing the structure of the weighting, you form the $\mathbf{H}_i$ matrix through the following:

$$\mathbf{H}_i = \hat{\mathbf{v}}_i\hat{\mathbf{v}}_i^{'}$$

You then complete the calculation as previously shown. The PROC PANEL option GMM2 specifies this estimation.

The case of multiple right-hand-side variables illustrates more clearly the power of Arellano and Bond (1991); Arellano and Bover (1995).

Considering the general case you have:

$$y_{it} = \sum_{l=1}^{maxlag}\phi_l y_{i(t-l)} + \boldsymbol{\beta}\mathbf{X}_i + \gamma_i + \alpha_t + \epsilon_{it}$$

It is clear that lags of the dependent variable are both not exogenous and correlated to the fixed effects. However, the independent variables can fall into one of several categories. An independent variable can be

correlated[1] and exogenous, uncorrelated and exogenous, correlated and predetermined, and uncorrelated and predetermined. The category in which an independent variable is found influences when or whether it becomes a suitable instrument. Note, however, that neither PROC PANEL nor Arellano and Bond require that a regressor be an instrument or that an instrument be a regressor.

First, suppose that the variables are all correlated with the individual effects γ_i. Consider the question of exogenous or predetermined. An exogenous variable is not correlated with the error term $\epsilon_{it} - \epsilon_{i,t-1}$ in the differenced equations. Therefore, all observations (on the exogenous variable) become valid instruments at all time periods. If the model has only one instrument and it happens to be exogenous, then the optimal instrument matrix looks like,

$$\mathbf{Z}_i = \begin{pmatrix} x_{i1} \cdots x_{iT} & 0 & 0 & 0 & 0 \\ 0 & x_{i1} \cdots x_{iT} & 0 & 0 & 0 \\ 0 & 0 & x_{i1} \cdots x_{iT} & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & x_{i1} \cdots x_{iT} \end{pmatrix}$$

The situation for the predetermined variables becomes a little more difficult. A predetermined variable is one whose future realizations can be correlated to current shocks in the dependent variable. With such an understanding, it is admissible to allow all current and lagged realizations as instruments. In other words you have,

$$\mathbf{Z}_i = \begin{pmatrix} x_{i1} & 0 & 0 & 0 & 0 \\ 0 & x_{i1}x_{i2} & 0 & 0 & 0 \\ 0 & 0 & x_{i1} \cdots x_{i3} & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & x_{i1} \cdots x_{i(T-1)} \end{pmatrix}$$

When the data contain a mix of endogenous, exogenous, and predetermined variables, the instrument matrix is formed by combining the three. For example, the third observation would have one observation on the dependent variable as an instrument, three observations on the predetermined variables as instruments, and all observations on the exogenous variables.

Now consider some variables, denoted as x_{1it}, that are not correlated with the individual effects γ_i. There is yet another set of moment restrictions that can be used. An uncorrelated variable means that the variable's level is not affected by the individual specific effect. You write the preceding general model as

$$y_{it} = \sum_{l=1}^{maxlag} \phi_l y_{i(t-l)} + \Sigma_{k=1}^{K} \beta_k x_{itk} + \alpha_t + \mu_{it}$$

where $\mu_{it} = \gamma_i + \epsilon_{it}$.

Because the variables are uncorrelated with γ_i and thus uncorrelated with the error term μ_{it} in the level equations, you can use the difference and level equations to perform a system estimation. That is, the uncorrelated variables imply moment restrictions on the level equations. Given the previously used restrictions for the equations in first differences, there are T extra restrictions. For predetermined variables, Arellano

[1] In this section, "correlated" means correlated with the individual effects and "uncorrelated" means uncorrelated with the individual effects.

and Bond (1991) use the extra restrictions $E\left(\mu_{i2}x^{p}_{1i1}\right) = 0$ and $E\left(\mu_{it}x^{p}_{1it}\right) = 0$ for $t = 2, \ldots, T$. The instrument matrix becomes

$$\mathbf{Z}^{*}_{i} = \begin{pmatrix} \mathbf{Z}_i & 0 & 0 & 0 & \cdots & 0 \\ 0 & x^{p}_{1i1} & x^{p}_{1i2} & 0 & \cdots & 0 \\ 0 & 0 & 0 & x^{p}_{1i3} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & x^{p}_{1iT} \end{pmatrix}$$

For exogenous variables x^{e}_{1it} Arellano and Bond (1991) use $E\left(T^{-1}\sum_{s=1}^{T}\mu_{is}x^{e}_{1it}\right) = 0$. PROC PANEL uses the same ones as the predetermined variables—that is, $E\left(\mu_{i2}x^{e}_{1i1}\right) = 0$ and $E\left(\mu_{it}x^{e}_{1it}\right) = 0$ for $t = 2, \ldots, T$. If you denote the new instrument matrix by using the full complement of instruments available by an asterisk and if both x^{p} and x^{e} are uncorrelated, then you have

$$\mathbf{Z}^{*}_{i} = \begin{pmatrix} \mathbf{Z}_i & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & x^{p}_{i1} & x^{e}_{i1} & x^{p}_{i2} & x^{e}_{i2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & x^{p}_{i3} & x^{e}_{i3} & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & x^{p}_{iT} & x^{e}_{iT} \end{pmatrix}$$

When the lagged dependent variable is included as the explanatory variable (as in the dynamic panel data models), Blundell and Bond (1998) suggest the system GMM to use $T - 2$ extra-moment restrictions, which use the lagged differences as the instruments for the level:

$$E\left(\mu_{it}\Delta y_{i,t-1}\right) = 0 \qquad \text{for } t = 3, \ldots, T$$

This additional set of moment conditions are required by DEPVAR(DIFF) option. The corresponding instrument matrix is

$$\mathbf{Z}^{y}_{li} = \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 \\ 0 & \Delta y_{i2} & 0 & \cdots & 0 \\ 0 & 0 & \Delta y_{i3} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \Delta y_{i(T-1)} \end{pmatrix}$$

Blundell and Bond (1998) argue that the system GMM that uses these extra conditions significantly increases the efficiency of the estimator, especially under strong serial correlation in the dependent variables.[2]

Except for those GMM-type instruments, PROC PANEL can also handle standard instruments by using the lists that you specify in the LEVELEQ= and DIFFEQ= options. Denote l_{it} and d_{it} as the standard instruments that are specified for the level equation and differenced equation, respectively. The additional moment restrictions are $E\left(\mu_{it}l_{it}\right) = 0$ for $t = 1, \ldots, T$ for level equations and $E\left(\Delta\epsilon_{it}d_{it}\right) = 0$ for $t = 2, \ldots, T$ for differenced equations. The instrument matrix for the level and differenced equations are $\mathbf{Z}_{li}$

[2]This happens when $\phi \to 1$ or as $\sigma^2_{\gamma}/\sigma^2_{\epsilon} \to \infty$. In this case, the lagged dependent variables $y_{i(t-l)}$ become weak instruments for the differenced variables Δy_{it}.

and $\mathbf{Z}_{di}$, respectively, as follows:

$$\mathbf{Z}_{li} = \begin{pmatrix} l_{i1} & 0 & 0 & 0 & 0 \\ 0 & l_{i2} & 0 & 0 & 0 \\ 0 & 0 & l_{i3} & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & l_{iT} \end{pmatrix}$$

$$\mathbf{Z}_{di} = \begin{pmatrix} d_{i1} & 0 & 0 & 0 & 0 \\ 0 & d_{i2} & 0 & 0 & 0 \\ 0 & 0 & d_{i3} & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & d_{iT} \end{pmatrix}$$

To put the differenced and level equations together, for the system GMM estimator, the instrument matrix can be constructed as

$$\mathbf{Z}_i = \begin{pmatrix} \mathbf{Z}_{di} & 0 & 0 & 0 & 0 \\ 0 & \mathbf{Z}^{e}_{li} & \mathbf{Z}^{p}_{li} & \mathbf{Z}_{li} & \mathbf{Z}^{y}_{li} \end{pmatrix}$$

where $\mathbf{Z}^{e}_{li}$ and $\mathbf{Z}^{p}_{li}$ correspond to the exogenous and predetermined uncorrelated variables, respectively.

The formation of the initial weighting matrix becomes somewhat problematic. If you denote the new weighting matrix with an asterisk, then you can write

$$\mathbf{A}^{*}_{\mathbf{N}} = \left(\frac{1}{N} \sum_{i}^{N} \mathbf{Z}^{*'}_{i} \mathbf{H}^{*}_{i} \mathbf{Z}^{*}_{i} \right)^{-1}$$

where

$$\mathbf{H}^{*}_{i} = \begin{pmatrix} \mathbf{H}_i & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

To finish, you write out the two equations (or two stages) that are estimated,

$$\Delta y_{it} = \boldsymbol{\beta}^{*} \Delta \mathbf{S}_{it} + \alpha_t - \alpha_{t-1} + \nu_{it} \qquad y_{it} = \boldsymbol{\beta}^{*} \mathbf{S}_{it} + \gamma_i + \alpha_t + \epsilon_{it}$$

where $\mathbf{S}_{it}$ is the matrix of all explanatory variables—lagged endogenous, exogenous, and predetermined.

Let $\mathbf{y}^{*}_{it}$ be given by

$$\mathbf{y}^{*}_{it} = \begin{pmatrix} \Delta y_{it} \\ y_{it} \end{pmatrix} \quad \boldsymbol{\beta}^{*} = \begin{pmatrix} \boldsymbol{\phi} & \boldsymbol{\beta} \end{pmatrix} \quad \mathbf{S}^{*}_{it} = \begin{pmatrix} \Delta \mathbf{S}_{it} \\ \mathbf{S}_{it} \end{pmatrix} \quad \mathbf{e}^{*}_{i} = \begin{pmatrix} \boldsymbol{\nu}_i \\ \boldsymbol{\mu}_i = \boldsymbol{\epsilon}_i + \gamma_i \end{pmatrix}$$

Using the preceding information, you can get the one-step GMM estimator,

$$\hat{\boldsymbol{\beta}}^{*}_{\mathbf{1}} = \left[\left(\sum_i \mathbf{S}^{*'}_{i} \mathbf{Z}^{*}_{i} \right) \mathbf{A}^{*}_{\mathbf{N}} \left(\sum_i \mathbf{Z}^{*'}_{i} \mathbf{S}^{*}_{i} \right) \right]^{-1} \left(\sum_i \mathbf{S}^{*'}_{i} \mathbf{Z}^{*}_{i} \right) \mathbf{A}^{*}_{\mathbf{N}} \left(\sum_i \mathbf{Z}^{*'}_{i} \mathbf{y}^{*}_{i} \right)$$

If the GMM2 or ITGMM option is not specified in the MODEL statement, estimation terminates here. If it terminates, you can obtain the following information.

Variance of the error term comes from the second-stage (level) equations—that is,

$$\sigma^2 = \frac{\hat{\boldsymbol{\mu}}'\hat{\boldsymbol{\mu}}}{M-p} = \frac{\left(y_{it} - \hat{\boldsymbol{\beta}}_1^* \mathbf{S}_{it}\right)' \left(y_{it} - \hat{\boldsymbol{\beta}}_1^* \mathbf{S}_{it}\right)}{M-p}$$

where p is the number of regressors and M is the number of observations as defined before.

The variance covariance matrix can be obtained from

$$\left[\left(\sum_i \mathbf{S}_i^{*'} \mathbf{Z}_i^*\right) \mathbf{A}_\mathbf{N}^* \left(\sum_i \mathbf{Z}_i^{*'} \mathbf{S}_i^*\right)\right]^{-1} \sigma^2$$

Alternatively, you can obtain a robust estimate of the variance covariance matrix by specifying the ROBUST option in the MODEL statement. Without further reestimation of the model, the $\mathbf{H}_i^*$ matrix is recalculated as

$$\mathbf{H}_{i,2}^* = \begin{pmatrix} \hat{\boldsymbol{\nu}}_i \hat{\boldsymbol{\nu}}_i' & 0 \\ 0 & \hat{\boldsymbol{\mu}}_i \hat{\boldsymbol{\mu}}_i' \end{pmatrix}$$

And the weighting matrix becomes

$$\mathbf{A}_\mathbf{N}^* \left(\hat{\boldsymbol{\beta}}_1^*\right) = \left(\frac{1}{N} \sum_i^N \mathbf{Z}_i^{*'} \mathbf{H}_{i,2}^* \mathbf{Z}_i^*\right)^{-1}$$

Using the preceding information, you construct the robust covariance matrix from the following.

Let $\mathbf{G}$ denote a temporary matrix,

$$\mathbf{G} = \left[\left(\sum_i \mathbf{S}_i^{*'} \mathbf{Z}_i^*\right) \mathbf{A}_\mathbf{N}^* \left(\sum_i \mathbf{Z}_i^{*'} \mathbf{S}_i^*\right)\right]^{-1} \left(\sum_i \mathbf{S}_i^{*'} \mathbf{Z}_i^*\right) \mathbf{A}_\mathbf{N}^*$$

The robust covariance estimate of $\hat{\boldsymbol{\beta}}_1^*$ is

$$\mathbf{V}^r \left(\hat{\boldsymbol{\beta}}_1^*\right) = \mathbf{G} \mathbf{A}_\mathbf{N}^{*-1} \left(\hat{\boldsymbol{\beta}}_1^*\right) \mathbf{G}'$$

Alternatively, you can use the new weighting matrix to form an updated estimate of the regression parameters, as requested by the GMM2 option in the MODEL statement. In short,

$$\hat{\boldsymbol{\beta}}_2^* = \left[\left(\sum_i \mathbf{S}_i^{*'} \mathbf{Z}_i^*\right) \mathbf{A}_\mathbf{N}^* \left(\hat{\boldsymbol{\beta}}_1^*\right) \left(\sum_i \mathbf{Z}_i^{*'} \mathbf{S}_i^*\right)\right]^{-1} \left(\sum_i \mathbf{S}_i^{*'} \mathbf{Z}_i^*\right) \mathbf{A}_\mathbf{N}^* \left(\hat{\boldsymbol{\beta}}_1^*\right) \left(\sum_i \mathbf{Z}_i^{*'} \mathbf{y}_i^*\right)$$

The covariance estimate of the two-step $\hat{\boldsymbol{\beta}}_2^*$ becomes

$$V \left(\hat{\boldsymbol{\beta}}_2^*\right) = \left[\left(\sum_i \mathbf{S}_i^{*'} \mathbf{Z}_i^*\right) \mathbf{A}_\mathbf{N}^* \left(\hat{\boldsymbol{\beta}}_1^*\right) \left(\sum_i \mathbf{Z}_i^{*'} \mathbf{S}_i^*\right)\right]^{-1}$$

Similarly, you construct the robust covariance matrix from the following.

Let $\mathbf{G}_2$ denote a temporary matrix,

$$\mathbf{G}_2 = \left[\left(\sum_i \mathbf{S}_i^{*'}\mathbf{Z}_i^*\right)\mathbf{A}_\mathbf{N}^*\left(\hat{\boldsymbol{\beta}}_1^*\right)\left(\sum_i \mathbf{Z}_i^{*'}\mathbf{S}_i^*\right)\right]^{-1}\left(\sum_i \mathbf{S}_i^{*'}\mathbf{Z}_i^*\right)\mathbf{A}_\mathbf{N}^*\left(\hat{\boldsymbol{\beta}}_1^*\right)$$

The robust covariance estimate of $\hat{\boldsymbol{\beta}}_2^*$ is

$$\mathbf{V}^r\left(\hat{\boldsymbol{\beta}}_2^*\right) = \mathbf{G}_2\mathbf{A}_\mathbf{N}^{*-1}\left(\hat{\boldsymbol{\beta}}_2^*\right)\mathbf{G}_2'$$

According to Arellano and Bond (1991), Blundell and Bond (1998), and many others, two-step standard errors are unreliable. Therefore, researchers often base inference on two-step parameter estimates and one-step standard errors. Windmeijer (2005) derives a small-sample bias-corrected variance that uses the first-order Taylor series approximation of the two-step GMM estimator $\hat{\boldsymbol{\beta}}_2^*$ around the true value $\boldsymbol{\beta}^*$ as a function of the one-step GMM estimator $\hat{\boldsymbol{\beta}}_1^*$,

$$\begin{aligned}
\hat{\boldsymbol{\beta}}_2^* - \boldsymbol{\beta}^* &= \left[\left(\textstyle\sum_i \mathbf{S}_i^{*'}\mathbf{Z}_i^*\right)\mathbf{A}_\mathbf{N}^*\left(\hat{\boldsymbol{\beta}}_1^*\right)\left(\textstyle\sum_i \mathbf{Z}_i^{*'}\mathbf{S}_i^*\right)\right]^{-1}\left(\textstyle\sum_i \mathbf{S}_i^{*'}\mathbf{Z}_i^*\right)\mathbf{A}_\mathbf{N}^*\left(\hat{\boldsymbol{\beta}}_1^*\right)\left(\textstyle\sum_i \mathbf{Z}_i^{*'}\mathbf{e}_i^*\right) \\
&= \left[\left(\textstyle\sum_i \mathbf{S}_i^{*'}\mathbf{Z}_i^*\right)\mathbf{A}_\mathbf{N}^*(\boldsymbol{\beta}^*)\left(\textstyle\sum_i \mathbf{Z}_i^{*'}\mathbf{S}_i^*\right)\right]^{-1}\left(\textstyle\sum_i \mathbf{S}_i^{*'}\mathbf{Z}_i^*\right)\mathbf{A}_\mathbf{N}^*(\boldsymbol{\beta}^*)\left(\textstyle\sum_i \mathbf{Z}_i^{*'}\mathbf{e}_i^*\right) \\
&\quad + D_{\boldsymbol{\beta}^*,\mathbf{A}_\mathbf{N}^*(\boldsymbol{\beta}^*)}\left(\hat{\boldsymbol{\beta}}_1^* - \boldsymbol{\beta}^*\right) + O_p\left(N^{-1}\right)
\end{aligned}$$

where $D_{\boldsymbol{\beta}^*,\mathbf{A}_\mathbf{N}^*(\boldsymbol{\beta}^*)}$ is the first derivative of $\hat{\boldsymbol{\beta}}_2^* - \boldsymbol{\beta}^*$ with regard to $\boldsymbol{\beta}'$ evaluated at the true value $\boldsymbol{\beta}^*$. The kth column of D is

$$\begin{aligned}
&\{D_{\boldsymbol{\beta}^*,\mathbf{A}_\mathbf{N}^*(\boldsymbol{\beta}^*)}\}_k = \\
&\left[\left(\textstyle\sum_i \mathbf{S}_i^{*'}\mathbf{Z}_i^*\right)\mathbf{A}_\mathbf{N}^*(\boldsymbol{\beta}^*)\left(\textstyle\sum_i \mathbf{Z}_i^{*'}\mathbf{S}_i^*\right)\right]^{-1}\left(\textstyle\sum_i \mathbf{S}_i^{*'}\mathbf{Z}_i^*\right)\mathbf{A}_\mathbf{N}^*(\boldsymbol{\beta}^*)\frac{\partial\mathbf{A}_\mathbf{N}^{*-1}(\boldsymbol{\beta})}{\partial\beta_k}|_{\boldsymbol{\beta}^*}\mathbf{A}_\mathbf{N}^*(\boldsymbol{\beta}^*)\left(\textstyle\sum_i \mathbf{Z}_i^{*'}\mathbf{S}_i^*\right) \\
&\times\left[\left(\textstyle\sum_i \mathbf{S}_i^{*'}\mathbf{Z}_i^*\right)\mathbf{A}_\mathbf{N}^*(\boldsymbol{\beta}^*)\left(\textstyle\sum_i \mathbf{Z}_i^{*'}\mathbf{S}_i^*\right)\right]^{-1}\left(\textstyle\sum_i \mathbf{S}_i^{*'}\mathbf{Z}_i^*\right)\mathbf{A}_\mathbf{N}^*(\boldsymbol{\beta}^*)\left(\textstyle\sum_i \mathbf{Z}_i^{*'}\mathbf{e}_i^*\right) \\
&-\left[\left(\textstyle\sum_i \mathbf{S}_i^{*'}\mathbf{Z}_i^*\right)\mathbf{A}_\mathbf{N}^*(\boldsymbol{\beta}^*)\left(\textstyle\sum_i \mathbf{Z}_i^{*'}\mathbf{S}_i^*\right)\right]^{-1}\left(\textstyle\sum_i \mathbf{S}_i^{*'}\mathbf{Z}_i^*\right)\mathbf{A}_\mathbf{N}^*(\boldsymbol{\beta}^*)\frac{\partial\mathbf{A}_\mathbf{N}^{*-1}(\boldsymbol{\beta})}{\partial\beta_k}|_{\boldsymbol{\beta}^*}\mathbf{A}_\mathbf{N}^*(\boldsymbol{\beta}^*)\left(\textstyle\sum_i \mathbf{Z}_i^{*'}\mathbf{e}_i^*\right)
\end{aligned}$$

Because $\boldsymbol{\beta}^*$, $\mathbf{A}_\mathbf{N}^*(\boldsymbol{\beta}^*)$, and $\frac{\partial\mathbf{A}_\mathbf{N}^{*-1}(\boldsymbol{\beta})}{\partial\beta_k}|_{\boldsymbol{\beta}^*}$ are not feasible, you can replace them with their estimators, $\hat{\boldsymbol{\beta}}_2^*$, $\mathbf{A}_\mathbf{N}^*\left(\hat{\boldsymbol{\beta}}_1^*\right)$, and $\frac{\partial\mathbf{A}_\mathbf{N}^{*-1}(\boldsymbol{\beta})}{\partial\beta_k}|_{\hat{\boldsymbol{\beta}}_1^*}$, respectively. Denote $\hat{\mathbf{e}}_{i,2}^*$ as the second-stage error term by

$$\left[\left(\sum_i \mathbf{S}_i^{*'}\mathbf{Z}_i^*\right)\mathbf{A}_\mathbf{N}^*\left(\hat{\boldsymbol{\beta}}_1^*\right)\left(\sum_i \mathbf{Z}_i^{*'}\mathbf{S}_i^*\right)\right]^{-1}\left(\sum_i \mathbf{S}_i^{*'}\mathbf{Z}_i^*\right)\mathbf{A}_\mathbf{N}^*\left(\hat{\boldsymbol{\beta}}_1^*\right)\left(\sum_i \mathbf{Z}_i^{*'}\hat{\mathbf{e}}_{i,2}^*\right) = 0$$

and

$$\frac{\partial\mathbf{A}_\mathbf{N}^{*-1}(\boldsymbol{\beta})}{\partial\beta_k}|_{\boldsymbol{\beta}^*} = -\frac{1}{N}\sum_i \mathbf{Z}_i^{*'}\begin{pmatrix} \Delta\mathbf{S}_{i,k}\boldsymbol{\nu}_i' + \boldsymbol{\nu}_i\Delta\mathbf{S}_{i,k}' & 0 \\ 0 & \mathbf{S}_{i,k}\boldsymbol{\mu}_i' + \boldsymbol{\mu}_i\mathbf{S}_{i,k}' \end{pmatrix}\mathbf{Z}_i^*$$

The first part vanishes and leaves

$$\begin{aligned}
\{D_{\hat{\boldsymbol{\beta}}_2^*,\mathbf{A}_\mathbf{N}^*\left(\hat{\boldsymbol{\beta}}_1^*\right)}\}_k &= \tfrac{1}{N}\left[\left(\textstyle\sum_i \mathbf{S}_i^{*'}\mathbf{Z}_i^*\right)\mathbf{A}_\mathbf{N}^*\left(\hat{\boldsymbol{\beta}}_1^*\right)\left(\textstyle\sum_i \mathbf{Z}_i^{*'}\mathbf{S}_i^*\right)\right]^{-1}\left(\textstyle\sum_i \mathbf{S}_i^{*'}\mathbf{Z}_i^*\right)\mathbf{A}_\mathbf{N}^*\left(\hat{\boldsymbol{\beta}}_1^*\right) \\
&\quad\left(\textstyle\sum_i \mathbf{Z}_i^{*'}\begin{pmatrix} \Delta\mathbf{S}_{i,k}\hat{\boldsymbol{\nu}}_{i,1}' + \hat{\boldsymbol{\nu}}_{i,1}\Delta\mathbf{S}_{i,k}' & 0 \\ 0 & \mathbf{S}_{i,k}\hat{\boldsymbol{\mu}}_{i,1}' + \hat{\boldsymbol{\mu}}_{i,1}\mathbf{S}_{i,k}' \end{pmatrix}\mathbf{Z}_i^*\right)\mathbf{A}_\mathbf{N}^*\left(\hat{\boldsymbol{\beta}}_1^*\right)\left(\textstyle\sum_i \mathbf{Z}_i^{*'}\hat{\mathbf{e}}_{i,2}^*\right)
\end{aligned}$$

Plugging these into the Taylor expansion series yields

$$V^c\left(\hat{\boldsymbol{\beta}}_2^*\right) = V\left(\hat{\boldsymbol{\beta}}_2^*\right) + D_{\hat{\boldsymbol{\beta}}_2^*,\mathbf{A}_\mathbf{N}^*\left(\hat{\boldsymbol{\beta}}_1^*\right)} V\left(\hat{\boldsymbol{\beta}}_2^*\right) + V\left(\hat{\boldsymbol{\beta}}_2^*\right) D'_{\hat{\boldsymbol{\beta}}_2^*,\mathbf{A}_\mathbf{N}^*\left(\hat{\boldsymbol{\beta}}_1^*\right)} + D_{\hat{\boldsymbol{\beta}}_2^*,\mathbf{A}_\mathbf{N}^*\left(\hat{\boldsymbol{\beta}}_1^*\right)} \mathbf{V}^r\left(\hat{\boldsymbol{\beta}}_1^*\right) D'_{\hat{\boldsymbol{\beta}}_2^*,\mathbf{A}_\mathbf{N}^*\left(\hat{\boldsymbol{\beta}}_1^*\right)}$$

As a final note, it possible to iterate more than twice by specifying the ITGMM option. At each iteration, the parameter estimates and its variance-covariance matrix (standard or robust) can be constructed as the one-step and/or two-step GMM estimators. Such a multiple iteration should result in a more stable estimate of the covariance estimate. PROC PANEL allows two convergence criteria. Convergence can occur in the parameter estimates or in the weighting matrices. Let $\mathbf{A}_{\mathbf{N,k+1}}^*$ denote the robust covariance matrix from iteration k, which is used as the weighting matrix in iteration $k+1$. Iterate until

$$\max_{i,j \le \dim\left(\mathbf{A}_{\mathbf{N,k}}^*\right)} \frac{\left|\mathbf{A}_{\mathbf{N,k+1}}^*(i,j) - \mathbf{A}_{\mathbf{N,k}}^*(i,j)\right|}{\left|\mathbf{A}_{\mathbf{N,k}}^*(i,j)\right|} \le \text{ATOL}$$

or

$$\max_{i \le \dim\left(\boldsymbol{\beta}_k^*\right)} \frac{\left|\boldsymbol{\beta}_{k+1}^*(i) - \boldsymbol{\beta}_k^*(i)\right|}{\left|\boldsymbol{\beta}_k^*(i)\right|} \le \text{BTOL}$$

where ATOL is the tolerance for convergence in the weighting matrix and BTOL is the tolerance for convergence in the parameter estimate matrix. The default convergence criteria is BTOL = 1E–8 for PROC PANEL.

Specification Testing For Dynamic Panel

Specification tests under GMM in PROC PANEL generally follow Arellano and Bond (1991). The first test available is a Sargan/Hansen test of over-identification. The test for a one-step estimation is constructed as

$$\left(\sum_i \eta_i' \mathbf{Z}_\mathbf{i}^*\right) \mathbf{A}_\mathbf{N}^* \left(\sum_i \mathbf{Z}_\mathbf{i}^{*'} \eta_i\right) \sigma^2$$

where η_i is the stacked error term (of the differenced equation and level equation).

When the robust weighting matrix is used, the test statistic is computed as

$$\left(\sum_i \eta_i' \mathbf{Z}_\mathbf{i}^*\right) \mathbf{A}_{\mathbf{N,2}}^* \left(\sum_i \mathbf{Z}_\mathbf{i}^{*'} \eta_i\right)$$

This definition of the Sargan test is used for all iterated estimations. The Sargan test is distributed as a χ^2 with degrees of freedom equal to the number of moment conditions minus the number of parameters.

In addition to the Sargan test, PROC PANEL tests for autocorrelation in the residuals. These tests are distributed as standard normal. PROC PANEL tests the hypothesis that the autocorrelation of the lth lag is significant.

Define $\boldsymbol{\omega}_l$ as the lag of the differenced error, with zero padding for the missing values generated. Symbolically,

$$\boldsymbol{\omega}_{l,i} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \boldsymbol{v}_{i,2} \\ \vdots \\ \boldsymbol{v}_{i,T-1-l} \end{pmatrix}$$

You define the constant k_0 as

$$k_0(l) = \sum_i \boldsymbol{\omega}'_{l,i} \boldsymbol{\nu}_i$$

You next define the constant k_1 as

$$k_1(l) = \sum_i \boldsymbol{\omega}'_{l,i} \mathbf{H}_i \boldsymbol{\omega}_{l,i}$$

Note that the choice of $\mathbf{H}_i$ is dependent on the stage of estimation. If the estimation is first stage, then you would use the matrix with twos along the main diagonal, and minus ones along the primary subdiagonals. In a robust estimation or multi-step estimation, this matrix would be formed from the outer product of the residuals (from the previous step).

Define the constant k_2 as

$$k_2(l) = -2 \left(\sum_i \boldsymbol{\omega}'_{l,i} \Delta \mathbf{S}_i \right) \mathbf{G} \left(\sum_i \Delta \mathbf{S}'_i \mathbf{Z}_i \right) \mathbf{A}_{\mathbf{N},\mathbf{k}} \left(\sum_i \mathbf{Z}'_i \mathbf{H}_i \boldsymbol{\omega}_{l,i} \right)$$

The matrix $\mathbf{G}$ is defined as

$$\mathbf{G} = \left[\left(\sum_i \Delta \mathbf{S}^{*\prime}_i \mathbf{Z}^*_i \right) \mathbf{A}^*_{\mathbf{N},\mathbf{k}} \left(\sum_i \mathbf{Z}^{*\prime}_i \Delta \mathbf{S}^*_i \right) \right]^{-1}$$

The constant k_3 is defined as

$$k_3(l) = \left(\sum_i \boldsymbol{\omega}'_{l,i} \Delta \mathbf{S}_i \right) V(\boldsymbol{\beta}^*) \left(\sum_i \Delta \mathbf{S}'_i \boldsymbol{\omega}_{l,i} \right)$$

Using the four quantities, the test for autoregressive structure in the differenced residual is

$$m(l) = \frac{k_0(l)}{\sqrt{k_1(l) + k_2(l) + k_3(l)}}$$

The m statistic is distributed as a normal random variable with mean zero and standard deviation of one.

Instrument Choice

Arellano and Bond's technique is a very useful method for dealing with any autoregressive characteristics in the data. However, there is one caveat to consider. Too many instruments bias the estimator to the within estimate. Furthermore, many instruments make this technique not scalable. The weighting matrix becomes very large, so every operation that involves it becomes more computationally intensive. The PANEL procedure enables you to specify a bandwidth for instrument selection. For example, specifying MAXBAND=10 means that at most there will be ten time observations for each variable that enters as an instrument. The default is to follow the Arellano-Bond methodology.

In specifying a maximum bandwidth, you can also specify the selection of the time observations. There are three possibilities: leading, trailing (default), and centered. The exact consequence of choosing any of those possibilities depends on the variable type (correlated, exogenous, or predetermined) and the time period of the current observation.

If the MAXBAND option is specified, then the following is true under any selection criterion (let t be the time subscript for the current observation). The first observation for the endogenous variable (as instrument) is $\max(t - \text{MAXBAND}, 1)$ and the last instrument is $t - 2$. The first observation for a predetermined variable is $\max(t - \text{MAXBAND}, 1)$ and the last is $t - 1$. The first and last observation for an exogenous variable is given in the following list:

- *Trailing*: If $t < \text{MAXBAND}$, then the first instrument is for the first time period and the last observation is MAXBAND. Otherwise, if $t \geq \text{MAXBAND}$, then the first observation is $t - \text{MAXBAND} + 1$ and the last instrument to enter is t.

- *Centered*: If $t \leq \frac{\text{MAXBAND}}{2}$, then the first observation is the first time period and the last observation is MAXBAND. If $t > T - \frac{\text{MAXBAND}}{2}$, then the first instrument included is $T - \text{MAXBAND} + 1$ and the last observation is T. If $\frac{\text{MAXBAND}}{2} < t \leq T - \frac{\text{MAXBAND}}{2}$, then the first included instrument is $t - \frac{\text{MAXBAND}}{2} + 1$ and the last observation is $t + \frac{\text{MAXBAND}}{2}$. If the MAXBAND value is an odd number, the procedure decrements by one.

- *Leading* : If $t > T - \text{MAXBAND}$, then the first instrument corresponds to time period $T - \text{MAXBAND} + 1$ and the last observation is T. Otherwise, if $t \leq T - \text{MAXBAND}$, then the first observation is t and the last observation is $t + \text{MAXBAND} + 1$.

The PANEL procedure enables you to include dummy variables to deal with the presence of time effects that are not captured by including the lagged dependent variable. The dummy variables directly affect the level equations. However, this implies that the difference of the dummy variable for time period t and $t - 1$ enters the difference equation. The first usable observation occurs at $t = 3$. If the level equation is not used in the estimation, then there is no way to identify the dummy variables. Selecting the TIME option gives the same result as that which would be obtained by creating dummy variables in the data set and using those in the regression.

The PANEL procedure gives you several options when it comes to missing values and unbalanced panel. By default, any time period for which there are missing values is skipped. The corresponding rows and columns of $\mathbf{H}$ matrices are zeroed, and the calculation is continued. Alternatively, you can elect to replace missing values and missing observations with zeros (ZERO), the overall mean of the series (OAM), the cross-sectional mean (CSM), or the time series mean (TSM).

Linear Hypothesis Testing

For a linear hypothesis of the form $\mathbf{R}\,\beta = \mathbf{r}$ where $\mathbf{R}$ is $J \times K$ and $\mathbf{r}$ is $J \times 1$, the F-statistic with $J, M - K$ degrees of freedom is computed as

$$(\mathbf{R}\beta - \mathbf{r})^{'}[\mathbf{R}\hat{\mathbf{V}}\mathbf{R}']^{-1}(\mathbf{R}\beta - \mathbf{r})$$

However, it is also possible to write the F statistic as

$$F = \frac{(\hat{\mathbf{u}}_{*}^{'}\hat{\mathbf{u}}_{*} - \hat{\mathbf{u}}^{'}\hat{\mathbf{u}})/J}{\hat{\mathbf{u}}^{'}\hat{\mathbf{u}}/(M - K)}$$

where

- $\hat{\mathbf{u}}_*$ is the residual vector from the restricted regression
- $\hat{\mathbf{u}}$ is the residual vector from the unrestricted regression
- J is the number of restrictions
- $(M - K)$ are the degrees of freedom, M is the number of observations, and K is the number of parameters in the model

The Wald, likelihood ratio (LR) and LaGrange multiplier (LM) tests are all related to the F test. You use this relationship of the F test to the likelihood ratio and LaGrange multiplier tests. The Wald test is calculated from its definition.

The Wald test statistic is:

$$W = (\mathbf{R}\beta - \mathbf{r})'[\mathbf{R}\hat{\mathbf{V}}\mathbf{R}']^{-1}(\mathbf{R}\beta - \mathbf{r})$$

The advantage of calculating Wald in this manner is that it enables you to substitute a heteroscedasticity-corrected covariance matrix for the matrix $\mathbf{V}$. PROC PANEL makes such a substitution if you request the HCCME option in the MODEL statement.

The likelihood ratio is:

$$LR = M \ln\left[1 + \frac{1}{M-K} JF\right]$$

The LaGrange multiplier test statistic is:

$$LM = M\left[\frac{JF}{M - K + JF}\right]$$

where JF represents the number of restrictions multiplied by the result of the F test.

Note that only the Wald is changed when the HCCME option is selected. The LR and LM tests are unchanged.

The distribution of these test statistics is the χ^2 with degrees of freedom equal to the number of restrictions imposed (J). The three tests are asymptotically equivalent, but they have differing small sample properties. Greene (2000, p. 392) and Davidson and MacKinnon (1993, pp. 456–458) discuss the small sample properties of these statistics.

Heteroscedasticity-Corrected Covariance Matrices

The HCCME= option in the MODEL statement selects the type of heteroscedasticity-consistent covariance matrix. In the presence of heteroscedasticity, the covariance matrix has a complicated structure that can result in inefficiencies in the OLS estimates and biased estimates of the covariance matrix. The variances for cross-sectional and time dummy variables and the covariances with or between the dummy variables are not corrected for heteroscedasticity in the one-way and two-way models. Whether or not HCCME is specified, they are the same. For the two-way models, the variance and the covariances for the intercept are not corrected.[3]

[3]The dummy variables are removed by the within transformations, so their variances and covariances cannot be calculated the same way as the other regressors. They are recovered by the formulas listed in the sections "One-Way Fixed-Effects Model" on page 1833 and "Two-Way Fixed-Effects Model" on page 1835. The formulas assume homoscedasticity, so they do not apply when HCCME is specified. Therefore, standard errors, variances, and covariances are reported only when the HCCME option is ignored. HCCME standard errors for dummy variables and intercept can be calculated by the dummy variable approach with the pooled model.

Consider the simple linear model:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

This discussion parallels the discussion in Davidson and MacKinnon 1993, pp. 548–562. For panel data models, we apply HCCME on the transformed data($\tilde{y}$ and $\tilde{\mathbf{X}}$). In other words, we first remove the random or fixed effects through transforming/demean the data[4], then correct heteroscedasticity (also auto-correlation with HAC option) in the residual. The assumptions that make the linear regression best linear unbiased estimator (BLUE) are $E(\epsilon) = 0$ and $E(\boldsymbol{\epsilon}\boldsymbol{\epsilon}^{'}) = \Omega$, where Ω has the simple structure $\sigma^2\mathbf{I}$. Heteroscedasticity results in a general covariance structure, so that it is not possible to simplify Ω. The result is the following:

$$\tilde{\boldsymbol{\beta}} = (\mathbf{X}^{'}\mathbf{X})^{-1}\mathbf{X}^{'}\mathbf{y} = (\mathbf{X}^{'}\mathbf{X})^{-1}\mathbf{X}^{'}(\mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}) = \beta + (\mathbf{X}^{'}\mathbf{X})^{-1}\mathbf{X}^{'}\boldsymbol{\epsilon}$$

As long as the following is true, then you are assured that the OLS estimate is consistent and unbiased:

$$plim_{n\to\infty}\left(\frac{1}{n}\mathbf{X}^{'}\boldsymbol{\epsilon}\right) = 0$$

If the regressors are nonrandom, then it is possible to write the variance of the estimated $\boldsymbol{\beta}$ as the following:

$$\mathrm{Var}\left(\boldsymbol{\beta} - \tilde{\boldsymbol{\beta}}\right) = (\mathbf{X}^{'}\mathbf{X})^{-1}\mathbf{X}^{'}\Omega\mathbf{X}(\mathbf{X}^{'}\mathbf{X})^{-1}$$

The effect of structure in the covariance matrix can be ameliorated by using generalized least squares (GLS), provided that Ω^{-1} can be calculated. Using Ω^{-1}, you premultiply both sides of the regression equation,

$$L^{-1}\mathbf{y} = L^{-1}\mathbf{X}_{\boldsymbol{\epsilon}} + L^{-1}\boldsymbol{\epsilon}$$

where L denotes the Cholesky root of Ω. (that is, $\Omega = LL'$ with L lower triangular).

The resulting GLS $\boldsymbol{\beta}$ is

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^{'}\Omega^{-1}\mathbf{X})^{-1}\mathbf{X}^{'}\Omega^{-1}\mathbf{y}$$

Using the GLS $\boldsymbol{\beta}$, you can write

$$\begin{aligned}\hat{\boldsymbol{\beta}} &= (\mathbf{X}^{'}\Omega^{-1}\mathbf{X})^{-1}\mathbf{X}^{'}\Omega^{-1}\mathbf{y}\\ &= (\mathbf{X}^{'}\Omega^{-1}\mathbf{X})^{-1}\mathbf{X}^{'}(\Omega^{-1}\mathbf{X}_{\boldsymbol{\epsilon}} + \Omega^{-1}\boldsymbol{\epsilon})\\ &= \beta + (\mathbf{X}^{'}\Omega^{-1}\mathbf{X})^{-1}\mathbf{X}^{'}\Omega^{-1}\boldsymbol{\epsilon}\end{aligned}$$

The resulting variance expression for the GLS estimator is

$$\begin{aligned}\mathrm{Var}\left(\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}\right) &= (\mathbf{X}^{'}\Omega^{-1}\mathbf{X})^{-1}\mathbf{X}^{'}\Omega^{-1}\boldsymbol{\epsilon}\boldsymbol{\epsilon}'\Omega^{-1}\mathbf{X}(\mathbf{X}^{'}\Omega^{-1}\mathbf{X})^{-1}\\ &= (\mathbf{X}^{'}\Omega^{-1}\mathbf{X})^{-1}\mathbf{X}^{'}\Omega^{-1}\Omega\Omega^{-1}\mathbf{X}(\mathbf{X}^{'}\Omega^{-1}\mathbf{X})^{-1}\\ &= (\mathbf{X}^{'}\Omega^{-1}\mathbf{X})^{-1}\end{aligned}$$

[4]Please refer to "One-Way Fixed-Effects Model" on page 1833, "Two-Way Fixed-Effects Model" on page 1835, "One-Way Random-Effects Model" on page 1841, and "Two-Way Random-Effects Model" on page 1844 for details about transforming the data.

The difference in variance between the OLS estimator and the GLS estimator can be written as

$$(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\Omega\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1} - (\mathbf{X}'\Omega^{-1}\mathbf{X})^{-1}$$

By the Gauss-Markov theorem, the difference matrix must be positive definite under most circumstances (zero if OLS and GLS are the same, when the usual classical regression assumptions are met). Thus, OLS is not efficient under a general error structure. It is crucial to realize that OLS does not produce biased results. It would suffice if you had a method for estimating a consistent covariance matrix and you used the OLS $\boldsymbol{\beta}$. Estimation of the Ω matrix is certainly not simple. The matrix is square and has M^2 elements; unless some sort of structure is assumed, it becomes an impossible problem to solve. However, the heteroscedasticity can have quite a general structure. White (1980) shows that it is not necessary to have a consistent estimate of Ω. On the contrary, it suffices to calculate an estimate of the middle expression. That is, you need an estimate of:

$$\Lambda = \mathbf{X}'\Omega\mathbf{X}$$

This matrix, Λ, is easier to estimate because its dimension is K. PROC PANEL provides the following classical HCCME estimators for Λ:

The matrix is approximated by:

- HCCME=N0:

  $$\sigma^2\mathbf{X}'\mathbf{X}$$

  This is the simple OLS estimator. If you do not specify the HCCME= option, PROC PANEL defaults to this estimator.

- HCCME=0:

  $$\sum_{i=1}^{N}\sum_{t=1}^{T_i}\hat{\epsilon}_{it}^2\mathbf{x}_{it}\mathbf{x}_{it}^{'}$$

  where N is the number of cross sections and T_i is the number of observations in ith cross section. The $\mathbf{x}_{it}^{'}$ is from the tth observation in the ith cross section, constituting the $(\sum_{j=1}^{i-1} T_j + t)$th row of the matrix $\mathbf{X}$. If the CLUSTER option is specified, one extra term is added to the preceding equation so that the estimator of matrix Λ is

  $$\sum_{i=1}^{N}\sum_{t=1}^{T_i}\hat{\epsilon}_{it}^2\mathbf{x}_{it}\mathbf{x}_{it}^{'} + \sum_{i=1}^{N}\sum_{t=1}^{T_i}\sum_{s=1}^{t-1}\hat{\epsilon}_{it}\hat{\epsilon}_{is}\left(\mathbf{x}_{it}\mathbf{x}_{is}^{'} + \mathbf{x}_{is}\mathbf{x}_{it}^{'}\right)$$

  The formula is the same as the robust variance matrix estimator in Wooldridge (2002, p. 152) and it is derived under the assumptions of section 7.3.2 of Wooldridge (2002).

- HCCME=1:

  $$\frac{M}{M-K}\sum_{i=1}^{N}\sum_{t=1}^{T_i}\hat{\epsilon}_{it}^2\mathbf{x}_{it}\mathbf{x}_{it}^{'}$$

  where M is the total number of observations, $\sum_{j=1}^{N} T_j$, and K is the number of parameters. With the CLUSTER option, the estimator becomes

  $$\frac{M}{M-K}\sum_{i=1}^{N}\sum_{t=1}^{T_i}\hat{\epsilon}_{it}^2\mathbf{x}_{it}\mathbf{x}_{it}^{'} + \frac{M}{M-K}\sum_{i=1}^{N}\sum_{t=1}^{T_i}\sum_{s=1}^{t-1}\hat{\epsilon}_{it}\hat{\epsilon}_{is}\left(\mathbf{x}_{it}\mathbf{x}_{is}^{'} + \mathbf{x}_{is}\mathbf{x}_{it}^{'}\right)$$

The formula is similar to the robust variance matrix estimator in Wooldridge (2002, p. 152) with the heteroskedasticity adjustment term $M/(M-K)$.

- HCCME=2:

$$\sum_{i=1}^{N}\sum_{t=1}^{T_i}\frac{\hat{\epsilon}_{it}^2}{1-\hat{h}_{it}}\mathbf{x}_{it}\mathbf{x}_{it}^{'}$$

The $\hat{h}_{it}$ term is the $(\sum_{j=1}^{i-1}T_j+t)$th diagonal element of the hat matrix. The expression for $\hat{h}_{it}$ is $\mathbf{x}_{it}^{'}(\mathbf{X}^{'}\mathbf{X})^{-1}\mathbf{x}_{it}$. The hat matrix attempts to adjust the estimates for the presence of influence or leverage points. With the CLUSTER option, the estimator becomes

$$\sum_{i=1}^{N}\sum_{t=1}^{T_i}\frac{\hat{\epsilon}_{it}^2}{1-\hat{h}_{it}}\mathbf{x}_{it}\mathbf{x}_{it}^{'}+2\sum_{i=1}^{N}\sum_{t=1}^{T_i}\sum_{s=1}^{t-1}\frac{\hat{\epsilon}_{it}}{\sqrt{1-\hat{h}_{it}}}\frac{\hat{\epsilon}_{is}}{\sqrt{1-\hat{h}_{is}}}\left(\mathbf{x}_{it}\mathbf{x}_{is}^{'}+\mathbf{x}_{is}\mathbf{x}_{it}^{'}\right)$$

The formula is similar to the robust variance matrix estimator in Wooldridge (2002, p. 152) with the heteroskedasticity adjustment.

- HCCME=3:

$$\sum_{i=1}^{N}\sum_{t=1}^{T_i}\frac{\hat{\epsilon}_{it}^2}{(1-\hat{h}_{it})^2}\mathbf{x}_{it}\mathbf{x}_{it}^{'}$$

With the CLUSTER option, the estimator becomes

$$\sum_{i=1}^{N}\sum_{t=1}^{T_i}\frac{\hat{\epsilon}_{it}^2}{(1-\hat{h}_{it})^2}\mathbf{x}_{it}\mathbf{x}_{it}^{'}+2\sum_{i=1}^{N}\sum_{t=1}^{T_i}\sum_{s=1}^{t-1}\frac{\hat{\epsilon}_{it}}{1-\hat{h}_{it}}\frac{\hat{\epsilon}_{is}}{1-\hat{h}_{is}}\left(\mathbf{x}_{it}\mathbf{x}_{is}^{'}+\mathbf{x}_{is}\mathbf{x}_{it}^{'}\right)$$

The formula is similar to the robust variance matrix estimator in Wooldridge (2002, p. 152) with the heteroskedasticity adjustment.

- HCCME=4: PROC PANEL includes this option for the calculation of the Arellano (1987) version of the White (1980) HCCME in the panel setting. Arellano's insight is that there are N covariance matrices in a panel, and each matrix corresponds to a cross section. Forming the White HCCME for each panel, you need to take only the average of those N estimators that yield Arellano. The details of the estimation follow. First, you arrange the data such that the first cross section occupies the first T_i observations. You treat the panels as separate regressions with the form:

$$\mathbf{y}_i=\alpha_i\mathbf{i}+\mathbf{X}_{is}\tilde{\boldsymbol{\beta}}+\boldsymbol{\epsilon}_i$$

The parameter estimates $\tilde{\boldsymbol{\beta}}$ and α_i are the result of least squares dummy variables (LSDV) or within estimator regressions, and $\mathbf{i}$ is a vector of ones of length T_i. The estimate of the ith cross section's $\mathbf{X}^{'}\Omega\mathbf{X}$ matrix (where the s subscript indicates that no constant column has been suppressed to avoid confusion) is $\mathbf{X}_i^{'}\Omega\mathbf{X}_i$. The estimate for the whole sample is:

$$\mathbf{X}_s^{'}\Omega\mathbf{X}_s=\sum_{i=1}^{N}\mathbf{X}_i^{'}\Omega\mathbf{X}_i$$

The Arellano standard error is in fact a White-Newey-West estimator with constant and equal weight on each component. In the between estimators, selecting HCCME=4 returns the HCCME=0 result since there is no 'other' variable to group by.

In their discussion, Davidson and MacKinnon (1993, p. 554) argue that HCCME=1 should always be preferred to HCCME=0. Although HCCME=3 is generally preferred to 2 and 2 is preferred to 1, the calculation of HCCME=1 is as simple as the calculation of HCCME=0. Therefore, it is clear that HCCME=1 is preferred when the calculation of the hat matrix is too tedious.

All HCCME estimators have well-defined asymptotic properties. The small sample properties are not well-known, and care must exercised when sample sizes are small.

The HCCME estimator of $\mathrm{Var}(\boldsymbol{\beta})$ is used to drive the covariance matrices for the fixed effects and the LaGrange multiplier standard errors. Robust estimates of the covariance matrix for $\boldsymbol{\beta}$ imply robust covariance matrices for all other parameters.

Heteroscedasticity- and Autocorrelation-Consistent Covariance Matrices

The HAC option in the MODEL statement selects the type of heteroscedasticity- and autocorrelation-consistent covariance matrix. As with the HCCME option, an estimator of the middle expression Λ in sandwich form is needed. With the HAC option, it is estimated as

$$\Lambda_{\mathrm{HAC}} = a\sum_{i=1}^{N}\sum_{t=1}^{T_i}\hat{\epsilon}_{it}^2\mathbf{x}_{it}\mathbf{x}_{it}^{'} + a\sum_{i=1}^{N}\sum_{t=1}^{T_i}\sum_{s=1}^{t-1}k(\frac{s-t}{b})\hat{\epsilon}_{it}\hat{\epsilon}_{is}\left(\mathbf{x}_{it}\mathbf{x}_{is}^{'} + \mathbf{x}_{is}\mathbf{x}_{it}^{'}\right)$$

, where $k(.)$ is the real-valued kernel function[5], b is the bandwidth parameter, and a is the adjustment factor of small sample degrees of freedom (that is, $a = 1$ if the ADJUSTDF option is not specified and otherwise $a = NT/(NT-k)$, where k is the number of parameters including dummy variables). The types of kernel functions are listed in Table 27.3.

Table 27.3 Kernel Functions

Kernel Name	Equation
Bartlett	$k(x) = \begin{cases} 1-\lvert x\rvert & \lvert x\rvert \le 1 \\ 0 & \text{otherwise} \end{cases}$
Parzen	$k(x) = \begin{cases} 1-6x^2+6\lvert x\rvert^3 & 0 \le \lvert x\rvert \le 1/2 \\ 2(1-\lvert x\rvert)^3 & 1/2 \le \lvert x\rvert \le 1 \\ 0 & \text{otherwise} \end{cases}$
Quadratic spectral	$k(x) = \frac{25}{12\pi^2x^2}\left(\frac{\sin(6\pi x/5)}{6\pi x/5} - \cos(6\pi x/5)\right)$
Truncated	$k(x) = \begin{cases} 1 & \lvert x\rvert \le 1 \\ 0 & \text{otherwise} \end{cases}$
Tukey-Hanning	$k(x) = \begin{cases} (1+\cos(\pi x))/2 & \lvert x\rvert \le 1 \\ 0 & \text{otherwise} \end{cases}$

When the BANDWIDTH=ANDREWS option is specified, the bandwidth parameter is estimated as shown in Table 27.4.

[5]The HCCME=0 with CLUSTER option sets $k(.) = 1$.

Table 27.4 Bandwidth Parameter Estimation

Kernel Name	Bandwidth Parameter
Bartlett	$b = 1.1447(\alpha(1)T)^{1/3}$
Parzen	$b = 2.6614(\alpha(2)T)^{1/5}$
Quadratic spectral	$b = 1.3221(\alpha(2)T)^{1/5}$
Truncated	$b = 0.6611(\alpha(2)T)^{1/5}$
Tukey-Hanning	$b = 1.7462(\alpha(2)T)^{1/5}$

Let $\{g_{ait}\}$ denote each series in $\{g_{it} = \hat{\epsilon}_{it}\mathbf{x}_{it}\}$, and let (ρ_a, σ_a^2) denote the corresponding estimates of the autoregressive and innovation variance parameters of the AR(1) model on $\{g_{ait}\}$, $a = 1, ..., k$, where the AR(1) model is parameterized as $g_{ait} = \rho g_{ait-1} + \epsilon_{ait}$ with $Var(\epsilon_{ait}) = \sigma_a^2$. The $\alpha(1)$ and $\alpha(2)$ are estimated with the following formulas:

$$\alpha(1) = \frac{\sum_{a=1}^{k} \frac{4\rho_a^2\sigma_a^4}{(1-\rho_a)^6(1+\rho_a)^2}}{\sum_{a=1}^{k} \frac{\sigma_a^4}{(1-\rho_a)^4}} \qquad \alpha(2) = \frac{\sum_{a=1}^{k} \frac{4\rho_a^2\sigma_a^4}{(1-\rho_a)^8}}{\sum_{a=1}^{k} \frac{\sigma_a^4}{(1-\rho_a)^4}}$$

When you specify BANDWIDTH=NEWEYWEST94, according to Newey and West (1994) the bandwidth parameter is estimated as shown in Table 27.5.

Table 27.5 Bandwidth Parameter Estimation

Kernel Name	Bandwidth Parameter
Bartlett	$b = 1.1447(\{s_1/s_0\}^2T)^{1/3}$
Parzen	$b = 2.6614(\{s_1/s_0\}^2T)^{1/5}$
Quadratic spectral	$b = 1.3221(\{s_1/s_0\}^2T)^{1/5}$
Truncated	$b = 0.6611(\{s_1/s_0\}^2T)^{1/5}$
Tukey-Hanning	$b = 1.7462(\{s_1/s_0\}^2T)^{1/5}$

The s_1 and s_0 are estimated with the following formulas:

$$s_1 = 2\sum_{j=1}^{n} j\sigma_j \qquad s_0 = \sigma_0 + 2\sum_{j=1}^{n} \sigma_j$$

where n is the lag selection parameter and is determined by kernels, as listed in Table 27.6.

Table 27.6 Lag Selection Parameter Estimation

Kernel Name	Lag Selection Parameter
Bartlett	$n = c(T/100)^{2/9}$
Parzen	$n = c(T/100)^{4/25}$
Quadratic Spectral	$n = c(T/100)^{2/25}$
Truncated	$n = c(T/100)^{1/5}$
Tukey-Hanning	$n = c(T/100)^{1/5}$

The c in Table 27.6 is specified by the C= option; by default, C=12.

The σ_j is estimated with the equation

$$\sigma_j = T^{-1} \sum_{t=j+1}^{T} \left(\sum_{a=i}^{k} g_{at} \sum_{a=i}^{k} g_{at-j} \right), j = 0, ..., n$$

where g_{at} is the same as in the Andrews method and i is 1 if the NOINT option in the MODEL statement is specified, and 2 otherwise.

When you specify BANDWIDTH=SAMPLESIZE, the bandwidth parameter is estimated with the equation

$$b = \begin{cases} \lfloor \gamma T^r + c \rfloor & \text{if BANDWIDTH=SAMPLESIZE(INT) option is specified} \\ \gamma T^r + c & \text{otherwise} \end{cases}$$

where T is the sample size, $\lfloor x \rfloor$ is the largest integer less than or equal to x, and γ, r, and c are values specified by BANDWIDTH=SAMPLESIZE(GAMMA=, RATE=, CONSTANT=) options, respectively.

If the PREWHITENING option is specified in the MODEL statement, g_{it} is prewhitened by the VAR(1) model,

$$g_{it} = A_i g_{i,t-1} + w_{it}$$

Then Λ_{HAC} is calculated by

$$\Lambda_{\mathrm{HAC}} = a \sum_{i=1}^{N} \left\{ \left(\sum_{t=1}^{T_i} w_{it} w_{it}' + \sum_{t=1}^{T_i} \sum_{s=1}^{t-1} k(\frac{s-t}{b}) \left(w_{it} w_{is}' + w_{is} w_{it}' \right) \right) (I - A_i)^{-1} ((I - A_i)^{-1})' \right\}$$

R-Square

The conventional R-square measure is inappropriate for all models that the PANEL procedure estimates by using GLS because a number outside the [0,1] range might be produced. Hence, a generalization of the R-square measure is reported. The following goodness-of-fit measure (Buse 1973) is reported:

$$R^2 = 1 - \frac{\hat{\mathbf{u}}' \hat{\mathbf{V}}^{-1} \hat{\mathbf{u}}}{\mathbf{y}' \mathbf{D}' \hat{\mathbf{V}}^{-1} \mathbf{D} \mathbf{y}}$$

where $\hat{\mathbf{u}}$ are the residuals of the transformed model, $\hat{\mathbf{u}} = \mathbf{y} - \mathbf{X}(\mathbf{X}'\hat{\mathbf{V}}^{-1}\mathbf{X})^{-1}\mathbf{X}'\hat{\mathbf{V}}^{-1}\mathbf{y}$,

and $\mathbf{D} = \mathbf{I}_M - \mathbf{j}_M\mathbf{j}_M'(\frac{\hat{\mathbf{V}}^{-1}}{\mathbf{j}_M'\hat{\mathbf{V}}^{-1}\mathbf{j}_M})$.

This is a measure of the proportion of the transformed sum of squares of the dependent variable that is attributable to the influence of the independent variables.

If there is no intercept in the model, the corresponding measure (Theil 1961) is

$$R^2 = 1 - \frac{\hat{\mathbf{u}}'\hat{\mathbf{V}}^{-1}\hat{\mathbf{u}}}{\mathbf{y}'\hat{\mathbf{V}}^{-1}\mathbf{y}}$$

However, the fixed-effects models are somewhat different. In the case of a fixed-effects model, the choice of including or excluding an intercept becomes merely a choice of classification. Suppressing the intercept in the FIXONE or FIXONETIME case merely changes the name of the intercept to a fixed effect. It makes no sense to redefine the R-square measure since nothing material changes in the model. Similarly, for the FIXTWO model there is no reason to change the R-square measure. In the case of the FIXONE, FIXONETIME, and FIXTWO models, the R-square is defined as the Theil (1961) R-square as shown in the preceding equation. This makes intuitive sense since you are regressing a transformed (demeaned) series on transformed regressors, excluding a constant. In other words, you are looking at 1 minus the sum of squared errors divided by the sum of squares of the (transformed) dependent variable.

In the case of OLS estimation, both of the R-square formulas given here reduce to the usual R-square formula.

Specification Tests

The PANEL procedure outputs the results of one specification test for fixed effects and two specification tests for random effects.

For fixed effects, let β_f be the n dimensional vector of fixed-effects parameters. The specification test reported is the conventional F statistic for the hypothesis $\beta_f = \mathbf{0}$. The F statistic with $n, M - K$ degrees of freedom is computed as

$$\hat{\beta}_f\hat{\mathbf{S}}_f^{-1}\hat{\beta}_f/n$$

where $\hat{\mathbf{S}}_f$ is the estimated covariance matrix of the fixed-effects parameters.

Hausman (1978) specification test or m statistic can be used to test hypotheses in terms of bias or inconsistency of an estimator. This test was also proposed by Wu (1973) and further extended in Hausman and Taylor (1982). Hausman's m statistic is as follows.

Consider two estimators, $\hat{\beta}_a$ and $\hat{\beta}_b$, which under the null hypothesis are both consistent, but only $\hat{\beta}_a$ is asymptotically efficient. Under the alternative hypothesis, only $\hat{\beta}_b$ is consistent. The m statistic is

$$m = (\hat{\beta}_b - \hat{\beta}_a)'(\hat{\mathbf{S}}_b - \hat{\mathbf{S}}_a)^{-1}(\hat{\beta}_b - \hat{\beta}_a)$$

where $\hat{\mathbf{S}}_b$ and $\hat{\mathbf{S}}_a$ are consistent estimates of the asymptotic covariance matrices of $\hat{\beta}_b$ and $\hat{\beta}_a$. Then m is distributed χ^2 with k degrees of freedom, where k is the dimension of $\hat{\beta}_a$ and $\hat{\beta}_b$.

In the random-effects specification, the null hypothesis of no correlation between effects and regressors implies that the OLS estimates of the slope parameters are consistent and inefficient but the GLS estimates of

the slope parameters are consistent and efficient. This facilitates a Hausman specification test. The reported χ^2 statistic has degrees of freedom equal to the number of slope parameters. If the null hypothesis holds, the random-effects specification should be used.

Breusch and Pagan (1980) lay out a LaGrange multiplier test for random effects based on the simple OLS (pooled) estimator. If $\hat{u}_{it}$ is the itth residual from the OLS regression, then the Breusch-Pagan (BP) test for one-way random effects is

$$BP = \frac{NT}{2(T-1)}\left[\frac{\sum_{i=1}^{N}\left[\sum_{t=1}^{T}\hat{u}_{it}\right]^2}{\sum_{i=1}^{N}\sum_{t=1}^{T}\hat{u}_{it}^2} - 1\right]^2$$

The BP test generalizes to the case of a two-way random-effects model (Greene 2000, p. 589). Specifically,

$$\begin{aligned} BP2 &= \frac{NT}{2(T-1)}\left[\frac{\sum_{i=1}^{n}\left[\sum_{t=1}^{T}\hat{u}_{it}\right]^2}{\sum_{i=1}^{N}\sum_{t=1}^{T}\hat{u}_{it}^2} - 1\right]^2 \\ &+ \frac{NT}{2(N-1)}\left[\frac{\sum_{t=1}^{T}\left[\sum_{i=1}^{N}\hat{u}_{it}\right]^2}{\sum_{i=1}^{N}\sum_{t=1}^{T}\hat{u}_{it}^2} - 1\right]^2 \end{aligned}$$

is distributed as a χ^2 statistic with two degrees of freedom. Since the BP2 test generalizes (nests the BP test) the test for random effects, the absence of random effects (nonrejection of the null of no random effects) in the BP2 is a fairly clear indication that there will probably not be any one-way effects either. In both cases (BP and BP2), the residuals are obtained from a pooled regression. There is very little extra cost in selecting both the BP and BP2 test. Notice that in the case of just groupwise heteroscedasticity, the BP2 test approaches BP. In the case of time based heteroscedasticity, the BP2 test reduces to a BP test of time effects. In the case of unbalanced panels, neither the BP nor BP2 statistics are valid.

Finally, you should be aware that the BP option generates different results depending on whether the estimation is FIXONE or FIXONETIME. Specifically, under the FIXONE estimation technique, the BP tests for cross-sectional random effects. Under the FIXONETIME estimation, the BP tests for time random effects.

While the Hausman statistic is automatically generated, you request Breusch-Pagan via the BP or BP2 option (see Baltagi 2008 for details).

Panel Data Poolability Test

The null hypothesis of poolability assumes homogeneous slope coefficients. An F test can be applied to test for the poolability across cross sections in panel data models.

F Test

For the unrestricted model, run a regression for each cross section and save the sum of squared residuals as SSE_u. For the restricted model, save the sum of squared residuals as SSE_r. If the test applies to all coefficients (including the constant), then the restricted model is the pooled model (OLS); if the test applies

to coefficients other than the constant, then the restricted model is the fixed one-way model with cross-sectional fixed effects. If N and T denote the number of cross sections and time periods, then the number of observations is $n = NT$.[6] Let k be the number of regressors except the constant. The degree of freedom for the unrestricted model is $df_u = n - N(k+1)$. If the constant is restricted to be the same, the degree of freedom for the restricted model is $df_r = n - k - 1$ and the number of restrictions is $q = (N-1)(k+1)$. If the restricted model is the fixed one-way model, the degree of freedom is $df_r = n - k - N$ and the number of restrictions is $q = (N-1)k$. So the F test is

$$F = \frac{(SSE_r - SSE_u)/q}{SSE_u/df_u} \sim F(q, df_u)$$

For large N and T, you can use a chi-square distribution to approximate the limiting distribution, namely, $qF \implies \chi^2(q)$. The error term is assumed to be homogeneous; therefore, $\epsilon \sim \mathcal{N}\left(0, \sigma^2 I_n\right)$, and an OLS regression is sufficient. The test is the same as the Chow test (Chow 1960) extended to N linear regressions.

LR Test

Zellner (1962) also proved that the likelihood ratio test for null hypothesis of poolability can be based on the F statistic. The likelihood ratio can be expressed as $LR = -2log\left((1 + qF/df_u)^{-NT/2}\right) \implies LR = qF + O\left(n^{-1}\right)$. Under H_0, LR is asymptotically distributed as a chi-square with q degrees of freedom.

Panel Data Cross-Sectional Dependence Test

Breusch-Pagan LM Test

Breusch and Pagan (1980) propose a Lagrange multiplier (LM) statistic to test the null hypothesis of zero cross-sectional error correlations. Let e_{it} be the OLS estimate of the error term u_{it} under the null hypothesis. Then the pairwise cross-sectional correlations can be estimated by the sample counterparts $\hat{\rho}_{ij}$,

$$\hat{\rho}_{ij} = \hat{\rho}_{ji} = \frac{\sum_{t=\underline{T}_{ij}}^{\overline{T}_{ij}} e_{it} e_{jt}}{\sqrt{\sum_{t=\underline{T}_{ij}}^{\overline{T}_{ij}} e_{it}^2} \sqrt{\sum_{t=\underline{T}_{ij}}^{\overline{T}_{ij}} e_{jt}^2}}$$

where $\underline{T}_{ij}$ and $\overline{T}_{ij}$ are the lower bound and upper bound, respectively, which mark the overlap time periods for the cross sections i and j. If the panel is balanced, $\underline{T}_{ij} = 1$ and $\overline{T}_{ij} = T$. Let T_{ij} denote the number of overlapped time periods ($T_{ij} = \overline{T}_{ij} - \underline{T}_{ij} + 1$). Then the Breusch-Pagan LM test statistic can be constructed as

$$\mathrm{BP} = \sum_{i=1}^{N} \sum_{j=i+1}^{N} T_{ij} \hat{\rho}_{ij}^2$$

When N is fixed and $T_{ij} \to \infty$, $\mathrm{BP} \to \chi^2(N(N-1)/2)$. So the test is not applicable as $N \to \infty$.

Because $\hat{\rho}_{ij}^2, i = 1, \ldots, N-1, j = i+1, \ldots, N$, are asymptotically independent under the null hypothesis of zero cross-sectional correlation, $T_{ij}\hat{\rho}_{ij}^2 \to \chi^2(1)$. Then the following modified Breusch-Pagan LM

[6] For the unbalanced panel, the number of time series T_i might be different. The number of observations needs to be redefined accordingly.

statistic can be considered to test for cross-sectional dependence:

$$\mathrm{BP}s = \sqrt{\frac{1}{N\,(N-1)}} \sum_{i=1}^{N} \sum_{j=i+1}^{N} \left(T_{ij}\,\hat{\rho}_{ij}^{2} - 1\right)$$

Under the null hypothesis, $\mathrm{BPs} \rightarrow \mathcal{N}\,(0,1)$ as $T_{ij} \rightarrow \infty$, and then $N \rightarrow \infty$. But because $E\left(T_{ij}\,\hat{\rho}_{ij}^{2} - 1\right)$ is not correctly centered at zero for finite T_{ij}, the test is likely to exhibit substantial size distortion for large N and small T_{ij}.

Pesaran CD and CD*p* Test

Pesaran (2004) proposes a cross-sectional dependence test that is also based on the pairwise correlation coefficients $\hat{\rho}_{ij}$,

$$\mathrm{CD} = \sqrt{\frac{2}{N\,(N-1)}} \sum_{i=1}^{N} \sum_{j=i+1}^{N} \sqrt{T_{ij}}\,\hat{\rho}_{ij}$$

The test statistic has a zero mean for fixed N and T_{ij} under a wide class of panel data models, including stationary or unit root heterogeneous dynamic models that are subject to multiple breaks. For each $i \neq j$, as $T_{ij} \rightarrow \infty$, $\sqrt{T_{ij}}\,\hat{\rho}_{ij} \Longrightarrow \mathcal{N}\,(0,1)$. Therefore, for N and T_{ij} tending to infinity in any order, $\mathrm{CD} \Longrightarrow \mathcal{N}\,(0,1)$.

To enhance the power against the alternative hypothesis of local dependence, Pesaran (2004) proposes the CD*p* test. Local dependence is defined with respect to a weight matrix, $\mathbf{W} = \left(w_{ij}\right)$. Therefore, the test can be applied only if the cross-sectional units can be given an ordering that remains immutable over time. Under the alternative hypothesis of a *p*th-order local dependence, the CD statistic can be generalized to a local CD test, CD*p*,

$$\begin{aligned} \mathrm{CD}p &= \sqrt{\tfrac{2}{p(2N-p-1)}} \left(\textstyle\sum_{s=1}^{p} \sum_{i=s+1}^{N} \sqrt{T_{i,i-s}}\,\hat{\rho}_{i,i-s}\right) \\ &= \sqrt{\tfrac{2}{p(2N-p-1)}} \left(\textstyle\sum_{s=1}^{p} \sum_{i=1}^{N-s} \sqrt{T_{i,i+s}}\,\hat{\rho}_{i,i+s}\right) \end{aligned}$$

where $p = 1, \ldots, N-1$. When $p = N-1$, CD*p* reduces to the original CD test. Under the null hypothesis of zero cross-sectional dependence, the CD*p* statistic is centered at zero for fixed N and $T_{i,i-s} > k+1$, and $\mathrm{CD}p \Longrightarrow \mathcal{N}\,(0,1)$ as $N \rightarrow \infty$ and $T_{i,i+s} \rightarrow \infty$.

Panel Data Unit Root Tests

Levin, Lin, and Chu (2002)

Levin, Lin, and Chu (2002) propose a panel unit root test for the null hypothesis of unit root against a homogeneous stationary hypothesis. The model is specified as

$$\Delta y_{it} = \delta y_{it-1} + \sum_{L=1}^{p_i} \theta_{iL}\,\Delta y_{it-L} + \alpha_{mi}\,d_{mt} + \varepsilon_{it} \qquad m = 1,2,3$$

Three models are considered: (1) $d_{1t} = \phi$ (the empty set) with no individual effects, (2) $d_{2t} = \{1\}$ in which the series y_{it} has an individual-specific mean but no time trend, and (3) $d_{3t} = \{1, t\}$ in which the series

y_{it} has an individual-specific mean and linear and individual-specific time trend. The panel unit root test evaluates the null hypothesis of $H_0 : \delta = 0$, for all i, against the alternative hypothesis $H_1 : \delta < 0$ for all i. The lag order p_i is unknown and is allowed to vary across individuals. It can be selected by the methods that are described in the section "Lag Order Selection in the ADF Regression" on page 1878. Denote the selected lag orders as $\hat{p}_i$. The test is implemented in three steps.

Step 1 The ADF regressions are implemented for each individual i, and then the orthogonalized residuals are generated and normalized. That is, the following model is estimated:

$$\Delta y_{it} = \delta_i y_{it-1} + \sum_{L=1}^{\hat{p}_i} \theta_{iL} \Delta y_{it-L} + \alpha_{mi} d_{mt} + \varepsilon_{it} \qquad m = 1, 2, 3$$

The two orthogonalized residuals are generated by the following two auxiliary regressions:

$$\Delta y_{it} = \sum_{L=1}^{\hat{p}_i} \theta_{iL} \Delta y_{it-L} + \alpha_{mi} d_{mi} + e_{it}$$

$$y_{it-1} = \sum_{L=1}^{\hat{p}_i} \theta_{iL} \Delta y_{it-L} + \alpha_{mi} d_{mi} + v_{it-1}$$

The residuals are saved at $\hat{e}_{it}$ and $\hat{v}_{it-1}$, respectively. To remove heteroscedasticity, the residuals $\hat{e}_{it}$ and $\hat{v}_{it-1}$ are normalized by the regression standard error from the ADF regression. Denote the standard error as $\hat{\sigma}^2_{\varepsilon i} = \sum_{t=\hat{p}_i+2}^{T} \left(\hat{e}_{it} - \hat{\delta}_i \hat{v}_{it-1} \right)^2 / (T - p_i - 1)$, and normalize residuals as

$$\tilde{e}_{it} = \frac{\hat{e}_{it}}{\hat{\sigma}_{\varepsilon i}}, \qquad \tilde{v}_{it-1} = \frac{\hat{v}_{it-1}}{\hat{\sigma}_{\varepsilon i}}$$

Step 2 The ratios of long-run to short-run standard deviations of Δy_{it} are estimated. Denote the ratios and the long-run variances as s_i and σ_{yi}, respectively. The long-run variances are estimated by the HAC (heteroscedasticity- and autocorrelation-consistent) estimators, which are described in the section "Long-Run Variance Estimation" on page 1878. Then the ratios are estimated by $\hat{s}_i = \hat{\sigma}_{yi} / \hat{\sigma}_{\varepsilon i}$. Let the average standard deviation ratio be $S_N = (1/N) \sum_{i=1}^{N} s_i$, and let its estimator be $\hat{S}_N = (1/N) \sum_{i=1}^{N} \hat{s}_i$.

Step 3 The panel test statistics are calculated. To calculate the t statistic and the adjusted t statistic, the following equation is estimated:

$$\tilde{e}_{it} = \delta \tilde{v}_{it-1} + \tilde{\varepsilon}_{it}$$

The total number of observations is $N\tilde{T}$, with $\bar{\hat{p}} = \sum_{i=1}^{N} \hat{p}_i / N, \tilde{T} = T - \bar{\hat{p}} - 1$. The standard t statistic for testing $\delta = 0$ is $t_\delta = \hat{\delta} / STD(\hat{\delta})$, with OLS estimator $\hat{\delta}$ and standard deviation $STD(\hat{\delta})$. However, the standard t statistic diverges to negative infinity for models (2) and (3). Let $\hat{\sigma}_{\tilde{\varepsilon}}$ be the root mean square error from the step 3 regression, and denote it as

$$\hat{\sigma}^2_{\tilde{\varepsilon}} = \left[\frac{1}{N\tilde{T}} \sum_{i=1}^{N} \sum_{t=2+\hat{p}_i}^{T} (\tilde{e}_{it} - \hat{\delta} \tilde{v}_{it-1})^2 \right]$$

Levin, Lin, and Chu (2002) propose the following adjusted t statistic:

$$t_{\delta}^{*} = \frac{t_{\delta} - N\tilde{T}\hat{S}_{N}\hat{\sigma}_{\tilde{\varepsilon}}^{-2}STD(\hat{\delta})\mu_{m\tilde{T}}^{*}}{\sigma_{m\tilde{T}}^{*}}$$

The mean and standard deviation adjustments ($\mu_{m\tilde{T}}^{*}, \sigma_{m\tilde{T}}^{*}$) depend on the time series dimension $\tilde{T}$ and model specification m, which can be found in Table 2 of Levin, Lin, and Chu (2002). The adjusted t statistic converges to the standard normal distribution. Therefore, the standard normal critical values are used in hypothesis testing.

Lag Order Selection in the ADF Regression

The methods for selecting the individual lag orders in the ADF regressions can be divided into two categories: selection based on information criteria and selection via sequential testing.

Lag Selection Based on Information Criteria In this method, the following information criteria can be applied to lag order selection: AIC, SBC, HQIC (HQC), and MAIC. As with other model selection applications, the lag order is selected from 0 to the maximum p_{max} to minimize the objective function, plus a penalty term, which is a function of the number of parameters in the regression. Let k be the number of parameters and T_o be the number of effective observations. For regression models, the objective function is $T_o log(SSR/T_o)$, where SSR is the sum of squared residuals. For AIC, the penalty term equals $2k$. For SBC, this term is $k log T_o$. For HQIC, it is $2ck log\,[log(T_o)]$ with c being a constant greater than 1.[7] For MAIC, the penalty term equals $2(\tau_T(k) + k)$, where

$$\tau_T(k) = (SSR/T_o)^{-1}\hat{\delta}^2 \sum_{t=p_{max}+2}^{T} y_{t-1}^2$$

and $\hat{\delta}$ is the estimated coefficient of the lagged dependent variable y_{t-1} in the ADF regression.

Lag Selection via Sequential Testing In this method, the lag order estimation is based on the statistical significance of the estimated AR coefficients. Hall (1994) proposed general-to-specific (GS) and specific-to-general (SG) strategies. Levin, Lin, and Chu (2002) recommend the first strategy, following Campbell and Perron (1991). In the GS modeling strategy, starting with the maximum lag order p_{max}, the t test for the largest lag order in $\hat{\theta}_i$ is performed to determine whether a smaller lag order is preferred. Specifically, when the null of $\hat{\theta}_{iL} = 0$ is not rejected given the significance level (5%), a smaller lag order is preferred. This procedure continues until a statistically significant lag order is reached. On the other hand, the SG modeling strategy starts with lag order 0 and moves toward the maximum lag order p_{max}.

Long-Run Variance Estimation

The long-run variance of Δy_{it} is estimated by a HAC-type estimator. For model (1), given the lag truncation parameter $\bar{K}$ and kernel weights $w_{\bar{K}L}$, the formula is

$$\hat{\sigma}_{yi}^2 = \frac{1}{T-1}\sum_{t=2}^{T}\Delta y_{it}^2 + 2\sum_{L=1}^{\bar{K}} w_{\bar{K}L}\left[\frac{1}{T-1}\sum_{t=2+L}^{T}\Delta y_{it}\Delta y_{it-L}\right]$$

[7] In practice c is set to 1, following the literature (Hannan and Quinn 1979; Hall 1994).

To achieve consistency, the lag truncation parameter must satisfy $\bar{K}/T \to 0$ and $\bar{K} \to \infty$ as $T \to \infty$. Levin, Lin, and Chu (2002) suggest $\bar{K} = \left\lfloor 3.21T^{1/3} \right\rfloor$. The weights $w_{\bar{K}L}$ depend on the kernel function. Andrews (1991) proposes data-driven bandwidth (lag truncation parameter + 1 if integer-valued) selection procedures to minimize the asymptotic mean squared error (MSE) criterion. For details about the kernel functions and Andrews (1991) data-driven bandwidth selection procedure, see the section "Heteroscedasticity- and Autocorrelation-Consistent Covariance Matrices" on page 1870 for details. Because Levin, Lin, and Chu (2002) truncate the bandwidth as an integer, when LLCBAND is specified as the BANDWIDTH option, it corresponds to BANDWIDTH $= \left\lfloor 3.21T^{1/3} \right\rfloor + 1$. Furthermore, kernel weights $w_{\bar{K}L} = k(L/(\bar{K}+1))$ with kernel function $k(\cdot)$.

For model (2), the series Δy_{it} is demeaned individual by individual first. Therefore, Δy_{it} is replaced by $\Delta y_{it} - \overline{\Delta y_{it}}$, where $\overline{\Delta y_{it}}$ is the mean of Δy_{it} for individual i. For model (3) with individual fixed effects and time trend, both the individual mean and trend should be removed before the long-run variance is estimated. That is, first regress Δy_{it} on $\{1, t\}$ for each individual and save the residual $\widetilde{\Delta y_{it}}$, and then replace Δy_{it} with the residual.

Cross-Sectional Dependence via Time-Specific Aggregate Effects

The Levin, Lin, and Chu (2002) testing procedure is based on the assumption of cross-sectional independence. It is possible to relax this assumption and allow for a limited degree of dependence via time-specific aggregate effects. Let θ_t denote the time-specific aggregate effects; then the data generating process (DGP) becomes

$$\Delta y_{it} = \delta y_{it-1} + \sum_{L=1}^{p_i} \theta_{iL} \Delta y_{it-L} + \alpha_{mi} d_{mt} + \theta_t + \varepsilon_{it} \qquad m = 4, 5$$

Two more models are considered: (4) $d_{1t} = \phi$ (the empty set) with no individual effects, but with time effects, and (5) $d_{2t} = \{1\}$ in which the series y_{it} has an individual-specific mean and time-specific mean.

By subtracting the time averages $\bar{y}_t = \sum_{i=1}^{N} y_{it}$ from the observed dependent variable y_{it}, or equivalently, by including the time-specific intercepts θ_t in the ADF regression, the cross-sectional dependence is removed. The impact of a single aggregate common factor that has an identical impact on all individuals but changes over time can also be removed in this way. After cross-sectional dependence is removed, the three-step procedure is applied to calculate the Levin, Lin, and Chu (2002) adjusted t statistic.

Deterministic Variables

Three deterministic variables can be included in the model for the first-stage estimation: CS_FixedEffects (cross-sectional fixed effects), TS_FixedEffects (time series fixed effects), and TimeTrend (individual linear time trend). When a linear time trend is included, the individual fixed effects are also included. Otherwise the time trend is not identified. Moreover, if the time fixed effects are included, the time trend is not identified either. Therefore, we have 5 identified models: model (1), no deterministic variables; model (2), CS_FixedEffects; model (3), CS_FixedEffects and TimeTrend; model (4), TS_FixedEffects; model (5), CS_FixedEffects TS_FixedEffects. PROC PANEL outputs the test results for all 5 model specifications.

Im, Pesaran, and Shin (2003)

To test for the unit root in heterogeneous panels, Im, Pesaran, and Shin (2003) propose a standardized t-bar test statistic based on averaging the (augmented) Dickey-Fuller statistics across the groups. The limiting distribution is standard normal. The stochastic process y_{it} is generated by the first-order autoregressive

process. If $\Delta y_{it} = y_{it} - y_{i,t-1}$, the data generating process can be expressed as in LLC:

$$\Delta y_{it} = \beta_i y_{it-1} + \sum_{j=1}^{p_i} \rho_{ij} \Delta y_{i,t-j} + \alpha_{mi} d_{mt} + \varepsilon_{it} \qquad m = 1, 2, 3$$

Unlike the DGP in LLC, β_i is allowed to differ across groups. The null hypothesis of unit roots is

$$H_0 : \beta_i = 0 \quad \text{for all } i$$

against the heterogeneous alternative,

$$H_1 : \beta_i < 0 \quad \text{for } i = 1, \ldots, N_1, \qquad \beta_i = 0 \quad \text{for } i = N_1 + 1, \ldots, N$$

The Im, Pesaran, and Shin test also allows for some (but not all) of the individual series to have unit roots under the alternative hypothesis. But the fraction of the individual processes that are stationary is positive, $lim_{N\to\infty} N_1/N = \delta \in (0, 1]$. The t-bar statistic, denoted by $t\text{-bar}_{NT}$, is formed as a simple average of the individual t statistics for testing the null hypothesis of $\beta_i = 0$. If $t_{iT}\,(p_i, \rho_i)$ is the standard t statistic, then

$$t\text{-bar}_{NT} = \frac{1}{N} \sum_{i=1}^{N} t_{iT}\,(p_i, \rho_i)$$

If $T \to \infty$, then for each i the t statistic (without time trend) converges to the Dickey-Fuller distribution, η_i, defined by

$$\eta_i = \frac{\frac{1}{2}\{[W_i(1)]^2 - 1\} - W_i(1) \int_0^1 W_i(u)du}{\int_0^1 [W_i(u)]^2 du - [\int_0^1 W_i(u)du]^2}$$

where W_i is the standard Brownian motion. The limiting distribution is different when a time trend is included in the regression (Hamilton 1994, p. 499). The mean and variance of the limiting distributions are reported in Nabeya (1999). The standardized t-bar statistic satisfies

$$Z_{tbar}(p, \rho) = \frac{\sqrt{N}\{t\text{-bar}_{NT} - E(\eta)\}}{\sqrt{Var(\eta)}} \implies \mathcal{N}(0, 1)$$

where the standard normal is the sequential limit with $T \to \infty$ followed by $N \to \infty$. To obtain better finite sample approximations, Im, Pesaran, and Shin (2003) propose standardizing the t-bar statistic by means and variances of $t_{iT}(p_i, 0)$ under the null hypothesis $\beta_i = 0$. The alternative standardized t-bar statistic is

$$W_{tbar}(p, \rho) = \frac{\sqrt{N}\{t\text{-bar}_{NT} - \sum_{i=1}^{N} E[t_{iT}(p_i, 0)|\beta_i = 0]/N\}}{\sqrt{\sum_{i=1}^{N} Var[t_{iT}(p_i, 0)|\beta_i = 0]/N}} \implies \mathcal{N}(0, 1)$$

Im, Pesaran, and Shin (2003) simulate the values of $E[t_{iT}(p_i, 0)|\beta_i = 0]$ and $Var[t_{iT}(p_i, 0)|\beta_i = 0]$ for different values of T and p. The lag order in the ADF regression can be selected by the same method as in Levin, Lin, and Chu (2002). See the section "Lag Order Selection in the ADF Regression" on page 1878 for details.

When T is fixed, Im, Pesaran, and Shin (2003) assume serially uncorrelated errors, $p_i = 0$; t_{iT} is likely to have finite second moment, which is not established in the paper. The t statistic is modified by imposing

the null hypothesis of a unit root. Denote $\tilde{\sigma}_{iT}$ as the estimated standard error from the restricted regression ($\beta_i = 0$),

$$\tilde{t}\text{-bar}_{NT} = \sum_{i=1}^{N} \tilde{t}_{it}/N = \sum_{i=1}^{N} \left[\hat{\beta}_{iT} \left(y'_{i,-1} M_\tau y_{i,-1} \right)^{1/2} / \tilde{\sigma}_{iT} \right] / N$$

where $\hat{\beta}_{iT}$ is the OLS estimator of β_i (unrestricted model), $\tau_T = (1, 1, \ldots, 1)'$, $M_\tau = I_T - \tau_T \left(\tau'_T \tau_T \right) \tau'_T$, and $y_{i,-1} = \left(y_{i0}, y_{i1}, \ldots, y_{i,T-1} \right)'$ Under the null hypothesis, the standardized $\tilde{t}$-bar statistic converges to a standard normal variate,

$$Z_{\tilde{t}bar} = \frac{\sqrt{N} \{ \tilde{t}\text{-bar}_{NT} - E\left(\tilde{t}_T \right) \}}{\sqrt{Var\left(\tilde{t}_T \right)}} \implies \mathcal{N}(0, 1)$$

where $E\left(\tilde{t}_T \right)$ and $Var\left(\tilde{t}_T \right)$ are the mean and variance of $\tilde{t}_{iT}$, respectively. The limit is taken as $N \to \infty$ and T is fixed. Their values are simulated for finite samples without a time trend. The $Z_{\tilde{t}bar}$ is also likely to converge to standard normal.

When N and T are both finite, an exact test that assumes no serial correlation can be used. The critical values of $t\text{-bar}_{NT}$ and $\tilde{t}\text{-bar}_{NT}$ are simulated.

Similar as in section "Levin, Lin, and Chu (2002)" on page 1876, it is possible to relax this assumption of cross-sectional independence and allow for a limited degree of dependence via time-specific aggregate effects. Two more models (model 4 and model 5) with time fixed effects are considered. See section "Cross-Sectional Dependence via Time-Specific Aggregate Effects" on page 1879 for details.

Combination Tests

Combining the observed significance levels (p-values) from N independent tests of the unit root null hypothesis was proposed by Maddala and Wu (1999); Choi (2001). Suppose G_i is the test statistic to test the unit root null hypothesis for individual $i = 1, \ldots, N$, and $F(\cdot)$ is the cdf (cumulative distribution function) of the asymptotic distribution as $T \to \infty$. Then the asymptotic p-value is defined as

$$p_i = F\left(G_i \right)$$

There are different ways to combine these p-values. The first one is the inverse chi-square test (Fisher 1932); this test is referred to as P test in Choi (2001) and λ in Maddala and Wu (1999):

$$P = -2 \sum_{i=1}^{N} \ln\left(p_i \right)$$

When the test statistics $\{G_i\}_{i=1,\ldots,N}$ are continuous, $\{p_i\}_{i=1,\ldots,N}$ are independent uniform $(0, 1)$ variables. Therefore, $P \Rightarrow \chi^2_{2N}$ as $T \to \infty$ and N fixed. But as $N \to \infty$, P diverges to infinity in probability. Therefore, it is not applicable for large N. To derive a nondegenerate limiting distribution, the P test (Fisher test with $N \to \infty$) should be modified to

$$P_m = \sum_{i=1}^{N} \left(-2\ln\left(p_i \right) - 2 \right) / 2\sqrt{N} = -\sum_{i=1}^{N} \left(\ln\left(p_i \right) + 1 \right) / \sqrt{N}$$

Under the null as $T_i \to \infty$,[8] and then $N \to \infty$, $P_m \Rightarrow \mathcal{N}(0,1)$.[9]

The second way of combining individual p-values is the inverse normal test,

$$Z = \sum_{i=1}^{N} \Phi^{-1}(p_i)$$

where $\Phi(\cdot)$ is the standard normal cdf. When $T_i \to \infty$, $Z \Rightarrow \mathcal{N}(0,1)$ as N is fixed. When N and T_i are both large, the sequential limit is also standard normal if $T_i \to \infty$ first and $N \to \infty$ next.

The third way of combining p-values is the logit test,

$$L^* = \sqrt{k}L = \sqrt{k}\sum_{i=1}^{N} \ln\left(\frac{p_i}{1-p_i}\right)$$

where $k = 3(5N+4)/\left(\pi^2 N(5N+2)\right)$. When $T_i \to \infty$ and N is fixed, $L^* \Rightarrow t_{5N+4}$. In other words, the limiting distribution is the t distribution with degree of freedom $5N+4$. The sequential limit is $L^* \Rightarrow \mathcal{N}(0,1)$ as $T_i \to \infty$ and then $N \to \infty$. Simulation results in Choi (2001) suggest that the Z test outperforms other combination tests. For the time series unit root test G_i, Maddala and Wu (1999) apply the augmented Dickey-Fuller test. According to Choi (2006), the Elliott, Rothenberg, and Stock (1996) Dickey-Fuller generalized least squares (DF-GLS) test brings significant size and power advantages in finite samples.

Similar as in section "Levin, Lin, and Chu (2002)" on page 1876, it is possible to relax this assumption of cross-sectional independence and allow for a limited degree of dependence via time-specific aggregate effects. Two more models (model 4 and model 5) with time fixed effects are considered. See section "Cross-Sectional Dependence via Time-Specific Aggregate Effects" on page 1879 for details.

Breitung's Unbiased Tests

To account for the nonzero mean of the t statistic in the OLS detrending case, bias-adjusted t statistics were proposed by: Levin, Lin, and Chu (2002); Im, Pesaran, and Shin (2003). The bias corrections imply a severe loss of power. Breitung and associates take an alternative approach to avoid the bias, by using alternative estimates of the deterministic terms (Breitung and Meyer 1994; Breitung 2000; Breitung and Das 2005). The DGP is the same as in the Im, Pesaran, and Shin approach. When serial correlation is absent, for model (2) with individual specific means, the constant terms are estimated by the initial values y_{i1}. Therefore, the series y_{it} is adjusted by subtracting the initial value. The equation becomes

$$\Delta y_{it} = \delta^* \left(y_{i,t-1} - y_{i1}\right) + v_{it}$$

For model (3) with individual specific means and time trends, the time trend can be estimated by $\hat{\beta}_i = (T-1)^{-1}(y_{iT} - y_{i1})$. The levels can be transformed as

$$\tilde{y}_{it} = y_{it} - y_{i1} - \hat{\beta}_i t = y_{it} - y_{i1} - t(y_{iT} - y_{i1})/(T-1)$$

The Helmert transformation is applied to the dependent variable to remove the mean of the differenced variable:

$$\Delta y^*_{it} = \sqrt{\frac{T-t}{T-t+1}}\left[\Delta y_{it} - \left(\Delta y_{i,t+1} + \ldots + \Delta y_{iT}\right)/(T-t)\right]$$

[8] The time series length T is subindexed by $i = 1, \ldots, N$ because the panel can be unbalanced.

[9] Choi (2001) also points out that the joint limit result where N and $\{T_i\}_{i=1,\ldots,N}$ go to infinity simultaneously is the same as the sequential limit, but it requires more moment conditions.

The transformed model is

$$\Delta y_{it}^{*} = \delta^{*} \tilde{y}_{i,t-1} + v_{it}$$

The pooled t statistic has a standard normal distribution. Therefore, no adjustment is needed for the t statistic. To adjust for heteroscedasticity across cross sections, Breitung (2000) proposes a UB (unbiased) statistic based on the transformed data,

$$UB = \frac{\sum_{i=1}^{N} \sum_{t=2}^{T} \Delta y_{it}^{*} \tilde{y}_{i,t-1} / \sigma_i^2}{\sqrt{\sum_{i=1}^{N} \sum_{t=2}^{T} \tilde{y}_{i,t-1}^2 / \sigma_i^2}}$$

where $\sigma_i^2 = E\left(\Delta y_{it} - \beta_i\right)^2$. When σ_i^2 is unknown, it can be estimated as

$$\hat{\sigma}_i^2 = \sum_{t=2}^{T} \left(\Delta y_{it} - \sum_{t=2}^{T} \Delta y_{it} / (T-1) \right)^2 / (T-2)$$

The UB statistic has a standard normal limiting distribution as $T \to \infty$ followed by $N \to \infty$ sequentially. To account for the short-run dynamics, Breitung and Das (2005) suggest applying the test to the prewhitened series, $\hat{y}_{it}$. For model (1) and model (2) (constant-only case), they suggested the same method as in step 1 of Levin, Lin, and Chu (2002).[10] For model (3) (with a constant and linear time trend), the prewhitened series can be obtained by running the following restricted ADF regression under the null hypothesis of a unit root ($\delta = 0$) and no intercept and linear time trend ($\mu_i = 0, \beta_i = 0$):

$$\Delta y_{it} = \sum_{L=1}^{\hat{p}_i} \theta_{iL} \Delta y_{it-L} + \mu_i + \varepsilon_{it}$$

where $\hat{p}_i$ is a consistent estimator of the true lag order p_i and can be estimated by the procedures listed in the section "Lag Order Selection in the ADF Regression" on page 1878. For LLC and IPS tests, the lag orders are selected by running the ADF regressions. But for Breitung and his coauthors' tests, the restricted ADF regressions are used to be consistent with the prewhitening method. Let $\left(\hat{\mu}_i, \hat{\theta}_{iL}\right)$ be the estimated coefficients.[11] The prewhitened series can be obtained by

$$\Delta \hat{y}_{it} = \Delta y_{it} - \sum_{L=1}^{\hat{p}_i} \hat{\theta}_{iL} \Delta y_{it-L}$$

and

$$\hat{y}_{it} = y_{it} - \sum_{L=1}^{\hat{p}_i} \hat{\theta}_{iL} y_{it-L}$$

[10] See the section "Levin, Lin, and Chu (2002)" on page 1876 for details. The only difference is the standard error estimate $\hat{\sigma}_{\varepsilon i}^2$. Breitung suggests using $T - p_i - 2$ instead of $T - p_i - 1$ as in LLC to normalize the standard error.

[11] Breitung (2000) suggests the approach in step 1 of Levin, Lin, and Chu (2002), while Breitung and Das (2005) suggest the prewhitening method as described above. In Breitung's code, to be consistent with the papers, different approaches are adopted for model (2) and (3). Meanwhile, for the order of variable transformation and prewhitening, in model (2), the initial values are deducted (variable transformation) first, and then the prewhitening was applied. For model (3), the order is reversed. The series is prewhitened and then transformed to remove the mean and linear time trend.

The transformed series are random walks under the null hypothesis,

$$\Delta \hat{y}_{it} = \delta \hat{y}_{i,t-1} + v_{it}$$

where $y_{is} = 0$ for $s < 0$. When the cross-section units are independent, the t statistic converges to standard normal under the null, as $T \to \infty$ followed by $N \to \infty$,

$$t_{OLS} = \frac{\sum_{i=1}^{N} \sum_{t=2}^{T} y_{i,t-1} \Delta y_{it}}{\hat{\sigma} \sqrt{\sum_{i=1}^{N} \sum_{t=2}^{T} y_{i,t-1}^2}} \implies \mathcal{N}(0,1)$$

where $\hat{\sigma}^2 = \sum_{i=1}^{N} \sum_{t=2}^{T} \left(\Delta y_{it} - \hat{\delta} y_{i,t-1} \right)^2 / N\,(T-1)$ with OLS estimator $\hat{\delta}$.
To take account for cross-sectional dependence, Breitung and Das (2005) propose the robust t statistic and a GLS version of the test statistic. Let $v_t = (v_{1t}, \ldots, v_{Nt})'$ be the error vector for time t, and let $\Omega = E\left(v_t v_t'\right)$ be a positive definite matrix with eigenvalues $\lambda_1 \geq \ldots \geq \lambda_N$. Let $y_t = (y_{1t}, \ldots, y_{Nt})'$ and $\Delta y_t = (\Delta y_{1t}, \ldots, \Delta y_{Nt})'$. The model can be written as a SUR-type system of equations,

$$\Delta y_t = \delta y_{t-1} + v_t$$

The unknown covariance matrix Ω can be estimated by its sample counterpart,

$$\hat{\Omega} = \sum_{t=2}^{T} \left(\Delta y_t - \hat{\delta} y_{t-1} \right) \left(\Delta y_t - \hat{\delta} y_{t-1} \right)' / (T-1)$$

The sequential limit $T \to \infty$ followed by $N \to \infty$ of the standard t statistic t_{OLS} is normal with mean 0 and variance $v_\Omega = \lim_{N\to\infty} \mathrm{tr}\left(\Omega^2/N\right) / (\mathrm{tr}\Omega/N)^2$. The variance v_Ω can be consistently estimated by $\hat{v}_{\hat{\delta}} = \left(\sum_{t=2}^{T} y_{t-1}' \hat{\Omega} y_{t-1} \right) / \left(\sum_{t=2}^{T} y_{t-1}' y_{t-1} \right)^2$. Thus the robust t statistic can be calculated as

$$t_{rob} = \frac{\delta}{\hat{v}_{\hat{\delta}}} = \frac{\sum_{t=2}^{T} y_{t-1}' \Delta y_t}{\sqrt{\sum_{t=2}^{T} y_{t-1}' \hat{\Omega} y_{t-1}}} \implies \mathcal{N}(0,1)$$

as $T \to \infty$ followed by $N \to \infty$ under the null hypothesis of random walk. Since the finite sample distribution can be quite different, Breitung and Das (2005) list the 1%, 5%, and 10% critical values for different N's.

When $T > N$, a (feasible) GLS estimator is applied; it is asymptotically more efficient than the OLS estimator. The data are transformed by multiplying $\hat{\Omega}^{-1/2}$ as defined before, $\hat{z}_t = \hat{\Omega}^{-1/2} y_t$. Thus the model is transformed into

$$\Delta \hat{z}_t = \delta \hat{z}_{t-1} + e_t$$

The feasible GLS (FGLS) estimator of δ and the corresponding t statistic are obtained by estimating the transformed model by OLS and denoted by $\hat{\delta}_{GLS}$ and t_{GLS}, respectively:

$$t_{GLS} = \frac{\sum_{t=2}^{T} y_{t-1}' \hat{\Omega}^{-1} \Delta y_t}{\sqrt{\sum_{t=2}^{T} y_{t-1}' \hat{\Omega}^{-1} y_{t-1}}} \implies \mathcal{N}(0,1)$$

Similar as in section "Levin, Lin, and Chu (2002)" on page 1876, it is possible to relax this assumption of cross-sectional independence and allow for a limited degree of dependence via time-specific aggregate effects. Two more models (model 4 and model 5) with time fixed effects are considered. See section "Cross-Sectional Dependence via Time-Specific Aggregate Effects" on page 1879 for details.

Hadri (2000) Stationarity Tests

Hadri (2000) adopts a component representation where an individual time series is written as a sum of a deterministic trend, a random walk, and a white-noise disturbance term. Under the null hypothesis of stationary, the variance of the random walk equals 0. Specifically, two models are considered:

- For model (1), the time series y_{it} is stationary around a level r_{i0},

$$y_{it} = r_{it} + \epsilon_{it} \quad i = 1, \ldots, N, \quad t = 1, \ldots, T$$

- For model (2), y_{it} is trend stationary,

$$y_{it} = r_{it} + \beta_i t + \epsilon_{it} \quad i = 1, \ldots, N, \quad t = 1, \ldots, T$$

  where r_{it} is the random walk component,

$$r_{it} = r_{it-1} + u_{it} \quad i = 1, \ldots, N, \quad t = 1, \ldots, T$$

  The initial values of the random walks, $\{r_{i0}\}_{i=1,\ldots,N}$, are assumed to be fixed unknowns and can be considered as heterogeneous intercepts. The errors ϵ_{it} and u_{it} satisfy $\epsilon_{it} \sim \text{iid}\mathcal{N}\left(0, \sigma_\epsilon^2\right)$, $u_{it} \sim \text{iid}\mathcal{N}\left(0, \sigma_u^2\right)$ and are mutually independent.

The null hypothesis of stationarity is $H_0 : \sigma_u^2 = 0$ against the alternative random walk hypothesis $H_1 : \sigma_u^2 > 0$.

In matrix form, the models can be written as

$$y_i = X_i \beta_i + e_i$$

where $y_i' = (y_{i1}, \ldots, y_{iT})$, $e_i' = (e_{i1}, \ldots, e_{iT})$ with $e_{it} = \sum_{j=1}^{t} u_{ij} + \epsilon_{it}$, and $X_i = (\iota_T, a_T)$ with ι_T being a $T \times 1$ vector of ones, $a_T' = (1, \ldots, T)$, and $\beta_i' = (r_{i0}, \beta_i)$.

Let $\hat{\epsilon}_{it}$ be the residuals from the regression of y_i on X_i; then the LM statistic is

$$LM = \frac{\frac{1}{N}\sum_{i=1}^{N} \frac{1}{T^2}\sum_{t=1}^{T} S_{it}^2}{\hat{\sigma}_\epsilon^2}$$

where $S_{it} = \sum_{j=1}^{t} \hat{\epsilon}_{ij}$ is the partial sum of the residuals and $\hat{\sigma}_\epsilon^2$ is a consistent estimator of σ_ϵ^2 under the null hypothesis of stationarity. With some regularity conditions,

$$LM \xrightarrow{p} E\left[\int_0^1 V^2(r)\, dr\right]$$

where $V(r)$ is a standard Brownian bridge in model (1) and a second-level Brownian bridge in model (2). Let $W(r)$ be a standard Wiener process (Brownian motion),

$$V(r) = \begin{cases} W(r) - rW(1) & \text{for model (1)} \\ W(r) + \left(2r - 3r^2\right) W(1) + 6r(r-1)\int_0^1 W(s)\, ds & \text{for model (2)} \end{cases}$$

The mean and variance of the random variable $\int V^2$ can be calculated by using the characteristic functions,

$$\xi = E\left[\int_0^1 V^2(r)\, dr\right] = \begin{cases} \frac{1}{6} & \text{for model (1)} \\ \frac{1}{15} & \text{for model (2)} \end{cases}$$

and

$$\zeta^2 = var\left[\int_0^1 V^2(r)\,dr\right] = \begin{cases} \frac{1}{45} & \text{for model (1)} \\ \frac{11}{6300} & \text{for model (2)} \end{cases}$$

The LM statistics can be standardized to obtain the standard normal limiting distribution,

$$Z = \frac{\sqrt{N}\,(LM - \xi)}{\zeta} \Longrightarrow \mathcal{N}(0,1)$$

Consistent Estimator of σ_ϵ^2

Hadri's (2000) test can be applied to the general case of heteroscedasticity and serially correlated disturbance errors. Under homoscedasticity and serially uncorrelated errors, σ_ϵ^2 can be estimated as

$$\hat{\sigma}_\epsilon^2 = \sum_{i=1}^{N}\sum_{t=1}^{T} \hat{\epsilon}_{it}^2 / N\,(T-k)$$

where k is the number of regressors. Therefore, $k = 1$ for model (1) and $k = 2$ for model (2).

When errors are heteroscedastic across individuals, the standard errors $\sigma_{\epsilon,i}^2$ can be estimated by $\hat{\sigma}_{\epsilon,i}^2 = \sum_{t=1}^{T} \hat{\epsilon}_{it}^2 / (T-k)$ for each individual i and the LM statistic needs to be modified to

$$LM = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\frac{1}{T^2}\sum_{t=1}^{T} S_{it}^2}{\hat{\sigma}_{\epsilon,i}^2}\right)$$

To allow for temporal dependence over t, σ_ϵ^2 has to be replaced by the long-run variance of ϵ_{it}, which is defined as $\sigma^2 = \sum_{i=1}^{N} \lim_{T\to\infty} T^{-1}\left(S_{iT}^2\right)/N$. A HAC estimator can be used to consistently estimate the long-run variance σ^2. For more information, see the section "Long-Run Variance Estimation" on page 1878.

Similar as in section "Levin, Lin, and Chu (2002)" on page 1876, it is possible to relax this assumption of cross-sectional independence and allow for a limited degree of dependence via time-specific aggregate effects. One more models (model 3) with time fixed effects are considered. See section "Cross-Sectional Dependence via Time-Specific Aggregate Effects" on page 1879 for details.

Harris and Tzavalis (1999) Panel Unit Root Tests

Harris and Tzavalis (1999) derive the panel unit root test under fixed T and large N. Five models are considered as in Levin, Lin, and Chu (2002). Model (1) is the homogeneous panel,

$$y_{it} = \varphi y_{it-1} + v_{it}$$

Under the null hypothesis, $\varphi = 1$. For model (2), each series is a unit root process with a heterogeneous drift,

$$y_{it} = \alpha_i + \varphi y_{it-1} + v_{it}$$

Model (3) includes heterogeneous drifts and linear time trends,

$$y_{it} = \alpha_i + \beta_i t + \varphi y_{it-1} + v_{it}$$

Similar as in section "Levin, Lin, and Chu (2002)" on page 1876, it is possible to relax this assumption of cross-sectional independence and allow for a limited degree of dependence via time-specific aggregate effects.

Two more models (model 4 and model 5) with time fixed effects are considered. See section "Cross-Sectional Dependence via Time-Specific Aggregate Effects" on page 1879 for details.

Let $\hat{\varphi}$ be the OLS estimator of φ; then

$$\hat{\varphi} - 1 = \left[\sum_{i=1}^{N} y'_{i,-1} Q_T y_{i,-1}\right]^{-1} \cdot \left[\sum_{i=1}^{N} y'_{i,-1} Q_T v_i\right]$$

where $y_{i,-1} = (y_{i0}, \ldots, y_{iT-1})$, $v'_i = (v_{i1}, \ldots, v_{iT})$, and Q_T is the projection matrix. For model (1), there are no regressors other than the lagged dependent value, so Q_T is the identity matrix I_T. For model (2), a constant is included, so $Q_T = I_T - e_T e'_T/T$ with e_T a $T \times 1$ column of ones. For model (3), a constant and time trend are included. Thus $Q_T = I_T - Z_T \left(Z'_T Z_T\right)^{-1} Z'_T$, where $Z_T = (e_T, \tau_T)$ and $\tau_T = (1, \ldots, T)'$.

When $y_{i0} = 0$ in model (1) under the null hypothesis, as $N \to \infty$

$$\sqrt{NT\,(T-1)/2}\,(\hat{\varphi} - 1) \xrightarrow{y_{i0}=0, H_0} \mathcal{N}(0, 1)$$

As $T \to \infty$, it becomes $T\sqrt{N}\,(\hat{\varphi} - 1) \overset{H_0}{\Longrightarrow} \mathcal{N}(0, 2)$.

When the drift is absent in model (2), $\alpha_i = 0$, under the null hypothesis, as $N \to \infty$

$$\sqrt{\frac{5N\,(T+1)^3\,(T-1)}{3\left(17T^2 - 20T + 17\right)}}\left(\hat{\varphi} - 1 + \frac{3}{(T+1)}\right) \xrightarrow{\alpha_i=0, H_0} \mathcal{N}(0, 1)$$

As $T \to \infty$, $\left(T\sqrt{N}\,(\hat{\varphi} - 1) + 3\sqrt{N}\right)/\sqrt{51/5} \overset{H_0}{\Longrightarrow} \mathcal{N}(0, 1)$.

When the time trend is absent in model (3), $\beta_i = 0$, under the null hypothesis, as $N \to \infty$

$$\sqrt{\frac{112N\,(T+2)^3\,(T-2)}{15\left(193T^2 - 728T + 1147\right)}})\left(\hat{\varphi} - 1 + \frac{15}{2\,(T+2)}\right) \xrightarrow{\beta_i=0, H_0} \mathcal{N}(0, 1)$$

When $T \to \infty$, $\left(T\sqrt{N}\,(\hat{\varphi} - 1) + 7.5\sqrt{N}\right)/\sqrt{2895/112} \overset{H_0}{\Longrightarrow} \mathcal{N}(0, 1)$.

Lagrange Multiplier (LM) Tests for Cross-Sectional and Time Effects

For random one-way and two-way error component models, the Lagrange multiplier test for the existence of cross-sectional or time effects or both is based on the residuals from the restricted model (that is, the pooled model). For more information about the Breusch-Pagan LM test, see the section "Specification Tests" on page 1873.

Honda (1985) and Honda (1991) UMP Test and Moulton and Randolph (1989) SLM Test

The Breusch-Pagan LM test is two-sided when the variance components are nonnegative. For a one-sided alternative hypothesis, Honda (1985) suggests a uniformly most powerful (UMP) LM test for $H_0^1 : \sigma_\gamma^2 = 0$ (no cross-sectional effects) that is based on the pooled estimator. The alternative is

the one-sided $H_1^1 : \sigma_\gamma^2 > 0$. Let $\hat{u}_{it}$ be the residual from the simple pooled OLS regression and $d = \left(\sum_{i=1}^{N}\left[\sum_{t=1}^{T}\hat{u}_{it}\right]^2\right) / \left(\sum_{i=1}^{N}\sum_{t=1}^{T}\hat{u}_{it}^2\right)$. Then the test statistic is defined as

$$J \equiv \sqrt{\frac{NT}{2(T-1)}}\,[d-1] \xrightarrow{H_0^1} \mathcal{N}(0,1)$$

The square of J is equivalent to the Breusch and Pagan (1980) LM test statistic. Moulton and Randolph (1989) suggest an alternative standardized Lagrange multiplier (SLM) test to improve the asymptotic approximation for Honda's one-sided LM statistic. The SLM test's asymptotic critical values are usually closer to the exact critical values than are those of the LM test. The SLM test statistic standardizes Honda's statistic by its mean and standard deviation. The SLM test statistic is

$$\mathrm{S} \equiv \frac{J - E(J)}{\sqrt{\mathrm{Var}(J)}} = \frac{d - E(d)}{\sqrt{\mathrm{Var}(d)}} \rightarrow \mathcal{N}(0,1)$$

Let $D = I_N \bigotimes J_T$, where J_T is the $T \times T$ square matrix of 1s. The mean and variance can be calculated by the formulas

$$E(d) = \mathrm{Tr}(DM_Z)/(n-k)$$

$$\mathrm{Var}(d) = 2\{(n-k)\mathrm{Tr}\,(DM_Z)^2 - [\mathrm{Tr}\,(DM_Z)]^2\}/((n-k)^2(n-k+2))$$

where Tr denotes the trace of a particular matrix, Z represents the regressors in the pooled model, $n = NT$ is the number of observations, k is the number of regressors, and $M_Z = I_n - Z(Z'Z)^{-1}Z'$. To calculate $\mathrm{Tr}(DM_Z)$, let $Z = \left(Z_1', Z_2', \ldots, Z_N\right)'$. Then

$$\mathrm{Tr}(DM_Z) = NT - \mathrm{Tr}\left(J_T \sum_{i=1}^{N}\left[Z_i\left(\sum_{j=1}^{N} Z_j' Z_j\right)^{-1} Z_i'\right]\right)$$

To test for $H_0^2 : \sigma_\alpha^2 = 0$ (no time effects), define $d2 = \left(\sum_{t=1}^{T}\left[\sum_{i=1}^{N}\hat{u}_{it}\right]^2\right) / \left(\sum_{t=1}^{T}\sum_{i=1}^{N}\hat{u}_{it}^2\right)$. Then the test statistic is modified as

$$J2 \equiv \sqrt{\frac{NT}{2(N-1)}}\,[d2-1] \xrightarrow{H_0^2} \mathcal{N}(0,1)$$

$J2$ can be standardized by $D = J_N \bigotimes I_T$, and other parameters are unchanged. Therefore,

$$\mathrm{S2} \equiv \frac{J2 - E(J2)}{\sqrt{\mathrm{Var}(J2)}} = \frac{d2 - E(d2)}{\sqrt{\mathrm{Var}(d2)}} \rightarrow \mathcal{N}(0,1)$$

To test for $H_0^3 : \sigma_\gamma^2 = 0, \sigma_\alpha^2 = 0$ (no cross-sectional and time effects), the test statistic is $J3 = (J + J2)/\sqrt{2}$ and $D = \sqrt{n/(T-1)}\,(I_N \bigotimes J_T)/2 + \sqrt{n/(N-1)}\,(J_N \bigotimes I_T)/2$. To standardize, define $d3 = \sqrt{n/(T-1)}d/2 + \sqrt{n/(N-1)}(d2)/2$,

$$\mathrm{S3} \equiv \frac{J3 - E(J3)}{\sqrt{\mathrm{Var}(J3)}} = \frac{d3 - E(d3)}{\sqrt{\mathrm{Var}(d3)}} \rightarrow \mathcal{N}(0,1)$$

King and Wu (1997) LMMP Test and the SLM Test

King and Wu (1997) derive the locally mean most powerful (LMMP) one-sided test for H_0^1 and H_0^2, which coincides with the Honda (1985) UMP test. Baltagi, Chang, and Li (1992) extend the King and Wu (1997) test for H_0^3 as follows:

$$\mathrm{KW} \equiv \frac{\sqrt{T-1}}{\sqrt{N+T-2}}\mathrm{J} + \frac{\sqrt{N-1}}{\sqrt{N+T-2}}\mathrm{J2} \xrightarrow{H_0^3} \mathcal{N}(0,1)$$

For the standardization, use $D = I_N \bigotimes J_T + J_N \bigotimes I_T$. Define $d_{kw} = d + d2$; then

$$\mathrm{S_{kw}} \equiv \frac{KW - E(KW)}{\sqrt{\mathrm{Var}(KW)}} = \frac{d_{kw} - E(d_{kw})}{\sqrt{\mathrm{Var}(d_{kw})}} \to \mathcal{N}(0,1)$$

Gourieroux, Holly, and Monfort (1982) LM Test

If one or both variance components (σ_γ^2 and σ_α^2) are small and close to 0, the test statistics J and $J2$ can be negative. Baltagi, Chang, and Li (1992) follow Gourieroux, Holly, and Monfort (1982) and propose a one-sided LM test for H_0^3, which is immune to the possible negative values of J and $J2$. The test statistic is

$$\mathrm{GHM} \equiv \begin{cases} J^2 + (J2)^2 & \text{if } J > 0, J2 > 0 \\ J^2 & \text{if } J > 0, J2 \le 0 \\ (J2)^2 & \text{if } J \le 0, J2 > 0 \\ 0 & \text{if } J \le 0, J2 \le 0 \end{cases} \xrightarrow{H_0^3} \left(\frac{1}{4}\right)\chi^2(0) + \left(\frac{1}{2}\right)\chi^2(1) + \left(\frac{1}{4}\right)\chi^2(2)$$

where $\chi^2(0)$ is the unit mass at the origin.

Tests for Serial Correlation and Cross-Sectional Effects

The presence of cross-sectional effects causes serial correlation in the errors. Therefore, serial correlation is often tested jointly with cross-sectional effects. Joint and conditional tests for both serial correlation and cross-sectional effects have been covered extensively in the literature.

Baltagi and Li Joint LM Test for Serial Correlation and Random Cross-Sectional Effects

Baltagi and Li (1991) derive the LM test statistic, which jointly tests for zero first-order serial correlation and random cross-sectional effects under normality and homoscedasticity. The test statistic is independent of the form of serial correlation, so it can be used with either AR(1) or MA(1) error terms. The null hypothesis is a white noise component: $H_0^1 : \sigma_\gamma^2 = 0, \theta = 0$ for MA(1) with MA coefficient θ or $H_0^2 : \sigma_\gamma^2 = 0, \rho = 0$ for AR(1) with AR coefficient ρ. The alternative is either a one-way random-effects model (cross-sectional) or first-order serial correlation AR(1) or MA(1) in errors or both. Under the null hypothesis, the model can be estimated by the pooled estimation (OLS). Denote the residuals as $\hat{u}_{it}$. The test statistic is

$$\mathrm{BL91} = \frac{NT^2}{2(T-1)(T-2)}\left[A^2 - 4AB + 2TB^2\right] \xrightarrow{H_0^{1,2}} \chi^2(2)$$

where

$$A = \frac{\sum_{i=1}^N \left(\sum_{t=1}^T \hat{u}_{it}\right)^2}{\sum_{i=1}^N \sum_{t=1}^T \hat{u}_{it}^2} - 1, \qquad B = \frac{\sum_{i=1}^N \sum_{t=2}^T \hat{u}_{it}\hat{u}_{i,t-1}}{\sum_{i=1}^N \sum_{t=1}^T \hat{u}_{it}^2}$$

Wooldridge Test for the Presence of Unobserved Effects

Wooldridge (2002, sec. 10.4.4) suggests a test for the absence of an unobserved effect. Under the null hypothesis $H_0 : \sigma_\gamma^2 = 0$, the errors u_{it} are serially uncorrelated. To test $H_0 : \sigma_\gamma^2 = 0$, Wooldridge (2002) proposes to test for AR(1) serial correlation. The test statistic that he proposes is

$$W = \frac{\sum_{i=1}^{N} \sum_{t=1}^{T-1} \sum_{s=t+1}^{T} \hat{u}_{it}\hat{u}_{is}}{\left[\sum_{i=1}^{N} \left(\sum_{t=1}^{T-1} \sum_{s=t+1}^{T} \hat{u}_{it}\hat{u}_{is}\right)^2\right]^{1/2}} \rightarrow \mathcal{N}(0,1)$$

where $\hat{u}_{it}$ are the pooled OLS residuals. The test statistic W can detect many types of serial correlation in the error term u, so it has power against both the one-way random-effects specification and the serial correlation in error terms.

Bera, Sosa Escudero, and Yoon Modified Rao's Score Test in the Presence of Local Misspecification

Bera, Sosa Escudero, and Yoon (2001) point out that the standard specification tests, such as the Honda (1985) test described in the section "Honda (1985) and Honda (1991) UMP Test and Moulton and Randolph (1989) SLM Test" on page 1887, are not valid when they test for either cross-sectional random effects or serial correlation without considering the presence of the other effects. They suggest a modified Rao's score (RS) test. When A and B are defined as in Baltagi and Li (1991), the test statistic for testing serial correlation under random cross-sectional effects is

$$\mathrm{RS}_\rho^* = \frac{NT^2 (B - A/T)^2}{(T-1)(1-2/T)}$$

Baltagi and Li (1991, 1995) derive the conventional RS test when the cross-sectional random effects is assumed to be absent:

$$\mathrm{RS}_\rho = \frac{NT^2 B^2}{T-1}$$

Symmetrically, to test for the cross-sectional random effects in the presence of serial correlation, the modified Rao's score test statistic is

$$\mathrm{RS}_\mu^* = \frac{NT (A - 2B)^2}{2(T-1)(1-2/T)}$$

and the conventional Rao's score test statistic is given in Breusch and Pagan (1980). The test statistics are asymptotically distributed as $\chi^2(1)$.

Because $\sigma_\gamma^2 > 0$, the one-sided test is expected to lead to more powerful tests. The one-sided test can be derived by taking the signed square root of the two-sided statistics:

$$\mathrm{RSO}_\mu^* = \sqrt{\frac{NT}{2(T-1)(1-2/T)}}\,(A - 2B) \rightarrow \mathcal{N}(0,1)$$

Baltagi and Li (1995) LM Test for First-Order Correlation under Fixed Effects

The two-sided LM test statistic for testing a white noise component in a fixed one-way model ($H_0^5 : \theta = 0$ or $H_0^6 : \rho = 0$, given that γ_i are fixed effects) is

$$\mathrm{BL95} = \frac{NT^2}{T-1}\left(\frac{\sum_{i=1}^{N}\sum_{t=2}^{T}\hat{u}_{it}\hat{u}_{i,t-1}}{\sum_{i=1}^{N}\sum_{t=1}^{T}\hat{u}_{it}^2}\right)^2$$

where $\hat{u}_{it}$ are the residuals from the fixed one-way model (FIXONE). The LM test statistic is asymptotically distributed as χ_1^2 under the null hypothesis. The one-sided LM test with alternative hypothesis $\rho > 0$ is

$$\mathrm{BL95}_2 = \sqrt{\frac{NT^2}{T-1}}\frac{\sum_{i=1}^{N}\sum_{t=2}^{T}\hat{u}_{it}\hat{u}_{i,t-1}}{\sum_{i=1}^{N}\sum_{t=1}^{T}\hat{u}_{it}^2}$$

which is asymptotically distributed as standard normal.

Durbin-Watson Statistic

Bhargava, Franzini, and Narendranathan (1982) propose a test of the null hypothesis of no serial correlation ($H_0^6 : \rho = 0$) against the alternative ($H_1^6 : 0 < |\rho| < 1$) by the Durbin-Watson statistic,

$$d_\rho = \frac{\sum_{i=1}^{N}\sum_{t=2}^{T}\left(\hat{u}_{it} - \hat{u}_{i,t-1}\right)^2}{\sum_{i=1}^{N}\sum_{t=1}^{T}\hat{u}_{it}^2}$$

where $\hat{u}_{it}$ are the residuals from the fixed one-way model (FIXONE).

The test statistic d_ρ is a locally most powerful invariant test in the neighborhood of $\rho = 0$. Some of the upper and lower bounds are listed in Bhargava, Franzini, and Narendranathan (1982). To test against a positive correlation ($\rho > 0$) for very large N, you can simply detect whether $d_\rho < 2$. However, for small-to-moderate N, the mechanics of the Durbin-Watson test produce a region of uncertainty as to whether to reject the null hypothesis. The output contains two p-values: The first, Pr < DWLower, treats the uncertainty region as a rejection region. The second, Pr > DWUpper, is more conservative and treats the uncertainty region as a failure-to-reject region. You can think of these two p-values as bounds on the exact p-value.

Berenblut-Webb Statistic

Bhargava, Franzini, and Narendranathan (1982) suggest using the Berenblut-Webb statistic, which is a locally most powerful invariant test in the neighborhood of $\rho = 1$. The test statistic is

$$\mathrm{g}_\rho = \frac{\sum_{i=1}^{N}\sum_{t=2}^{T}\Delta\tilde{u}_{i,t}^2}{\sum_{i=1}^{N}\sum_{t=1}^{T}\hat{u}_{it}^2}$$

where $\Delta\tilde{u}_{it}$ are the residuals from the first-difference estimation. The upper and lower bounds are the same as for the Durbin-Watson statistic d_ρ and produce two p-values, one conservative and one anti-conservative.

Testing for Random Walk Null Hypothesis

You can also use the Durbin-Watson and Berenblut-Webb statistics to test the random walk null hypothesis, with the bounds that are listed in Bhargava, Franzini, and Narendranathan (1982). For more information about these statistics, see the sections "Durbin-Watson Statistic" on page 1891 and "Berenblut-Webb Statistic"

on page 1891. Bhargava, Franzini, and Narendranathan (1982) also propose the R_ρ statistic to test the random walk null hypothesis $\rho = 1$ against the stationary alternative $|\rho| < 1$. Let $\mathbf{F}^* = I_N \otimes \mathbf{F}$, where $\mathbf{F}$ is a $(T-1)(T-1)$ symmetric matrix that has the following elements:

$$\mathbf{F}_{tt'} = \left(T - t'\right) t/T \quad if t' \geq t \quad \left(t, t' = 1, \ldots, T-1\right)$$

The test statistic is

$$\begin{aligned} R_\rho &= \Delta\tilde{U}'\Delta\tilde{U} / \Delta\tilde{U}' F^* \Delta\tilde{U} \\ &= \frac{\sum_{i=1}^{N}\sum_{t=2}^{T}\Delta\tilde{u}_{i,t}^2}{\left[\sum_{i=1}^{N}\sum_{t=2}^{T}(t-1)(T-t+1)\Delta\tilde{u}_{i,t}^2 + 2\sum_{i=1}^{N}\sum_{t=2}^{T-1}\sum_{t'=t+1}^{T}(T-t'+1)(t-1)\Delta\tilde{u}_{i,t}\Delta\tilde{u}_{i,t'}\right]/T} \end{aligned}$$

The statistics R_ρ, g_ρ, and d_ρ can be used with the same bounds. They satisfy $R_\rho \leq g_\rho \leq d_\rho$, and they are equivalent for large panels.

Troubleshooting

You need to follow some guidelines when you use PROC PANEL for analysis. For each cross section, PROC PANEL requires at least two time series observations that have nonmissing values for all model variables. There should be at least two cross sections for each time point in the data. If these two conditions are not met, then an error message is printed in the log that states that there is only one cross section or time series observation and further computations will be terminated. You must provide adequate data for an estimation method to produce results, and you should check the log for any errors that are related to data.

If PROC PANEL uses the Parks method and the number of cross sections is greater than the number of time series observations per cross section, then PROC PANEL produces an error message that states that the ϕ matrix is singular. This is analogous to seemingly unrelated regression that has fewer observations than equations in the model. To avoid the problem, reduce the number of cross sections.

Your data set could have multiple observations for each time ID within a particular cross section. However, you can use PROC PANEL only in cases where you have only a single observation for each time ID within each cross section. In such a case, after you have sorted the data, an error warning is printed in the log that states that the data have not been sorted in ascending sequence with respect to time series ID.

The cause of the error is due to multiple observations for each time ID for a given cross section. PROC PANEL allows only one observation for each time ID within each cross section.

The following data set shown in Figure 27.2 illustrates the preceding instance with the correct representation.

Figure 27.2 Single Observation for Each Time Series

Obs	firm	year	production	cost
1	1	1955	5.36598	1.14867
2	1	1960	6.03787	1.45185
3	1	1965	6.37673	1.52257
4	1	1970	6.93245	1.76627
5	2	1955	6.54535	1.35041
6	2	1960	6.69827	1.71109
7	2	1965	7.40245	2.09519
8	2	1970	7.82644	2.39480

In this case, you can observe that there are no multiple observations with respect to a given time series ID within a cross section. This is the correct representation of a data set where PROC PANEL is applicable.

If for state ID 1 you have two observations for the year=1955, then PROC PANEL produces the following error message:

"The data set is not sorted in ascending sequence with respect to time series ID. The current time period has year=1955 and the previous time period has year=1955 in cross section firm=1."

A data set similar to the previous example with multiple observations for the YEAR=1955 is shown in Figure 27.3; this data set results in an error message due to multiple observations while using PROC PANEL.

Figure 27.3 Multiple Observations for Each Time Series

Obs	firm	year	production	cost
1	1	1955	5.36598	1.14867
2	1	1955	6.37673	1.52257
3	1	1960	6.03787	1.45185
4	1	1970	6.93245	1.76627
5	2	1955	6.54535	1.35041
6	2	1960	6.69827	1.71109
7	2	1965	7.40245	2.09519
8	2	1970	7.82644	2.39480

In order to use PROC PANEL, you need to aggregate the data so that you have unique time ID values within each cross section. One possible way to do this is to run a PROC MEANS on the input data set and compute the mean of all the variables by FIRM and YEAR, and then use the output data set.

Creating ODS Graphics

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*).

Before you create graphs, ODS Graphics must be enabled (for example, with the ODS GRAPHICS ON statement). For more information about enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" in that chapter.

The overall appearance of graphs is controlled by ODS styles. Styles and other aspects of using ODS Graphics are discussed in the section "A Primer on ODS Statistical Graphics" in that chapter.

This section describes the use of ODS for creating graphics with the PANEL procedure. The table below lists the graph names, the plot descriptions, and the options used.

Table 27.7 ODS Graphics Produced by PROC PANEL

ODS Graph Name	Plot Description	Plots=Option
DiagnosticsPanel	All applicable plots listed below	
ResidualPlot	Plot of the residuals	RESIDUAL, RESID
FitPlot	Predicted versus actual plot	FITPLOT
QQPlot	Plot of the quantiles of the residuals	QQ
ResidSurfacePlot	Surface plot of the residuals	RESIDSURFACE
PredSurfacePlot	Surface plot of the predicted values	PREDSURFACE
ActSurfacePlot	Surface plot of the actual values	ACTSURFACE
ResidStackPlot	Stack plot of the residuals	RESIDSTACK, RESSTACK
ResidHistogram	Plot of the histogram of residuals	RESIDUALHISTOGRAM, RESIDHISTOGRAM

OUTPUT OUT= Data Set

PROC PANEL writes the initial data of the estimated model, predicted values, and residuals to an output data set when the OUTPUT OUT= statement is specified. The OUT= data set contains the following variables:

MODELL is a character variable that contains the label for the MODEL statement if a label is specified.

METHOD is a character variable that identifies the estimation method.

MODLNO is the number of the model estimated.

ACTUAL contains the value of the dependent variable.

WEIGHT contains the weighing variable.

CSID is the value of the cross section ID.

TSID is the value of the time period in the dynamic model.

regressors are the values of regressor variables specified in the MODEL statement.

name if PRED= *name1* and/or RESIDUAL= *name2* options are specified, then *name1* and *name2* are the columns of predicted values of dependent variable and residuals of the regression, respectively.

OUTEST= Data Set

PROC PANEL writes the parameter estimates to an output data set when the OUTEST= option is specified. The OUTEST= data set contains the following variables:

MODEL is a character variable that contains the label for the MODEL statement if a label is specified.

METHOD is a character variable that identifies the estimation method.

TYPE is a character variable that identifies the type of observation. Values of the _TYPE_ variable are CORRB, COVB, CSPARMS, STD, and the type of model estimated. The CORRB observation contains correlations of the parameter estimates, the COVB observation contains covariances of the parameter estimates, the CSPARMS observation contains cross-sectional parameter estimates, the STD observation indicates the row of standard deviations of the corresponding coefficients, and the type of model estimated observation contains the parameter estimates.

NAME is a character variable that contains the name of a regressor variable for COVB and CORRB observations and is left blank for other observations. The _NAME_ variable is used in conjunction with the _TYPE_ values COVB and CORRB to identify rows of the correlation or covariance matrix.

DEPVAR is a character variable that contains the name of the response variable.

MSE is the mean square error of the transformed model.

CSID is the value of the cross section ID for CSPARMS observations. The _CSID_ variable is used with the _TYPE_ value CSPARMS to identify the cross section for the first-order autoregressive parameter estimate contained in the observation. The _CSID_ variable is missing for observations with other _TYPE_ values. (Currently, only the _A_1 variable contains values for CSPARMS observations.)

VARCS is the variance component estimate due to cross sections. The _VARCS_ variable is included in the OUTEST= data set when a one-way or two-way random effects models is estimated.

VARTS is the variance component estimate due to time series. The _VARTS_ variable is included in the OUTEST= data set when a two-way random effects model is estimated.

VARERR is the variance component estimate due to error. The _VARERR_ variable is included in the OUTEST= data set when a one-way or two-way random effects models is estimated.

_A_1 is the first-order autoregressive parameter estimate. The _A_1 variable is included in the OUTEST= data set when the PARKS option is specified. The values of _A_1 are cross-sectional parameters, meaning that they are estimated for each cross section separately. The _A_1 variable has a value only for _TYPE_=CSPARMS observations. The cross section to which the estimate belongs is indicated by the _CSID_ variable.

Intercept is the intercept parameter estimate. (Intercept is missing for models when the NOINT option is specified.)

regressors are the regressor variables specified in the MODEL statement. The regressor variables in the OUTEST= data set contain the corresponding parameter estimates for the model identified by _MODEL_ for _TYPE_=PARMS observations, and the corresponding covariance or correlation matrix elements for _TYPE_=COVB and _TYPE_=CORRB observations. The response variable contains the value–1 for the _TYPE_=PARMS observation for its model.

OUTTRANS= Data Set

PROC PANEL writes the transformed series to an output data set. That is, if the user selects FIXONE, FIXONETIME, or RANONE and supplies the OUTTRANS = option, the transformed dependent variable

and independent variables are written out to a SAS data set; other variables in the input data set are copied unchanged.

Say that your data set contains variables y, x1, x2, x3, and z2. The following statements result in a SAS data set:

```
proc panel data=datain outtrans=dataout;
   id cs ts;
   model y = x1 x2 x3 / fixone;
run;
```

First, z2 is copied over. Then _Int, x1, x2, y, and x3, are replaced with their mean deviates (from cross sections). Furthermore, two new variables are created.

MODELL is the model's label (if it exists).

METHOD is the model's transformation type. In the FIXONE case, this is _FIXONE_ or _FIXONETIME_. If the model RANONE model is selected, the _METHOD_ variable is either _Ran1FB_, _Ran1WK_, _Ran1WH_, or _Ran1NL_, depending on the variance component estimators chosen.

Printed Output

For each MODEL statement, the printed output from PROC PANEL includes the following:

- a model description, which gives the estimation method used, the model statement label if specified, the number of cross sections and the number of observations in each cross section, and the order of moving average error process for the DASILVA option. For fixed-effects model analysis, an F test for the absence of fixed effects is produced, and for random-effects model analysis, a Hausman test is used for the appropriateness of the random-effects specification.
- the estimates of the underlying error structure parameters
- the regression parameter estimates and analysis. For each regressor, this includes the name of the regressor, the degrees of freedom, the parameter estimate, the standard error of the estimate, a t statistic for testing whether the estimate is significantly different from 0, and the significance probability of the t statistic.

Optionally, PROC PANEL prints the following:

- the covariance and correlation of the resulting regression parameter estimates for each model and assumed error structure
- the $\hat{\Phi}$ matrix that is the estimated contemporaneous covariance matrix for the PARKS option

ODS Table Names

PROC PANEL assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in Table 27.8.

Table 27.8 ODS Tables Produced in PROC PANEL

ODS Table Name	Description	Option
ODS Tables Created by the MODEL Statement		
ModelDescription	Model description	Default
FitStatistics	Fit statistics	Default
FixedEffectsTest	F test for no fixed effects	FIXONE,FIXTWO, FIXONETIME
ParameterEstimates	Parameter estimates	Default
CovB	Covariances of parameter estimates	COVB
CorrB	Correlations of parameter estimates	CORRB
VarianceComponents	Variance component estimates	RANONE, RANTWO, DASILVA
RandomEffectsTest	Hausman test for random effects	RANONE, RANTWO
HausmanTest	Hausman specification test	HTAYLOR, AMACURDY
AR1Estimates	First-order autoregressive parameter estimates	RHO(PARKS)
BFNTest	R_ρ statistic for serial correlation	BFN
BL91Test	Baltagi and Li joint LM test	BL91
BL95Test	Baltagi and Li (1995) LM test	BL95
BreuschPaganTest	Breusch-Pagan one-way test	BP
BreuschPaganTest2	Breusch-Pagan two-way test	BP2
BSYTest	Bera, Sosa Escudero, and Yoon modified RS test	BSY
BWTest	Berenblut-Webb statistic for serial correlation	BW
DWTest	Durbin-Watson statistic for serial correlation	DW
GHMTest	Gourieroux, Holly, and Monfort two-way test	GHM
HondaTest	Honda one-way test	HONDA
HondaTest2	Honda two-way test	HONDA2
KingWuTest	King and Wu two-way test	KW
WOOLDTest	Wooldridge (2002) test for unobserved effects	WOOLDRIDGE02
CDTestResults	Cross-sectional dependence test	CDTEST
CDpTestResults	Local cross-sectional dependence test	CDTEST

Table 27.8 (continued)

ODS Table Name	Description	Option
Sargan	Sargan's test for overidentification	GMM1, GMM2, ITGMM
ARTest	Autoregression test for the residuals	GMM1, GMM2, ITGMM
IterHist	Iteration history	ITPRINT(ITGMM)
ConvergenceStatus	Convergence status of iterated GMM estimator	ITGMM
EstimatedPhiMatrix	Estimated phi matrix	PARKS
EstimatedAutocovariances	Estimates of autocovariances	DASILVA
LLCResults	LLC panel unit root test	UROOTTEST
IPSResults	IPS panel unit root test	UROOTTEST
CTResults	Combination test for panel unit root	UROOTTEST
HadriResults	Hadri panel stationarity test	UROOTTEST
HTResults	Harris and Tzavalis panel unit root test	UROOTTEST
BRResults	Breitung panel unit root test	UROOTTEST
URootdetail	Panel unit root test intermediate results	UROOTTEST
PTestResults	Poolability test for panel data	POOLTEST
ODS Tables Created by the COMPARE Statement		
StatComparisonTable	Comparison of model fit statistics	
ParameterComparisonTable	Comparison of model parameter estimates, standard errors, and *t* tests	
ODS Tables Created by the TEST Statement		
TestResults	Test results	

Examples: PANEL Procedure

Example 27.1: Analyzing Demand for Liquid Assets

In this example, the demand equations for liquid assets are estimated. The demand function for the demand deposits is estimated under three error structures while demand equations for time deposits and savings and loan (S&L) association shares are calculated using the Parks method. The data for seven states (CA, DC, FL, IL, NY, TX, and WA) are selected out of 49 states. See Feige (1964) for data description. All variables were transformed via natural logarithm. The data set A is shown below.

```
data a;
   length state $ 2;
   input state $ year d t s y rd rt rs;
   label d = 'Per Capita Demand Deposits'
         t = 'Per Capita Time Deposits'
         s = 'Per Capita S & L Association Shares'
         y = 'Permanent Per Capita Personal Income'
         rd = 'Service Charge on Demand Deposits'
         rt = 'Interest on Time Deposits'
         rs = 'Interest on S & L Association Shares';
datalines;
CA   1949  6.2785  6.1924  4.4998  7.2056 -1.0700  0.1080  1.0664
CA   1950  6.4019  6.2106  4.6821  7.2889 -1.0106  0.1501  1.0767
CA   1951  6.5058  6.2729  4.8598  7.3827 -1.0024  0.4008  1.1291
CA   1952  6.4785  6.2729  5.0039  7.4000 -0.9970  0.4492  1.1227
CA   1953  6.4118  6.2538  5.1761  7.4200 -0.8916  0.4662  1.2110
CA   1954  6.4520  6.2971  5.3613  7.4478 -0.6951  0.4756  1.1924

   ... more lines ...
```

As shown in the following statements, the SORT procedure is used to sort the data into the required time series cross-sectional format; then PROC PANEL analyzes the data.

```
proc sort data=a;
   by state year;
run;

proc panel data=a;
   model d = y rd rt rs / rantwo vcomp = fb parks dasilva m=7;
   model t = y rd rt rs / parks;
   model s = y rd rt rs / parks;
   id state year;
run;
```

The income elasticities for liquid assets are greater than 1 except for the demand deposit income elasticity (0.692757) estimated by the Da Silva method. In Output 27.1.1, Output 27.1.2, and Output 27.1.3, the coefficient estimates (–0.29094, –0.43591, and –0.27736) of demand deposits (RD) imply that demand deposits increase significantly as the service charge is reduced. The price elasticities (0.227152 and 0.408066) for time deposits (RT) and S&L association shares (RS) have the expected sign. Thus an increase in the interest rate on time deposits or S&L shares will increase the demand for the corresponding liquid asset. Demand deposits and S&L shares appear to be substitutes (see Output 27.1.2, Output 27.1.3, and Output 27.1.5). Time deposits are also substitutes for S&L shares in the time deposit demand equation (see Output 27.1.4), while these liquid assets are independent of each other in Output 27.1.5 (insignificant coefficient estimate of RT, −0.02705). Demand deposits and time deposits appear to be weak complements in Output 27.1.3 and Output 27.1.4, while the cross elasticities between demand deposits and time deposits are not significant in Output 27.1.2 and Output 27.1.5.

Output 27.1.1 Demand for Demand Deposits, Fuller-Battese Variance Component with Two-Way Random–Effects Model

The PANEL Procedure
Fuller and Battese Variance Components (RanTwo)

Dependent Variable: d (Per Capita Demand Deposits)

Model Description	
Estimation Method	RanTwo
Number of Cross Sections	7
Time Series Length	11

Fit Statistics			
SSE	0.0795	**DFE**	72
MSE	0.0011	**Root MSE**	0.0332
R-Square	0.6786		

Variance Component Estimates	
Variance Component for Cross Sections	0.03427
Variance Component for Time Series	0.00026
Variance Component for Error	0.00111

Hausman Test for Random Effects			
Coefficients	**DF**	**m Value**	**Pr > m**
4	4	5.51	0.2385

Parameter Estimates

Variable	DF	Estimate	Standard Error	t Value	Pr > |t|	Label
Intercept	1	-1.23606	0.7252	-1.70	0.0926	Intercept
y	1	1.064058	0.1040	10.23	<.0001	Permanent Per Capita Personal Income
rd	1	-0.29094	0.0526	-5.53	<.0001	Service Charge on Demand Deposits
rt	1	0.039388	0.0278	1.42	0.1603	Interest on Time Deposits
rs	1	-0.32662	0.1140	-2.86	0.0055	Interest on S & L Association Shares

Output 27.1.2 Demand for Demand Deposits, Parks Method

The PANEL Procedure
Parks Method Estimation

Dependent Variable: d (Per Capita Demand Deposits)

Model Description	
Estimation Method	Parks
Number of Cross Sections	7
Time Series Length	11

Output 27.1.2 *continued*

Fit Statistics			
SSE	40.0198	**DFE**	72
MSE	0.5558	**Root MSE**	0.7455
R-Square	0.9263		

Parameter Estimates

Variable	DF	Estimate	Standard Error	t Value	Pr > |t|	Label
Intercept	1	-2.66565	0.4250	-6.27	<.0001	Intercept
y	1	1.222569	0.0573	21.33	<.0001	Permanent Per Capita Personal Income
rd	1	-0.43591	0.0272	-16.03	<.0001	Service Charge on Demand Deposits
rt	1	0.041237	0.0284	1.45	0.1505	Interest on Time Deposits
rs	1	-0.26683	0.0886	-3.01	0.0036	Interest on S & L Association Shares

Output 27.1.3 Demand for Demand Deposits, DaSilva Method

The PANEL Procedure
Da Silva Method Estimation

Dependent Variable: d (Per Capita Demand Deposits)

Model Description	
Estimation Method	DaSilva
Number of Cross Sections	7
Time Series Length	11
Order of MA Error Process	7

Fit Statistics			
SSE	21609.8923	**DFE**	72
MSE	300.1374	**Root MSE**	17.3245
R-Square	0.4995		

Variance Component Estimates	
Variance Component for Cross Sections	0.03063
Variance Component for Time Series	0.000148

Estimates of Autocovariances

Lag	Gamma
0	0.0008558553
1	0.0009081747
2	0.0008494797
3	0.0007889687
4	0.0013281983
5	0.0011091685
6	0.0009874973
7	0.0008462601

Output 27.1.3 *continued*

Parameter Estimates

Variable	DF	Estimate	Standard Error	t Value	Pr > |t|	Label
Intercept	1	1.281084	0.0824	15.55	<.0001	Intercept
y	1	0.692757	0.00677	102.40	<.0001	Permanent Per Capita Personal Income
rd	1	-0.27736	0.00274	-101.18	<.0001	Service Charge on Demand Deposits
rt	1	0.009378	0.00171	5.49	<.0001	Interest on Time Deposits
rs	1	-0.09942	0.00601	-16.53	<.0001	Interest on S & L Association Shares

Output 27.1.4 Demand for Time Deposits, Parks Method

The PANEL Procedure
Parks Method Estimation

Dependent Variable: t (Per Capita Time Deposits)

Model Description	
Estimation Method	Parks
Number of Cross Sections	7
Time Series Length	11

Fit Statistics			
SSE	34.5713	**DFE**	72
MSE	0.4802	**Root MSE**	0.6929
R-Square	0.9517		

Parameter Estimates

Variable	DF	Estimate	Standard Error	t Value	Pr > |t|	Label
Intercept	1	-5.33334	0.6780	-7.87	<.0001	Intercept
y	1	1.516344	0.1097	13.82	<.0001	Permanent Per Capita Personal Income
rd	1	-0.04791	0.0399	-1.20	0.2335	Service Charge on Demand Deposits
rt	1	0.227152	0.0449	5.06	<.0001	Interest on Time Deposits
rs	1	-0.42569	0.1708	-2.49	0.0150	Interest on S & L Association Shares

Output 27.1.5 Demand for Savings and Loan Shares, Parks Method

The PANEL Procedure
Parks Method Estimation

Dependent Variable: s (Per Capita S & L Association Shares)

Model Description	
Estimation Method	Parks
Number of Cross Sections	7
Time Series Length	11

Output 27.1.5 *continued*

Fit Statistics			
SSE	39.2550	**DFE**	72
MSE	0.5452	**Root MSE**	0.7384
R-Square	0.9017		

Parameter Estimates

Variable	DF	Estimate	Standard Error	t Value	Pr > |t|	Label
Intercept	1	-8.09632	1.0628	-7.62	<.0001	Intercept
y	1	1.832988	0.1567	11.70	<.0001	Permanent Per Capita Personal Income
rd	1	0.576723	0.0589	9.80	<.0001	Service Charge on Demand Deposits
rt	1	-0.02705	0.0423	-0.64	0.5242	Interest on Time Deposits
rs	1	0.408066	0.1478	2.76	0.0073	Interest on S & L Association Shares

Example 27.2: The Airline Cost Data: Fixtwo Model

The Christenson Associates airline data are a frequently cited data set (see Greene 2000). The data measure costs, prices of inputs, and utilization rates for six airlines over the time span 1970–1984. This example analyzes the log transformations of the cost, price and quantity, and the raw (not logged) capacity utilization measure. You speculate the following model:

$$\begin{aligned} ln\left(TC_{it}\right) = \alpha_N + \gamma_T &+ \left(\alpha_i - \alpha_N\right) + \left(\gamma_t - \gamma_T\right) \\ &+ \beta_1 ln\left(Q_{it}\right) + \beta_2 ln\left(PF_{it}\right) + \beta_3 LF_{it} + \epsilon_{it} \end{aligned}$$

where the α are the pure cross-sectional effects and γ are the time effects. The actual model speculated is highly nonlinear in the original variables. It would look like the following:

$$TC_{it} = exp\left(\alpha_i + \gamma_t + \beta_3 LF_{it} + \epsilon_{it}\right) Q_{it}^{\beta_1} PF_{it}^{\beta_2}$$

The data and preliminary SAS statements are:

```
data airline;
   input  Obs I T C Q PF LF;
   label obs = "Observation number";
   label I  = "Firm Number (CSID)";
   label T  = "Time period (TSID)";
   label Q  = "Output in revenue passenger miles (index)";
   label C  = "Total cost, in thousands";
   label PF = "Fuel price";
   label LF = "Load Factor (utilization index)";
datalines;

   ... more lines ...
```

```
data airline;
   set airline;
   lC = log(C);
   lQ = log(Q);
   lPF = log(PF);
   label lC = "Log Transformation of Costs";
   label lQ = "Log Transformation of Quantity";
   label lPF= "Log Transformation of Price of Fuel";
run;
```

The following statements fit the model:

```
proc panel data=airline printfixed;
   id i t;
   model lC = lQ lPF LF / fixtwo;
run;
```

First, you see the model's description in Output 27.2.1. The model is a two-way fixed-effects model. There are six cross sections and fifteen time observations.

Output 27.2.1 The Airline Cost Data—Model Description

The PANEL Procedure
Fixed Two-Way Estimates

Dependent Variable: lC (Log Transformation of Costs)

Model Description	
Estimation Method	FixTwo
Number of Cross Sections	6
Time Series Length	15

The R-square and degrees of freedom can be seen in Table 27.2.2. On the whole, you see a large R-square, so there is a reasonable fit. The degrees of freedom of the estimate are 90 minus 14 time dummy variables minus 5 cross section dummy variables and 4 regressors.

Output 27.2.2 The Airline Cost Data—Fit Statistics

Fit Statistics			
SSE	0.1768	DFE	67
MSE	0.0026	Root MSE	0.0514
R-Square	0.9984		

The F test for fixed effects is shown in Table 27.2.3. Testing the hypothesis that there are no fixed effects, you easily reject the null of poolability. There are group effects, or time effects, or both. The test is highly significant. OLS would not give reasonable results.

Output 27.2.3 The Airline Cost Data—Test for Fixed Effects

F Test for No Fixed Effects			
Num DF	Den DF	F Value	Pr > F
19	67	23.10	<.0001

Looking at the parameters, you see a more complicated pattern. Most of the cross-sectional effects are highly significant (with the exception of CS2). This means that the cross sections are significantly different from the sixth cross section. Many of the time effects show significance, but this is not uniform. It looks like the significance might be driven by a large 16th period effect, since the first six time effects are negative and of similar magnitude. The time dummy variables taper off in size and lose significance from time period 12 onward. There are many causes to which you could attribute this decay of time effects. The time period of the data spans the OPEC oil embargoes and the dissolution of the Civil Aeronautics Board (CAB). These two forces are two possible reasons to observe the decay and parameter instability. As for the regression parameters, you see that quantity affects cost positively, and the price of fuel has a positive effect, but load factors negatively affect the costs of the airlines in this sample. The somewhat disturbing result is that the fuel cost is not significant. If the time effects are proxies for the effect of the oil embargoes, then an insignificant fuel cost parameter would make some sense. If the dummy variables proxy for the dissolution of the CAB, then the effect of load factors is also not being precisely estimated.

Output 27.2.4 The Airline Cost Data—Parameter Estimates

Parameter Estimates

Variable	DF	Estimate	Standard Error	t Value	Pr > |t|	Label
CS1	1	0.174237	0.0861	2.02	0.0470	Cross Sectional Effect 1
CS2	1	0.111412	0.0780	1.43	0.1576	Cross Sectional Effect 2
CS3	1	-0.14354	0.0519	-2.77	0.0073	Cross Sectional Effect 3
CS4	1	0.18019	0.0321	5.61	<.0001	Cross Sectional Effect 4
CS5	1	-0.04671	0.0225	-2.08	0.0415	Cross Sectional Effect 5
TS1	1	-0.69286	0.3378	-2.05	0.0442	Time Series Effect 1
TS2	1	-0.63816	0.3321	-1.92	0.0589	Time Series Effect 2
TS3	1	-0.59554	0.3294	-1.81	0.0751	Time Series Effect 3
TS4	1	-0.54192	0.3189	-1.70	0.0939	Time Series Effect 4
TS5	1	-0.47288	0.2319	-2.04	0.0454	Time Series Effect 5
TS6	1	-0.42705	0.1884	-2.27	0.0267	Time Series Effect 6
TS7	1	-0.39586	0.1733	-2.28	0.0255	Time Series Effect 7
TS8	1	-0.33972	0.1501	-2.26	0.0269	Time Series Effect 8
TS9	1	-0.2718	0.1348	-2.02	0.0478	Time Series Effect 9
TS10	1	-0.22734	0.0763	-2.98	0.0040	Time Series Effect 10
TS11	1	-0.1118	0.0319	-3.50	0.0008	Time Series Effect 11
TS12	1	-0.03366	0.0429	-0.78	0.4354	Time Series Effect 12
TS13	1	-0.01775	0.0363	-0.49	0.6261	Time Series Effect 13
TS14	1	-0.01865	0.0305	-0.61	0.5430	Time Series Effect 14
Intercept	1	12.93834	2.2181	5.83	<.0001	Intercept
lQ	1	0.817264	0.0318	25.66	<.0001	Log Transformation of Quantity
lPF	1	0.168732	0.1635	1.03	0.3057	Log Transformation of Price of Fuel
LF	1	-0.88267	0.2617	-3.37	0.0012	Load Factor (utilization index)

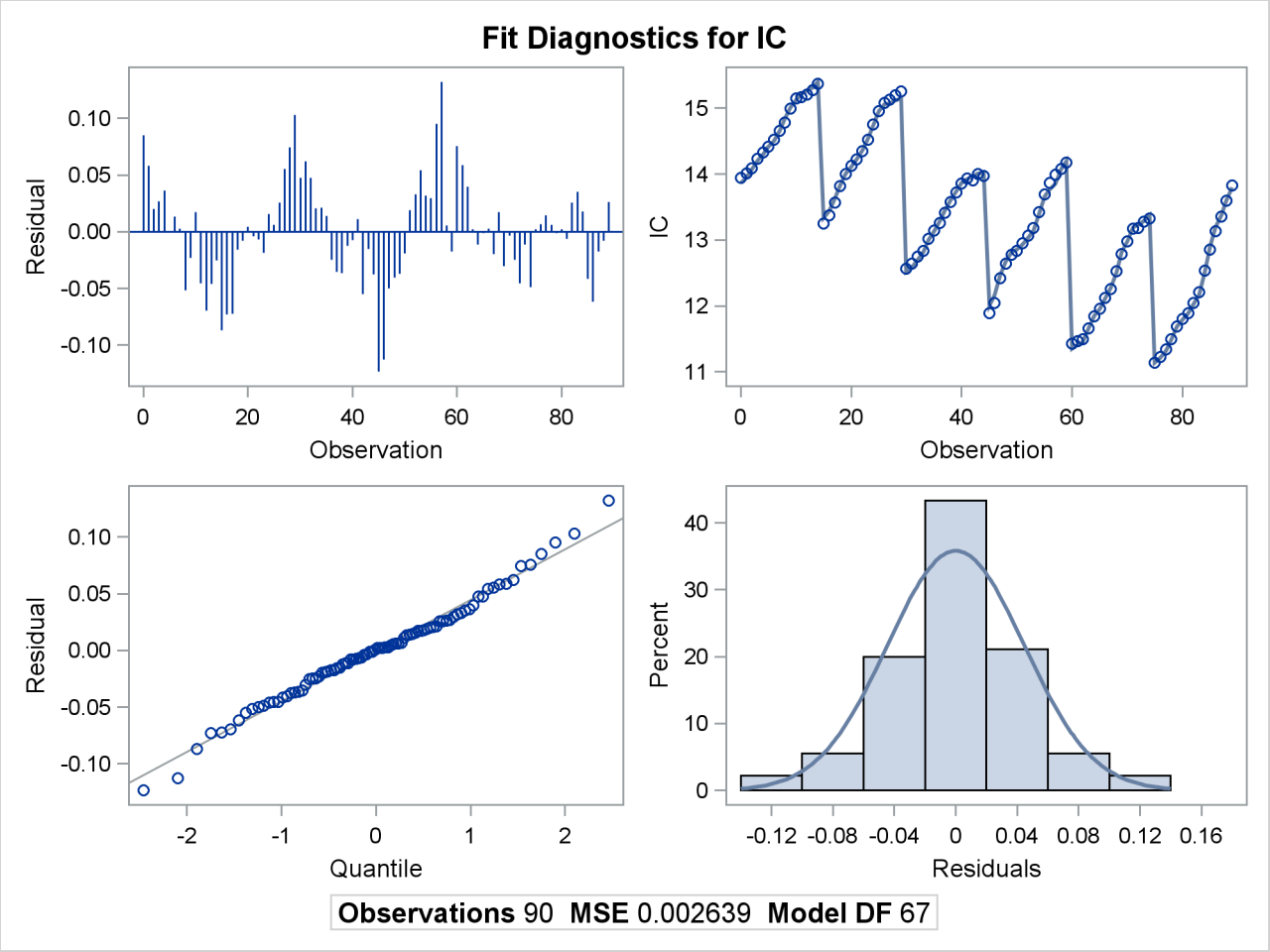
ODS Graphics Plots

ODS graphics plots can be obtained to graphically analyze the results. The following statements show how to generate the plots. If the PLOTS=ALL option is specified, all available plots are produced in two panels. For a complete list of options, see the section "Creating ODS Graphics" on page 1893.

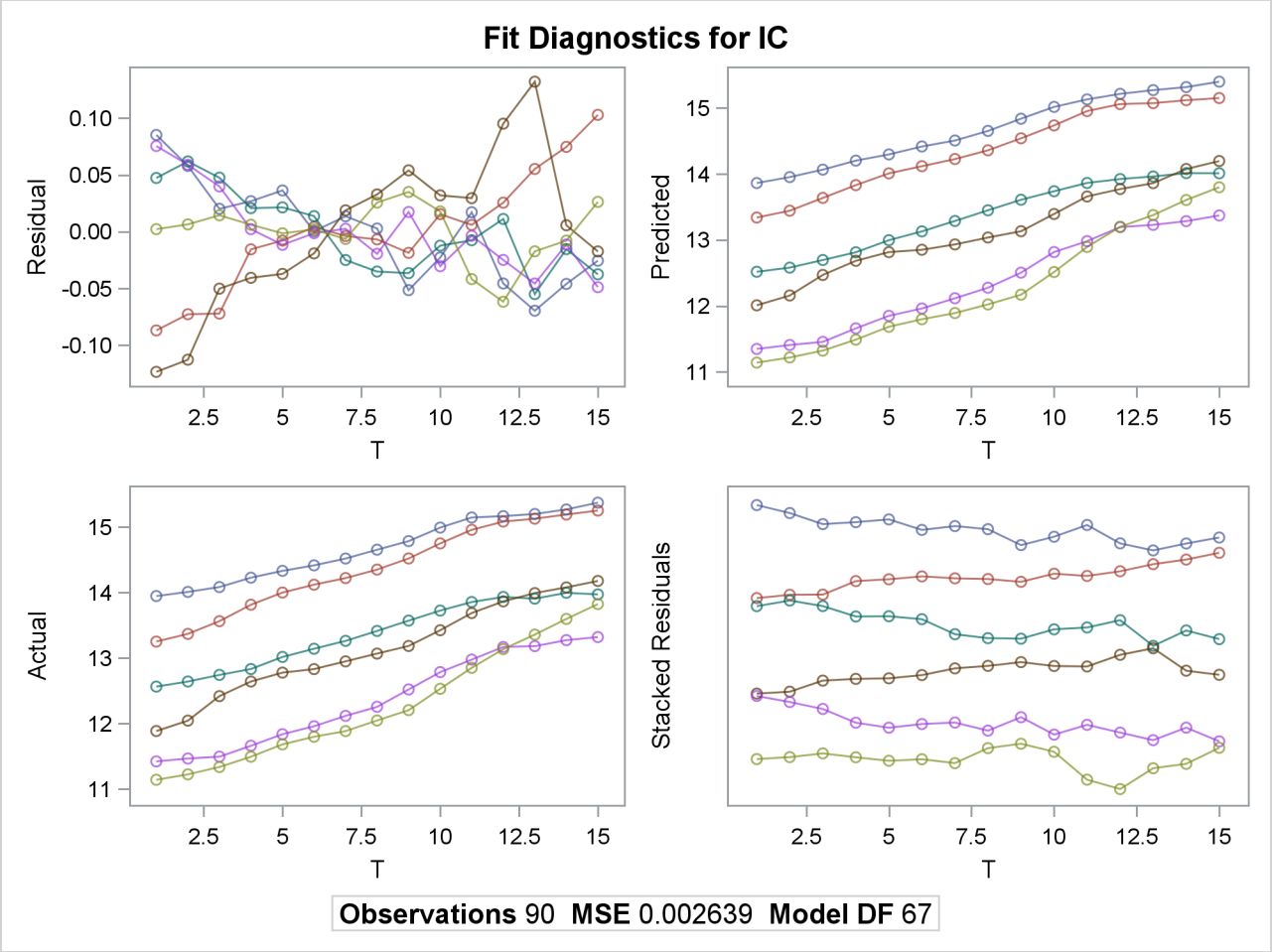
```
ods graphics on;
proc panel data=airline;
   id i t;
   model lC = lQ lPF LF / fixtwo plots = all;
run;
```

The preceding statements result in plots shown in Output 27.2.5 and Output 27.2.6.

Output 27.2.5 Diagnostic Panel 1



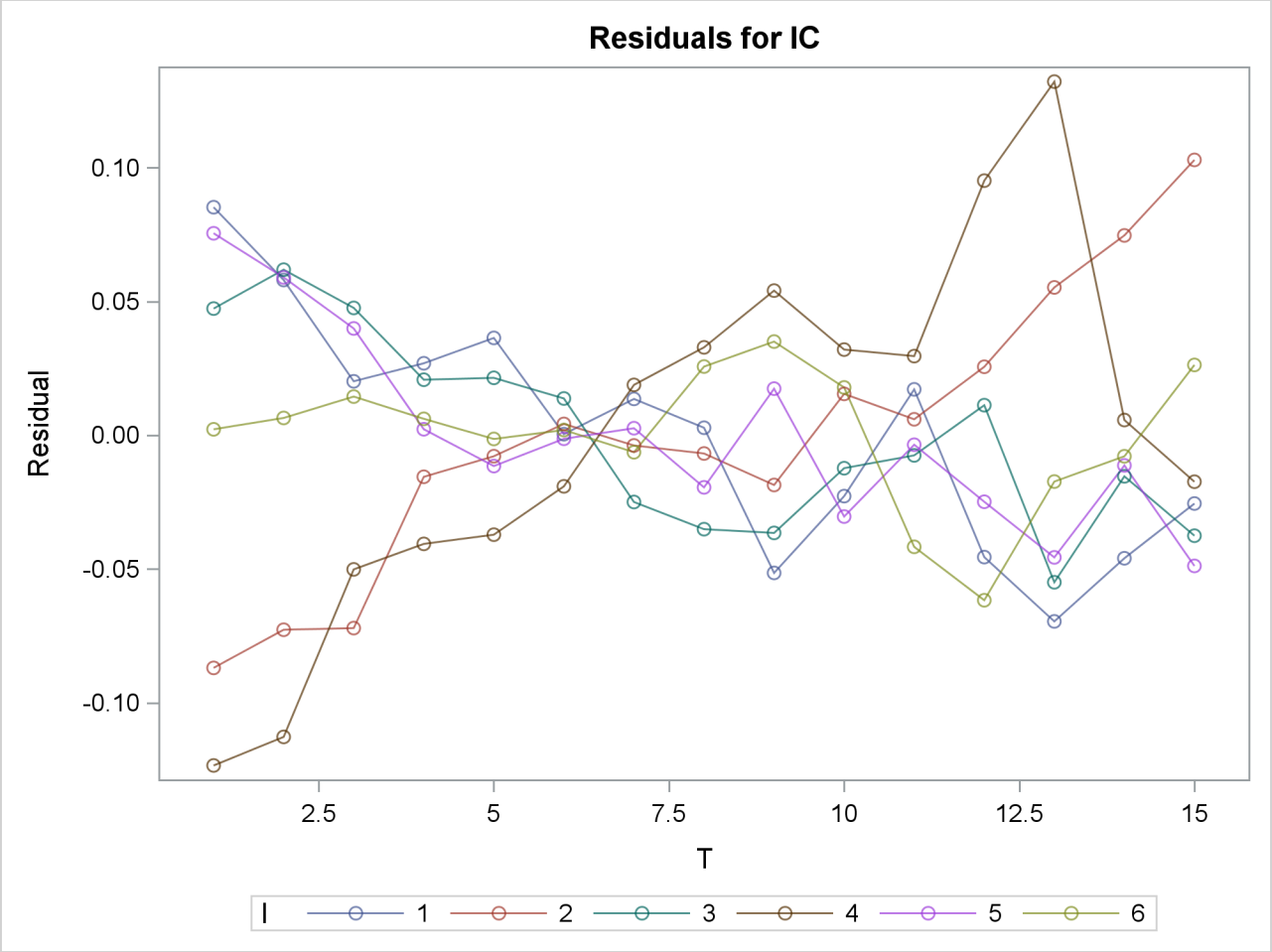
Output 27.2.6 Diagnostic Panel 2



The UNPACK and ONLY options produce individual detail images of paneled plots. The graph shown in Output 27.2.7 shows a detail plot of residuals by cross section. The packed version always puts all cross sections on one plot while the unpacked one shows the cross sections in groups of ten to avoid loss of detail.

```
proc panel data=airline;
   id i t;
   model lC = lQ lPF LF / fixtwo plots(unpack only) = residsurface;
run;
```

Output 27.2.7 Surface Plot of the Residual



Example 27.3: The Airline Cost Data: Further Analysis

Using the same data as in Example 27.2, you further investigate the 'true' effect of fuel prices. Specifically, you run the FixOne model, ignoring time effects. You specify the following statements in PROC PANEL to run this model:

```
proc panel data=airline;
   id i t;
   model lC = lQ lPF LF / fixone;
run;
```

The preceding statements result in Output 27.3.1. The fit seems to have deteriorated somewhat. The SSE rises from 0.1768 to 0.2926.

Output 27.3.1 The Airline Cost Data—Fit Statistics

The PANEL Procedure
Fixed One-Way Estimates

Dependent Variable: lC (Log Transformation of Costs)

Fit Statistics			
SSE	0.2926	**DFE**	81
MSE	0.0036	**Root MSE**	0.0601
R-Square	0.9974		

You still reject poolability based on the F test in Output 27.3.2 at all accepted levels of significance.

Output 27.3.2 The Airline Cost Data—Test for Fixed Effects

F Test for No Fixed Effects			
Num DF	**Den DF**	**F Value**	**Pr > F**
5	81	57.74	<.0001

The parameters change somewhat dramatically as shown in Output 27.3.3. The effect of fuel costs comes in very strong and significant. The load factor's coefficient increases, although not as dramatically. This suggests that the fixed time effects might be proxies for both the oil shocks and deregulation.

Output 27.3.3 The Airline Cost Data—Parameter Estimates

Parameter Estimates						
Variable	**DF**	**Estimate**	**Standard Error**	**t Value**	**Pr > |t|**	**Label**
Intercept	1	9.79304	0.2636	37.15	<.0001	Intercept
lQ	1	0.919293	0.0299	30.76	<.0001	Log Transformation of Quantity
lPF	1	0.417492	0.0152	27.47	<.0001	Log Transformation of Price of Fuel
LF	1	-1.07044	0.2017	-5.31	<.0001	Load Factor (utilization index)

Example 27.4: The Airline Cost Data: Random-Effects Models

This example continues to use the Christenson Associates airline data, which measures costs, prices of inputs, and utilization rates for six airlines over the time span 1970–1984. There are six cross sections and fifteen time observations. Here, you examine the different estimates that are generated from the one-way random-effects and two-way random-effects models, by using four different methods to estimate the variance components: Fuller and Battese, Wansbeek and Kapteyn, Wallace and Hussain, and Nerlove.

The data for this example are created by the PROC PANEL statements shown in Example 27.2. The following PROC PANEL statements generate the estimates:

```
proc panel data=airline;
   id I T;
   model "One-Way, FB"    lC = lQ lPF lF / ranone vcomp=fb;
   model "One-Way, WK"    lC = lQ lPF lF / ranone vcomp=wk;
   model "One-Way, WH"    lC = lQ lPF lF / ranone vcomp=wh;
   model "One-Way, NL"    lC = lQ lPF lF / ranone vcomp=nl;
   model "Two-Way, FB"    lC = lQ lPF lF / rantwo vcomp=fb;
   model "Two-Way, WK"    lC = lQ lPF lF / rantwo vcomp=wk;
   model "Two-Way, WH"    lC = lQ lPF lF / rantwo vcomp=wh;
   model "Two-Way, NL"    lC = lQ lPF lF / rantwo vcomp=nl;
   model "Pooled"         lC = lQ lPF lF / pooled;
   model "Between Groups" lC = lQ lPF lF / btwng;
   model "Between Times"  lC = lQ lPF lF / btwnt;
   compare / pstat(estimate) mstat(varcs varts varerr);
run;
```

The parameter estimates and variance components for all models and estimators are reported in Output 27.4.1 and Output 27.4.2. Both tables are created by the COMPARE statement.

Output 27.4.1 Parameter Estimates

The PANEL Procedure
Model Comparison

Dependent Variable: lC (Log Transformation of Costs)

Comparison of Model Parameter Estimates							
Variable		One-Way, FB RanOne	One-Way, WK RanOne	One-Way, WH RanOne	One-Way, NL RanOne	Two-Way, FB RanTwo	Two-Way, WK RanTwo
Intercept	Estimate	9.637027	9.629542	9.643869	9.640560	9.362705	9.643579
lQ	Estimate	0.908032	0.906926	0.909042	0.908554	0.866458	0.843341
lPF	Estimate	0.422199	0.422676	0.421766	0.421975	0.436160	0.409662
LF	Estimate	-1.064733	-1.064564	-1.064966	-1.064844	-0.980482	-0.926308

Comparison of Model Parameter Estimates						
Variable		Two-Way, WH RanTwo	Two-Way, NL RanTwo	Pooled Pooled	Between Groups BtwGrps	Between Times BtwTime
Intercept	Estimate	9.379328	9.972603	9.516907	85.809402	11.184905
lQ	Estimate	0.869214	0.838724	0.882740	0.782455	1.133318
lPF	Estimate	0.435317	0.382904	0.453978	-5.524011	0.334268
LF	Estimate	-0.985181	-0.913357	-1.627511	-1.750949	-1.350947

Output 27.4.2 Variance Component Estimates

The PANEL Procedure
Model Comparison

Dependent Variable: lC (Log Transformation of Costs)

Comparison of Model Statistics							
Statistic	**One-Way, FB RanOne**	**One-Way, WK RanOne**	**One-Way, WH RanOne**	**One-Way, NL RanOne**	**Two-Way, FB RanTwo**	**Two-Way, WK RanTwo**	**Two-Way, WH RanTwo**
Var due to Cross Sections	0.0182	0.0160	0.0187	0.0174	0.0174	0.0156	0.0187
Var due to Time Series					0.001081	0.0391	0.000854
Var due to Error	0.003612	0.003612	0.003280	0.003251	0.002639	0.002639	0.002502

Comparison of Model Statistics				
Statistic	**Two-Way, NL RanTwo**	**Pooled Pooled**	**Between Groups BtwGrps**	**Between Times BtwTime**
Var due to Cross Sections	0.0171			
Var due to Time Series	0.0591			
Var due to Error	0.001965	0.0155	0.0158	0.000508

In the random-effects model, individual constant terms are viewed as randomly distributed across cross-sectional units. They are not viewed as parametric shifts of the regression function, as they are in the fixed-effects model. This is appropriate when the sampled cross-sectional units are drawn from a large population. In this example, the six airlines are clearly a sample of all the airlines in the industry and not an exhaustive, or nearly exhaustive, list.

There are four techniques for computing the variance components in the one-way random-effects model. The method by Fuller and Battese (1974) (FB) uses a "fitting of constants" method to estimate them, the Wansbeek and Kapteyn (1989) (WK) method uses the true disturbances, the Wallace and Hussain (1969) (WH) method uses ordinary least squares residuals, and the Nerlove (1971) (NL) method uses one-way fixed-effects regression.

Looking at the estimates of the variance components for cross section and error in Output 27.4.2, you see that equal variance components for error for FB and WK are equal, whereas they are nearly equal for WH and NL.

All four techniques produce different variance components for cross sections. These estimates are then used to estimate the values of the parameters in Output 27.4.1. All the parameters appear to have similar and equally plausible estimates. Both the indices for output in revenue passenger miles (lQ) and fuel price (lPF) have small, positive effects on total costs, which you would expect. The load factor (LF) has a somewhat larger and negative effect on total costs, suggesting that as utilization increases, costs decrease.

Comparing the four two-way estimators, you find that the variance components for error produced by the FB and WK methods are equal, as they are in the one-way model. However, in this case, the WH and NL methods produce variance estimates that are dissimilar. The estimates of the variance component for cross sections are all different, but in a close range. The same cannot be said for the variance component for time series. As varied as each of the variance estimates might be, they produce parameter estimates that are similar and plausible. As in the one-way effects model, the indices for output (lQ) and fuel price (lPF) are

small and positive. The load factor (LF) estimates are all negative and somewhat smaller in magnitude than the estimates that are produced in the one-way model. During the time the data were collected, the Civil Aeronautics Board dissolved, so it is possible that the time dummy variables are proxies for this dissolution. The dissolution would lead to the decay of time effects and an imprecise estimation of the effects of the load factors, even though the estimates are statistically significant.

The pooled estimates give you something to compare the random-effects estimates against. You see that signs and magnitudes of output and fuel price are similar but that the magnitude of the load factor coefficient is somewhat larger under pooling. Because the model appears to have both cross-sectional and time series effects, the pooled model should not be used.

Finally, you examine the between estimators. For the between-groups estimate, you are looking at each airline's data averaged across time. You see in Output 27.4.1 that the between-groups parameter estimates are radically different from all other parameter estimates. This difference could indicate that the time component is not being appropriately handled with this technique. For the between-times estimate, you are looking at the average across all airlines in each time period. In this case, the parameter estimates are of the same sign and closer in magnitude to the previously computed estimates. Both the output and load factor effects appear to have more bearing on total costs.

Example 27.5: Panel Study of Income Dynamics (PSID): Hausman-Taylor Models

Cornwell and Rupert (1988) analyze data from the Panel Study of Income Dynamics (PSID), an income study of 595 individuals over the seven-year period, 1976–1982 inclusive. Of particular interest is the effect of additional schooling on wages. The analysis here replicates that of Baltagi (2008, sec. 7.5), where it is surmised that covariate correlation with individual effects makes a standard random-effects model inadequate.

The following statements create the PSID data:

```
data psid;
   input id t lwage wks south smsa ms exp exp2 occ ind union fem blk ed;
   label id    = 'Person ID'
         t     = 'Time'
         lwage = 'Log(wages)'
         wks   = 'Weeks worked'
         south = '1 if resides in the South'
         smsa  = '1 if resides in SMSA'
         ms    = '1 if married'
         exp   = 'Years full-time experience'
         exp2  = 'exp squared'
         occ   = '1 if blue-collar occupation'
         ind   = '1 if manufacturing'
         union = '1 if union contract'
         fem   = '1 if female'
         blk   = '1 if black'
         ed    = 'Years of education';
datalines;
1    1  5.5606799126  32  1  0  1  3   9     0  0  0  0  0  9
1    2  5.7203102112  43  1  0  1  4   16    0  0  0  0  0  9
1    3  5.9964499474  40  1  0  1  5   25    0  0  0  0  0  9
```

```
1    4  5.9964499474  39  1  0  1  6   36    0  0  0  0  0  9
1    5  6.0614600182  42  1  0  1  7   49    0  1  0  0  0  9
1    6  6.1737899780  35  1  0  1  8   64    0  1  0  0  0  9
1    7  6.2441701889  32  1  0  1  9   81    0  1  0  0  0  9
2    1  6.1633100510  34  0  0  1  30  900   1  0  0  0  0  11
2    2  6.2146100998  27  0  0  1  31  961   1  0  0  0  0  11
2    3  6.2634000778  33  0  0  1  32  1024  1  1  1  0  0  11
2    4  6.5439100266  30  0  0  1  33  1089  1  1  0  0  0  11
2    5  6.6970300674  30  0  0  1  34  1156  1  1  0  0  0  11
2    6  6.7912201881  37  0  0  1  35  1225  1  1  0  0  0  11
2    7  6.8156399727  30  0  0  1  36  1296  1  1  0  0  0  11

   ... more lines ...
```

You begin by fitting a one-way random effects model:

```
proc panel data=psid;
   id id t;
   model lwage = wks south smsa ms exp exp2 occ
                          ind union fem blk ed / ranone;
run;
```

The output is shown in Output 27.5.1. The coefficient on the variable ED (which represents years of education) estimates that an additional year of schooling is associated with about a 10.7% increase in wages. However, the results of the Hausman test for random effects show a serious violation of the random-effects assumptions, namely that the regressors are independent of both error components.

Output 27.5.1 One-Way Random Effects Estimation

The PANEL Procedure
Fuller and Battese Variance Components (RanOne)

Dependent Variable: lwage (Log(wages))

Model Description	
Estimation Method	RanOne
Number of Cross Sections	595
Time Series Length	7

Variance Component Estimates	
Variance Component for Cross Sections	0.100553
Variance Component for Error	0.023102

Hausman Test for Random Effects			
Coefficients	DF	m Value	Pr > m
9	9	5288.98	<.0001

Output 27.5.1 *continued*

Parameter Estimates

Variable	DF	Estimate	Standard Error	t Value	Pr > |t|	Label
Intercept	1	4.030811	0.1044	38.59	<.0001	Intercept
wks	1	0.000954	0.000740	1.29	0.1971	Weeks worked
south	1	-0.00788	0.0281	-0.28	0.7795	1 if resides in the South
smsa	1	-0.02898	0.0202	-1.43	0.1517	1 if resides in SMSA
ms	1	-0.07067	0.0224	-3.16	0.0016	1 if married
exp	1	0.087726	0.00281	31.27	<.0001	Years full-time experience
exp2	1	-0.00076	0.000062	-12.31	<.0001	exp squared
occ	1	-0.04293	0.0162	-2.65	0.0081	1 if blue-collar occupation
ind	1	0.00381	0.0172	0.22	0.8242	1 if manufacturing
union	1	0.058121	0.0169	3.45	0.0006	1 if union contract
fem	1	-0.30791	0.0572	-5.38	<.0001	1 if female
blk	1	-0.21995	0.0660	-3.33	0.0009	1 if black
ed	1	0.10742	0.00642	16.73	<.0001	Years of education

An alternative could be a fixed-effects (FIXONE) model, but that would not permit estimation of the coefficient for ED, which does not vary within individuals. A compromise is the Hausman-Taylor model, for which you stipulate a set of covariates that are correlated with the individual effects (but uncorrelated with the observation-level errors). You specify the correlated variables in the CORRELATED= option in the INSTRUMENTS statement:

```
proc panel data=psid;
   id id t;
   instruments correlated = (wks ms exp exp2 union ed);
   model lwage = wks south smsa ms exp exp2 occ
                          ind union fem blk ed / htaylor;
run;
```

The results are shown in Output 27.5.2. The table of parameter estimates has an added column, Type, that identifies which regressors are assumed to be correlated with individual effects (C) and which regressors do not vary within cross sections (TI). It was stated previously that the Hausman-Taylor model is a compromise between fixed-effects and random-effects models, and you can think of the compromise this way: You want to fit a random-effects model, but the correlated (C) variables make that model invalid. So you fall back to the consistent fixed-effects model, but then the time-invariant (TI) variables are the problem because they will be dropped from that model. The solution is to use the Hausman-Taylor estimator.

The estimation results show that an additional year of schooling is now associated with a 13.8% increase in wages. Also presented is a Hausman test that compares this model to the fixed-effects model. As was the case previously when you fit the random-effects model, you can think of the Hausman test as a referendum on the assumptions you are making. For this estimation, it seems that your choice of variables to treat as correlated is adequate. It also seems to hold true that any correlation is with the individual-level effects, and not the observation-level errors.

Output 27.5.2 Hausman-Taylor Estimation

The PANEL Procedure
Hausman and Taylor Model for Correlated Individual Effects (HTaylor)

Dependent Variable: lwage (Log(wages))

Variance Component Estimates	
Variance Component for Cross Sections	0.886993
Variance Component for Error	0.023044

Hausman Test against Fixed Effects			
Coefficients	DF	m Value	Pr > m
9	3	5.26	0.1539

Parameter Estimates

Variable	Type		DF	Estimate	Standard Error	t Value	Pr > |t|	Label
Intercept			1	2.912726	0.2837	10.27	<.0001	Intercept
wks	C		1	0.000837	0.000600	1.40	0.1627	Weeks worked
south			1	0.00744	0.0320	0.23	0.8159	1 if resides in the South
smsa			1	-0.04183	0.0190	-2.21	0.0274	1 if resides in SMSA
ms	C		1	-0.02985	0.0190	-1.57	0.1159	1 if married
exp	C		1	0.113133	0.00247	45.79	<.0001	Years full-time experience
exp2	C		1	-0.00042	0.000055	-7.67	<.0001	exp squared
occ			1	-0.0207	0.0138	-1.50	0.1331	1 if blue-collar occupation
ind			1	0.013604	0.0152	0.89	0.3720	1 if manufacturing
union	C		1	0.032771	0.0149	2.20	0.0280	1 if union contract
fem		TI	1	-0.13092	0.1267	-1.03	0.3014	1 if female
blk		TI	1	-0.28575	0.1557	-1.84	0.0665	1 if black
ed	C	TI	1	0.137944	0.0212	6.49	<.0001	Years of education

C: correlated with the individual effects
TI: constant (time-invariant) within cross sections

At its core, the Hausman-Taylor estimator is an instrumental variables regression, where the instruments are derived from regressors that are assumed to be uncorrelated with the individual effects. Technically it is the cross-sectional means of these variables that need to be uncorrelated, not the variables themselves.

The Amemiya-MaCurdy model is a close relative of the Hausman-Taylor model. The only difference between the two is that the Amemiya-MaCurdy model makes the added assumption that the regressors (and not just their means) are uncorrelated with the individual effects. By making that assumption, the Amemiya-MaCurdy model can take advantage of a more efficient set of instrumental variables.

The following statements fit the Amemiya-MaCurdy model:

```
proc panel data=psid;
   id id t;
   instruments correlated = (wks ms exp exp2 union ed);
   model lwage = wks south smsa ms exp exp2 occ
                          ind union fem blk ed / amacurdy;
run;
```

The results are shown in Output 27.5.3. Little is changed from the Hausman-Taylor model. The Hausman test compares the Amemiya-MaCurdy model to the Hausman-Taylor model (not the fixed-effects model as previously) and shows that the one additional assumption is acceptable. You even gained a bit of efficiency in the process; compare the standard deviations of the coefficient on the variable ED from both models.

Output 27.5.3 Amemiya-MaCurdy Estimation

The PANEL Procedure
Amemiya and MaCurdy Model for Correlated Individual Effects (AMaCurdy)

Dependent Variable: lwage (Log(wages))

Variance Component Estimates	
Variance Component for Cross Sections	0.886993
Variance Component for Error	0.023044

Hausman Test against Hausman-Taylor			
Coefficients	DF	m Value	Pr > m
13	13	14.67	0.3287

Parameter Estimates

Variable	Type		DF	Estimate	Standard Error	t Value	Pr > |t|	Label
Intercept			1	2.927338	0.2751	10.64	<.0001	Intercept
wks	C		1	0.000838	0.000599	1.40	0.1622	Weeks worked
south			1	0.007282	0.0319	0.23	0.8197	1 if resides in the South
smsa			1	-0.04195	0.0189	-2.21	0.0269	1 if resides in SMSA
ms	C		1	-0.03009	0.0190	-1.59	0.1127	1 if married
exp	C		1	0.11297	0.00247	45.76	<.0001	Years full-time experience
exp2	C		1	-0.00042	0.000055	-7.72	<.0001	exp squared
occ			1	-0.02085	0.0138	-1.51	0.1299	1 if blue-collar occupation
ind			1	0.013629	0.0152	0.89	0.3709	1 if manufacturing
union	C		1	0.032475	0.0149	2.18	0.0293	1 if union contract
fem		TI	1	-0.13201	0.1266	-1.04	0.2972	1 if female
blk		TI	1	-0.2859	0.1555	-1.84	0.0660	1 if black
ed	C	TI	1	0.137205	0.0206	6.67	<.0001	Years of education

C: correlated with the individual effects
TI: constant (time-invariant) within cross sections

Finally, you should realize that the Hausman-Taylor and Amemiya-MaCurdy estimators are not cure-alls for correlated individual effects. Estimation tacitly relies on the uncorrelated regressors being sufficient to predict the correlated regressors. Otherwise, you run into the problem of weak instruments. If you have weak instruments, you will obtain biased estimates that have very large standard errors. However, that does not seem to be the case here.

Example 27.6: The Cigarette Sales Data: Dynamic Panel Estimation with GMM

In this example, a dynamic panel demand model for cigarette sales is estimated. It illustrates the application of the method described in the section "Dynamic Panel Estimators" on page 1855. The data are a panel from

46 American states over the period 1963–92. For data description see: Baltagi and Levin (1992); (Baltagi 2008, sec. 8.8). All variables were transformed by taking the natural logarithm. The data set CIGAR is shown in the following statements:

```
data cigar;
   input state year price pop pop_16 cpi ndi sales pimin;
   label
   state  = 'State abbreviation'
   year   = 'YEAR'
   price  = 'Price per pack of cigarettes'
   pop    = 'Population'
   pop_16 = 'Population above the age of 16'
   cpi    = 'Consumer price index with (1983=100)'
   ndi    = 'Per capita disposable income'
   sales  = 'Cigarette sales in packs per capita'
   pimin  = 'Minimum price in adjoining states per pack of cigarettes';
datalines;
1 63 28.6 3383 2236.5 30.6 1558.3045298 93.9 26.1
1 64 29.8 3431 2276.7 31.0 1684.0732025 95.4 27.5
1 65 29.8 3486 2327.5 31.5 1809.8418752 98.5 28.9
1 66 31.5 3524 2369.7 32.4 1915.1603572 96.4 29.5
1 67 31.6 3533 2393.7 33.4 2023.5463678 95.5 29.6
1 68 35.6 3522 2405.2 34.8 2202.4855362 88.4 32
1 69 36.6 3531 2411.9 36.7 2377.3346665 90.1 32.8
1 70 39.6 3444 2394.6 38.8 2591.0391591 89.8 34.3
1 71 42.7 3481 2443.5 40.5 2785.3159706 95.4 35.8
1 72 42.3 3511 2484.7 41.8 3034.8082969 101.1 37.4

   ... more lines ...
```

The following statements sort the data by STATE and YEAR variables:

```
proc sort data=cigar;
   by state year;
run;
```

Next, logarithms of the variables required for regression estimation are calculated, as shown in the following statements:

```
data cigar;
   set cigar;
   lsales = log(sales);
   lprice = log(price);
   lndi = log(ndi);
   lpimin = log(pimin);
   label lprice = 'Log price per pack of cigarettes';
   label lndi = 'Log per capita disposable income';
   label lsales = 'Log cigarette sales in packs per capita';
   label lpimin = 'Log minimum price in adjoining states
                     per pack of cigarettes';
run;
```

The following statements create the CIGAR_LAG data set with lagged variable for each cross section:

```
proc panel data=cigar;
   id state year;
   clag lsales(1) / out=cigar_lag;
run;

data cigar_lag;
   set cigar_lag;
   label lsales_1 = 'Lagged log cigarette sales in packs per capita';
run;
```

Finally, the model is estimated by a two step GMM method. Five lags (MAXBAND=5) of the dependent variable are used as instruments. NOLEVELS options is specified to avoid use of level equations, as shown in the following statements:

```
proc panel data=cigar_lag;
   inst depvar;
   model lsales = lsales_1 lprice lndi lpimin
         / gmm2 nolevels maxband=5 noint;
   id state year;
run;
```

Output 27.6.1 Estimation with GMM

The PANEL Procedure
GMM: First Differences Transformation

Dependent Variable: lsales (Log cigarette sales in packs per capita)

Model Description	
Estimation Method	GMM2
Number of Cross Sections	46
Time Series Length	30
Estimate Stage	2
Maximum Number of Time Periods (MAXBAND)	5

Fit Statistics			
SSE	2187.5988	DFE	1284
MSE	1.7037	Root MSE	1.3053

Parameter Estimates

Variable	DF	Estimate	Standard Error	t Value	Pr > |t|
lsales_1	1	0.572219	0.000981	583.51	<.0001
lprice	1	-0.23464	0.00306	-76.56	<.0001
lndi	1	0.232673	0.000392	593.69	<.0001
lpimin	1	-0.08299	0.00328	-25.29	<.0001

If the theory suggests that there are other valid instruments, PREDETERMINED, EXOGENOUS and CORRELATED options can also be used.

Example 27.7: Using the FLATDATA Statement

Sometimes the data can be found in compressed form, where each line consists of all observations for the dependent and independent variables for the cross section. To illustrate, suppose you have a data set with 20 cross sections where each cross section consists of observations for six time periods. Each time period has values for dependent and independent variables $Y_1 \ldots Y_6$ and $X_1 \ldots X_6$. The cs and num variables represent other character and numeric variables that are constant across each cross section.

The observations for first five cross sections along with other variables are shown in Output 27.7.1. In this example, *i* represents the cross section. The time period is identified by the subscript on the *Y* and *X* variables; it ranges from 1 to 6.

Output 27.7.1 Compressed Data Set

Obs	i	cs	num	X_1	X_2	X_3	X_4	X_5	X_6	Y_1	Y_2
1	1	CS1	-1.56058	0.40268	0.91951	0.69482	-2.28899	-1.32762	1.92348	2.30418	2.11850
2	2	CS2	0.30989	1.01950	-0.04699	-0.96695	-1.08345	-0.05180	0.30266	4.50982	3.73887
3	3	CS3	0.85054	0.60325	0.71154	0.66168	-0.66823	-1.87550	0.55065	4.07276	4.89621
4	4	CS4	-0.18885	-0.64946	-1.23355	0.04554	-0.24996	0.09685	-0.92771	2.40304	1.48182
5	5	CS5	-0.04761	-0.79692	0.63445	-2.23539	-0.37629	-0.82212	-0.70566	3.58092	6.08917

Obs	Y_3	Y_4	Y_5	Y_6
1	2.66009	-4.94104	-0.83053	5.01359
2	1.44984	-1.02996	2.78260	1.73856
3	3.90470	1.03437	0.54598	5.01460
4	2.70579	3.82672	4.01117	1.97639
5	3.08249	4.26605	3.65452	0.81826

Since the PANEL procedure cannot work directly with the data in compressed form, the FLATDATA statement can be used to transform the data. The OUT= option can be used to output transformed data to a data set.

```
proc panel data=flattest;
   flatdata indid=i tsname="t" base=(X Y)
            keep=( cs num seed )  / out=flat_out;
   id i t;
   model y = x / fixone noint;
run;
```

First, six observations for the uncompressed data set and results for the one-way fixed-effects model fitted are shown in Output 27.7.2 and Output 27.7.3.

Output 27.7.2 Uncompressed Data Set

Obs	I	t	X	Y	CS	NUM
1	1	1	0.40268	2.30418	CS1	-1.56058
2	1	2	0.91951	2.11850	CS1	-1.56058
3	1	3	0.69482	2.66009	CS1	-1.56058
4	1	4	-2.28899	-4.94104	CS1	-1.56058
5	1	5	-1.32762	-0.83053	CS1	-1.56058
6	1	6	1.92348	5.01359	CS1	-1.56058

Output 27.7.3 Estimation with the FLATDATA Statement

The PANEL Procedure
Fixed One-Way Estimates

Dependent Variable: Y

Parameter Estimates

Variable	DF	Estimate	Standard Error	t Value	Pr > |t|	Label
X	1	2.010753	0.1217	16.52	<.0001	

References

Amemiya, T., and MaCurdy, T. E. (1986). "Instrumental-Variable Estimation of an Error-Components Model." *Econometrica* 54:869–880.

Andrews, D. W. K. (1991). "Heteroscedasticity and Autocorrelation Consistent Covariance Matrix Estimation." *Econometrica* 59:817–858.

Arellano, M. (1987). "Computing Robust Standard Errors for Within-Groups Estimators." *Oxford Bulletin of Economics and Statistics* 49:431–434.

Arellano, M., and Bond, S. (1991). "Some Tests of Specification for Panel Data: Monte Carlo Evidence and an Application to Employment Equations." *Review of Economic Studies* 58:277–297.

Arellano, M., and Bover, O. (1995). "Another Look at the Instrumental Variable Estimation of Error-Components Models." *Journal of Econometrics* 68:29–51.

Baltagi, B. H. (2008). *Econometric Analysis of Panel Data*. 4th ed. Chichester, UK: John Wiley & Sons.

Baltagi, B. H., and Chang, Y. (1994). "Incomplete Panels: A Comparative Study of Alternative Estimators for the Unbalanced One-Way Error Component Regression Model." *Journal of Econometrics* 62:67–89.

Baltagi, B. H., Chang, Y. J., and Li, Q. (1992). "Monte Carlo Results on Several New and Existing Tests for the Error Component Model." *Journal of Econometrics* 54:95–120.

Baltagi, B. H., and Levin, D. (1992). "Cigarette Taxation: Raising Revenues and Reducing Consumption." *Structural Change and Economic Dynamics* 3:321–335.

Baltagi, B. H., and Li, Q. (1991). "A Joint Test for Serial Correlation and Random Individual Effects." *Statistics and Probability Letters* 11:277–280.

Baltagi, B. H., and Li, Q. (1995). "Testing AR(1) against MA(1) Disturbances in an Error Component Model." *Journal of Econometrics* 68:133–151.

Baltagi, B. H., Song, S. H., and Jung, B. C. (2002). "A Comparative Study of Alternative Estimators for the Unbalanced Two-Way Error Component Regression Model." *Econometrics Journal* 5:480–493.

Bera, A. K., Sosa Escudero, W., and Yoon, M. (2001). "Tests for the Error Component Model in the Presence of Local Misspecification." *Journal of Econometrics* 101:1–23.

Bhargava, A., Franzini, L., and Narendranathan, W. (1982). "Serial Correlation and Fixed Effects Model." *Review of Economic Studies* 49:533–549.

Blundell, R., and Bond, S. (1998). "Initial Conditions and Moment Restrictions in Dynamic Panel Data Models." *Journal of Econometrics* 87:115–143.

Breitung, J. (2000). "The Local Power of Some Unit Root Tests for Panel Data." In *Nonstationary Panels, Panel Cointegration, and Dynamic Panels*, edited by B. H. Baltagi, 161–178. Vol. 15 of Advances in Econometrics. Amsterdam: JAI Press.

Breitung, J., and Das, S. (2005). "Panel Unit Root Tests under Cross-Sectional Dependence." *Statistica Neerlandica* 59:414–433.

Breitung, J., and Meyer, W. (1994). "Testing for Unit Roots in Panel Data: Are Wages on Different Bargaining Levels Cointegrated?" *Applied Economics* 26:353–361.

Breusch, T. S., and Pagan, A. R. (1980). "The Lagrange Multiplier Test and Its Applications to Model Specification in Econometrics." *Review of Econometric Studies* 47:239–253.

Buse, A. (1973). "Goodness of Fit in Generalized Least Squares Estimation." *American Statistician* 27:106–108.

Campbell, J. Y., and Perron, P. (1991). "Pitfalls and Opportunities: What Macroeconomists Should Know about Unit Roots." In *NBER Macroeconomics Annual*, edited by O. Blanchard, and S. Fisher, 141–201. Cambridge, MA: MIT Press.

Choi, I. (2001). "Unit Root Tests for Panel Data." *Journal of International Money and Finance* 20:249–272.

Choi, I. (2006). "Nonstationary Panels." In *Econometric Theory*, edited by T. C. Mills, and K. Patterson, 511–539. Vol. 1 of Palgrave Handbook of Econometrics. Basingstoke, UK: Palgrave Macmillan.

Chow, G. (1960). "Tests of Equality between Sets of Coefficients in Two Linear Regressions." *Econometrica* 28:531–534.

Cornwell, C., and Rupert, P. (1988). "Efficient Estimation with Panel Data: An Empirical Comparison of Instrumental Variables Estimators." *Journal of Applied Econometrics* 3:149–155.

Da Silva, J. G. C. (1975). *The Analysis of Cross-Sectional Time Series Data*. Ph.D. diss., Department of Statistics, North Carolina State University.

Davidson, R., and MacKinnon, J. G. (1993). *Estimation and Inference in Econometrics*. New York: Oxford University Press.

Davis, P. (2002). "Estimating Multi-way Error Components Models with Unbalanced Data Structures." *Journal of Econometrics* 106:67–95.

Elliott, G., Rothenberg, T. J., and Stock, J. H. (1996). "Efficient Tests for an Autoregressive Unit Root." *Econometrica* 64:813–836.

Feige, E. L. (1964). *The Demand for Liquid Assets: A Temporal Cross-Section Analysis*. Englewood Cliffs, NJ: Prentice-Hall.

Feige, E. L., and Swamy, P. A. (1974). "A Random Coefficient Model of the Demand for Liquid Assets." *Journal of Money, Credit, and Banking* 6:241–252.

Fisher, R. A. (1932). *Statistical Methods for Research Workers*. 4th ed. London: Oliver & Boyd.

Fuller, W. A., and Battese, G. E. (1974). "Estimation of Linear Models with Crossed-Error Structure." *Journal of Econometrics* 2:67–78.

Gardner, R. (1998). "Unobservable Individual Effects in Unbalanced Panel Data." *Economics Letters* 58:39–42.

Gourieroux, C., Holly, A., and Monfort, A. (1982). "Likelihood Ratio Test, Wald Test, and Kuhn-Tucker Test in Linear Models with Inequality Constraints on the Regression Parameters." *Econometrica* 50:63–80.

Greene, W. H. (1990). *Econometric Analysis*. New York: Macmillan.

Greene, W. H. (2000). *Econometric Analysis*. 4th ed. Upper Saddle River, NJ: Prentice-Hall.

Hadri, K. (2000). "Testing for Stationarity in Heterogeneous Panel Data." *Econometrics Journal* 3:148–161.

Hall, A. R. (1994). "Testing for a Unit Root with Pretest Data Based Model Selection." *Journal of Business and Economic Statistics* 12:461–470.

Hamilton, J. D. (1994). *Time Series Analysis*. Princeton, NJ: Princeton University Press.

Hannan, E. J., and Quinn, B. G. (1979). "The Determination of the Order of an Autoregression." *Journal of the Royal Statistical Society, Series B* 41:190–195.

Harris, R. D. F., and Tzavalis, E. (1999). "Inference for Unit Roots in Dynamic Panels Where the Time Dimension Is Fixed." *Journal of Econometrics* 91:201–226.

Hausman, J. A. (1978). "Specification Tests in Econometrics." *Econometrica* 46:1251–1271.

Hausman, J. A., and Taylor, W. E. (1981). "Panel Data and Unobservable Individual Effects." *Econometrica* 49:1377–1398.

Hausman, J. A., and Taylor, W. E. (1982). "A Generalized Specification Test." *Economics Letters* 8:239–245.

Honda, Y. (1985). "Testing the Error Components Model with Non-normal Disturbances." *Review of Economic Studies* 52:681–690.

Honda, Y. (1991). "A Standardized Test for the Error Components Model with the Two-Way Layout." *Economics Letters* 37:125–128.

Hsiao, C. (1986). *Analysis of Panel Data*. Cambridge: Cambridge University Press.

Im, K. S., Pesaran, M. H., and Shin, Y. (2003). "Testing for Unit Root in Heterogeneous Panels." *Journal of Econometrics* 115:53–74.

Judge, G. G., Griffiths, W. E., Hill, R. C., Lütkepohl, H., and Lee, T.-C. (1985). *The Theory and Practice of Econometrics*. 2nd ed. New York: John Wiley & Sons.

King, M. L., and Wu, P. X. (1997). "Locally Optimal One-Sided Tests for Multiparameter Hypotheses." *Econometric Reviews* 16:131–156.

Kmenta, J. (1971). *Elements of Econometrics*. New York: Macmillan.

LaMotte, L. R. (1994). "A Note on the Role of Independence in *t* Statistics Constructed from Linear Statistics in Regression Models." *American Statistician* 48:238–240.

Levin, A., Lin, C.-F., and Chu, C. S. (2002). "Unit Root Tests in Panel Data: Asymptotic and Finite-Sample Properties." *Journal of Econometrics* 108:1–24.

Maddala, G. S. (1977). *Econometrics*. New York: McGraw-Hill.

Maddala, G. S., and Wu, S. (1999). "A Comparative Study of Unit Root Tests with Panel Data and a New Simple Test." *Oxford Bulletin of Economics and Statistics* 61:631–652.

Moulton, B. R., and Randolph, W. C. (1989). "Alternative Tests of the Error Components Model." *Econometrica* 57:685–693.

Nabeya, S. (1999). "Asymptotic Moments of Some Unit Root Test Statistics in the Null Case." *Econometric Theory* 15:139–149.

Nerlove, M. (1971). "Further Evidence on the Estimation of Dynamic Relations from a Time Series of Cross Sections." *Econometrica* 39:359–382.

Newey, W. K., and West, D. W. (1994). "Automatic Lag Selection in Covariance Matrix Estimation." *Review of Economic Studies* 61:631–653.

Ng, S., and Perron, P. (2001). "Lag Length Selection and the Construction of Unit Root Tests with Good Size and Power." *Econometrica* 69:1519–1554.

Parks, R. W. (1967). "Efficient Estimation of a System of Regression Equations When Disturbances Are Both Serially and Contemporaneously Correlated." *Journal of the American Statistical Association* 62:500–509.

Pesaran, M. H. (2004). "General Diagnostic Tests for Cross Section Dependence in Panels." Institute for the Study of Labor (IZA) Discussion Paper No. 1240, CESifo Working Paper No. 1229, Department of Applied Economics, University of Cambridge.

Phillips, P. C. B., and Perron, P. (1988). "Testing for a Unit Root in Time Series Regression." *Biometrika* 75:335–346.

Roy, S. N. (1957). *Some Aspects of Multivariate Analysis*. New York: John Wiley & Sons.

Searle, S. R. (1971). "Topics in Variance Component Estimation." *Biometrics* 26:1–76.

Seely, J. (1969). *Estimation in Finite-Dimensional Vector Spaces with Application to the Mixed Linear Model*. Ph.D. diss., Iowa State University.

Seely, J. (1970a). "Linear Spaces and Unbiased Estimation." *Annals of Mathematical Statistics* 41:1725–1734.

Seely, J. (1970b). "Linear Spaces and Unbiased Estimation—Application to the Mixed Linear Model." *Annals of Mathematical Statistics* 41:1735–1748.

Seely, J., and Soong, S. (1971). *A Note on MINQUE's and Quadratic Estimability*. Corvallis: Oregon State University.

Seely, J., and Zyskind, G. (1971). "Linear Spaces and Minimum Variance Unbiased Estimation." *Annals of Mathematical Statistics* 42:691–703.

Stock, J. H., and Watson, M. W. (2002). *Introduction to Econometrics*. 3rd ed. Reading, MA: Addison-Wesley.

Theil, H. (1961). *Economic Forecasts and Policy*. 2nd ed. Amsterdam: North-Holland.

Wallace, T., and Hussain, A. (1969). "The Use of Error Components Model in Combining Cross Section with Time Series Data." *Econometrica* 37:55–72.

Wansbeek, T., and Kapteyn, A. (1989). "Estimation of the Error-Components Model with Incomplete Panels." *Journal of Econometrics* 41:341–361.

White, H. (1980). "A Heteroskedasticity-Consistent Covariance Matrix Estimator and a Direct Test for Heteroskedasticity." *Econometrica* 48:817–838.

Windmeijer, F. (2005). "A Finite Sample Correction for the Variance of Linear Efficient Two-Step GMM Estimators." *Journal of Econometrics* 126:25–51.

Wooldridge, J. M. (2002). *Econometric Analysis of Cross Section and Panel Data*. Cambridge, MA: MIT Press.

Wu, D. M. (1973). "Alternative Tests of Independence between Stochastic Regressors and Disturbances." *Econometrica* 41:733–750.

Zellner, A. (1962). "An Efficient Method of Estimating Seemingly Unrelated Regressions and Tests for Aggregation Bias." *Journal of the American Statistical Association* 57:348–368.

Chapter 28
The PDLREG Procedure

Contents

Overview: PDLREG Procedure **1925**
Getting Started: PDLREG Procedure **1926**
Introductory Example 1927
Syntax: PDLREG Procedure **1929**
Functional Summary 1929
PROC PDLREG Statement 1931
BY Statement 1931
MODEL Statement 1931
OUTPUT Statement 1934
RESTRICT Statement 1935
Details: PDLREG Procedure **1936**
Missing Values 1936
Polynomial Distributed Lag Estimation 1936
Autoregressive Error Model Estimation 1938
OUT= Data Set 1938
Printed Output 1938
ODS Graphics 1939
Examples: PDLREG Procedure **1940**
Example 28.1: Industrial Conference Board Data 1940
Example 28.2: Money Demand Model 1943
References **1946**

Overview: PDLREG Procedure

The PDLREG procedure estimates regression models for time series data in which the effects of some of the regressor variables are distributed across time. The distributed lag model assumes that the effect of an input variable X on an output Y is distributed over time. If you change the value of X at time t, Y will experience some immediate effect at time t, and it will also experience a delayed effect at times $t+1$, $t+2$, and so on up to time $t+p$ for some limit p.

The regression model supported by PROC PDLREG can include any number of regressors with distribution lags and any number of covariates. (Simple regressors without lag distributions are called covariates.) For example, the two-regressor model with a distributed lag effect for one regressor is written

$$y_t = \alpha + \sum_{i=0}^{p} \beta_i x_{t-i} + \gamma z_t + u_t$$

Here, x_t is the regressor with a distributed lag effect, z_t is a simple covariate, and u_t is an error term.

The distribution of the lagged effects is modeled by Almon lag polynomials. The coefficients b_i of the lagged values of the regressor are assumed to lie on a polynomial curve. That is,

$$b_i = \alpha_0^* + \sum_{j=1}^{d} \alpha_j^* i^j$$

where $d(\leq p)$ is the degree of the polynomial. For the numerically efficient estimation, the PDLREG procedure uses *orthogonal polynomials*. The preceding equation can be transformed into orthogonal polynomials:

$$b_i = \alpha_0 + \sum_{j=1}^{d} \alpha_j f_j(i)$$

where $f_j(i)$ is a polynomial of degree j in the lag length i, and α_j is a coefficient estimated from the data.

The PDLREG procedure supports endpoint restrictions for the polynomial. That is, you can constrain the estimated polynomial lag distribution curve so that $b_{-1} = 0$ or $b_{p+1} = 0$, or both. You can also impose linear restrictions on the parameter estimates for the covariates.

You can specify a minimum degree and a maximum degree for the lag distribution polynomial, and the procedure fits polynomials for all degrees in the specified range. (However, if distributed lags are specified for more that one regressor, you can specify a range of degrees for only one of them.)

The PDLREG procedure can also test for autocorrelated residuals and perform autocorrelated error correction by using the autoregressive error model. You can specify any order autoregressive error model and can specify several different estimation methods for the autoregressive model, including exact maximum likelihood.

The PDLREG procedure computes generalized Durbin-Watson statistics to test for autocorrelated residuals. For models with lagged dependent variables, the procedure can produce Durbin h and Durbin t statistics. You can request significance level p-values for the Durbin-Watson, Durbin h, and Durbin t statistics. See Chapter 9, "The AUTOREG Procedure," for details about these statistics.

The PDLREG procedure assumes that the input observations form a time series. Thus, the PDLREG procedure should be used only for ordered and equally spaced time series data.

Getting Started: PDLREG Procedure

Use the MODEL statement to specify the regression model. The PDLREG procedure's MODEL statement is written like MODEL statements in other SAS regression procedures, except that a regressor can be followed by a lag distribution specification enclosed in parentheses.

For example, the following MODEL statement regresses Y on X and Z and specifies a distributed lag for X:

```
model y = x(4,2) z;
```

The notation X(4,2) specifies that the model includes X and 4 lags of X, with the coefficients of X and its lags constrained to follow a second-degree (quadratic) polynomial. Thus, the regression model specified by this MODEL statement is

$$y_t = a + b_0 x_t + b_1 x_{t-1} + b_2 x_{t-2} + b_3 x_{t-3} + b_4 x_{t-4} + c z_t + u_t$$

$$b_i = \alpha_0 + \alpha_1 f_1(i) + \alpha_2 f_2(i)$$

where $f_1(i)$ is a polynomial of degree 1 in i and $f_2(i)$ is a polynomial of degree 2 in i.

Lag distribution specifications are enclosed in parentheses and follow the name of the regressor variable. The general form of the lag distribution specification is

regressor-name (length, degree, minimum-degree, end-constraint)

where

length	is the length of the lag distribution—that is, the number of lags of the regressor to use.
degree	is the degree of the distribution polynomial.
minimum-degree	is an optional minimum degree for the distribution polynomial.
end-constraint	is an optional endpoint restriction specification, which can have the value FIRST, LAST, or BOTH.

If the *minimum-degree* option is specified, the PDLREG procedure estimates models for all degrees between *minimum-degree* and *degree*.

Introductory Example

The following statements generate simulated data for variables Y and X. Y depends on the first three lags of X, with coefficients .25, .5, and .25. Thus, the effect of changes of X on Y takes effect 25% after one period, 75% after two periods, and 100% after three periods.

```
data test;
   xl1 = 0; xl2 = 0; xl3 = 0;
   do t = -3 to 100;
      x = ranuni(1234);
      y = 10 + .25 * xl1 + .5 * xl2 + .25 * xl3
             + .1 * rannor(1234);
      if t > 0 then output;
      xl3 = xl2; xl2 = xl1; xl1 = x;
   end;
run;
```

The following statements use the PDLREG procedure to regress Y on a distributed lag of X. The length of the lag distribution is 4, and the degree of the distribution polynomial is specified as 3.

```
proc pdlreg data=test;
   model y = x( 4, 3 );
run;
```

The PDLREG procedure first prints a table of statistics for the residuals of the model, as shown in Figure 28.1. See Chapter 9, "The AUTOREG Procedure," for an explanation of these statistics.

Figure 28.1 Residual Statistics

The PDLREG Procedure

Dependent Variable	y

Ordinary Least Squares Estimates			
SSE	0.86604442	**DFE**	91
MSE	0.00952	**Root MSE**	0.09755
SBC	-156.72612	**AIC**	-169.54786
MAE	0.07761107	**AICC**	-168.88119
MAPE	0.73971576	**HQC**	-164.3651
Durbin-Watson	1.9920	**Total R-Square**	0.7711

The PDLREG procedure next prints a table of parameter estimates, standard errors, and t tests, as shown in Figure 28.2.

Figure 28.2 Parameter Estimates

Parameter Estimates					
Variable	**DF**	**Estimate**	**Standard Error**	**t Value**	**Approx Pr > |t|**
Intercept	1	10.0030	0.0431	231.87	<.0001
x0**	1	0.4406	0.0378	11.66	<.0001
x1**	1	0.0113	0.0336	0.34	0.7377
x2**	1	-0.4108	0.0322	-12.75	<.0001
x3**	1	0.0331	0.0392	0.84	0.4007

The table in Figure 28.2 shows the model intercept and the estimated parameters of the lag distribution polynomial. The parameter labeled X**0 is the constant term, α_0, of the distribution polynomial. X**1 is the linear coefficient, α_1; X**2 is the quadratic coefficient, α_2; and X**3 is the cubic coefficient, α_3.

The parameter estimates for the distribution polynomial are not of interest in themselves. Since the PDLREG procedure does not print the orthogonal polynomial basis that it constructs to represent the distribution polynomial, these coefficient values cannot be interpreted.

However, because these estimates are for an orthogonal basis, you can use these results to test the degree of the polynomial. For example, this table shows that the X**3 estimate is not significant; the p-value for its t ratio is 0.4007, while the X**2 estimate is highly significant ($p < .0001$). This indicates that a second-degree polynomial might be more appropriate for this data set.

The PDLREG procedure next prints the lag distribution coefficients and a graphical display of these coefficients, as shown in Figure 28.3.

Figure 28.3 Coefficients and Graph of Estimated Lag Distribution

Estimate of Lag Distribution

Variable	Estimate	Standard Error	t Value	Approx Pr > |t|	-0.04	0.4167
x(0)	-0.040150	0.0360	-1.12	0.2677	|***|	
x(1)	0.324241	0.0307	10.55	<.0001	|**********|	
x(2)	0.416661	0.0239	17.45	<.0001	|**********|	
x(3)	0.289482	0.0315	9.20	<.0001	|**********	
x(4)	-0.004926	0.0365	-0.13	0.8929	|	

The lag distribution coefficients are the coefficients of the lagged values of X in the regression model. These coefficients lie on the polynomial curve defined by the parameters shown in Figure 28.2. Note that the estimated values for X(1), X(2), and X(3) are highly significant, while X(0) and X(4) are not significantly different from 0. These estimates are reasonably close to the true values used to generate the simulated data.

The graphical display of the lag distribution coefficients plots the estimated lag distribution polynomial reported in Figure 28.2. The roughly quadratic shape of this plot is another indication that a third-degree distribution curve is not needed for this data set.

Syntax: PDLREG Procedure

The following statements can be used with the PDLREG procedure:

PROC PDLREG *option* **;**
BY *variables* **;**
MODEL *dependents = effects / options* **;**
OUTPUT OUT= *SAS-data-set keyword = variables* **;**
RESTRICT *restrictions* **;**

Functional Summary

The statements and options used with the PDLREG procedure are summarized in the following table.

Table 28.1 PDLREG Functional Summary

Description	Statement	Option
Data Set Options		
specify the input data set	PDLREG	DATA=
write predicted values to an output data set	OUTPUT	OUT=
BY-Group Processing		
specify BY-group processing	BY	
Printing Control Options		
request all print options	MODEL	ALL
print transformed coefficients	MODEL	COEF

Table 28.1 *continued*

Description	Statement	Option
print correlations of the estimates	MODEL	CORRB
print covariances of the estimates	MODEL	COVB
print DW statistics up to order *j*	MODEL	DW=*j*
print the marginal probability of DW statistics	MODEL	DWPROB
print inverse of Toeplitz matrix	MODEL	GINV
print inverse of the crossproducts matrix	MODEL	I
print details at each iteration step	MODEL	ITPRINT
print Durbin *t* statistic	MODEL	LAGDEP
print Durbin *h* statistic	MODEL	LAGDEP=
suppress printed output	MODEL	NOPRINT
print partial autocorrelations	MODEL	PARTIAL
print standardized parameter estimates	MODEL	STB
print crossproducts matrix	MODEL	XPX
Model Estimation Options		
specify order of autoregressive process	MODEL	NLAG=
suppress intercept parameter	MODEL	NOINT
specify convergence criterion	MODEL	CONVERGE=
specify maximum number of iterations	MODEL	MAXITER=
specify estimation method	MODEL	METHOD=
Output Control Options		
specify confidence limit size	OUTPUT	ALPHACLI=
specify confidence limit size for structural predicted values	OUTPUT	ALPHACLM=
output transformed intercept variable	OUTPUT	CONSTANT=
output lower confidence limit for predicted values	OUTPUT	LCL=
output lower confidence limit for structural predicted values	OUTPUT	LCLM=
output predicted values	OUTPUT	P=
output predicted values of the structural part	OUTPUT	PM=
output residuals from the predicted values	OUTPUT	R=
output residuals from the structural predicted values	OUTPUT	RM=
output transformed variables	OUTPUT	TRANSFORM=
output upper confidence limit for the predicted values	OUTPUT	UCL=
output upper confidence limit for the structural predicted values	OUTPUT	UCLM=

PROC PDLREG Statement

PROC PDLREG *option* **;**

The PROC PDLREG statement has the following option:

DATA= *SAS-data-set*
specifies the name of the SAS data set containing the input data. If you do not specify the DATA= option, the most recently created SAS data set is used.

In addition, you can place any of the following MODEL statement options in the PROC PDLREG statement, which is equivalent to specifying the option for every MODEL statement: ALL, COEF, CONVERGE=, CORRB, COVB, DW=, DWPROB, GINV, ITPRINT, MAXITER=, METHOD=, NOINT, NOPRINT, and PARTIAL.

BY Statement

BY *variables* **;**

A BY statement can be used with PROC PDLREG to obtain separate analyses on observations in groups defined by the BY variables.

MODEL Statement

MODEL *dependent = effects / options* **;**

The MODEL statement specifies the regression model. The keyword MODEL is followed by the dependent variable name, an equal sign, and a list of independent effects. Only one MODEL statement is allowed.

Every variable in the model must be a numeric variable in the input data set. Specify an independent effect with a variable name optionally followed by a polynomial lag distribution specification.

Specifying Independent Effects

The general form of an effect is

variable (length, degree, minimum-degree, constraint)

The term in parentheses following the variable name specifies a polynomial distributed lag (PDL) for the variable. The PDL specification is as follows:

length	specifies the number of lags of the variable to include in the lag distribution.
degree	specifies the maximum degree of the distribution polynomial. If not specified, the degree defaults to the lag length.
minimum-degree	specifies the minimum degree of the polynomial. By default *minimum-degree* is the same as *degree*.
constraint	specifies endpoint restrictions on the polynomial. The value of *constraint* can be FIRST, LAST, or BOTH. If a value is not specified, there are no endpoint restrictions.

If you do not specify the *degree* or *minimum-degree* parameter, but you do specify endpoint restrictions, you must use commas to show which parameter, *degree* or *minimum-degree*, is left out.

MODEL Statement Options

The following options can appear in the MODEL statement after a slash (/).

ALL
prints all the matrices computed during the analysis of the model.

COEF
prints the transformation coefficients for the first p observations. These coefficients are formed from a scalar multiplied by the inverse of the Cholesky root of the Toeplitz matrix of autocovariances.

CORRB
prints the matrix of estimated correlations between the parameter estimates.

COVB
prints the matrix of estimated covariances between the parameter estimates.

DW= *j*
prints the generalized Durbin-Watson statistics up to the order of j. The default is DW=1. When you specify the LAGDEP or LAGDEP=*name* option, the Durbin-Watson statistic is not printed unless you specify the DW= option.

DWPROB
prints the marginal probability of the Durbin-Watson statistic.

CONVERGE= *value*
sets the convergence criterion. If the maximum absolute value of the change in the autoregressive parameter estimates between iterations is less than this amount, then convergence is assumed. The default is CONVERGE=.001.

GINV
prints the inverse of the Toeplitz matrix of autocovariances for the Yule-Walker solution.

I
prints $(\mathbf{X}'\mathbf{X})^{-1}$, the inverse of the crossproducts matrix for the model; or, if restrictions are specified, it prints $(\mathbf{X}'\mathbf{X})^{-1}$ adjusted for the restrictions.

ITPRINT
prints information on each iteration.

LAGDEP

LAGDV
prints the t statistic for testing residual autocorrelation when regressors contain lagged dependent variables.

LAGDEP= *name*

LAGDV= *name*

prints the Durbin *h* statistic for testing the presence of first-order autocorrelation when regressors contain the lagged dependent variable whose name is specified as LAGDEP=*name*. When the *h* statistic cannot be computed, the asymptotically equivalent *t* statistic is given.

MAXITER= *number*

sets the maximum number of iterations allowed. The default is MAXITER=50.

METHOD= *value*

specifies the type of estimates for the autoregressive component. The values of the METHOD= option are as follows:

METHOD=ML	specifies the maximum likelihood method.
METHOD=ULS	specifies unconditional least squares.
METHOD=YW	specifies the Yule-Walker method.
METHOD=ITYW	specifies iterative Yule-Walker estimates.

The default is METHOD=ML if you specified the LAGDEP or LAGDEP= option; otherwise, METHOD=YW is the default.

NLAG= *m*

NLAG= *(number-list)*

specifies the order of the autoregressive process or the subset of autoregressive lags to be fit. If you do not specify the NLAG= option, PROC PDLREG does not fit an autoregressive model.

NOINT

suppresses the intercept parameter from the model.

NOPRINT

suppresses the printed output.

PARTIAL

prints partial autocorrelations if the NLAG= option is specified.

STB

prints standardized parameter estimates. Sometimes known as a standard partial regression coefficient, a *standardized parameter estimate* is a parameter estimate multiplied by the standard deviation of the associated regressor and divided by the standard deviation of the regressed variable.

XPX

prints the crossproducts matrix, $\mathbf{X}'\mathbf{X}$, used for the model. $\mathbf{X}$ refers to the transformed matrix of regressors for the regression.

OUTPUT Statement

OUTPUT *OUT= SAS-data-set keyword= option . . .* **;**

The OUTPUT statement creates an output SAS data set with variables as specified by the following keyword options. See the section "Predicted Values" in Chapter 9, "The AUTOREG Procedure," for a description of the associated computations for these options.

ALPHACLI= *number*
sets the confidence limit size for the estimates of future values of the current realization of the response time series to *number*, where *number* is less than one and greater than zero. The resulting confidence interval has 1-*number* confidence. The default value for *number* is 0.05, corresponding to a 95% confidence interval.

ALPHACLM= *number*
sets the confidence limit size for the estimates of the structural or regression part of the model to *number*, where *number* is less than one and greater than zero. The resulting confidence interval has 1-*number* confidence. The default value for *number* is 0.05, corresponding to a 95% confidence interval.

OUT= *SAS-data-set*
names the output data.

The following specifications are of the form *KEYWORD=names*, where *KEYWORD=* specifies the statistic to include in the output data set and *names* gives names to the variables that contain the statistics.

CONSTANT= *variable*
writes the transformed intercept to the output data set.

LCL= *name*
requests that the lower confidence limit for the predicted value (specified in the PREDICTED= option) be added to the output data set under the name given.

LCLM= *name*
requests that the lower confidence limit for the structural predicted value (specified in the PREDICTEDM= option) be added to the output data set under the name given.

PREDICTED= *name*

P= *name*
stores the predicted values in the output data set under the name given.

PREDICTEDM= *name*

PM= *name*
stores the structural predicted values in the output data set under the name given. These values are formed from only the structural part of the model.

RESIDUAL= *name*

R= *name*
stores the residuals from the predicted values based on both the structural and time series parts of the model in the output data set under the name given.

RESIDUALM= *name*

RM= *name*
requests that the residuals from the structural prediction be given.

TRANSFORM= *variables*
requests that the specified variables from the input data set be transformed by the autoregressive model and put in the output data set. If you need to reproduce the data suitable for reestimation, you must also transform an intercept variable. To do this, transform a variable that only takes the value 1 or use the CONSTANT= option.

UCL= *name*
stores the upper confidence limit for the predicted value (specified in the PREDICTED= option) in the output data set under the name given.

UCLM= *name*
stores the upper confidence limit for the structural predicted value (specified in the PREDICTEDM= option) in the output data set under the name given.

For example, the SAS statements

```
proc pdlreg data=a;
   model y=x1 x2;
   output out=b p=yhat r=resid;
run;
```

create an output data set named B. In addition to the input data set variables, the data set B contains the variable YHAT, whose values are predicted values of the dependent variable Y, and RESID, whose values are the residual values of Y.

RESTRICT Statement

RESTRICT *equation , . . . , equation* **;**

The RESTRICT statement places restrictions on the parameter estimates for covariates in the preceding MODEL statement. A parameter produced by a distributed lag cannot be restricted with the RESTRICT statement.

Each restriction is written as a linear equation. If you specify more than one restriction in a RESTRICT statement, the restrictions are separated by commas.

You can refer to parameters by the name of the corresponding regressor variable. Each name used in the equation must be a regressor in the preceding MODEL statement. Use the keyword INTERCEPT to refer to the intercept parameter in the model.

RESTRICT statements can be given labels. You can use labels to distinguish results for different restrictions in the printed output. Labels are specified as follows:

label : **RESTRICT** *...*

The following is an example of the use of the RESTRICT statement, in which the coefficients of the regressors X1 and X2 are required to sum to 1:

```
proc pdlreg data=a;
   model y = x1 x2;
   restrict x1 + x2 = 1;
run;
```

Parameter names can be multiplied by constants. When no equal sign appears, the linear combination is set equal to 0. Note that the parameters associated with the variables are restricted, not the variables themselves. Here are some examples of valid RESTRICT statements:

```
restrict x1 + x2 = 1;
restrict x1 + x2 - 1;
restrict 2 * x1 = x2 + x3 , intercept + x4 = 0;
restrict x1 = x2 = x3 = 1;
restrict 2 * x1 - x2;
```

Restricted parameter estimates are computed by introducing a Lagrangian parameter λ for each restriction (Pringle and Rayner 1971). The estimates of these Lagrangian parameters are printed in the parameter estimates table. If a restriction cannot be applied, its parameter value and degrees of freedom are listed as 0.

The Lagrangian parameter, λ, measures the sensitivity of the SSE to the restriction. If the restriction is changed by a small amount ϵ, the SSE is changed by $2\lambda\epsilon$.

The *t* ratio tests the significance of the restrictions. If λ is zero, the restricted estimates are the same as the unrestricted ones.

You can specify any number of restrictions in a RESTRICT statement, and you can use any number of RESTRICT statements. The estimates are computed subject to all restrictions specified. However, restrictions should be consistent and not redundant.

Details: PDLREG Procedure

Missing Values

The PDLREG procedure skips any observations at the beginning of the data set that have missing values. The procedure uses all observations with nonmissing values for all the independent and dependent variables such that the lag distribution has sufficient nonmissing lagged independent variables.

Polynomial Distributed Lag Estimation

The simple finite distributed lag model is expressed in the form

$$y_t = \alpha + \sum_{i=0}^{p} \beta_i x_{t-i} + \epsilon_t$$

When the lag length (p) is long, severe multicollinearity can occur. Use the Almon or *polynomial distributed lag* model to avoid this problem, since the relatively low-degree d ($\leq p$) polynomials can capture the true lag distribution. The lag coefficient can be written in the Almon polynomial lag

$$\beta_i = \alpha_0^* + \sum_{j=1}^{d} \alpha_j^* i^j$$

Emerson (1968) proposed an efficient method of constructing orthogonal polynomials from the preceding polynomial equation as

$$\beta_i = \alpha_0 + \sum_{j=1}^{d} \alpha_j f_j(i)$$

where $f_j(i)$ is a polynomial of degree j in the lag length i. The polynomials $f_j(i)$ are chosen so that they are orthogonal:

$$\sum_{i=1}^{n} w_i f_j(i) f_k(i) = \begin{cases} 1 & \text{if } j = k \\ 0 & \text{if } j \neq k \end{cases}$$

where w_i is the weighting factor, and $n = p + 1$. PROC PDLREG uses the equal weights ($w_i = 1$) for all i. To construct the orthogonal polynomials, the following recursive relation is used:

$$f_j(i) = (A_j i + B_j) f_{j-1}(i) - C_j f_{j-2}(i) j = 1, \ldots, d$$

The constants A_j, B_j, and C_j are determined as follows:

$$\begin{aligned} A_j &= \left\{ \sum_{i=1}^{n} w_i i^2 f_{j-1}^2(i) - \left(\sum_{i=1}^{n} w_i i f_{j-1}^2(i) \right)^2 \right. \\ &\quad \left. - \left(\sum_{i=1}^{n} w_i i f_{j-1}(i) f_{j-2}(i) \right)^2 \right\}^{-1/2} \\ B_j &= -A_j \sum_{i=1}^{n} w_i i f_{j-1}^2(i) \\ C_j &= A_j \sum_{i=1}^{n} w_i i f_{j-1}(i) f_{j-2}(i) \end{aligned}$$

where $f_{-1}(i) = 0$ and $f_0(i) = 1/\sqrt{\sum_{i=1}^{n} w_i}$.

PROC PDLREG estimates the orthogonal polynomial coefficients, $\alpha_0, \ldots, \alpha_d$, to compute the coefficient estimate of each independent variable (X) with distributed lags. For example, if an independent variable is specified as X(9,3), a third-degree polynomial is used to specify the distributed lag coefficients. The third-degree polynomial is fit as a constant term, a linear term, a quadratic term, and a cubic term. The four terms are constructed to be orthogonal. In the output produced by the PDLREG procedure for this case, parameter estimates with names X**0, X**1, X**2, and X**3 correspond to $\hat{\alpha}_0, \hat{\alpha}_1, \hat{\alpha}_2$, and $\hat{\alpha}_3$, respectively. A test using the t statistic and the approximate p-value ("Approx Pr > $|t|$") associated with X**3 can determine whether a second-degree polynomial rather than a third-degree polynomial is appropriate. The estimates of the 10 lag coefficients associated with the specification X(9,3) are labeled X(0), X(1), X(2), X(3), X(4), X(5), X(6), X(7), X(8), and X(9).

Autoregressive Error Model Estimation

The PDLREG procedure uses the same autoregressive error model estimation methods as the AUTOREG procedure. These two procedures share the same computational resources for computing estimates. See Chapter 9, "The AUTOREG Procedure," for details about estimation methods for autoregressive error models.

OUT= Data Set

The OUT= data set produced by the PDLREG procedure's OUTPUT statement is similar in form to the OUT= data set produced by the AUTOREG procedure. See Chapter 9, "The AUTOREG Procedure," for details on the OUT= data set.

Printed Output

The PDLREG procedure prints the following items:

1. the name of the dependent variable
2. the ordinary least squares (OLS) estimates
3. the estimates of autocorrelations and of the autocovariance, and if line size permits, a graph of the autocorrelation at each lag. The autocorrelation for lag 0 is 1. These items are printed if you specify the NLAG= option.
4. the partial autocorrelations if the PARTIAL and NLAG= options are specified. The first partial autocorrelation is the autocorrelation for lag 1.
5. the preliminary mean square error, which results from solving the Yule-Walker equations if you specify the NLAG= option
6. the estimates of the autoregressive parameters, their standard errors, and the ratios of estimates to standard errors (t) if you specify the NLAG= option
7. the statistics of fit for the final model if you specify the NLAG= option. These include the error sum of squares (SSE), the degrees of freedom for error (DFE), the mean square error (MSE), the root mean square error (Root MSE), the mean absolute error (MAE), the mean absolute percentage error (MAPE), the Schwarz information criterion (SBC), the Akaike's information criterion (AIC), Akaike's information criterion corrected(AICC), the regression R^2 (Regress R-Square), the total R^2 (Total R-Square), and the Durbin-Watson statistic (Durbin-Watson). See Chapter 9, "The AUTOREG Procedure," for details of the regression R^2 and the total R^2.
8. the parameter estimates for the structural model (B), a standard error estimate, the ratio of estimate to standard error (t), and an approximation to the significance probability for the parameter being 0 ("Approx Pr $> |t|$")
9. a plot of the lag distribution (estimate of lag distribution)
10. the covariance matrix of the parameter estimates if the COVB option is specified

ODS Graphics

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*).

Before you create graphs, ODS Graphics must be enabled (for example, with the ODS GRAPHICS ON statement). For more information about enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" in that chapter.

The overall appearance of graphs is controlled by ODS styles. Styles and other aspects of using ODS Graphics are discussed in the section "A Primer on ODS Statistical Graphics" in that chapter.

PROC PDLREG assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in the following table.

Table 28.2 ODS Tables Produced in PROC PDLREG

ODS Table Name	Description	Option
ODS Tables Created by the MODEL Statement		
ARParameterEstimates	Estimates of autoregressive parameters	NLAG=
CholeskyFactor	Cholesky root of gamma	NLAG= and ALL
Coefficients	Coefficients for first NLAG observations	NLAG= and (COEF or ALL)
ConvergenceStatus	Convergence status table	default
CorrB	Correlation of parameter estimates	CORRB
CorrGraph	Estimates of autocorrelations	NLAG=
CovB	Covariance of parameter estimates	COVB
DependenceEquations	Linear dependence equation	
Dependent	Dependent variable	default
DWTest	Durbin-Watson statistics	DW=
DWTestProb	Durbin-Watson statistics and p-values	DW= DWPROB
ExpAutocorr	Expected autocorrelations	{NLAG= and (COEF or ALL)} or {NLAG=($l_1 \ldots l_m$) where $l_m > m$}
FitSummary	Summary of regression	default
GammaInverse	Gamma inverse	NLAG= and (GINV or ALL)
IterHistory	Iteration history	ITPRINT
LagDist	Lag distribution	default
ParameterEstimates	Parameter estimates	default
ParameterEstimatesGivenAR	Parameter estimates assuming AR parameters are given	NLAG=
PartialAutoCorr	Partial autocorrelation	PARTIAL

Table 28.2 *continued*

ODS Table Name	Description	Option
PreMSE	Preliminary MSE	NLAG=
XPXIMatrix	$(\mathbf{X}'\mathbf{X})^{-1}$ matrix	XPX
XPXMatrix	$\mathbf{X}'\mathbf{X}$ matrix	XPX
YWIterSSE	Yule-Walker iteration sum of squared error	METHOD=ITYW
ODS Tables Created by the RESTRICT Statement		
Restrict	Restriction table	default

Examples: PDLREG Procedure

Example 28.1: Industrial Conference Board Data

In this example, a second-degree Almon polynomial lag model is fit to a model with a five-period lag, and dummy variables are used for quarter effects. The PDL model is estimated using capital appropriations data series for the period 1952 to 1967. The estimation model is written

$$CE_t = a_0 + b_1 Q1_t + b_2 Q2_t + b_3 Q3_t$$
$$+c_0 CA_t + c_1 CA_{t-1} + \ldots + c_5 CA_{t-5}$$

where CE represents capital expenditures and CA represents capital appropriations.

```
title 'National Industrial Conference Board Data';
title2 'Quarterly Series - 1952Q1 to 1967Q4';

data a;
   input ce ca @@;
   qtr = mod( _n_-1, 4 ) + 1;
   q1  = qtr=1;
   q2  = qtr=2;
   q3  = qtr=3;
datalines;
   2072 1660 2077 1926 2078 2181 2043 1897 2062 1695

   ... more lines ...


proc pdlreg data=a;
   model ce = q1 q2 q3 ca(5,2) / dwprob;
run;
```

The printed output produced by the PDLREG procedure is shown in Output 28.1.1. The small Durbin-Watson test indicates autoregressive errors.

Output 28.1.1 Printed Output Produced by PROC PDLREG

National Industrial Conference Board Data
Quarterly Series - 1952Q1 to 1967Q4

The PDLREG Procedure

Dependent Variable ce

Ordinary Least Squares Estimates

SSE	1205186.4	**DFE**	48
MSE	25108	**Root MSE**	158.45520
SBC	733.84921	**AIC**	719.797878
MAE	107.777378	**AICC**	722.180856
MAPE	3.71653891	**HQC**	725.231641
Durbin-Watson	0.6157	**Total R-Square**	0.9834

Parameter Estimates

Variable	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	210.0109	73.2524	2.87	0.0061
q1	1	-10.5515	61.0634	-0.17	0.8635
q2	1	-20.9887	59.9386	-0.35	0.7277
q3	1	-30.4337	59.9004	-0.51	0.6137
ca0**	1	0.3760	0.007318	51.38	<.0001
ca1**	1	0.1297	0.0251	5.16	<.0001
ca2**	1	0.0247	0.0593	0.42	0.6794

Estimate of Lag Distribution

Variable	Estimate	Standard Error	t Value	Approx Pr > |t|	0	0.2444
ca(0)	0.089467	0.0360	2.49	0.0165	|**********	|
ca(1)	0.104317	0.0109	9.56	<.0001	|**********	|
ca(2)	0.127237	0.0255	5.00	<.0001	|**********	|
ca(3)	0.158230	0.0254	6.24	<.0001	|**********	|
ca(4)	0.197294	0.0112	17.69	<.0001	|**********	|
ca(5)	0.244429	0.0370	6.60	<.0001	|**********	|

The following statements use the REG procedure to fit the same polynomial distributed lag model. A DATA step computes lagged values of the regressor X, and RESTRICT statements are used to impose the polynomial lag distribution. Refer to Judge et al. (1985, pp. 357–359) for the restricted least squares estimation of the Almon distributed lag model.

```
data b;
   set a;
   ca_1 = lag( ca );
   ca_2 = lag2( ca );
   ca_3 = lag3( ca );
   ca_4 = lag4( ca );
   ca_5 = lag5( ca );
run;
```

```
proc reg data=b;
   model  ce = q1 q2 q3 ca ca_1 ca_2 ca_3 ca_4 ca_5;
   restrict   - ca + 5*ca_1 - 10*ca_2 + 10*ca_3 - 5*ca_4 +   ca_5;
   restrict     ca - 3*ca_1 +  2*ca_2 +  2*ca_3 - 3*ca_4 +   ca_5;
   restrict  -5*ca + 7*ca_1 +  4*ca_2 -  4*ca_3 - 7*ca_4 + 5*ca_5;
run;
```

The REG procedure output is shown in Output 28.1.2.

Output 28.1.2 Printed Output Produced by PROC REG

National Industrial Conference Board Data
Quarterly Series - 1952Q1 to 1967Q4

The REG Procedure
Model: MODEL1
Dependent Variable: ce

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	6	71343377	11890563	473.58	<.0001
Error	48	1205186	25108		
Corrected Total	54	72548564			

Root MSE	158.45520	R-Square	0.9834
Dependent Mean	3185.69091	Adj R-Sq	0.9813
Coeff Var	4.97397		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	210.01094	73.25236	2.87	0.0061
q1	1	-10.55151	61.06341	-0.17	0.8635
q2	1	-20.98869	59.93860	-0.35	0.7277
q3	1	-30.43374	59.90045	-0.51	0.6137
ca	1	0.08947	0.03599	2.49	0.0165
ca_1	1	0.10432	0.01091	9.56	<.0001
ca_2	1	0.12724	0.02547	5.00	<.0001
ca_3	1	0.15823	0.02537	6.24	<.0001
ca_4	1	0.19729	0.01115	17.69	<.0001
ca_5	1	0.24443	0.03704	6.60	<.0001
RESTRICT	-1	623.63242	12697	0.05	0.9614*
RESTRICT	-1	18933	44803	0.42	0.6772*
RESTRICT	-1	10303	18422	0.56	0.5814*

*** Probability computed using beta distribution.**

Example 28.2: Money Demand Model

This example estimates the demand for money by using the following dynamic specification:

$$m_t = a_0 + b_0 m_{t-1} + \sum_{i=0}^{5} c_i y_{t-i} + \sum_{i=0}^{2} d_i r_{t-i} + \sum_{i=0}^{3} f_i p_{t-i} + u_t$$

where

m_t = log of real money stock (M1)

y_t = log of real GNP

r_t = interest rate (commercial paper rate)

p_t = inflation rate

c_i, d_i, and f_i ($i > 0$) are coefficients for the lagged variables

The following DATA step reads the data and transforms the real money and real GNP variables using the natural logarithm. Refer to Balke and Gordon (1986) for a description of the data.

```
data a;
   input m1 gnp gdf r @@;
   m    = log( 100 * m1 / gdf );
   lagm = lag( m );
   y    = log( gnp );
   p    = log( gdf / lag( gdf ) );
   date = intnx( 'qtr', '1jan1968'd, _n_-1 );
   format date yyqc6.;
   label m    = 'Real Money Stock (M1)'
         lagm = 'Lagged Real Money Stock'
         y    = 'Real GNP'
         r    = 'Commercial Paper Rate'
         p    = 'Inflation Rate';
datalines;
187.15  1036.22    81.18     5.58

   ... more lines ...
```

Output 28.2.1 shows a partial list of the data set.

Output 28.2.1 Partial List of the Data Set A

National Industrial Conference Board Data
Quarterly Series - 1952Q1 to 1967Q4

Obs	date	m	lagm	y	r	p
1	1968:1	5.44041	.	6.94333	5.58	.
2	1968:2	5.44732	5.44041	6.96226	6.08	0.011513
3	1968:3	5.45815	5.44732	6.97422	5.96	0.008246
4	1968:4	5.46492	5.45815	6.97661	5.96	0.014865
5	1969:1	5.46980	5.46492	6.98855	6.66	0.011005

The regression model is written for the PDLREG procedure with a MODEL statement. The LAGDEP= option is specified to test for the serial correlation in disturbances since regressors contain the lagged dependent variable LAGM.

```
title 'Money Demand Estimation using Distributed Lag Model';
title2 'Quarterly Data - 1968Q2 to 1983Q4';

proc pdlreg data=a;
   model m = lagm y(5,3) r(2, , ,first) p(3,2) / lagdep=lagm;
run;
```

The estimated model is shown in Output 28.2.2 and Output 28.2.3.

Output 28.2.2 Parameter Estimates

Money Demand Estimation using Distributed Lag Model
Quarterly Data - 1968Q2 to 1983Q4

The PDLREG Procedure

Dependent Variable	m
	Real Money Stock (M1)

Ordinary Least Squares Estimates			
SSE	0.00169815	DFE	48
MSE	0.0000354	Root MSE	0.00595
SBC	-404.60169	AIC	-427.4546
MAE	0.00383648	AICC	-421.83758
MAPE	0.07051345	HQC	-418.53375
		Total R-Square	0.9712

Output 28.2.2 *continued*

Parameter Estimates

Variable	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	-0.1407	0.2625	-0.54	0.5943
lagm	1	0.9875	0.0425	23.21	<.0001
y0**	1	0.0132	0.004531	2.91	0.0055
y1**	1	-0.0704	0.0528	-1.33	0.1891
y2**	1	0.1261	0.0786	1.60	0.1154
y3**	1	-0.4089	0.1265	-3.23	0.0022
r0**	1	-0.000186	0.000336	-0.55	0.5816
r1**	1	0.002200	0.000774	2.84	0.0065
r2**	1	0.000788	0.000249	3.16	0.0027
p0**	1	-0.6602	0.1132	-5.83	<.0001
p1**	1	0.4036	0.2321	1.74	0.0885
p2**	1	-1.0064	0.2288	-4.40	<.0001

Restriction	DF	L Value	Standard Error	t Value	Approx Pr > |t|
r(-1)	-1	0.0164	0.007275	2.26	0.0223

Output 28.2.3 Estimates for Lagged Variables

Estimate of Lag Distribution

Variable	Estimate	Standard Error	t Value	Approx Pr > |t|	-0.196 0 0.2686
y(0)	0.268619	0.0910	2.95	0.0049	| |**********|
y(1)	-0.196484	0.0612	-3.21	0.0024	|**********| |
y(2)	-0.163148	0.0537	-3.04	0.0038	| **********| |
y(3)	0.063850	0.0451	1.42	0.1632	| |****** |
y(4)	0.179733	0.0588	3.06	0.0036	| |********** |
y(5)	-0.120276	0.0679	-1.77	0.0827	| *********| |

Estimate of Lag Distribution

Variable	Estimate	Standard Error	t Value	Approx Pr > |t|	-0.001 0 0.0018
r(0)	-0.001341	0.000388	-3.45	0.0012	|**********| |
r(1)	-0.000751	0.000234	-3.22	0.0023	| *********| |
r(2)	0.001770	0.000754	2.35	0.0230	| |**********|

Estimate of Lag Distribution

Variable	Estimate	Standard Error	t Value	Approx Pr > |t|	-1.104 0 0.2634
p(0)	-1.104051	0.2027	-5.45	<.0001	|**********| |
p(1)	0.082892	0.1257	0.66	0.5128	| |*** |
p(2)	0.263391	0.1381	1.91	0.0624	| |********|
p(3)	-0.562556	0.2076	-2.71	0.0093	| **********| |

References

Balke, N. S., and Gordon, R. J. (1986). "Historical Data." In *The American Business Cycle*, edited by R. J. Gordon, 781–850. Chicago: University of Chicago Press.

Emerson, P. L. (1968). "Numerical Construction of Orthogonal Polynomials from a General Recurrence Formula." *Biometrics* 24:695–701.

Gallant, A. R., and Goebel, J. J. (1976). "Nonlinear Regression with Autoregressive Errors." *Journal of the American Statistical Association* 71:961–967.

Harvey, A. C. (1981). *The Econometric Analysis of Time Series*. New York: John Wiley & Sons.

Johnston, J. (1972). *Econometric Methods*. 2nd ed. New York: McGraw-Hill.

Judge, G. G., Griffiths, W. E., Hill, R. C., Lütkepohl, H., and Lee, T.-C. (1985). *The Theory and Practice of Econometrics*. 2nd ed. New York: John Wiley & Sons.

Park, R. E., and Mitchell, B. M. (1980). "Estimating the Autocorrelated Error Model with Trended Data." *Journal of Econometrics* 13:185–201.

Pringle, R. M., and Rayner, A. A. (1971). *Generalized Inverse Matrices with Applications to Statistics*. New York: Hafner Publishing.

Chapter 29
The QLIM Procedure

Contents

Overview: QLIM Procedure **1948**
Getting Started: QLIM Procedure **1949**
Introductory Example: Binary Probit and Logit Models 1951
Syntax: QLIM Procedure **1955**
Functional Summary 1956
PROC QLIM Statement 1959
BAYES Statement 1964
BOUNDS Statement 1970
BY Statement 1971
CLASS Statement 1971
ENDOGENOUS Statement 1971
FREQ Statement 1975
HETERO Statement 1975
INIT Statement 1976
MODEL Statement 1976
NLOPTIONS Statement 1978
OUTPUT Statement 1978
PRIOR Statement 1979
RANDOM Statement (Experimental) 1980
RESTRICT Statement 1982
TEST Statement 1983
WEIGHT Statement 1984
Details: QLIM Procedure **1985**
Ordinal Discrete Choice Modeling 1985
Limited Dependent Variable Models 1988
Stochastic Frontier Production and Cost Models 1991
Heteroscedasticity and Box-Cox Transformation 1993
Bivariate Limited Dependent Variable Modeling 1995
Selection Models 1996
Multivariate Limited Dependent Models 1999
Variable Selection 2000
Tests on Parameters 2000
Endogeneity and Instrumental Variables 2001
Panel Data Analysis 2008
Bayesian Analysis 2011
Prior Distributions 2018

Automated MCMC 2021
Marginal Likelihood 2022
Standard Distributions 2024
Output to SAS Data Set 2027
OUTEST= Data Set 2031
Naming 2032
ODS Table Names 2033
ODS Graphics 2035
Examples: QLIM Procedure **2036**
Example 29.1: Ordered Data Modeling 2036
Example 29.2: Tobit Analysis 2039
Example 29.3: Bivariate Probit Analysis 2041
Example 29.4: Sample Selection Model 2042
Example 29.5: Sample Selection Model with Truncation and Censoring 2043
Example 29.6: Types of Tobit Models 2045
Example 29.7: Stochastic Frontier Models 2051
Example 29.8: Bayesian Modeling 2056
References **2062**

Overview: QLIM Procedure

The QLIM (qualitative and limited dependent variable model) procedure analyzes univariate and multivariate limited dependent variable models in which dependent variables take discrete values or in which dependent variables are observed only in a limited range of values. These models include logit, probit, tobit, selection, and multivariate models. The multivariate model can contain discrete choice and limited endogenous variables in addition to continuous endogenous variables.

The QLIM procedure supports the following models:

- linear regression model with heteroscedasticity
- Box-Cox regression with heteroscedasticity
- probit with heteroscedasticity
- logit with heteroscedasticity
- tobit (censored and truncated) with heteroscedasticity
- bivariate probit
- bivariate tobit
- sample selection and switching regression models

- multivariate limited dependent variables
- stochastic frontier production and cost models

In the linear regression models with heteroscedasticity, the assumption that error variance is constant across observations is relaxed. The QLIM procedure allows for a number of different linear and nonlinear variance specifications. Another way to make the linear model more appropriate to fit the data and reduce skewness is to apply Box-Cox transformation. If the nature of the data is such that the dependent variable is discrete and it takes only two possible values, ordinary least squares (OLS) estimates are inconsistent. The QLIM procedure offers probit and logit models to overcome these estimation problems. Assumptions about the error variance can also be relaxed in order to estimate probit or logit with heteroscedasticity.

The QLIM procedure also offers a class of models in which the dependent variable is censored or truncated from below or above or both. When a continuous dependent variable is observed only within a certain range and values outside this range are not available, the QLIM procedure offers a class of models that adjust for truncation. In some cases, the dependent variable is continuous only in a certain range and all values outside this range are reported as being on its boundary. For example, if it is not possible to observe negative values, the value of the dependent variable is reported as equal to 0. Because the data are censored, OLS results are inconsistent, and it cannot be guaranteed that the predicted values from the model fall in the appropriate region.

Most of the models in the QLIM procedure can be extended to accommodate bivariate and multivariate scenarios. The assumption that one variable is observed only if another variable takes on certain values lead to the introduction of sample selection models. If the dependent variables are mutually exclusive and observed only for certain ranges of the selection variable, the sample selection can be extended to include cases of switching regression. Stochastic frontier production and cost models allow for random shocks of the production or cost. They include a systematic positive component in the error term that adjusts for technological or cost inefficiency.

The QLIM procedure can use the maximum likelihood method and the Bayesian method for both univariate and multivariate models. Initial starting values for the nonlinear optimizations are typically calculated by OLS.

Getting Started: QLIM Procedure

The QLIM procedure is similar in use to the other regression or simultaneous equations model procedures in the SAS System. For example, the following statements are used to estimate a binary choice model by using the probit probability function:

```
proc qlim data=a;
   model y = x1;
   endogenous y ~ discrete;
run;
```

The response variable, y, is numeric and has discrete values. PROC QLIM enables the user to specify the type of endogenous variables in the ENDOGENOUS statement. The binary probit model can be also specified as follows:

```
model y = x1 / discrete;
```

When multiple endogenous variables are specified in the QLIM procedure, these equations are estimated as a system. Multiple endogenous variables can be specified with one MODEL statement in the QLIM procedure when these models have the same exogenous variables:

```
model y1 y2 = x1 x2 / discrete;
```

The preceding specification is equivalent to the following statements:

```
proc qlim data=a;
   model y1 = x1 x2;
   model y2 = x1 x2;
   endogenous y1 y2 ~ discrete;
run;
```

Some equations in multivariate models can be continuous while other equations can be discrete. A bivariate model with a discrete and a continuous equation is specified as follows:

```
proc qlim data=a;
   model y1 = x1 x2;
   model y2 = x3 x4;
   endogenous y1 ~ discrete;
run;
```

The standard tobit model is estimated by specifying the endogenous variable to be truncated or censored. The limits of the dependent variable can be specified with the CENSORED or TRUNCATED option in the ENDOGENOUS or MODEL statement when the data are limited by specific values or variables. For example, the two-limit censored model requires two variables that contain the lower (bottom) and upper (top) bound:

```
proc qlim data=a;
   model y = x1 x2 x3;
   endogenous y ~ censored(lb=bottom ub=top);
run;
```

The bounds can be numbers if they are fixed for all observations in the data set. For example, the standard tobit model can be specified as follows:

```
proc qlim data=a;
   model y = x1 x2 x3;
   endogenous y ~ censored(lb=0);
run;
```

Introductory Example: Binary Probit and Logit Models

The following example illustrates the use of PROC QLIM. The data were originally published by Mroz (1987) and downloaded from Wooldridge (2002). This data set is based on a sample of 753 married white women. The dependent variable is a discrete variable of labor force participation (inlf). Explanatory variables are the number of children ages 5 or younger (kidslt6), the number of children ages 6 to 18 (kidsge6), the woman's age (age), the woman's years of schooling (educ), wife's labor experience (exper), square of experience (expersq), and the family income excluding the wife's wage (nwifeinc). The program (with data values omitted) is as follows:

```
/*-- Binary Probit --*/
proc qlim data=mroz plots=predicted;
   model inlf = nwifeinc educ exper expersq
                age kidslt6 kidsge6  / discrete;
run;
```

Results of this analysis are shown in the following four figures. In the first table, shown in Figure 29.1, PROC QLIM provides frequency information about each choice. In this example, 428 women participate in the labor force (inlf =1).

Figure 29.1 Choice Frequency Summary

Binary Data

The QLIM Procedure

Discrete Response Profile of inlf		
Index	Value	Total Frequency
1	0	325
2	1	428

The second table is the estimation summary table shown in Figure 29.2. Included are the number of dependent variables, names of dependent variables, the number of observations, the log-likelihood function value, the maximum absolute gradient, the number of iterations, AIC, and Schwarz criterion.

Figure 29.2 Fit Summary Table of Binary Probit

Model Fit Summary	
Number of Endogenous Variables	1
Endogenous Variable	inlf
Number of Observations	753
Log Likelihood	-401.30219
Maximum Absolute Gradient	0.0000669
Number of Iterations	15
Optimization Method	Quasi-Newton
AIC	818.60439
Schwarz Criterion	855.59691

Goodness-of-fit measures are displayed in Figure 29.3. All measures except McKelvey-Zavoina's definition are based on the log-likelihood function value. The likelihood ratio test statistic has chi-square distribution conditional on the null hypothesis that all slope coefficients are zero. In this example, the likelihood ratio statistic is used to test the hypothesis that kidslt6 =kidge6 =age =educ =exper =expersq =nwifeinc = 0.

Figure 29.3 Goodness of Fit

Goodness-of-Fit Measures

Measure	Value	Formula
Likelihood Ratio (R)	227.14	2 * (LogL - LogL0)
Upper Bound of R (U)	1029.7	- 2 * LogL0
Aldrich-Nelson	0.2317	R / (R+N)
Cragg-Uhler 1	0.2604	1 - exp(-R/N)
Cragg-Uhler 2	0.3494	(1-exp(-R/N)) / (1-exp(-U/N))
Estrella	0.2888	1 - (1-R/U)^(U/N)
Adjusted Estrella	0.2693	1 - ((LogL-K)/LogL0)^(-2/N*LogL0)
McFadden's LRI	0.2206	R / U
Veall-Zimmermann	0.4012	(R * (U+N)) / (U * (R+N))
McKelvey-Zavoina	0.4025	

N = # of observations, K = # of regressors

The parameter estimates and standard errors are shown in Figure 29.4.

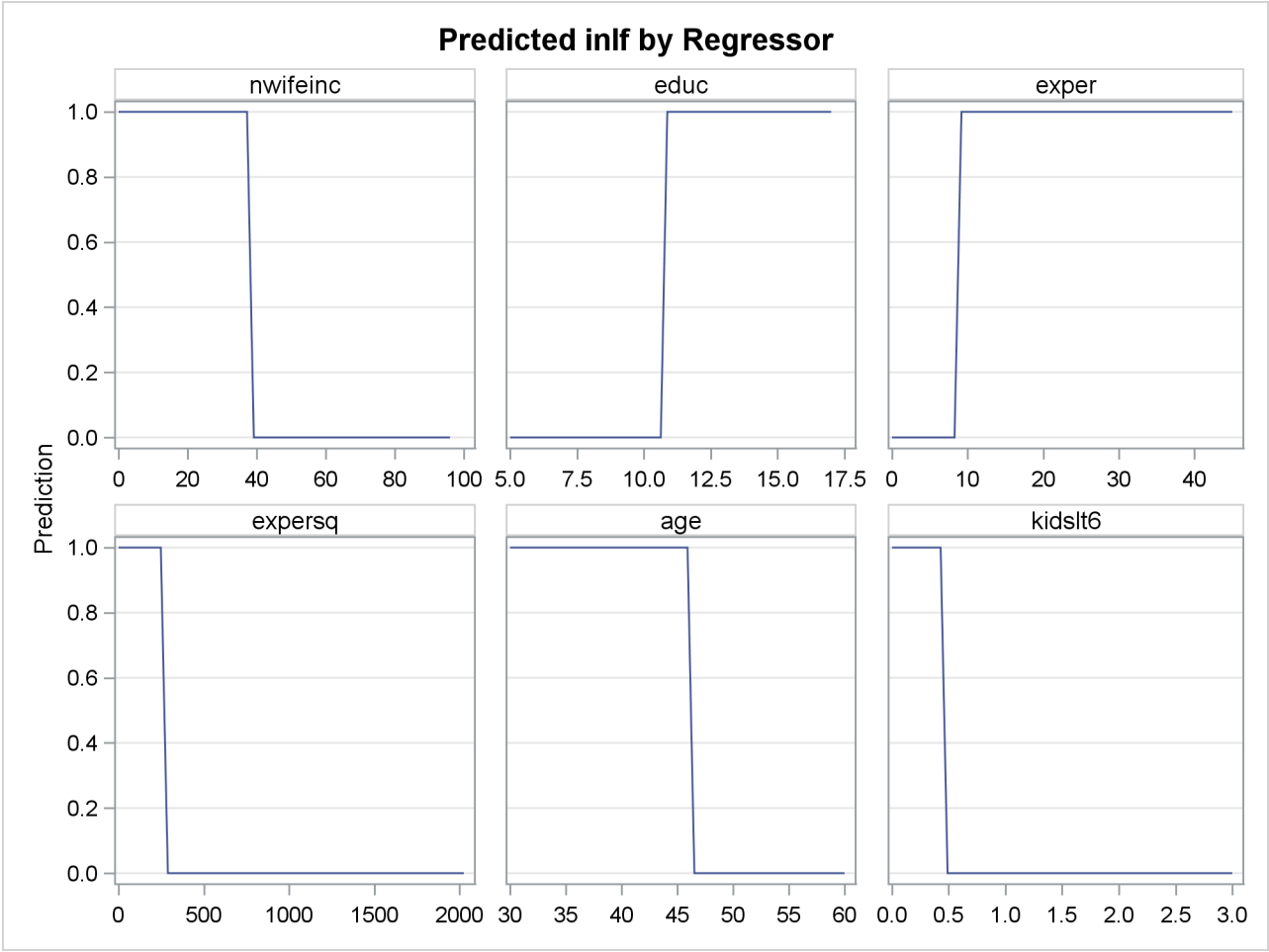
Figure 29.4 Parameter Estimates of Binary Probit

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	0.270077	0.508590	0.53	0.5954
nwifeinc	1	-0.012024	0.004840	-2.48	0.0130
educ	1	0.130905	0.025255	5.18	<.0001
exper	1	0.123348	0.018720	6.59	<.0001
expersq	1	-0.001887	0.000600	-3.14	0.0017
age	1	-0.052853	0.008477	-6.24	<.0001
kidslt6	1	-0.868329	0.118519	-7.33	<.0001
kidsge6	1	0.036005	0.043477	0.83	0.4076

Finally, the QLIM procedure profiles the predicted outcome with respect to the regressors. For example, Output 29.5 shows the predicted values profiled with respect to nwifeinc, educ, exper, expersq, age, and kidslt6.

Figure 29.5 Predictions by Regressors: nwifeinc, educ, exper, expersq, age, and kidslt6



When the error term has a logistic distribution, the binary logit model is estimated. To specify a logistic distribution, add D=LOGIT option as follows:

```
/*-- Binary Logit --*/
proc qlim data=mroz;
   model inlf = nwifeinc educ exper expersq
                age kidslt6 kidsge6  / discrete(d=logit);
run;
```

The estimated parameters are shown in Figure 29.6.

Figure 29.6 Parameter Estimates of Binary Logit

Binary Data

The QLIM Procedure

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	0.425452	0.860365	0.49	0.6210
nwifeinc	1	-0.021345	0.008421	-2.53	0.0113
educ	1	0.221170	0.043441	5.09	<.0001
exper	1	0.205870	0.032070	6.42	<.0001
expersq	1	-0.003154	0.001017	-3.10	0.0019
age	1	-0.088024	0.014572	-6.04	<.0001
kidslt6	1	-1.443354	0.203575	-7.09	<.0001
kidsge6	1	0.060112	0.074791	0.80	0.4215

The heteroscedastic logit model can be estimated using the HETERO statement. If the variance of the logit model is a function of the family income level excluding wife's income (nwifeinc), the variance can be specified as

$$\mathrm{Var}(\epsilon_i) = \sigma^2 \exp(\gamma * \mathrm{nwifeinc}_i)$$

where σ^2 is normalized to 1 because the dependent variable is discrete. The following SAS statements estimate the heteroscedastic logit model:

```
/*-- Binary Logit with Heteroscedasticity --*/
proc qlim data=mroz;
  model inlf = nwifeinc educ exper expersq
               age kidslt6 kidsge6 / discrete(d=logit);
  hetero inlf ~ nwifeinc / noconst;
run;
```

The parameter estimate, γ, of the heteroscedasticity variable is listed as _H.nwifeinc; see Figure 29.7.

Figure 29.7 Parameter Estimates of Binary Logit with Heteroscedasticity

Binary Data

The QLIM Procedure

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	0.510445	0.983538	0.52	0.6038
nwifeinc	1	-0.026778	0.012108	-2.21	0.0270
educ	1	0.255547	0.061728	4.14	<.0001
exper	1	0.234105	0.046639	5.02	<.0001
expersq	1	-0.003613	0.001236	-2.92	0.0035
age	1	-0.100878	0.021491	-4.69	<.0001
kidslt6	1	-1.645206	0.311296	-5.29	<.0001
kidsge6	1	0.066941	0.085633	0.78	0.4344
_H.nwifeinc	1	0.013280	0.013606	0.98	0.3291

Syntax: QLIM Procedure

The following statements are available in the QLIM procedure:

PROC QLIM *options* ;
BAYES *< options >* ;
BOUNDS *bound1 < , bound2 . . . >* ;
BY *variables* ;
CLASS *variables* ;
FREQ *variable* ;
ENDOGENOUS *variables ~ options* ;
HETERO *dependent variables ~ exogenous variables / options* ;
INIT *initvalue1 < , initvalue2 . . . >* ;
MODEL *dependent variables = regressors / options* ;
NLOPTIONS *options* ;
OUTPUT *options* ;
PRIOR *variables ~ distributions* ;
RANDOM *random-effects < / options >* ;
RESTRICT *restriction1 < , restriction2 . . . >* ;
TEST *options* ;
WEIGHT *variable* ;

At least one MODEL statement is required. If more than one MODEL statement is used, the QLIM procedure estimates a system of models. If a FREQ or WEIGHT statement is specified more than once, the variable specified in the first instance is used. Main effects and higher-order terms can be specified in the MODEL statement, as in the GLM procedure and PROBIT procedure in SAS/STAT. If a CLASS statement is used, it must precede the MODEL statement.

Functional Summary

Table 29.1 summarizes the statements and options used with the QLIM procedure.

Table 29.1 PROC QLIM Functional Summary

Description	Statement	Option
Data Set Options		
Specifies the input data set	QLIM	DATA=
Writes parameter estimates to an output data set	QLIM	OUTEST=
Writes predictions to an output data set	OUTPUT	OUT=
Declaring the Role of Variables		
Specifies BY-group processing	BY	
Specifies classification variables	CLASS	
Specifies a frequency variable	FREQ	
Specifies a weight variable	WEIGHT	NONORMALIZE
Printing Control Options		
Requests all printing options	QLIM	PRINTALL
Prints correlation matrix of the estimates	QLIM	CORRB
Prints covariance matrix of the estimates	QLIM	COVB
Prints a summary iteration listing	QLIM	ITPRINT
Suppresses the normal printed output	QLIM	NOPRINT
Plotting Options		
Displays plots	QLIM	PLOTS=
Options to Control the Optimization Process		
Specifies the optimization method	QLIM	METHOD=
Specifies the optimization options	NLOPTIONS	see Chapter 7, "Nonlinear Optimization Methods,"
Sets initial values for parameters	INIT	
Specifies upper and lower bounds for the parameter estimates	BOUNDS	
Specifies linear restrictions on the parameter estimates	RESTRICT	
Model Estimation Options		
Specifies options specific to Box-Cox transformation	MODEL	BOXCOX()
Suppresses the intercept parameter	MODEL	NOINT
Specifies variable selection	MODEL	SELECTVAR=( )
Specifies the type of random number generators	MODEL	RANDNUM=
Specifies that initial values are generated using random numbers	MODEL	RANDOMINIT

Table 29.1 *continued*

Description	Statement	Option
Specifies a seed for pseudo-random number generation	QLIM	SEED=
Specifies the number of draws for Monte Carlo integration	QLIM	NDRAW=
Specifies the method to calculate parameter covariance	QLIM	COVEST=
Requests estimation by Heckman's two-step method	QLIM	HECKIT
Integration Method Options for Random-Effects Models		
Requests the simulation method	RANDOM	METHOD=SIMULATION()
Requests the Gauss-Hermite quadrature method	RANDOM	METHOD=HERMITE()
Requests the Halton sequence method	RANDOM	METHOD=HALTON()
Bayesian MCMC Options		
Controls the aggregation of multiple posterior chains	BAYES	AGGREGATION=
Automates the initialization of the MCMC algorithm	BAYES	AUTOMCMC()
Specifies the initial values of the MCMC	INIT	
Evaluates the marginal likelihood	BAYES	MARGINLIKE
Specifies the maximum number of tuning phases	BAYES	MAXTUNE=
Specifies the minimum number of tuning phases	BAYES	MINTUNE=
Specifies the number of burn-in iterations	BAYES	NBI=
Specifies the number of iterations during the sampling phase	BAYES	NMC=
Specifies the number of samples for the prior predictive analysis	BAYES	NMCPRIOR=
Specifies the number of threads to use during the sampling phase	BAYES	NTRDS=
Specifies the number of iterations during the tuning phase	BAYES	NTU=
Controls options for constructing the initial proposal covariance matrix	BAYES	PROPCOV=
Specifies the sampling scheme	BAYES	SAMPLING=
Specifies the random number generator seed	BAYES	SEED=
Prints the time required for the MCMC sampling	BAYES	SIMTIME
Controls the thinning of the Markov chain	BAYES	THIN=
Bayesian Summary Statistics and Convergence Diagnostics		
Displays convergence diagnostics	BAYES	DIAGNOSTICS=
Displays summary statistics of the posterior samples	BAYES	STATISTICS=
Bayesian Prior and Posterior Samples		
Specifies a SAS data set for the posterior samples	BAYES	OUTPOST=
Specifies a SAS data set for the prior samples	BAYES	OUTPRIOR=

Table 29.1 *continued*

Description	Statement	Option
Bayesian Analysis		
Specifies normal prior distribution	PRIOR	NORMAL(MEAN=, VAR=)
Specifies gamma prior distribution	PRIOR	GAMMA(SHAPE=, SCALE=)
Specifies square root gamma prior distribution	PRIOR	SQGAMMA(SHAPE=, SCALE=)
Specifies inverse gamma prior distribution	PRIOR	IGAMMA(SHAPE=, SCALE=)
Specifies square root inverse gamma prior distribution	PRIOR	SQIGAMMA(SHAPE=, SCALE=)
Specifies uniform prior distribution	PRIOR	UNIFORM(MIN=, MAX=)
Specifies beta prior distribution	PRIOR	BETA(SHAPE1=, SHAPE2=, MIN=, MAX=)
Specifies *t* prior distribution	PRIOR	T(LOCATION=, DF=)
Endogenous Variable Options		
Specifies discrete variable	ENDOGENOUS	DISCRETE()
Specifies censored variable	ENDOGENOUS	CENSORED()
Specifies truncated variable	ENDOGENOUS	TRUNCATED()
Specifies variable selection condition	ENDOGENOUS	SELECT()
Specifies stochastic frontier variable	ENDOGENOUS	FRONTIER()
Endogeneity and Overidentification Test Options		
Requests the variable addition test for endogeneity	ENDOGENOUS	ENDOTEST()
Requests the overidentification test	ENDOGENOUS	OVERID()
Heteroscedasticity Model Options		
Specifies the function for heteroscedasticity models	HETERO	LINK=
Squares the function for heteroscedasticity models	HETERO	SQUARE
Specifies no constant for heteroscedasticity models	HETERO	NOCONST
Output Control Options		
Outputs predicted values	OUTPUT	PREDICTED
Outputs structured part	OUTPUT	XBETA
Outputs residuals	OUTPUT	RESIDUAL
Outputs error standard deviation	OUTPUT	ERRSTD
Outputs marginal effects	OUTPUT	MARGINAL
Outputs probability for the current response	OUTPUT	PROB
Outputs probability for all responses	OUTPUT	PROBALL
Outputs expected value	OUTPUT	EXPECTED
Outputs conditional expected value	OUTPUT	CONDITIONAL
Outputs inverse Mills ratio	OUTPUT	MILLS
Outputs technical efficiency measures	OUTPUT	TE1
	OUTPUT	TE2
Includes covariances in the OUTEST= data set	QLIM	COVOUT

Table 29.1 *continued*

Description	Statement	Option
Includes correlations in the OUTEST= data set	QLIM	CORROUT
Test Request Options		
Requests Wald, Lagrange multiplier, and likelihood ratio tests	TEST	ALL
Requests the WALD test	TEST	WALD
Requests the Lagrange multiplier test	TEST	LM
Requests the likelihood ratio test	TEST	LR

PROC QLIM Statement

PROC QLIM *options* **;**

The following options can be used in the PROC QLIM statement.

Data Set Options

DATA=*SAS-data-set*
specifies the input SAS data set. If the DATA= option is not specified, PROC QLIM uses the most recently created SAS data set.

Output Data Set Options

OUTEST=*SAS-data-set*
writes the parameter estimates to an output data set.

COVOUT
writes the covariance matrix for the parameter estimates to the OUTEST= data set. This option is valid only if the OUTEST= option is specified.

CORROUT
writes the correlation matrix for the parameter estimates to the OUTEST= data set. This option is valid only if the OUTEST= option is specified.

Printing Options

NOPRINT
suppresses the normal printed output but does not suppress error listings. If NOPRINT option is set, then any other print option is turned off.

PRINTALL
turns on all the printing-control options. The options set by PRINTALL are COVB and CORRB.

CORRB
prints the correlation matrix of the parameter estimates.

COVB
prints the covariance matrix of the parameter estimates.

ITPRINT
prints the initial parameter estimates, convergence criteria, and all constraints of the optimization. At each iteration, objective function value, step size, maximum gradient, and slope of search direction are printed as well.

Model Estimation Options

COVEST=*covariance-option*
specifies the method to calculate the covariance matrix of parameter estimates. The supported covariance types are as follows:

OP	specifies the covariance from the outer product matrix.
HESSIAN	specifies the covariance from the inverse Hessian matrix.
QML	specifies the covariance from the outer product and Hessian matrices (the quasi-maximum likelihood estimates).

The default is COVEST=HESSIAN.

HECKIT < (*heckit-options***) >**
requests that the selection model be estimated by Heckman's two-step estimation method. You must specify exactly two MODEL statements when you use the HECKIT option. One of the models must be a binary probit model; therefore, you must specify the DISCRETE option in the MODEL or ENDOGENOUS statement. You base the selection on the binary probit model for the second model; therefore, you must specify the SELECT option for this model.

You can specify one or both of the following *heckit-options*:

SECONDSTAGE=OLS | ML
specifies the estimation method of the second stage of Heckman's two-step method. SECONDSTAGE=OLS requests the ordinary least squares method for the second stage. If you specify SECONDSTAGE=OLS, then the model of interest—that is, the model that uses the SELECT option—must be linear and contain a continuous dependent variable. Therefore, you cannot specify the DISCRETE, CENSORED, or TRUNCATED option along with the SELECT option for the model of interest. When you specify the SECONDSTAGE=OLS option, you cannot test or restrict the parameters of the model of interest. However, you can test or restrict the parameters of the selection model—that is, the model that defines the selection rule.

If you specify SECONDSTAGE=ML, then PROC QLIM uses the maximum likelihood method in the second stage, as it does in the first stage. When you specify SECONDSTAGE=ML, the model of interest can be nonlinear. Moreover, you can also use the TEST or RESTRICT statement to test or restrict the parameters of the model of interest.

By default, SECONDSTAGE=OLS.

UNCORRECTED
requests the conventional OLS standard errors when the second-stage estimation method is the ordinary least squares method. If you do not specify the UNCORRECTED option, PROC QLIM reports the corrected OLS standard errors. For more information about the corrected standard errors, see the section "Heckman's Two-Step Selection Method" on page 1997.

If you specify both the UNCORRECTED and SECONDSTAGE=ML options, PROC QLIM ignores the UNCORRECTED option, because the UNCORRECTED option is related to the OLS standard errors.

NDRAW=*value*
specifies the number of draws for Monte Carlo integration.

SEED=*value*
specifies a seed for pseudo-random number generation in Monte Carlo integration.

Optimization Process Control Options

PROC QLIM uses the nonlinear optimization (NLO) subsystem to perform nonlinear optimization tasks. You can use any of the NLO options in the NLOPTIONS statement. For more information, see Chapter 7, "Nonlinear Optimization Methods."

METHOD=*value*
specifies the optimization method. If this option is specified, it overwrites the TECH= option in NLOPTIONS statement. Valid values are as follows:

CONGRA	performs a conjugate-gradient optimization
DBLDOG	performs a version of double-dogleg optimization
NEWRAP	performs a Newton-Raphson optimization combining a line-search algorithm with ridging
NMSIMP	performs a Nelder-Mead simplex optimization
NONE	specifies that no optimization be performed beyond using the ordinary least squares method to compute the parameter estimates
NRRIDG	performs a Newton-Raphson optimization with ridging
QUANEW	performs a quasi-Newton optimization
TRUREG	performs a trust region optimization

The default method is METHOD=QUANEW.

Plotting Options

PLOTS<(*global-plot-options***)> = ***plot-request* | *(plot-requests)***
controls the display of plots. By default, the plots are displayed in panels unless the UNPACK *global-plot-option* is specified. When you specify only one *plot-request*, you can omit the parentheses around the *plot-request*.

Global Plot Options

You can specify the following *global-plot-options*:

ONLY
displays only the requested plot.

PRIOR
displays the prior predictive graph that is associated with the requested posterior predictive plot BAYESPRED. This option is available only for Bayesian analysis.

UNPACKPANEL
UNPACK
specifies that all paneled plots be unpacked, meaning that each plot in a panel is displayed separately.

Plot Requests

You can specify the following *plot-requests*:

ALL
specifies all types of available plots.

AUTOCORR< (LAGS=*n*) >
displays the autocorrelation function plots for the parameters. This *plot-request* is available only for Bayesian analysis. The optional LAGS= suboption the number (up to lag *n*) of autocorrelations to be plotted in the AUTOCORR plot. If this suboption is not specified, autocorrelations are plotted up to lag 50.

BAYESDIAG
displays the TRACE, AUTOCORR, and DENSITY plots. This *plot-request* is available only for Bayesian analysis.

BAYESPRED
displays the predictive analysis. The predictive analysis takes into account the variability of the error term, whereas the PREDICTED *plot-request* does not. The BAYESPRED *plot-request* is available only for Bayesian analysis.

BAYESSUM
displays the posterior distribution, the prior distribution, and the maximum likelihood estimates. This *plot-request* is available only for Bayesian analysis.

CONDITIONAL
displays the conditional expected values for continuous endogenous variables. Each contributing regressor is set equal to its mean, except for the parameter that is reported on the X axis. This *plot-request* is not available for Bayesian analysis.

DENSITY< (FRINGE) >
displays the kernel density plots for the parameters. This *plot-request* is available only for Bayesian analysis. If you specify the FRINGE suboption, a fringe plot is created on the X axis of the kernel density plot. This *plot-request* is available only for Bayesian analysis.

ERRSTD
displays the error standard deviation versus observed regressors when you also specify a HETERO statement. This *plot-request* is not available for Bayesian analysis.

EXPECTED
displays the expected values for continuous endogenous variables. Each contributing regressor is set equal to its mean, except for the parameter that is reported on the X axis. This *plot-request* is not available for Bayesian analysis.

MARGINAL
displays the marginal effects. Each contributing regressor is set equal to its mean, except for the parameter that is reported on the X axis. This *plot-request* is not available for Bayesian analysis.

MILLS
displays the inverse Mills ratio. Each contributing regressor is set equal to its mean, except for the parameter that is reported on the X axis. This *plot-request* is not available for Bayesian analysis.

NONE
suppresses all diagnostic plots.

PREDICTED
displays the model predicted values. Each contributing regressor is set equal to its mean, except for the parameter that is reported on the X axis. This *plot-request* is not available for Bayesian analysis.

PROB
displays the predicted response probability. Each contributing regressor is set equal to its mean, except for the parameter that is reported on the X axis. This *plot-request* is not available for Bayesian analysis.

PROBALL
displays the predicted probabilities for each level of the response. Each contributing regressor is set equal to its mean, except for the parameter that is reported on the X axis. This *plot-request* is not available for Bayesian analysis.

PROFLIK
displays the profiled log likelihood. Each profiled graph is obtained by setting all the parameters to their maximum likelihood estimate except for the profiling parameter. The profiling parameter takes values on a predefined grid that is determined by the maximum likelihood estimate of the corresponding standard deviation. When a restricted optimization is requested, the profiled log likelihood plots depict the behavior of the profiled log likelihood around the restricted MLE without imposing the actual restrictions.

RESIDUAL
displays the residuals versus observed regressors. This *plot-request* is not available for Bayesian analysis.

TE1
displays the technical efficiency for the stochastic frontier model as suggested by Battese and Coelli (1988). Each contributing regressor is set equal to its mean, except for the parameter that is reported on the X axis. This *plot-request* is not available for Bayesian analysis.

TE2
displays the technical efficiency for the stochastic frontier model as suggested by Jondrow et al. (1982). Each contributing regressor is set equal to its mean, except for the parameter that is reported on the X axis. This *plot-request* is not available for Bayesian analysis.

TRACE< (SMOOTH) >
displays the trace plots for the parameters. This *plot-request* is available only for Bayesian analysis. The SMOOTH suboption displays a fitted penalized B-spline curve for each TRACE plot.

XBETA
displays the structural part on the right-hand side of the model. Each contributing regressor is set equal to its mean, except for the parameter that is reported on the X axis. This is not available for Bayesian analysis.

BAYES Statement

BAYES < *options* > **;**

The BAYES statement controls the Metropolis sampling scheme that is used to obtain samples from the posterior distribution of the underlying model and data.

AGGREGATION=WEIGHTED | UNWEIGHTED (Experimental)
specifies how multiple posterior samples should be aggregated. AGGREGATION=WEIGHTED implements a weighted resampling scheme for the aggregation of multiple posterior chains. You can use this option when the posterior distribution is characterized by several very distinct posterior modes. AGGREGATION=UNWEIGHTED aggregates multiple posterior chains without any adjustment. You can use this option when the posterior distribution is characterized by one or few relatively close posterior modes. By default, AGGREGATION=UNWEIGHTED. For more information, see the section "Aggregation of Multiple Chains" on page 2013.

AUTOMCMC < =(*automcmc-options***) >** (Experimental)
specifies an algorithm for the auto-initialization of the MCMC sampling algorithm. For more information, see the section "Automated Initialization of MCMC" on page 2014.

ACCURACY= (*accuracy-options***)**
customizes the behavior of the AUTOMCMC algorithm when you are searching for an accurate representation of the posterior distribution. You can specify the following *accuracy-options*:

ATTEMPTS=*number*
specifies the maximum number of attempts that is required in order to obtain accurate samples from the posterior distribution. By default,ATTEMPTS=10.

TARGETESS=*number*
requests that the accuracy search be based on the effective sample size (ESS) analysis. If you specify this option, you must also specify the minimum *number* of effective samples.

TARGETSTATS<=(*targetstats-option***)>**
requests that the accuracy search be based on the analysis of the posterior mean and a posterior quantile of interest. You can customize the behavior of the analysis of the posterior

mean by adjusting HEIDELBERGER sub-options. You can customize the behavior of the analysis of the posterior quantile by adjusting the RAFTERY sub-options. If you specify TARGETSTATS, you can also specify how the Raftery-Lewis test should be interpreted by using the following option:

RLLIMITS=(LB=*number* **UB=***number***)**
specifies a region where the search for the optimal sample size depends directly on the Raftery-Lewis test. By default, RLLIMITS (LB=10000 UB=300000).

TOL=*value*
specifies the proportion of parameters that are required to be accurate. By default, TOL=0.95.

MAXNMC=*number*
specifies the maximum number of posterior samples that the AUTOMCMC option allows. By default, MAXNMC=700000.

RANDINIT < =(*randinit-options***) >**
specifies random starting points for the MCMC algorithm. The starting points can be sampled around the maximum likelihood estimate and around the prior mean. You can specify the following *randinit-options*:

MULTIPLIER= (*value***)**
specifies the radius of the area where the starting points are sampled. For the starting points that are sampled around the maximum likelihood estimate, the radius equals the standard deviation of the maximum likelihood estimate multiplied by the multiplier value. For the starting points that are sampled around the prior mean, the radius equals the standard deviation of the prior distribution multiplied by the multiplier value. By default, MULTIPLIER=2.

PROPORTION= (*value***)**
specifies the proportion of starting points that are sampled around the maximum likelihood estimate and around the prior mean. By default, PROPORTION=0, which implies that all the initial points are sampled around the maximum likelihood estimate. If you use choose to sample starting points around the prior mean, the convergence of the MCMC algorithm could be very slow.

STATIONARITY= (*stationarity-options***)**
customizes the behavior of the AUTOMCMC algorithm when you are trying to sample from the posterior distribution. You can specify the following *stationarity-options*:

ATTEMPTS=*number*
specifies the maximum number of attempts that are required in order to obtain stationary samples from the posterior distribution. By default, ATTEMPTS=10.

TOL=*value*
specifies the proportion of parameter whose samples must to be stationary. By default, TOL=0.95.

DIAGNOSTICS=ALL | NONE | *(keyword-list)*

DIAG=ALL | NONE | *(keyword-list)*

controls which diagnostics are produced. All the following diagnostics are produced with DIAGNOSTICS=ALL. If you do not want any of these diagnostics, specify DIAGNOSTICS=NONE. If you want some but not all of the diagnostics, or if you want to change certain settings of these diagnostics, specify a subset of the following keywords. By default, DIAGNOSTICS=NONE.

AUTOCORR <(LAGS= *numeric-list*) **>**

computes the autocorrelations at lags that are specified in the *numeric-list*. Elements in the *numeric-list* are truncated to integers, and repeated values are removed. If the LAGS= option is not specified, autocorrelations of lags 1, 5, 10, and are computed.

AUTOMCMCSUM

produces a summary table for the AUTOMCMC (automatic MCMC) sampling tool is used.

ESS

computes Carlin's estimate of the effective sample size, the correlation time, and the efficiency of the chain for each parameter.

GEWEKE <(*geweke-options***) >**

computes the Geweke spectral density diagnostics, which are essentially a two-sample t test between the first f_1 portion and the last f_2 portion of the chain. The default is $f_1 = 0.1$ and $f_2 = 0.5$, but you can choose other fractions by using the following *geweke-options*:

FRAC1=*value*

specifies the fraction f_1 for the first window.

FRAC2=*value*

specifies the fraction f_2 for the second window.

HEIDELBERGER <(*heidel-options***) >**

computes the Heidelberger and Welch diagnostic for each variable, which consists of a stationarity test of the null hypothesis that the sample values form a stationary process. If the stationarity test is not rejected, a halfwidth test is then carried out. Optionally, you can specify one or more of the following *heidel-options*:

SALPHA=*value*

specifies the α level ($0 < \alpha < 1$) for the stationarity test.

HALPHA=*value*

specifies the α level ($0 < \alpha < 1$) for the halfwidth test.

EPS=*value*

specifies a positive number ϵ such that if the halfwidth is less than ϵ times the sample mean of the retained iterates, the halfwidth test is passed.

MCSE

MCERROR

computes the Monte Carlo standard error for each parameter. The Monte Carlo standard error, which measures the simulation accuracy, is the standard error of the posterior mean estimate and is calculated as the posterior standard deviation divided by the square root of the effective sample size.

RAFTERY<(*raftery-options*)>
computes the Raftery and Lewis diagnostics, which evaluate the accuracy of the estimated quantile ($\hat{\theta}_Q$ for a given $Q \in (0,1)$) of a chain. $\hat{\theta}_Q$ can achieve any degree of accuracy when the chain is allowed to run for a long time. The computation is stopped when the estimated probability $\hat{P}_Q = \Pr(\theta \leq \hat{\theta}_Q)$ reaches within $\pm R$ of the value Q with probability S; that is, $\Pr(Q - R \leq \hat{P}_Q \leq Q + R) = S$. The following *raftery-options* enable you to specify Q, R, S, and a precision level ϵ for the test:

QUANTILE | Q=*value*
specifies the order (a value between 0 and 1) of the quantile of interest. The default is 0.025.

ACCURACY | R=*value*
specifies a small positive number as the margin of error for measuring the accuracy of estimation of the quantile. The default is 0.005.

PROBABILITY | S=*value*
specifies the probability of attaining the accuracy of the estimation of the quantile. The default is 0.95.

EPSILON | EPS=*value*
specifies the tolerance level (a small positive number) for the stationary test. The default is 0.001.

MARGINLIKE<(NSIM=*number***)>** (Experimental)
evaluates of the logarithm of the marginal likelihood. Two estimates are produced: the cross entropy estimate and the harmonic mean. The cross entropy estimate is based on an importance sampling algorithm. You can specify the number of importance samples in the NSIM=*number* option. By default NSIM=10,000. For more information, see the section "Marginal Likelihood" on page 2022.

MAXTUNE=*number*
specifies the maximum number of tuning phases. The default is 24.

MINTUNE=*number*
specifies the minimum number of tuning phases. The default is 2.

NBI=*number*
specifies the number of burn-in iterations before the chains are saved. The default is 1,000.

NMC=*number*
specifies the number of iterations after the burn-in. The default is 1,000.

NMCPRIOR=*number*
specifies the number of samples for the prior predictive analysis when PLOTS(PRIOR)=BAYESPRED is requested. The default is 10,000.

NTRDS=*number*

THREADS=*number*
specifies the number of threads to be used. The number of threads cannot exceed the number of computer cores available. Each core samples the number of iterations that is specified by the NMC option. The default is 1.

NTU=*number*
specifies the number of samples for each tuning phase. The default is 500.

OUTPOST=*SAS-data-set*
names the SAS data set to contain the posterior samples. Alternatively, you can create the output data set by specifying an ODS OUTPUT statement as follows:

ODS OUTPUT POSTERIORSAMPLE = < *SAS-data-set* > **;**

OUTPRIOR=*SAS-data-set*
names the SAS data set to contain the prior samples used to generate the prior predictive analysis when you request the prior predictive plots. Alternatively, you can create the output data set by specifying an ODS OUTPUT statement as follows:

ODS OUTPUT PRIORSAMPLE = < *SAS-data-set* > **;**

PROPCOV=*value*
specifies the method used in constructing the initial covariance matrix for the Metropolis-Hastings algorithm. The QUANEW and NMSIMP methods find numerically approximated covariance matrices at the optimum of the posterior density function with respect to all continuous parameters. The tuning phase starts at the optimized values; in some problems, this can greatly increase convergence performance. If the approximated covariance matrix is not positive definite, then an identity matrix is used instead. You can specify the following values:

CONGRA
performs a conjugate-gradient optimization.

DBLDOG
performs a version of double-dogleg optimization.

NEWRAP
performs a Newton-Raphson optimization that combines a line-search algorithm with ridging.

NMSIMP
performs a Nelder-Mead simplex optimization.

NRRIDG
performs a Newton-Raphson optimization with ridging.

QUANEW
performs a quasi-Newton optimization.

TRUREG
performs a trust-region optimization.

SAMPLING=MULTIMETROPOLIS | UNIMETROPOLIS | MODELMETROPOLIS
specifies how to sample from the posterior distribution. SAMPLING=MULTIMETROPOLIS implements a Metropolis sampling scheme on a single block that contains all the parameters of the model. SAMPLING=MODELMETROPOLIS implements a Metropolis sampling scheme on multiple blocks: one block for each model (all the parameters of the model) plus a block for all the correlation parameters across the models. SAMPLING=UNIMETROPOLIS implements a Metropolis sampling scheme on multiple blocks, one for each parameter of the model. By default, SAMPLING=MULTIMETROPOLIS.

SEED=*number*

specifies an integer seed in the range 1 to $2^{31} - 1$ for the random number generator in the simulation. Specifying a seed enables you to reproduce identical Markov chains for the same specification. If you do not specify the SEED= option, or if you specify a nonpositive seed, a random seed is derived from the time of day.

SIMTIME

prints the time required for the MCMC sampling.

STATISTICS < (*global-options***) > = ALL | NONE |** *keyword* **|** *(keyword-list)*

STATS < (*global-options***) > = ALL | NONE |** *keyword* **|** *(keyword-list)*

controls the number of posterior statistics produced. Specifying STATISTICS=ALL is equivalent to specifying STATISTICS= (CORR COV INTERVAL PRIOR SUMMARY). If you do not want any posterior statistics, specify STATISTICS=NONE. The default is STATISTICS=(SUMMARY INTERVAL). You can specify the following *global-options*:

ALPHA=*numeric-list*

controls the probabilities of the credible intervals. The ALPHA= values must be between 0 and 1. Each ALPHA= value produces a pair of 100(1–ALPHA)% equal-tail and HPD intervals for each parameter. The default is ALPHA=0.05, which yields the 95% credible intervals for each parameter.

PERCENT=*numeric-list*

requests the percentile points of the posterior samples. The PERCENT= values must be between 0 and 100. The default is PERCENT=25, 50, 75, which yields the 25th, 50th, and 75th percentile points, respectively, for each parameter.

You can specify the following *keywords*:

CORR

produces the posterior correlation matrix.

COV

produces the posterior covariance matrix.

INTERVAL

produces equal-tail credible intervals and HPD intervals. The default is to produce the 95% equal-tail credible intervals and 95% HPD intervals, but you can use the ALPHA= *global-option* to request intervals of any probabilities.

NONE

suppresses printing of all summary statistics.

PRIOR

produces a summary table of the prior distributions used in the Bayesian analysis.

SUMMARY

produces the means, standard deviations, and percentile points (25th, 50th, and 75th) for the posterior samples. You can use the global PERCENT= *global-option* to request specific percentile points.

THIN=*number*

THINNING=*number*

controls the thinning of the Markov chain. Only one in every k samples is used when THIN=k, and if NBI=n_0 and NMC=n, the number of samples that are kept is

$$\left[\frac{n_0 + n}{k}\right] - \left[\frac{n_0}{k}\right]$$

where $[a]$ represents the integer part of the number a. The default is THIN=1.

BOUNDS Statement

BOUNDS *bound1* < , *bound2* . . . > **;**

The BOUNDS statement imposes simple boundary constraints on the parameter estimates. BOUNDS statement constraints refer to the parameters estimated by the QLIM procedure. Any number of BOUNDS statements can be specified.

Each *bound* is composed of parameters and constants and inequality operators. Parameters associated with regressor variables are referred to by the names of the corresponding regressor variables:

item operator item < *operator item* < *operator item* . . . > >

Each *item* is a constant, the name of a parameter, or a list of parameter names. See the section "Naming of Parameters" on page 2032 for more details on how parameters are named in the QLIM procedure. Each *operator* is '<', '>', '<=', or '>='.

Both the BOUNDS statement and the RESTRICT statement can be used to impose boundary constraints; however, the BOUNDS statement provides a simpler syntax for specifying these kinds of constraints. See the "RESTRICT Statement" on page 1982 for more information.

The following BOUNDS statement constrains the estimates of the parameters associated with the variable ttime and the variables x1 through x10 to be between 0 and 1. This example illustrates the use of parameter lists to specify boundary constraints.

```
bounds 0 < ttime x1-x10 < 1;
```

The following BOUNDS statement constrains the estimates of the correlation (_RHO) and sigma (_SIGMA) in the bivariate model:

```
bounds _rho >= 0, _sigma.y1 > 1, _sigma.y2 < 5;
```

The BOUNDS statement is not supported if a BAYES statement is also specified. In Bayesian analysis, the restrictions on parameters are usually introduced through the prior distribution.

BY Statement

BY *variables* **;**

A BY statement can be used with PROC QLIM to obtain separate analyses on observations in groups defined by the BY variables.

CLASS Statement

CLASS *variables* **;**

The CLASS statement names the classification variables to be used in the analysis. Classification variables can be either character or numeric.

Class levels are determined from the formatted values of the CLASS variables. Thus, you can use formats to group values into levels. See the discussion of the FORMAT procedure in *SAS Language Reference: Dictionary* for details.

ENDOGENOUS Statement

ENDOGENOUS *variables* ∼ *options* **;**

The ENDOGENOUS statement specifies the type of dependent variables that appear on the left-hand side of the equation. Endogenous variables listed refer to the dependent variables that appear on the left-hand side of the equation.

Discrete Variable Options

DISCRETE <(*discrete-options* **)>**
specifies that the endogenous variables in this statement are discrete. Valid *discrete-options* are as follows:

ORDER=DATA | FORMATTED | FREQ | INTERNAL
specifies the sorting order for the levels of the discrete variables specified in the ENDOGENOUS statement. This ordering determines which parameters in the model correspond to each level in the data. The following table shows how PROC QLIM interprets values of the ORDER= option.

Value of ORDER=	Levels Sorted By
DATA	Order of appearance in the input data set
FORMATTED	Formatted value
FREQ	Descending frequency count; levels with the most observations come first in the order
INTERNAL	Unformatted value

By default, ORDER=FORMATTED. For the values FORMATTED and INTERNAL, the sort order is machine dependent. For more information about sorting order, see the chapter on the SORT procedure in the *Base SAS Procedures Guide*.

DISTRIBUTION=NORMAL | LOGISTIC

DIST=NORMAL | LOGISTIC

D=NORMAL | LOGISTIC

specifies the cumulative distribution function used to model the response probabilities. DISTRIBUTION=NORMAL specifies the normal distribution for the probit model. DISTRIBUTION=LOGISTIC specifies the logistic distribution for the logit model.

By default, DISTRIBUTION=NORMAL.

If a multivariate model is specified, logistic distribution is not allowed. Only normal distribution is supported.

Censored Variable Options

CENSORED (*censored-options* **)**

specifies that the endogenous variables in this statement be censored. Valid *censored-options* are as follows:

LB=*value or variable*

LOWERBOUND=*value or variable*

specifies the lower bound of the censored variables. If *value* is missing or the value in *variable* is missing, no lower bound is set. By default, no lower bound is set.

UB=*value or variable*

UPPERBOUND=*value or variable*

specifies the upper bound of the censored variables. If *value* is missing or the value in *variable* is missing, no upper bound is set. By default, no upper bound is set.

Truncated Variable Options

TRUNCATED (*truncated-options* **)**

specifies that the endogenous variables in this statement be truncated. Valid *truncated-options* are as follows:

LB=*value or variable*

LOWERBOUND=*value or variable*

specifies the lower bound of the truncated variables. If *value* is missing or the value in *variable* is missing, no lower bound is set. By default, no lower bound is set.

UB=*value or variable*

UPPERBOUND=*value or variable*

specifies the upper bound of the truncated variables. If *value* is missing or the value in *variable* is missing, no upper bound is set. By default, no upper bound is set.

Stochastic Frontier Variable Options

FRONTIER < (*frontier-options* **) >**
specifies that the endogenous variable in this statement follow a production or cost frontier. Valid *frontier-options* are as follows:

TYPE=HALF | EXPONENTIAL | TRUNCATED
specifies the model type:

HALF	specifies a half-normal model.
EXPONENTIAL	specifies an exponential model.
TRUNCATED	specifies a truncated normal model.

PRODUCTION
specifies that the model estimated be a production function.

COST
specifies that the model estimated be a cost function.

If neither PRODUCTION nor COST option is specified, production function is estimated by default.

Selection Options

SELECT (*select-option* **)**
specifies selection criteria for sample selection model. The BAYES statement does not support the SELECT option. The *select-option* specifies the condition for the endogenous variable to be selected. It is written as a variable name, followed by an equality operator (=) or an inequality operator (<, >, <=, >=), followed by a number:

variable operator number

The *variable* is the endogenous variable that the selection is based on. The *operator* can be =, <, >, <= , or >=. Multiple *select-options* can be combined with the logic operators: AND, OR. The following example illustrates the use of the SELECT option:

```
endogenous y1 ~ select(z=0);
endogenous y2 ~ select(z=1 or z=2);
```

The SELECT option can be used together with the DISCRETE, CENSORED, or TRUNCATED option. For example:

```
endogenous y1 ~ select(z=0) discrete;
endogenous y2 ~ select(z=1) censored (lb=0);
endogenous y3 ~ select(z=1 or z=2) truncated (ub=10);
```

For more information about selection models with censoring or truncation, see the section "Selection Models" on page 1996.

Endogeneity and Overidentification Test Options

ENDOTEST (*regressors*)

requests the test of endogeneity for a list of regressors in the model. More specifically, this option tests the null hypothesis that the specified regressors are exogenous. Each of these regressors must also have a model of its own. The former model is considered the structural model, and the latter models are considered reduced form models.

The following example illustrates the use of the ENDOTEST option by testing whether the regressors $y2$ and $y3$ are endogenous in the model for $y1$:

```
proc qlim;
   model y1 = y2 y3 x1;
   model y2 = x1 x2 x3 x4 x5;
   model y3 = x1 x2 x3 x4 x5;
   endogenous y1 ~ endotest(y2 y3);
run;
```

The ENDOTEST option is not available when you specify the SELECT or FRONTIER option. You can specify the ENDOTEST option only once for each ENDOGENOUS statement.

For more information about the test for endogeneity, see the section "Test for Endogeneity" on page 2006.

OVERID (*variables*)

requests the overidentification test for a list of variables. These variables are the overidentifying instrumental variables that you provide from the reduced form models. For more information, see the section "Overidentification Test" on page 2007.

The following example illustrates the use of the OVERID option:

```
proc qlim;
   model y1 = y2 y3 x1;
   model y2 = x1 x2 x3 x4 x5;
   model y3 = x1 x2 x3 x4 x5;
   endogenous y1 ~ overid(y2.x4 y3.x5);
run;
```

The regressors $y2$ and $y3$ in the model for $y1$ are the endogenous variables. Therefore, each of these variables has its own models, which are considered reduced form models. The overidentifying instrumental variables are $x4$ and $x5$. If you specify the OVERID option as

```
endogenous y1 ~ overid(y2.x4 y2.x5);
```

then you consider only the regressor $y2$ to be endogenous, and the model for $y3$ is ignored during the testing process.

The OVERID option is not available when you specify the SELECT or FRONTIER option. You can specify the OVERID option only once for each ENDOGENOUS statement.

FREQ Statement

FREQ *variable* **;**

The FREQ statement identifies a variable that contains the frequency of occurrence of each observation. PROC QLIM treats each observation as if it appears n times, where n is the value of the FREQ variable for the observation. If it is not an integer, the frequency value is truncated to an integer. If the frequency value is less than 1 or missing, the observation is not used in the model fitting. When the FREQ statement is not specified, each observation is assigned a frequency of 1. If you specify more than one FREQ statement, then the first FREQ statement is used.

HETERO Statement

HETERO *dependent variables* ~ *exogenous variables* < / *options* > **;**

The HETERO statement specifies variables that are related to the heteroscedasticity of the residuals and the way these variables are used to model the error variance. The heteroscedastic regression model supported by PROC QLIM is

$$y_i = \mathbf{x}_i'\boldsymbol{\beta} + \epsilon_i$$

$$\epsilon_i \sim \mathrm{N}(0, \sigma_i^2)$$

See the section "Heteroscedasticity" on page 1993 for more details on the specification of functional forms.

LINK=*value*

The functional form can be specified using the LINK= option. The following option values are allowed:

EXP specifies the exponential link function

$$\sigma_i^2 = \sigma^2(1 + \exp(\mathbf{z}_i'\boldsymbol{\gamma}))$$

LINEAR specifies the linear link function

$$\sigma_i^2 = \sigma^2(1 + \mathbf{z}_i'\boldsymbol{\gamma})$$

When the LINK= option is not specified, the exponential link function is specified by default.

NOCONST

specifies that there be no constant in the exponential heteroscedasticity model.

$$\sigma_i^2 = \sigma^2\exp(\mathbf{z}_i'\boldsymbol{\gamma})$$

SQUARE

estimates the model by using the square of linear heteroscedasticity function. For example, you can specify the following heteroscedasticity function:

$$\sigma_i^2 = \sigma^2(1 + (\mathbf{z}_i'\boldsymbol{\gamma})^2)$$

```
model y = x1 x2 / discrete;
hetero y ~ z1 / link=linear square;
```

The option SQUARE does not apply to exponential heteroscedasticity function because the square of an exponential function of $\mathbf{z}_i'\boldsymbol{\gamma}$ is the same as the exponential of $2\mathbf{z}_i'\boldsymbol{\gamma}$. Hence the only difference is that all $\boldsymbol{\gamma}$ estimates are divided by two.

You can use the HETERO statement within a Bayesian framework, but you should do this carefully because convergence can be slower than in the homoscedastic case. For more information see "Priors for Heteroscedastic Models" on page 2019.

INIT Statement

INIT *initvalue1* < *, initvalue2 . . .* > **;**

The INIT statement sets initial values for parameters in the optimization. You can specify any number of INIT statements.

Each *initvalue* is written as a parameter or parameter list, followed by an optional equality operator (=), followed by a number:

parameter **<=>** *number*

If you also specify the BAYES statement, the INIT statement also initializes the Markov chain Monte Carlo (MCMC) algorithm. In particular, the INIT statement does one of the following:

- It initializes the tuning phase (this also includes the PROPCOV option).
- It initializes the sampling phase, if there is no tuning phase.

MODEL Statement

MODEL *dependent* = *regressors* < */ options* > **;**

The MODEL statement specifies the dependent variable and independent regressor variables for the regression model.

You can specify the following *options* after a slash (/).

LIMIT1=ZERO | VARYING
specifies the restriction of the threshold value of the first category when the ordinal probit or logit model is estimated. LIMIT1=ZERO is the default option. When LIMIT1=VARYING is specified, the threshold value is estimated.

NOINT
suppresses the intercept parameter.

Endogenous Variable Options

The endogenous variable options are the same as the options that you can specify in the ENDOGENOUS statement. If you specify an ENDOGENOUS statement, all endogenous options in the MODEL statement are ignored.

Endogeneity and Overidentification Test Options

The endogeneity and overidentification test options are the same as the options that you can specify in the ENDOGENOUS statement. If you specify an ENDOGENOUS statement, all endogeneity and overidentification test options in the MODEL statement are ignored.

BOXCOX Estimation Options

BOXCOX (*option-list* **)**

specifies options that are used for Box-Cox regression or regressor transformation. For example, the Box-Cox regression is specified as

```
model y = x1 x2 / boxcox(y=lambda,x1 x2)
```

PROC QLIM estimates the following Box-Cox regression model:

$$y_i^{(\lambda)} = \beta_0 + \beta_1 x_{1i}^{(\lambda_2)} + \beta_2 x_{2i}^{(\lambda_2)} + \epsilon_i$$

The *option-list* takes the form *variable-list* < = *varname* > separated by commas. The *variable-list* specifies that the list of variables have the same Box-Cox transformation; *varname* specifies the name of this Box-Cox coefficient. If *varname* is not specified, the coefficient is called _Lambda*i*, where *i* increments sequentially.

Variable Selection Options

SELECTVAR <=(*selectvar-option***)>**

enables variable selection. The *selectvar-option* specifies a variable selection method based on an information criterion. For more information, see the section "Variable Selection" on page 2000. You can specify the following *selectvar-options*:

DIRECTION=FORWARD | BACKWARD

specifies the searching algorithm to use in the variable selection method. By default, DIRECTION=FORWARD.

CRITER=AIC | SBC

specifies the information criterion to use for the variable selection. By default, CRITER=AIC.

MAXSTEPS=*value*

specifies the maximum number of steps that are allowed in the search algorithm. The default is 100.

LSTOP=*value*

specifies the stopping criterion. The *value* represents the percentage of decrease or increase in the AIC or SBC that is required for the algorithm to proceed; it must be a positive number less than 1. The default is 0.

RETAIN(*regressors***)**

specifies a list of regressors that are to be retained in any model that the variable selection process considers.

The following rules apply to how regressors are handled when you specify more than one MODEL statement and use the SELECTVAR option:

- If you do not specify the SELECTVAR option in a particular MODEL statement, then all regressors in the original model are included in any model that the variable selection algorithm considers. In other words, omitting the SELECTVAR option is equivalent to providing the option: SELECTVAR=(RETAIN(*all-regressors*)).
- If you specify the SELECTVAR option without any =(*option*) clause in a MODEL statement, then all regressors in that model (other than the intercept, if present) are eligible for potential exclusion as the variable selection process is executed.

The following example specifies 10 possible regressor candidates, out of which five are selected using the AIC criterions:

```
proc qlim data=one;
   model y = x1-x10 /selectvar=(direction=forward criter=AIC maxsteps=5);
run;
```

NLOPTIONS Statement

NLOPTIONS < *options* > **;**

PROC QLIM uses the nonlinear optimization (NLO) subsystem to perform nonlinear optimization tasks. For a list of all the options of the NLOPTIONS statement, see Chapter 7, "Nonlinear Optimization Methods."

OUTPUT Statement

OUTPUT < **OUT**=*SAS-data-set* > < *output-options* > **;**

The OUTPUT statement creates a new SAS data set containing all variables in the input data set and, optionally, the estimates of $\mathbf{x}'\boldsymbol{\beta}$, predicted value, residual, marginal effects, probability, standard deviation of the error, expected value, conditional expected value, technical efficiency measures, and inverse Mills ratio. When the response values are missing for the observation, all output estimates except residual are still computed as long as none of the explanatory variables is missing. This enables you to compute these statistics for prediction. You can specify only one OUTPUT statement.

Details on the specifications in the OUTPUT statement are as follows:

CONDITIONAL
outputs estimates of conditional expected values of continuous endogenous variables.

ERRSTD
outputs estimates of σ_j, the standard deviation of the error term.

EXPECTED
outputs estimates of expected values of continuous endogenous variables.

MARGINAL
outputs marginal effects.

MILLS
outputs estimates of inverse Mills ratios of censored or truncated continuous, binary discrete, and selection endogenous variables.

OUT=*SAS-data-set*
names the output data set.

PREDICTED
outputs estimates of predicted endogenous variables.

PROB
outputs estimates of probability of discrete endogenous variables taking the current observed responses.

PROBALL
outputs estimates of probability of discrete endogenous variables for all possible responses.

RESIDUAL
outputs estimates of residuals of continuous endogenous variables.

XBETA
outputs estimates of $\mathbf{x}'\boldsymbol{\beta}$.

TE1
outputs estimates of technical efficiency for each producer in the stochastic frontier model suggested by Battese and Coelli (1988).

TE2
outputs estimates of technical efficiency for each producer in the stochastic frontier model suggested by Jondrow et al. (1982).

PRIOR Statement

PRIOR _REGRESSORS | *parameter-list* ~ *distribution* **;**

The PRIOR statement specifies the prior distribution of the model parameters. You must specify a single parameter or a list of parameter, a tilde ~, and then a distribution with its parameters. Multiple PRIOR statements are allowed.

You can specify the following *distributions*:

NORMAL(MEAN=μ**, VAR=**σ^2**)**
specifies a normal distribution with parameters MEAN and VAR.

GAMMA(SHAPE=*a*, SCALE=*b*)
specifies a gamma distribution with parameters SHAPE and SCALE.

SQGAMMA(SHAPE=*a*, SCALE=*b*)
specifies a square root gamma distribution with parameters SHAPE and SCALE.

IGAMMA(SHAPE=*a*, SCALE=*b*)
specifies an inverse gamma distribution with parameters SHAPE and SCALE.

SQIGAMMA(SHAPE=*a*, SCALE=*b*)
specifies a square root inverse gamma distribution with parameters SHAPE and SCALE.

UNIFORM(MIN=*m*, MAX=*M*)
specifies a uniform distribution that is defined between MIN and MAX.

BETA(SHAPE1=*a*, SHAPE2=*b*, MIN=*m*, MAX=*M*)
specifies a beta distribution with parameters SHAPE1 and SHAPE2 and defined between MIN and MAX.

T(LOCATION=μ, DF=ν)
specifies a noncentral *t* distribution with DF degrees of freedom and location parameter equal to LOCATION.

See the section "Standard Distributions" on page 2024 for details about how to specify *distributions*.

You can specify the special keyword REGRESSORS to select all the parameters used in the linear regression component of the model.

RANDOM Statement (Experimental)

RANDOM *INTERCEPT* < */ options* > **;**

The RANDOM statement defines the random effects in the mixed model. Currently, you can specify only random intercepts as random effects; random coefficients are not allowed. If you have a panel data set, you can use the RANDOM statement to estimate random-effects models that include binomial probit, binomial logit, ordinal probit, ordinal logit, linear regression, Tobit, truncated regression, and stochastic frontier models.

You can specify only a single RANDOM statement, and if you specify a RANDOM statement, you can specify only one MODEL statement. The RANDOM statement is not supported if a BAYES statement is also specified.

You can abbreviate INTERCEPT as INT. You can specify the following *options* after a slash (/).

SUBJECT=*variable*
S=*variable*
determines the unique realizations of the random effects. In panel data, the *variable* identifies the cross-sectional units. For example, in panel data, *variable* might be household or country.

If you do not specify the SUBJECT= option, then *variable* is assumed to have a single realization; that is, there is no variation in the random effects. You should specify this option in order to have a true random-effects model.

The following statement illustrates the SUBJECT= option:

```
random int / subject=id;
```

METHOD=*method-options*

M=*method-options*

specifies the method of approximation to the integral that appears in the likelihood function. For more information about the integral and the integration methods, see the section "Random-Effects Models for Panel Data" on page 2008 and its subsections.

You can specify the following *method-options*:

HALTON < (*halton-options***) >**

HALT < (*halton-options***) >**

QMC < (*halton-options***) >**

requests a quasi–Monte Carlo integration method that uses the Halton sequence that is defined by the prime number 2. For information about how this series is generated, see the section "QMC Method Using the Halton Sequence" on page 2010.

You can specify the following *halton-options*:

HALTONDRAW=*value*

HDRAW=*value*

determines the number of elements that the Halton series has for each unique value of the subject variable. Therefore, the total number of elements in the Halton sequence is *value* times the number of unique values of the *variable* that is specified in the SUBJECT= option. For more information, see the subsection "QMC Method Using the Halton Sequence" on page 2010.

The default value of the HALTONDRAW= option is the number of unique values of the *variable* that is specified in the SUBJECT= option. For example, if you have a panel data set, the number of terms in the Halton sequence is the square of the number of cross-sections.

HALTONSTART=*value*

specifies the starting point of the Halton sequence, where *value* must be a positive integer. By default, HALTONSTART=11.

The following statement requests a Halton sequence that has 100 elements and does not discard any elements:

```
random int / subject=country method=halton(hdraw=100 haltonstart=1);
```

HERMITE < (QPOINTS=*value* **) >**

HERM < (QPOINTS=*value* **) >**

GAUSS < (QPOINTS=*value* **) >**

requests the Gauss-Hermite quadrature integration method. For more information, see the section "Approximation by Hermite Quadrature" on page 2010.

QPOINTS=*value* specifies the number of quadrature points to be used during evaluation of integral. By default QPOINTS=20.

The following statement illustrates this option:

```
random int / subject=states method=hermite(qpoints=4);
```

SIMULATION < (*simulation-options*) >

SIM < (*simulation-options*) >

requests simulation as the method of integration. For more information, see the section "Simulated Maximum Likelihood" on page 2009.

You can specify the following *simulation-options*:

NDRAW=*value*

specifies the number of draws for the simulation. You can also specify the number of draws in the NDRAW= option in the PROC QLIM statement. If you specify both, PROC QLIM uses the *value* in the RANDOM statement. If you do not specify any NDRAW= option, the default value is set to $N^{3/2}$, where N is the number of unique values of the subject variable. For example, for a panel data set, N is the number of cross-sections.

SEED=*value*

specifies the seed of the random draws, where *value* must be less than $2^{31} - 1$. You can also specify the seed in the SEED= option in the PROC QLIM statement. If you specify both, PROC QLIM uses the *value* in the RANDOM statement. If you do not specify a seed, or if you specify a value less than equal to zero, the seed is generated randomly.

The following statement illustrates this option:

```
random int / subject=id method=simulation(ndraw=1000 seed=12345);
```

By default, METHOD=HERMITE(QPOINTS=20).

RESTRICT Statement

RESTRICT *restriction1* < , *restriction2 . . .* > **;**

The RESTRICT statement is used to impose linear restrictions on the parameter estimates. Any number of RESTRICT statements can be specified, but the number of restrictions imposed is limited by the number of regressors.

Each *restriction* is written as an expression, followed by an equality operator (=) or an inequality operator (<, >, <=, >=), followed by a second expression:

expression operator expression

The *operator* can be =, <, >, <= , or >=. The operator and second expression are optional.

Restriction expressions can be composed of parameter names, multiplication ($*$), addition ($+$) and substitution ($-$) operators, and constants. Parameters named in restriction expressions must be among the parameters estimated by the model. Parameters associated with a regressor variable are referred to by the name of the corresponding regressor variable. The restriction expressions must be a linear function of the parameters.

The following is an example of the use of the RESTRICT statement:

```
proc qlim data=one;
   model y = x1-x10 / discrete;
   restrict x1*2 <= x2 + x3;
run;
```

The RESTRICT statement can also be used to impose cross-equation restrictions in multivariate models. The following RESTRICT statement imposes an equality restriction on coefficients of $x1$ in equation $y1$ and $x1$ in equation $y2$:

```
proc qlim data=one;
   model y1 = x1-x10;
   model y2 = x1-x4;
   endogenous y1 y2 ~ discrete;
   restrict y1.x1=y2.x1;
run;
```

The RESTRICT statement is not supported if a BAYES statement is also specified. In Bayesian analysis, the restrictions on parameters are usually introduced through the prior distribution.

TEST Statement

<'label':> **TEST** *<'string':> equation [,equation. . .] / options* **;**

The TEST statement performs Wald, Lagrange multiplier, and likelihood ratio tests of linear hypotheses about the regression parameters in the preceding MODEL statement. Each equation specifies a linear hypothesis to be tested. All hypotheses in one TEST statement are tested jointly. Variable names in the equations must correspond to regressors in the preceding MODEL statement, and each name represents the coefficient of the corresponding regressor. The keyword INTERCEPT refers to the coefficient of the intercept.

The following options can be specified in the TEST statement after the slash (/):

ALL
requests Wald, Lagrange multiplier, and likelihood ratio tests.

WALD
requests the Wald test.

LM
requests the Lagrange multiplier test.

LR
requests the likelihood ratio test.

The following illustrates the use of the TEST statement:

```
proc qlim;
   model y = x1 x2 x3;
   test x1 = 0, x2 * .5 + 2 * x3 = 0;
   test_int: test intercept = 0, x3 = 0;
run;
```

The first test investigates the joint hypothesis that

$$\beta_1 = 0$$

and

$$0.5\beta_2 + 2\beta_3 = 0$$

In case there is more than one MODEL statement in one QLIM procedure, then TEST statement is capable of testing cross-equation restrictions. Each parameter reference should be preceded by the name of the dependent variable of the particular model and the dot sign. For example,

```
proc qlim;
   model y1 = x1 x2 x3;
   model y2 = x3 x5 x6;
   test y1.x1 + y2.x6 = 1;
run;
```

This cross-equation test investigates the null hypothesis that

$$\beta_{1,1} + \beta_{2,3} = 1$$

in the system of equations

$$\begin{aligned} y_{1,i} &= \alpha_1 + \beta_{1,1}x_{1,i} + \beta_{1,2}x_{2,i} + \beta_{1,3}x_{3,i} \\ y_{2,i} &= \alpha_2 + \beta_{2,1}x_{3,i} + \beta_{2,2}x_{5,i} + \beta_{2,3}x_{6,i} \end{aligned}$$

Only linear equality restrictions and tests are permitted in PROC QLIM. Tests expressions can be composed only of algebraic operations involving the addition symbol (+), subtraction symbol (-), and multiplication symbol (*).

The TEST statement accepts labels that are reproduced in the printed output. TEST statement can be labeled in two ways. A TEST statement can be preceded by a label followed by a colon. Alternatively, the keyword TEST can be followed by a quoted string. If both are present, PROC QLIM uses the label preceding the colon. In the event no label is present, PROC QLIM automatically labels the tests.

You cannot specify both the TEST statement and the BAYES statement.

WEIGHT Statement

WEIGHT *variable* < / *option* > **;**

The WEIGHT statement specifies a variable to supply weighting values to use for each observation in estimating parameters. The log likelihood for each observation is multiplied by the corresponding weight variable value.

If the weight of an observation is nonpositive, that observation is not used in the estimation.

The following option can be added to the WEIGHT statement after a slash (*/*).

NONORMALIZE

specifies that the weights are required to be used as is. When this option is not specified, the weights are normalized so that they add up to the actual sample size. Weights w_i are normalized by multiplying them by $\frac{n}{\sum_{i=1}^{n} w_i}$, where n is the sample size.

Details: QLIM Procedure

Ordinal Discrete Choice Modeling

Binary Probit and Logit Model

The binary choice model is

$$y_i^* = \mathbf{x}_i'\boldsymbol{\beta} + \epsilon_i$$

where value of the latent dependent variable, y_i^*, is observed only as follows:

$$\begin{aligned} y_i &= 1 \quad \text{if } y_i^* > 0 \\ &= 0 \quad \text{otherwise} \end{aligned}$$

The disturbance, ϵ_i, of the probit model has standard normal distribution with the distribution function (CDF)

$$\Phi(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} \exp(-t^2/2)dt$$

The disturbance of the logit model has standard logistic distribution with the CDF

$$\Lambda(x) = \frac{\exp(x)}{1+\exp(x)} = \frac{1}{1+\exp(-x)}$$

The binary discrete choice model has the following probability that the event $\{y_i = 1\}$ occurs:

$$P(y_i = 1) = F(\mathbf{x}_i'\boldsymbol{\beta}) = \begin{cases} \Phi(\mathbf{x}_i'\boldsymbol{\beta}) & \text{(probit)} \\ \Lambda(\mathbf{x}_i'\boldsymbol{\beta}) & \text{(logit)} \end{cases}$$

The log-likelihood function is

$$\ell = \sum_{i=1}^{N} \left\{ y_i \log[F(\mathbf{x}_i'\boldsymbol{\beta})] + (1-y_i)\log[1-F(\mathbf{x}_i'\boldsymbol{\beta})] \right\}$$

where the CDF $F(x)$ is defined as $\Phi(x)$ for the probit model while $F(x) = \Lambda(x)$ for logit. The first order derivatives of the logit model are

$$\frac{\partial \ell}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{N} (y_i - \Lambda(\mathbf{x}_i'\boldsymbol{\beta}))\mathbf{x}_i$$

The probit model has more complicated derivatives

$$\frac{\partial \ell}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{N} \left\{ \frac{(2y_i-1)\phi\left[(2y_i-1)\mathbf{x}_i'\boldsymbol{\beta}\right]}{\Phi\left[(2y_i-1)\mathbf{x}_i'\boldsymbol{\beta}\right]} \right\} \mathbf{x}_i = \sum_{i=1}^{N} r_i \mathbf{x}_i$$

where

$$r_i = \frac{(2y_i-1)\phi\left[(2y_i-1)\mathbf{x}_i'\boldsymbol{\beta}\right]}{\Phi\left[(2y_i-1)\mathbf{x}_i'\boldsymbol{\beta}\right]}$$

Note that the logit maximum likelihood estimates are $\frac{\pi}{\sqrt{3}}$ times greater than probit maximum likelihood estimates, since the probit parameter estimates, $\boldsymbol{\beta}$, are standardized, and the error term with logistic distribution has a variance of $\frac{\pi^2}{3}$.

Ordinal Probit/Logit

When the dependent variable is observed in sequence with M categories, binary discrete choice modeling is not appropriate for data analysis. McKelvey and Zavoina (1975) proposed the ordinal (or ordered) probit model.

Consider the following regression equation:

$$y_i^* = \mathbf{x}_i'\boldsymbol{\beta} + \epsilon_i$$

where error disturbances, ϵ_i, have the distribution function F. The unobserved continuous random variable, y_i^*, is identified as M categories. Suppose there are $M+1$ real numbers, $\mu_0, \cdots, \mu_M$, where $\mu_0 = -\infty$, $\mu_1 = 0$, $\mu_M = \infty$, and $\mu_0 \le \mu_1 \le \cdots \le \mu_M$. Define

$$R_{i,j} = \mu_j - \mathbf{x}_i'\boldsymbol{\beta}$$

The probability that the unobserved dependent variable is contained in the jth category can be written as

$$P[\mu_{j-1} < y_i^* \le \mu_j] = F(R_{i,j}) - F(R_{i,j-1})$$

The log-likelihood function is

$$\ell = \sum_{i=1}^{N}\sum_{j=1}^{M} d_{ij} \log\left[F(R_{i,j}) - F(R_{i,j-1})\right]$$

where

$$d_{ij} = \begin{cases} 1 & \text{if} \mu_{j-1} < y_i \le \mu_j \\ 0 & \text{otherwise} \end{cases}$$

The first derivatives are written as

$$\frac{\partial \ell}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{N}\sum_{j=1}^{M} d_{ij}\left[\frac{f(R_{i,j-1}) - f(R_{i,j})}{F(R_{i,j}) - F(R_{i,j-1})}\mathbf{x}_i\right]$$

$$\frac{\partial \ell}{\partial \mu_k} = \sum_{i=1}^{N}\sum_{j=1}^{M} d_{ij}\left[\frac{\delta_{j,k} f(R_{i,j}) - \delta_{j-1,k} f(R_{i,j-1})}{F(R_{i,j}) - F(R_{i,j-1})}\right]$$

where $f(x) = \frac{dF(x)}{dx}$ and $\delta_{j,k} = 1$ if $j = k$, and $\delta_{j,k} = 0$ otherwise. When the ordinal probit is estimated, it is assumed that $F(R_{i,j}) = \Phi(R_{i,j})$. The ordinal logit model is estimated if $F(R_{i,j}) = \Lambda(R_{i,j})$. The first threshold parameter, μ_1, is estimated when the LIMIT1=VARYING option is specified. By default (LIMIT1=ZERO), so that $M-2$ threshold parameters $(\mu_2, \ldots, \mu_{M-1})$ are estimated.

The ordered probit models are analyzed by Aitchison and Silvey (1957), and Cox (1970) discussed ordered response data by using the logit model. They defined the probability that y_i^* belongs to jth category as

$$P[\mu_{j-1} < y_i \le \mu_j] = F(\mu_j + \mathbf{x}_i'\boldsymbol{\theta}) - F(\mu_{j-1} + \mathbf{x}_i'\boldsymbol{\theta})$$

where $\mu_0 = -\infty$ and $\mu_M = \infty$. Therefore, the ordered response model analyzed by Aitchison and Silvey can be estimated if the LIMIT1=VARYING option is specified. Note that $\boldsymbol{\theta} = -\boldsymbol{\beta}$.

Goodness-of-Fit Measures

The goodness-of-fit measures discussed in this section apply only to discrete dependent variable models.

McFadden (1974) suggested a likelihood ratio index that is analogous to the R^2 in the linear regression model:

$$R_M^2 = 1 - \frac{\ln L}{\ln L_0}$$

where L is the value of the maximum likelihood function and L_0 is the value of a likelihood function when regression coefficients except an intercept term are zero. It can be shown that L_0 can be written as

$$L_0 = \sum_{j=1}^{M} N_j \ln(\frac{N_j}{N})$$

where N_j is the number of responses in category j.

Estrella (1998) proposes the following requirements for a goodness-of-fit measure to be desirable in discrete choice modeling:

- The measure must take values in [0, 1], where 0 represents no fit and 1 corresponds to perfect fit.
- The measure should be directly related to the valid test statistic for significance of all slope coefficients.
- The derivative of the measure with respect to the test statistic should comply with corresponding derivatives in a linear regression.

Estrella's (1998) measure is written

$$R_{E1}^2 = 1 - \left(\frac{\ln L}{\ln L_0}\right)^{-\frac{2}{N}\ln L_0}$$

An alternative measure suggested by Estrella (1998) is

$$R_{E2}^2 = 1 - [(\ln L - K)/\ln L_0]^{-\frac{2}{N}\ln L_0}$$

where $\ln L_0$ is computed with null slope parameter values, N is the number observations used, and K represents the number of estimated parameters.

Other goodness-of-fit measures are summarized as follows:

$$R_{CU1}^2 = 1 - \left(\frac{L_0}{L}\right)^{\frac{2}{N}} \quad (\text{Cragg} - \text{Uhler1})$$

$$R_{CU2}^2 = \frac{1 - (L_0/L)^{\frac{2}{N}}}{1 - L_0^{\frac{2}{N}}} \quad (\text{Cragg} - \text{Uhler2})$$

$$R_A^2 = \frac{2(\ln L - \ln L_0)}{2(\ln L - \ln L_0) + N} \quad (\text{Aldrich} - \text{Nelson})$$

$$R_{VZ}^2 = R_A^2 \frac{2\ln L_0 - N}{2\ln L_0} \quad (\text{Veall} - \text{Zimmermann})$$

$$R_{MZ}^2 = \frac{\sum_{i=1}^{N}(\hat{y}_i - \bar{\hat{y}}_i)^2}{N + \sum_{i=1}^{N}(\hat{y}_i - \bar{\hat{y}}_i)^2} \quad (\text{McKelvey} - \text{Zavoina})$$

where $\hat{y}_i = \mathbf{x}_i'\hat{\boldsymbol{\beta}}$ and $\bar{\hat{y}}_i = \sum_{i=1}^{N} \hat{y}_i / N$.

Limited Dependent Variable Models

Censored Regression Models

When the dependent variable is censored, values in a certain range are all transformed to a single value. For example, the standard tobit model can be defined as

$$y_i^* = \mathbf{x}_i'\boldsymbol{\beta} + \epsilon_i$$

$$y_i = \begin{cases} y_i^* & \text{if } y_i^* > 0 \\ 0 & \text{if } y_i^* \le 0 \end{cases}$$

where $\epsilon_i \sim iidN(0, \sigma^2)$. The log-likelihood function of the standard censored regression model is

$$\ell = \sum_{i \in \{y_i = 0\}} \ln[1 - \Phi(\mathbf{x}_i'\boldsymbol{\beta}/\sigma)] + \sum_{i \in \{y_i > 0\}} \ln\left[\phi\left(\frac{y_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma}\right)/\sigma\right]$$

where $\Phi(\cdot)$ is the cumulative density function of the standard normal distribution and $\phi(\cdot)$ is the probability density function of the standard normal distribution.

The tobit model can be generalized to handle observation-by-observation censoring. The censored model on both of the lower and upper limits can be defined as

$$y_i = \begin{cases} R_i & \text{if } y_i^* \ge R_i \\ y_i^* & \text{if } L_i < y_i^* < R_i \\ L_i & \text{if } y_i^* \le L_i \end{cases}$$

The log-likelihood function can be written as

$$\begin{aligned} \ell \quad = \quad & \sum_{i \in \{L_i < y_i < R_i\}} \ln\left[\phi\left(\frac{y_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma}\right)/\sigma\right] + \sum_{i \in \{y_i = R_i\}} \ln\left[\Phi\left(-\frac{R_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma}\right)\right] + \\ & \sum_{i \in \{y_i = L_i\}} \ln\left[\Phi\left(\frac{L_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma}\right)\right] \end{aligned}$$

Log-likelihood functions of the lower- or upper-limit censored model are easily derived from the two-limit censored model. The log-likelihood function of the lower-limit censored model is

$$\ell = \sum_{i \in \{y_i > L_i\}} \ln\left[\phi\left(\frac{y_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma}\right)/\sigma\right] + \sum_{i \in \{y_i = L_i\}} \ln\left[\Phi\left(\frac{L_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma}\right)\right]$$

The log-likelihood function of the upper-limit censored model is

$$\ell = \sum_{i \in \{y_i < R_i\}} \ln\left[\phi\left(\frac{y_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma}\right)/\sigma\right] + \sum_{i \in \{y_i = R_i\}} \ln\left[1 - \Phi\left(\frac{R_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma}\right)\right]$$

Types of Tobit Models

Amemiya (1984) classified Tobit models into five types based on characteristics of the likelihood function. For notational convenience, let P denote a distribution or density function, y^*_{ji} is assumed to be normally distributed with mean $\mathbf{x}'_{ji}\boldsymbol{\beta}_j$ and variance σ^2_j.

Type 1 Tobit

The Type 1 Tobit model was already discussed in the preceding section.

$$\begin{aligned} y^*_{1i} &= \mathbf{x}'_{1i}\boldsymbol{\beta}_1 + u_{1i} \\ y_{1i} &= y^*_{1i} \quad \text{if } y^*_{1i} > 0 \\ &= 0 \quad \text{if } y^*_{1i} \leq 0 \end{aligned}$$

The likelihood function is characterized as $P(y_1 < 0)P(y_1)$.

Type 2 Tobit

The Type 2 Tobit model is defined as

$$\begin{aligned} y^*_{1i} &= \mathbf{x}'_{1i}\boldsymbol{\beta}_1 + u_{1i} \\ y^*_{2i} &= \mathbf{x}'_{2i}\boldsymbol{\beta}_2 + u_{2i} \\ y_{1i} &= 1 \quad \text{if } y^*_{1i} > 0 \\ &= 0 \quad \text{if } y^*_{1i} \leq 0 \\ y_{2i} &= y^*_{2i} \quad \text{if } y^*_{1i} > 0 \\ &= 0 \quad \text{if } y^*_{1i} \leq 0 \end{aligned}$$

where $(u_{1i}, u_{2i}) \sim N(0, \Sigma)$. The likelihood function is described as $P(y_1 < 0)P(y_1 > 0, y_2)$.

Type 3 Tobit

The Type 3 Tobit model is different from the Type 2 Tobit in that y^*_{1i} of the Type 3 Tobit is observed when $y^*_{1i} > 0$.

$$\begin{aligned} y^*_{1i} &= \mathbf{x}'_{1i}\boldsymbol{\beta}_1 + u_{1i} \\ y^*_{2i} &= \mathbf{x}'_{2i}\boldsymbol{\beta}_2 + u_{2i} \\ y_{1i} &= y^*_{1i} \quad \text{if } y^*_{1i} > 0 \\ &= 0 \quad \text{if } y^*_{1i} \leq 0 \\ y_{2i} &= y^*_{2i} \quad \text{if } y^*_{1i} > 0 \\ &= 0 \quad \text{if } y^*_{1i} \leq 0 \end{aligned}$$

where $(u_{1i}, u_{2i})' \sim iidN(0, \Sigma)$.

The likelihood function is characterized as $P(y_1 < 0)P(y_1, y_2)$.

Type 4 Tobit

The Type 4 Tobit model consists of three equations:

$$\begin{aligned}
y^*_{1i} &= \mathbf{x}'_{1i}\boldsymbol{\beta}_1 + u_{1i} \\
y^*_{2i} &= \mathbf{x}'_{2i}\boldsymbol{\beta}_2 + u_{2i} \\
y^*_{3i} &= \mathbf{x}'_{3i}\boldsymbol{\beta}_3 + u_{3i} \\
y_{1i} &= y^*_{1i} \quad \text{if } y^*_{1i} > 0 \\
&= 0 \quad \text{if } y^*_{1i} \le 0 \\
y_{2i} &= y^*_{2i} \quad \text{if } y^*_{1i} > 0 \\
&= 0 \quad \text{if } y^*_{1i} \le 0 \\
y_{3i} &= y^*_{3i} \quad \text{if } y^*_{1i} \le 0 \\
&= 0 \quad \text{if } y^*_{1i} > 0
\end{aligned}$$

where $(u_{1i}, u_{2i}, u_{3i})' \sim iidN(0, \Sigma)$. The likelihood function of the Type 4 Tobit model is characterized as $P(y_1 < 0, y_3)P(y_1, y_2)$.

Type 5 Tobit

The Type 5 Tobit model is defined as follows:

$$\begin{aligned}
y^*_{1i} &= \mathbf{x}'_{1i}\boldsymbol{\beta}_1 + u_{1i} \\
y^*_{2i} &= \mathbf{x}'_{2i}\boldsymbol{\beta}_2 + u_{2i} \\
y^*_{3i} &= \mathbf{x}'_{3i}\boldsymbol{\beta}_3 + u_{3i} \\
y_{1i} &= 1 \quad \text{if } y^*_{1i} > 0 \\
&= 0 \quad \text{if } y^*_{1i} \le 0 \\
y_{2i} &= y^*_{2i} \quad \text{if } y^*_{1i} > 0 \\
&= 0 \quad \text{if } y^*_{1i} \le 0 \\
y_{3i} &= y^*_{3i} \quad \text{if } y^*_{1i} \le 0 \\
&= 0 \quad \text{if } y^*_{1i} > 0
\end{aligned}$$

where $(u_{1i}, u_{2i}, u_{3i})'$ are from *iid* trivariate normal distribution. The likelihood function of the Type 5 Tobit model is characterized as $P(y_1 < 0, y_3)P(y_1 > 0, y_2)$.

Code examples for these models can be found in "Example 29.6: Types of Tobit Models" on page 2045.

Truncated Regression Models

In a truncated model, the observed sample is a subset of the population where the dependent variable falls in a certain range. For example, when neither a dependent variable nor exogenous variables are observed for $y^*_i < 0$, the truncated regression model can be specified.

$$\ell = \sum_{i \in \{y_i \ge 0\}} \left\{ -\ln \Phi(\mathbf{x}'_i\boldsymbol{\beta}/\sigma) + \ln \left[\frac{\phi((y_i - \mathbf{x}'_i\boldsymbol{\beta})/\sigma)}{\sigma} \right] \right\}$$

Two-limit truncation model is defined as

$$y_i = y^*_i \text{if } L_i \le y^*_i \le R_i$$

The log-likelihood function of the two-limit truncated regression model is

$$\ell = \sum_{i=1}^{N} \left\{ \ln \left[\phi(\frac{y_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma})/\sigma \right] - \ln \left[\Phi(\frac{R_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma}) - \Phi(\frac{L_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma}) \right] \right\}$$

The log-likelihood functions of the lower- and upper-limit truncation model are

$$\ell = \sum_{i=1}^{N} \left\{ \ln \left[\phi(\frac{y_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma})/\sigma \right] - \ln \left[1 - \Phi(\frac{L_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma}) \right] \right\} \quad \text{(lower)}$$

$$\ell = \sum_{i=1}^{N} \left\{ \ln \left[\phi(\frac{y_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma})/\sigma \right] - \ln \left[\Phi(\frac{R_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma}) \right] \right\} \quad \text{(upper)}$$

Stochastic Frontier Production and Cost Models

Stochastic frontier production models were first developed by Aigner, Lovell, and Schmidt (1977); Meeusen and van den Broeck (1977). Specification of these models allow for random shocks of the production or cost but also include a term for technological or cost inefficiency. Assuming that the production function takes a log-linear Cobb-Douglas form, the stochastic frontier production model can be written as

$$ln(y_i) = \beta_0 + \sum_n \boldsymbol{\beta}_n \ln(x_{ni}) + \epsilon_i$$

where $\epsilon_i = v_i - u_i$. The v_i term represents the stochastic error component and u_i is the nonnegative, technology inefficiency error component. The v_i error component is assumed to be distributed iid normal and independently from u_i. Given that $u_i > 0$, the error term, ϵ_i, is negatively skewed and represents technology inefficiency. For the stochastic frontier cost model, $\epsilon_i = v_i + u_i$. The v_i term represents the stochastic error component and u_i is the nonnegative, cost inefficiency error component. Given that $u_i > 0$, the error term, ϵ_i, is positively skewed and represents cost inefficiency. PROC QLIM models the u_i error component as a half normal, exponential, or truncated normal distribution.

The Normal-Half Normal Model

In case of the normal-half normal model, v_i is iid $N(0, \sigma_v^2)$, u_i is iid $N^+(0, \sigma_u^2)$ with v_i and u_i independent of each other. Given the independence of error terms, the joint density of v and u can be written as

$$f(u, v) = \frac{2}{2\pi\sigma_u\sigma_v} \exp\left\{ -\frac{u^2}{2\sigma_u^2} - \frac{v^2}{2\sigma_v^2} \right\}$$

Substituting $v = \epsilon + u$ into the preceding equation gives

$$f(u, \epsilon) = \frac{2}{2\pi\sigma_u\sigma_v} \exp\left\{ -\frac{u^2}{2\sigma_u^2} - \frac{(\epsilon + u)^2}{2\sigma_v^2} \right\}$$

Integrating u out to obtain the marginal density function of ϵ results in the following form:

$$\begin{aligned} f(\epsilon) &= \int_0^\infty f(u,\epsilon)du \\ &= \frac{2}{\sqrt{2\pi}\sigma}\left[1-\Phi\left(\frac{\epsilon\lambda}{\sigma}\right)\right]\exp\left\{-\frac{\epsilon^2}{2\sigma^2}\right\} \\ &= \frac{2}{\sigma}\phi\left(\frac{\epsilon}{\sigma}\right)\Phi\left(-\frac{\epsilon\lambda}{\sigma}\right) \end{aligned}$$

where $\lambda = \sigma_u/\sigma_v$ and $\sigma = \sqrt{\sigma_u^2+\sigma_v^2}$.

In the case of a stochastic frontier cost model, $v = \epsilon - u$ and

$$f(\epsilon) = \frac{2}{\sigma}\phi\left(\frac{\epsilon}{\sigma}\right)\Phi\left(\frac{\epsilon\lambda}{\sigma}\right)$$

The log-likelihood function for the production model with N producers is written as

$$\ln L = constant - N\ln\sigma + \sum_i \ln\Phi\left(-\frac{\epsilon_i\lambda}{\sigma}\right) - \frac{1}{2\sigma^2}\sum_i \epsilon_i^2$$

The Normal-Exponential Model

Under the normal-exponential model, v_i is iid $N(0,\sigma_v^2)$ and u_i is iid exponential with scale parameter σ_u. Given the independence of error term components u_i and v_i, the joint density of v and u can be written as

$$f(u,v) = \frac{1}{\sqrt{2\pi}\sigma_u\sigma_v}\exp\left\{-\frac{u}{\sigma_u}-\frac{v^2}{2\sigma_v^2}\right\}$$

The marginal density function of ϵ for the production function is

$$\begin{aligned} f(\epsilon) &= \int_0^\infty f(u,\epsilon)du \\ &= \left(\frac{1}{\sigma_u}\right)\Phi\left(-\frac{\epsilon}{\sigma_v}-\frac{\sigma_v}{\sigma_u}\right)\exp\left\{\frac{\epsilon}{\sigma_u}+\frac{\sigma_v^2}{2\sigma_u^2}\right\} \end{aligned}$$

and the marginal density function for the cost function is equal to

$$f(\epsilon) = \left(\frac{1}{\sigma_u}\right)\Phi\left(\frac{\epsilon}{\sigma_v}-\frac{\sigma_v}{\sigma_u}\right)\exp\left\{-\frac{\epsilon}{\sigma_u}+\frac{\sigma_v^2}{2\sigma_u^2}\right\}$$

The log-likelihood function for the normal-exponential production model with N producers is

$$\ln L = constant - N\ln\sigma_u + N\left(\frac{\sigma_v^2}{2\sigma_u^2}\right) + \sum_i\frac{\epsilon_i}{\sigma_u} + \sum_i \ln\Phi\left(-\frac{\epsilon_i}{\sigma_v}-\frac{\sigma_v}{\sigma_u}\right)$$

The Normal-Truncated Normal Model

The normal-truncated normal model is a generalization of the normal-half normal model by allowing the mean of u_i to differ from zero. Under the normal-truncated normal model, the error term component v_i is iid $N(0, \sigma_v^2)$ and u_i is iid $N^+(\mu, \sigma_u^2)$. The joint density of v_i and u_i can be written as

$$f(u,v) = \frac{1}{2\pi\sigma_u\sigma_v\Phi\left(\mu/\sigma_u\right)} \exp\left\{-\frac{(u-\mu)^2}{2\sigma_u^2} - \frac{v^2}{2\sigma_v^2}\right\}$$

The marginal density function of ϵ for the production function is

$$\begin{aligned} f(\epsilon) &= \int_0^\infty f(u,\epsilon)du \\ &= \frac{1}{\sqrt{2\pi}\sigma\Phi\left(\mu/\sigma_u\right)}\Phi\left(\frac{\mu}{\sigma\lambda} - \frac{\epsilon\lambda}{\sigma}\right)\exp\left\{-\frac{(\epsilon+\mu)^2}{2\sigma^2}\right\} \\ &= \frac{1}{\sigma}\phi\left(\frac{\epsilon+\mu}{\sigma}\right)\Phi\left(\frac{\mu}{\sigma\lambda} - \frac{\epsilon\lambda}{\sigma}\right)\left[\Phi\left(\frac{\mu}{\sigma_u}\right)\right]^{-1} \end{aligned}$$

and the marginal density function for the cost function is

$$f(\epsilon) = \frac{1}{\sigma}\phi\left(\frac{\epsilon-\mu}{\sigma}\right)\Phi\left(\frac{\mu}{\sigma\lambda} + \frac{\epsilon\lambda}{\sigma}\right)\left[\Phi\left(\frac{\mu}{\sigma_u}\right)\right]^{-1}$$

The log-likelihood function for the normal-truncated normal production model with N producers is

$$\begin{aligned} \ln L &= constant - N\ln\sigma - N\ln\Phi\left(\frac{\mu}{\sigma_u}\right) + \sum_i \ln\Phi\left(\frac{\mu}{\sigma\lambda} - \frac{\epsilon_i\lambda}{\sigma}\right) \\ &\quad -\frac{1}{2}\sum_i\left(\frac{\epsilon_i+\mu}{\sigma}\right)^2 \end{aligned}$$

For more detail on normal-half normal, normal-exponential, and normal-truncated models, see Kumbhakar and Lovell (2000); Coelli, Prasada Rao, and Battese (1998).

Heteroscedasticity and Box-Cox Transformation

Heteroscedasticity

If the variance of regression disturbance, (ϵ_i), is heteroscedastic, the variance can be specified as a function of variables

$$E(\epsilon_i^2) = \sigma_i^2 = f(\mathbf{z}_i'\boldsymbol{\gamma})$$

The following table shows various functional forms of heteroscedasticity and the corresponding options to request each model.

No.	Model	Options
1	$f(\mathbf{z}_i'\boldsymbol{\gamma}) = \sigma^2(1+\exp(\mathbf{z}_i'\gamma))$	LINK=EXP (default)
2	$f(\mathbf{z}_i'\boldsymbol{\gamma}) = \sigma^2\exp(\mathbf{z}_i'\gamma)$	LINK=EXP NOCONST
3	$f(\mathbf{z}_i'\boldsymbol{\gamma}) = \sigma^2(1+\sum_{l=1}^{L}\gamma_l z_{li})$	LINK=LINEAR
4	$f(\mathbf{z}_i'\boldsymbol{\gamma}) = \sigma^2(1+(\sum_{l=1}^{L}\gamma_l z_{li})^2)$	LINK=LINEAR SQUARE

For discrete choice models, σ^2 is normalized ($\sigma^2 = 1$) since this parameter is not identified. Note that in models 3 and 5, it may be possible that variances of some observations are negative. Although the QLIM procedure assigns a large penalty to move the optimization away from such region, it is possible that the optimization cannot improve the objective function value and gets locked in the region. Signs of such outcome include extremely small likelihood values or missing standard errors in the estimates. In models 2 and 6, variances are guaranteed to be greater or equal to zero, but it may be possible that variances of some observations are very close to zero. In these scenarios, standard errors may be missing. Models 1 and 4 do not have such problems. Variances in these models are always positive and never close to zero.

The heteroscedastic regression model is estimated using the following log-likelihood function:

$$\ell = -\frac{N}{2}\ln(2\pi) - \sum_{i=1}^{N}\frac{1}{2}\ln(\sigma_i^2) - \frac{1}{2}\sum_{i=1}^{N}(\frac{e_i}{\sigma_i})^2$$

where $e_i = y_i - \mathbf{x}_i'\boldsymbol{\beta}$.

Box-Cox Modeling

The Box-Cox transformation on x is defined as

$$x^{(\lambda)} = \begin{cases} \frac{x^{\lambda}-1}{\lambda} & \text{if}\lambda \neq 0 \\ \ln(x) & \text{if}\lambda = 0 \end{cases}$$

The Box-Cox regression model with heteroscedasticity is written as

$$\begin{aligned} y_i^{(\lambda_0)} &= \beta_0 + \sum_{k=1}^{K}\beta_k x_{ki}^{(\lambda_k)} + \epsilon_i \\ &= \mu_i + \epsilon_i \end{aligned}$$

where $\epsilon_i \sim N(0,\sigma_i^2)$ and transformed variables must be positive. In practice, too many transformation parameters cause numerical problems in model fitting. It is common to have the same Box-Cox transformation performed on all the variables — that is, $\lambda_0 = \lambda_1 = \cdots = \lambda_K$. It is required for the magnitude of transformed variables to be in the tolerable range if the corresponding transformation parameters are $|\lambda| > 1$.

The log-likelihood function of the Box-Cox regression model is written as

$$\ell = -\frac{N}{2}\ln(2\pi) - \sum_{i=1}^{N}\ln(\sigma_i) - \frac{1}{2\sigma_i^2}\sum_{i=1}^{N}e_i^2 + (\lambda_0 - 1)\sum_{i=1}^{N}\ln(y_i)$$

where $e_i = y_i^{(\lambda_0)} - \mu_i$.

When the dependent variable is discrete, censored, or truncated, the Box-Cox transformation can be applied only to explanatory variables.

Bivariate Limited Dependent Variable Modeling

The generic form of a bivariate limited dependent variable model is

$$\begin{aligned} y_{1i}^* &= \mathbf{x}_{1i}'\boldsymbol{\beta}_1 + \epsilon_{1i} \\ y_{2i}^* &= \mathbf{x}_{2i}'\boldsymbol{\beta}_2 + \epsilon_{2i} \end{aligned}$$

where the disturbances, ϵ_{1i} and ϵ_{2i}, have joint normal distribution with zero mean, standard deviations σ_1 and σ_2, and correlation of ρ. y_1^* and y_2^* are latent variables. The dependent variables y_1 and y_2 are observed if the latent variables y_1^* and y_2^* fall in certain ranges:

$$\begin{aligned} y_1 &= y_{1i} \text{ if } y_{1i}^* \in D_1(y_{1i}) \\ y_2 &= y_{2i} \text{ if } y_{2i}^* \in D_2(y_{2i}) \end{aligned}$$

D is a transformation from (y_{1i}^*, y_{2i}^*) to (y_{1i}, y_{2i}). For example, if $y1$ and $y2$ are censored variables with lower bound 0, then

$$\begin{aligned} y_1 &= y_{1i} \text{ if } y_{1i}^* > 0, \quad y_1 = 0 \text{ if } y_{1i}^* \le 0 \\ y_2 &= y_{2i} \text{ if } y_{2i}^* > 0, \quad y_2 = 0 \text{ if } y_{2i}^* \le 0 \end{aligned}$$

There are three cases for the log likelihood of (y_{1i}, y_{2i}). The first case is that $y_{1i} = y_{1i}^*$ and $y_{2i} = y_{2i}^*$. That is, this observation is mapped to one point in the space of latent variables. The log likelihood is computed from a bivariate normal density,

$$\ell_i = \ln\left[\phi_2\left(\frac{y_1 - \mathbf{x_1}'\boldsymbol{\beta}_1}{\sigma_1}, \frac{y_2 - \mathbf{x_2}'\boldsymbol{\beta}_2}{\sigma_2}, \rho\right)\right] - \ln\sigma_1 - \ln\sigma_2$$

where $\phi_2(u, v, \rho)$ is the density function for standardized bivariate normal distribution with correlation ρ,

$$\phi_2(u, v, \rho) = \frac{e^{-(1/2)(u^2+v^2-2\rho uv)/(1-\rho^2)}}{2\pi(1-\rho^2)^{1/2}}$$

The second case is that one observed dependent variable is mapped to a point of its latent variable and the other dependent variable is mapped to a segment in the space of its latent variable. For example, in the bivariate censored model specified, if observed $y1 > 0$ and $y2 = 0$, then $y1^* = y1$ and $y2^* \in (-\infty, 0]$. In general, the log likelihood for one observation can be written as follows (the subscript i is dropped for simplicity): If one set is a single point and the other set is a range, without loss of generality, let $D_1(y_1) = \{y_1\}$ and $D_2(y_2) = [L_2, R_2]$,

$$\begin{aligned} \ell_i &= \ln\left[\phi\left(\frac{y_1 - \mathbf{x_1}'\boldsymbol{\beta}_1}{\sigma_1}\right)\right] - \ln\sigma_1 \\ &+ \ln\left[\Phi\left(\frac{R_2 - \mathbf{x_2}'\boldsymbol{\beta}_2 - \rho\frac{y_1 - \mathbf{x_1}'\boldsymbol{\beta}_1}{\sigma_1}}{\sigma_2}\right) - \Phi\left(\frac{L_2 - \mathbf{x_2}'\boldsymbol{\beta}_2 - \rho\frac{y_1 - \mathbf{x_1}'\boldsymbol{\beta}_1}{\sigma_1}}{\sigma_2}\right)\right] \end{aligned}$$

where ϕ and Φ are the density function and the cumulative probability function for standardized univariate normal distribution.

The third case is that both dependent variables are mapped to segments in the space of latent variables. For example, in the bivariate censored model specified, if observed $y1 = 0$ and $y2 = 0$, then $y1^* \in (-\infty, 0]$ and $y2^* \in (-\infty, 0]$. In general, if $D_1(y_1) = [L_1, R_1]$ and $D_2(y_2) = [L_2, R_2]$, the log likelihood is

$$\ell_i = \ln \int_{\frac{L_1 - \mathbf{x_1}'\boldsymbol{\beta}_1}{\sigma_1}}^{\frac{R_1 - \mathbf{x_1}'\boldsymbol{\beta}_1}{\sigma_1}} \int_{\frac{L_2 - \mathbf{x_2}'\boldsymbol{\beta}_2}{\sigma_2}}^{\frac{R_2 - \mathbf{x_2}'\boldsymbol{\beta}_2}{\sigma_2}} \phi_2(u, v, \rho)\, du\, dv$$

Selection Models

In sample selection models, one or several dependent variables are observed when another variable takes certain values. For example, the standard Heckman selection model can be defined as

$$z_i^* = \mathbf{w}_i'\boldsymbol{\gamma} + u_i$$

$$z_i = \begin{cases} 1 & \text{if } z_i^* > 0 \\ 0 & \text{if } z_i^* \le 0 \end{cases}$$

$$y_i = \mathbf{x}_i'\boldsymbol{\beta} + \epsilon_i \quad \text{if } z_i = 1$$

where u_i and ϵ_i are jointly normal with 0 mean, standard deviations of 1 and σ, respectively, and correlation of ρ. Selection is based on the variable z, and y is observed when z has a value of 1. Least squares regression that uses the observed data of y produces inconsistent estimates of $\boldsymbol{\beta}$. The maximum likelihood method is used to estimate selection models. It is also possible to estimate these models by using Heckman's method, which is more computationally efficient. But it can be shown that the resulting estimates, although consistent, are not asymptotically efficient under a normality assumption. Moreover, this method often violates the constraint on the correlation coefficient $|\rho| \le 1$.

The log-likelihood function of the Heckman selection model is written as

$$\begin{aligned} \ell &= \sum_{i \in \{z_i = 0\}} \ln[1 - \Phi(\mathbf{w}_i'\boldsymbol{\gamma})] \\ &+ \sum_{i \in \{z_i = 1\}} \left\{ \ln \phi\left(\frac{y_i - \mathbf{x_i}'\boldsymbol{\beta}}{\sigma}\right) - \ln \sigma + \ln \Phi\left(\frac{\mathbf{w}_i'\boldsymbol{\gamma} + \rho \frac{y_i - \mathbf{x_i}'\boldsymbol{\beta}}{\sigma}}{\sqrt{1 - \rho^2}}\right) \right\} \end{aligned}$$

The selection can be based on only one variable, but the selection can lead to several variables. For example, selection is based on the variable z in the following switching regression model:

$$z_i^* = \mathbf{w}_i'\boldsymbol{\gamma} + u_i$$

$$z_i = \begin{cases} 1 & \text{if } z_i^* > 0 \\ 0 & \text{if } z_i^* \le 0 \end{cases}$$

$$\begin{aligned} y_{1i} &= \mathbf{x}_{1i}'\boldsymbol{\beta}_1 + \epsilon_{1i} \quad \text{if } z_i = 0 \\ y_{2i} &= \mathbf{x}_{2i}'\boldsymbol{\beta}_2 + \epsilon_{2i} \quad \text{if } z_i = 1 \end{aligned}$$

If $z = 0$, then y_1 is observed. If $z = 1$, then y_2 is observed. Because $y1$ and $y2$ are never observed at the same time, the correlation between y_1 and y_2 cannot be estimated. Only the correlation between z and y_1 and the correlation between z and y_2 can be estimated. This estimation uses the maximum likelihood method.

A brief example of the SAS statements for this model can be found in "Example 29.4: Sample Selection Model" on page 2042.

The Heckman selection model can include censoring or truncation. For a brief example of the SAS statements for these models see "Example 29.5: Sample Selection Model with Truncation and Censoring" on page 2043. The following example shows a variable y_i that is censored from below at zero:

$$z_i^* = \mathbf{w}_i'\boldsymbol{\gamma} + u_i$$

$$z_i = \begin{cases} 1 & \text{if } z_i^* > 0 \\ 0 & \text{if } z_i^* \le 0 \end{cases}$$

$$y_i^* = \mathbf{x}_i'\boldsymbol{\beta} + \epsilon_i \quad \text{if } z_i = 1$$

$$y_i = \begin{cases} y_i^* & \text{if } y_i^* > 0 \\ 0 & \text{if } y_i^* \le 0 \end{cases}$$

In this case, the log-likelihood function of the Heckman selection model needs to be modified as follows to include the censored region:

$$\begin{aligned} \ell &= \sum_{\{i\,|\,z_i=0\}} \ln[1-\Phi(\mathbf{w}_i'\boldsymbol{\gamma})] \\ &+ \sum_{\{i\,|\,z_i=1, y_i=y_i^*\}} \left\{ \ln\left[\phi\left(\frac{y_i - \mathbf{x_i}'\boldsymbol{\beta}}{\sigma}\right)\right] - \ln\sigma + \ln\left[\Phi\left(\frac{\mathbf{w}_i'\boldsymbol{\gamma} + \rho\frac{y_i - \mathbf{x_i}'\boldsymbol{\beta}}{\sigma}}{\sqrt{1-\rho^2}}\right)\right]\right\} \\ &+ \sum_{\{i\,|\,z_i=1, y_i=0\}} \ln \int_{-\infty}^{\frac{-\mathbf{x_i}'\boldsymbol{\beta}}{\sigma}} \int_{-\mathbf{w_i}'\gamma}^{\infty} \phi_2(u,v,\rho)\,du\,dv \end{aligned}$$

In case y_i is truncated from below at 0 instead of censored, the likelihood function can be written as

$$\begin{aligned} \ell &= \sum_{\{i\,|\,z_i=0\}} \ln[1-\Phi(\mathbf{w}_i'\boldsymbol{\gamma})] \\ &+ \sum_{\{i\,|\,z_i=1\}} \left\{ \ln\left[\phi\left(\frac{y_i - \mathbf{x_i}'\boldsymbol{\beta}}{\sigma}\right)\right] - \ln\sigma + \ln\left[\Phi\left(\frac{\mathbf{w}_i'\boldsymbol{\gamma} + \rho\frac{y_i - \mathbf{x_i}'\boldsymbol{\beta}}{\sigma}}{\sqrt{1-\rho^2}}\right)\right] - \ln\left[\Phi(\mathbf{x}_i'\boldsymbol{\beta}/\sigma)\right]\right\} \end{aligned}$$

Heckman's Two-Step Selection Method

Sample selection bias arises from nonrandom selection of the sample from the population. A classic example is using a sample of market wages for working women to estimate female labor supply function. This sample is nonrandom because it includes only the wages of women whose market wage exceeds their home wage at zero hours of work.

A simple selection model can be written as the latent model

$$z_i^* = \mathbf{w}_i'\boldsymbol{\gamma} + u_i$$

$$z_i = \begin{cases} 1 & \text{if } z_i^* > 0 \\ 0 & \text{if } z_i^* \le 0 \end{cases}$$

$$y_i = \mathbf{x}_i'\boldsymbol{\beta} + \epsilon_i \quad \text{if } z_i = 1$$

where u_i and ϵ_i are jointly normal with 0 mean, standard deviations of 1 and σ, respectively, and correlation of ρ. The dependent variable y_i (wage) is observed if the latent variable z_i^* (the difference between market wage and reservation wage) is positive or if the indicator variable z_i (labor force participation) is 1.

The model of interest that applies to the observations in the selected sample can be written as

$$E(y_i|\mathbf{x}_i, z_i = 1) = \mathbf{x}_i'\boldsymbol{\beta} + \rho\sigma\lambda(\mathbf{w}_i'\boldsymbol{\gamma})$$

where $\lambda(\mathbf{w}_i'\boldsymbol{\gamma}) = \phi(\mathbf{w}_i'\boldsymbol{\gamma})/\Phi(\mathbf{w}_i'\boldsymbol{\gamma})$. Hence, the following regression equation is valid for the observations for which $z_i = 1$:

$$y_i = \mathbf{x}_i'\boldsymbol{\beta} + \rho\sigma\lambda(\mathbf{w}_i'\boldsymbol{\gamma}) + v_i$$

Therefore, estimates of $\boldsymbol{\beta}$ that are obtained from the OLS regression of y on $\mathbf{x}$ by using the selected sample (that is, the sample for which $z_i = 1$) suffer from omitted variable bias if selection bias is really the case. Although maximum likelihood estimation of $\boldsymbol{\beta}$ is consistent and efficient, Heckman's two-step method is more frequently used. Heckman's two-step method can be requested by specifying the HECKIT option of the QLIM statement.

Heckman's two-step method is as follows:

1. Obtain $\hat{\boldsymbol{\gamma}}$, the estimate of the parameters of the probability that $z_i^* > 0$, by using regressors $\mathbf{w}_i$ and the binary dependent variable z_i by probit analysis for the full sample. Compute $\hat{\lambda}_i = \lambda(\mathbf{w}_i'\hat{\boldsymbol{\gamma}})$.
2. Obtain $\hat{\boldsymbol{\beta}}$ and $\hat{\beta}_\lambda$, the estimates of $\boldsymbol{\beta}$ and $\rho\sigma$, by least squares regression of y_i on $\mathbf{x}_i$ and $\hat{\lambda}_i$ by using observations on the selected subsample.

The standard least squares estimators of the population variance σ^2 and the variances of the estimated coefficients are incorrect. To test hypotheses, the correct ones need to be calculated. An estimator of σ^2 is

$$\hat{\sigma}^2 = \frac{1}{N_1}\sum_{i=1}^{N_1} e_i^2 + \hat{\beta}_\lambda^2 \frac{1}{N_1}\sum_{i=1}^{N_1} \hat{\delta}_i$$

where N_1 is the selected subsample size, e_i is the residual for the ith observation obtained from step 2, and $\hat{\delta}_i = \hat{\lambda}_i^2 + \hat{\lambda}_i \mathbf{w}_i'\hat{\boldsymbol{\gamma}}$. Let $\mathbf{X}_*$ be an $N_1 \times (K+1)$ matrix with ith row $[\mathbf{x}_i' \;\; \lambda_i]$, and define $\mathbf{W}$ similarly with ith row $\mathbf{w}_i'$. Then the estimator of the asymptotic covariance of $[\hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\beta}}_\lambda]$ is

$$\text{EstAsyVar}[\hat{\boldsymbol{\beta}}, \hat{\beta}_\lambda] = \hat{\sigma}^2[\mathbf{X}_*'\mathbf{X}_*]^{-1}[\mathbf{X}_*'(\mathbf{I} - \hat{\rho}^2\hat{\boldsymbol{\Delta}})\mathbf{X}_* + \mathbf{Q}][\mathbf{X}_*'\mathbf{X}_*]^{-1}$$

where $\hat{\rho}^2 = \hat{\beta}_\lambda^2/\hat{\sigma}^2$, $\hat{\boldsymbol{\Delta}} = \text{diag}(\hat{\delta}_i)$, and

$$\mathbf{Q} = \hat{\sigma}^2(\mathbf{X}_*'\hat{\boldsymbol{\Delta}}\mathbf{W})\text{Est.Asy.Var}(\hat{\boldsymbol{\gamma}})(\mathbf{W}'\hat{\boldsymbol{\Delta}}\mathbf{X}_*)$$

where Est.Asy.Var($\hat{\boldsymbol{\gamma}}$) is the estimator of the asymptotic covariance of the probit coefficients that are obtained in step 1. When you specify the HECKIT option, PROC QLIM uses a numerical estimated asymptotic variance.

When the HECKIT option is specified, PROC QLIM reports the corrected standard errors for $[\hat{\boldsymbol{\beta}}, \hat{\beta}_\lambda]$ automatically. However, if you need the conventional OLS standard errors, you can specify the HECKIT(UNCORRECTED) option.

In the selected regression model, when the coefficient of $\lambda(\mathbf{w}_i'\boldsymbol{\gamma})$ is 0, you do not need Heckman's two-step estimation method; a simple regression of y on $\mathbf{x}$ produces consistent estimates for $\boldsymbol{\beta}$, and the OLS standard errors are correct. Thus, a standard t test on $\hat{\beta}_\lambda$ (which uses the estimate from step 2 and the uncorrected standard errors) is a valid test of the null hypothesis of no selection bias.

Although Heckman's two-step method uses the OLS method in the second stage, you can request the ML method by specifying the HECKIT(SECONDSTAGE=ML) option. When the second-stage method is the ML method, the model for y_i can be nonlinear.

Multivariate Limited Dependent Models

The multivariate model is similar to bivariate models. The generic form of the multivariate limited dependent variable model is

$$\begin{aligned} y_{1i}^* &= \mathbf{x}_{1i}'\boldsymbol{\beta}_1 + \epsilon_{1i} \\ y_{2i}^* &= \mathbf{x}_{2i}'\boldsymbol{\beta}_2 + \epsilon_{2i} \\ &\cdots \\ y_{mi}^* &= \mathbf{x}_{mi}'\boldsymbol{\beta}_m + \epsilon_{mi} \end{aligned}$$

where m is the number of models to be estimated. The vector ϵ has multivariate normal distribution with mean 0 and variance-covariance matrix Σ. Similar to bivariate models, the likelihood may involve computing multivariate normal integrations. This is done using Monte Carlo integration. (See Genz (1992); Hajivassiliou and McFadden (1998).)

When the number of equations, N, increases in a system, the number of parameters increases at the rate of N^2 because of the correlation matrix. When the number of parameters is large, sometimes the optimization converges but some of the standard deviations are missing. This usually means that the model is over-parameterized. The default method for computing the covariance is to use the inverse Hessian matrix. The Hessian is computed by finite differences, and in over-parameterized cases, the inverse cannot be computed. It is recommended that you reduce the number of parameters in such cases. Sometimes using the outer product covariance matrix (COVEST=OP option) may also help.

Variable Selection

Variable Selection

Variable Selection

Variable selection uses either Akaike's information criterion (AIC) or the Schwartz Bayesian criterion (SBC) and either a forward selection method or a backward elimination method.

Forward selection starts from a small subset of variables. In each step, the variable that gives the largest decrease in the value of the information criterion specified in the CRITER= option (AIC or SBC) is added. The process stops when the next candidate to be added does not reduce the value of the information criterion by more than the amount specified in the LSTOP= option in the MODEL statement.

Backward elimination starts from a larger subset of variables. In each step, one variable is dropped based on the information criterion that is chosen.

Tests on Parameters

Tests on Parameters

Tests on Parameters

In general, the hypothesis tested can be written as

$$H_0 : \mathbf{h}(\theta) = 0$$

where $\mathbf{h}(\theta)$ is an r by 1 vector valued function of the parameters θ given by the r expressions specified in the TEST statement.

Let $\hat{V}$ be the estimate of the covariance matrix of $\hat{\theta}$. Let $\hat{\theta}$ be the unconstrained estimate of θ and $\tilde{\theta}$ be the constrained estimate of θ such that $h(\tilde{\theta}) = 0$. Let

$$A(\theta) = \partial h(\theta)/\partial\theta \mid_{\hat{\theta}}$$

Using this notation, the test statistics for the three kinds of tests are computed as follows.

The Wald test statistic is defined as

$$W = h^{'}(\hat{\theta}) \left(A(\hat{\theta})\hat{V}A^{'}(\hat{\theta}) \right)^{-1} h(\hat{\theta})$$

The Wald test is not invariant to reparameterization of the model (Gregory and Veall 1985, Gallant 1987, p. 219). For more information about the theoretical properties of the Wald test, see Phillips and Park (1988).

The Lagrange multiplier test statistic is

$$LM = \lambda^{'} A(\tilde{\theta})\tilde{V}A^{'}(\tilde{\theta})\lambda$$

where λ is the vector of Lagrange multipliers from the computation of the restricted estimate $\tilde{\theta}$.

The likelihood ratio test statistic is

$$LR = 2\left(L(\hat{\theta}) - L(\tilde{\theta})\right)$$

where $\tilde{\theta}$ represents the constrained estimate of θ and L is the concentrated log-likelihood value.

For each kind of test, under the null hypothesis the test statistic is asymptotically distributed as a χ^2 random variable with r degrees of freedom, where r is the number of expressions in the TEST statement. The p-values reported for the tests are computed from the $\chi^2(r)$ distribution and are only asymptotically valid.

Monte Carlo simulations suggest that the asymptotic distribution of the Wald test is a poorer approximation to its small sample distribution than that of the other two tests. However, the Wald test has the lowest computational cost, since it does not require computation of the constrained estimate $\tilde{\theta}$.

The following is an example of using the TEST statement to perform a likelihood ratio test:

```
proc qlim;
   model y = x1 x2 x3;
   test x1 = 0, x2 * .5 + 2 * x3 = 0 /lr;
run;
```

Endogeneity and Instrumental Variables

The PROC QLIM models such as qualitative response or limited dependent variable models assume that the errors are independent of the explanatory variables. If this assumption fails to hold, the distributional form that the likelihood is based on is misspecified and the obtained coefficients are inconsistent.

To begin, consider a linear model

$$y_i = y_i^* = \beta_0 + \beta_1 x_{1i} + \cdots + \beta_k x_{ki} + u_i$$

Assume that $E(u) = 0$, $\mathrm{Cov}(x_j, u) = 0$ for $j = 1, \ldots, k-1$, and $\mathrm{Cov}(x_k, u) = \rho \neq 0$. Therefore, x_k is endogenous. The endogeneity comes from many sources, such as x_k having measurement error or omitting a variable that is correlated with x_k. If you ignore the endogeneity, you can estimate this model in PROC QLIM as follows (assuming $k = 4$):

```
proc qlim data=a;
   model y = x1 x2 x3 x4;
run;
```

However, this approach produces inconsistent maximum likelihood estimates. To obtain consistent maximum likelihood estimates, you should consider the joint density of the dependent variable and the endogenous variables. To do this in PROC QLIM, you need at least one instrument—that is, an observable variable, z_1—that is not in the structural equation and that satisfies two conditions: z_1 is exogenous (that is, $\mathrm{Cov}(z_1, u) = 0$), and z_1 must be correlated with the endogenous regressor x_k. Then, you can model x_k as

$$x_{ki} = \pi_0 + \pi_1 x_{1i} + \cdots + \pi_{k-1} x_{(k-1)i} + \theta z_{1i} + \epsilon_i$$

You can now write this reduced form equation along with the structural equation to obtain the consistent maximum likelihood estimates as follows:

```
proc qlim data=a;
   model y = x1 x2 x3 x4;
   model x4 = x1 x2 x3 z1;
run;
```

Estimating the structural model together with the reduced form models for the endogenous explanatory variables gives you the full information maximum likelihood (FIML) estimates. Because of the linearity of the structural model, you can estimate it efficiently and more simply by using the two-stage least squares estimator. However, PROC QLIM handles nonlinear models such as qualitative response and limited dependent variable models, and in their estimation it maximizes the corresponding joint likelihood function (for more information and an application see Wooldridge (2010, Section 15.7.3)). In the case of endogeneity, when the reduced form models for the endogenous explanatory variables are written along with the structural model, PROC QLIM maximizes the likelihood function that is obtained from the joint density of the response variable and the endogenous explanatory variables. For example, consider the following censored regression model in which one of the explanatory variables is a continuous endogenous variable:

$$\begin{aligned} y_{1i}^{*} &= \alpha y_{2i} + \mathbf{z}_{1i}'\boldsymbol{\beta} + u_i \\ y_{2i} &= \mathbf{z}_i'\boldsymbol{\pi} + \epsilon_i \\ y_{1i} &= \begin{cases} y_{1i}^{*} & \text{if } y_{1i}^{*} > 0 \\ 0 & \text{if } y_{1i}^{*} \le 0 \end{cases} \end{aligned}$$

The exogenous explanatory variables are $\mathbf{z}_{1i}$, and the continuous endogenous explanatory variable is y_{2i}.

The likelihood function to maximize is

$$L = \prod_{i \in \{y_{1i} > 0\}} f(y_{1i}, y_{2i}) \cdot \prod_{i \in \{y_{1i} = 0\}} \int_{-\infty}^{0} f(y_{1i}^{*}, y_{2i}) dy_{1i}^{*}$$

where $f(y_{1i}^{*}, y_{2i})$ is the joint density of y_{1i}^{*} and y_{2i}. Note that y_{1i} is substituted for y_{1i}^{*} when $y_{1i} > 0$. If you assume $(u_i, \epsilon_i) \overset{iid}{\sim} N(\mathbf{0}, \boldsymbol{\Sigma})$ with $\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_u^2 & \eta \\ \eta & \sigma_\epsilon^2 \end{bmatrix}$, then, by using $f(y_{1i}^{*}, y_{2i}) = f(y_{1i}^{*} | y_{2i}) \cdot f(y_{2i})$, you can write the likelihood function for each i as a multiplication of two parts. The first part is the probability density function of the normal distribution with mean $\mathbf{z}_i'\boldsymbol{\pi}$ and variance σ_ϵ^2, and the second part follows a Tobit model that has latent mean $\alpha y_{2i} + \mathbf{z}_{1i}'\boldsymbol{\pi} + (\eta/\sigma_\epsilon^2)(y_{2i} - \mathbf{z}_i'\boldsymbol{\pi})$ and variance $\sigma_u^2 - (\eta^2/\sigma_\epsilon^2)$. Then, you can obtain the log-likelihood function by taking the log of this multiplication and summing over i (for more information, see Wooldridge (2002, Section 16.6.2)). This is the log-likelihood function that PROC QLIM maximizes. The parameters $(\hat{\alpha}, \hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\pi}}, \hat{\sigma}_u^2, \hat{\sigma}_\epsilon^2, \hat{\eta})$ that are obtained from this maximization are the FIML estimators. Assuming that the latent model includes two instrumental variables and two exogenous explanatory variables, you can estimate this model in PROC QLIM as follows:

```
proc qlim data=a;
   model y1 = y2 z11 z12 / censored(lb=0);
   model y2 = z11 z12 z21 z22;
run;
```

For simple examples like the preceding ones, you can derive the likelihood function easily. However, as the number of endogenous explanatory variables increases, if these variables have a discontinuous nature, if simultaneity among equations exists, or if a combination of these occurs, then the derivation of the likelihood function becomes cumbersome, or, in some cases, the likelihood function does not even have a closed analytical form.

PROC QLIM can handle endogeneity regardless of the nature of the endogenous explanatory variables for a single structural model. In the case of one endogenous explanatory variable, PROC QLIM reports the FIML estimates that are calculated by using the analytical likelihood function that is obtained from the joint

distribution of the dependent variable and the endogenous variable. When there is more than one endogenous explanatory variable, the analytical form of the likelihood function is usually not available; in this case PROC QLIM reports the simulated maximum likelihood estimates. For the simulated maximum likelihood estimation method, PROC QLIM uses the Geweke-Hajivassiliou-Keane (GHK) simulator (see, among others, Hajivassiliou, McFadden, and Ruud (1996)) to simulate the joint distribution of the dependent variable and the endogenous variables. The simulation is facilitated by assuming that the error terms in the latent models for the dependent variable and the endogenous explanatory variables are distributed as multivariate normal.

When you estimate a model in PROC QLIM, you can take the endogeneity into account by writing the structural model along with the reduced form models for each endogenous variable. Examples are provided in the following sections.

Probit Model with a Continuous Endogenous Explanatory Variable

Consider a probit model that contains a single endogenous explanatory variable in addition to two instruments and two exogenous explanatory variables. The model is

$$\begin{aligned} y_{1i}^* &= \alpha_1 y_{2i} + \beta_1 z_{1i} + \beta_2 z_{2i} + u_i \\ y_{2i}^* &= \pi_1 z_{1i} + \pi_2 z_{2i} + \pi_3 z_{3i} + \pi_4 z_{4i} + \epsilon_i \\ y_{1i} &= \begin{cases} 1 & \text{if } y_{1i}^* > 0 \\ 0 & \text{if } y_{1i}^* \le 0 \end{cases} \\ y_{2i} &= y_{2i}^* \end{aligned}$$

where $\mathrm{Cov}(u, \epsilon) = \eta$. You can estimate this model by using the following statements:

```
proc qlim data=a;
   model y1 = y2 z1 z2 / discrete;
   model y2 = z1 z2 z3 z4;
run;
```

Probit Model with a Binary Endogenous Explanatory Variable

Consider a probit model that contains a single binary endogenous explanatory variable in addition to two instruments and two exogenous explanatory variables. The model is

$$\begin{aligned} y_{1i}^* &= \alpha_1 y_{2i} + \beta_1 z_{1i} + \beta_2 z_{2i} + u_i \\ y_{2i}^* &= \pi_1 z_{1i} + \pi_2 z_{2i} + \pi_3 z_{3i} + \pi_4 z_{4i} + \epsilon_i \\ y_{1i} &= \begin{cases} 1 & \text{if } y_{1i}^* > 0 \\ 0 & \text{if } y_{1i}^* \le 0 \end{cases} \\ y_{2i} &= \begin{cases} 1 & \text{if } y_{2i}^* > 0 \\ 0 & \text{if } y_{2i}^* \le 0 \end{cases} \end{aligned}$$

where $\mathrm{Cov}(u, \epsilon) = \eta$. You can estimate this model by using the following statements:

```
proc qlim data=a;
   model y1 = y2 z1 z2 / discrete;
   model y2 = z1 z2 z3 z4 / discrete;
run;
```

Probit Model with a Censored Endogenous Explanatory Variable

Consider a probit model that contains a single censored (below zero) endogenous explanatory variable in addition to two instruments and two exogenous explanatory variables. The model is

$$
\begin{aligned}
y^*_{1i} &= \alpha_1 y_{2i} + \beta_1 z_{1i} + \beta_2 z_{2i} + u_i \\
y^*_{2i} &= \pi_1 z_{1i} + \pi_2 z_{2i} + \pi_3 z_{3i} + \pi_4 z_{4i} + \epsilon_i \\
y_{1i} &= \begin{cases} 1 & \text{if } y^*_{1i} > 0 \\ 0 & \text{if } y^*_{1i} \le 0 \end{cases} \\
y_{2i} &= \begin{cases} y^*_{2i} & \text{if } y^*_{2i} > 0 \\ 0 & \text{if } y^*_{2i} \le 0 \end{cases}
\end{aligned}
$$

where $\mathrm{Cov}(u, \epsilon) = \eta$. You can estimate this model by using the following statements:

```
proc qlim data=a;
   model y1 = y2 z1 z2 / discrete;
   model y2 = z1 z2 z3 z4 / censored(lb=0);
run;
```

Censored Regression Model with a Binary Endogenous Explanatory Variable

Consider a Type 1 Tobit model that contains a single binary endogenous explanatory variable in addition to two instruments and two exogenous explanatory variables. The model is

$$
\begin{aligned}
y^*_{1i} &= \alpha_1 y_{2i} + \beta_1 z_{1i} + \beta_2 z_{2i} + u_i \\
y^*_{2i} &= \pi_1 z_{1i} + \pi_2 z_{2i} + \pi_3 z_{3i} + \pi_4 z_{4i} + \epsilon_i \\
y_{1i} &= \begin{cases} y^*_{1i} & \text{if } y^*_{1i} > 0 \\ 0 & \text{if } y^*_{1i} \le 0 \end{cases} \\
y_{2i} &= \begin{cases} 1 & \text{if } y^*_{2i} > 0 \\ 0 & \text{if } y^*_{2i} \le 0 \end{cases}
\end{aligned}
$$

where $\mathrm{Cov}(u, \epsilon) = \eta$. You can estimate this model by using the following statements:

```
proc qlim data=a;
   model y1 = y2 z1 z2 / censored(lb=0);
   model y2 = z1 z2 z3 z4 / discrete;
run;
```

Censored Regression Model with Binary and Continuous Endogenous Explanatory Variables

Consider a Type 1 Tobit model that contain binary and continuous endogenous explanatory variables in addition to two instruments and two exogenous explanatory variables. The model is

$$
\begin{aligned}
y^*_{1i} &= \alpha_1 y_{21i} + \alpha_2 y_{22i} + \beta_1 z_{1i} + \beta_2 z_{2i} + u_i \\
y^*_{21i} &= \pi_{11} z_{1i} + \pi_{12} z_{2i} + \pi_{13} z_{3i} + \pi_{14} z_{4i} + \epsilon_{1i} \\
y^*_{22i} &= \pi_{21} z_{1i} + \pi_{22} z_{2i} + \pi_{23} z_{3i} + \pi_{24} z_{4i} + \epsilon_{2i} \\
y_{1i} &= \begin{cases} y^*_{1i} & \text{if } y^*_{1i} > 0 \\ 0 & \text{if } y^*_{1i} \le 0 \end{cases} \\
y_{21i} &= \begin{cases} 1 & \text{if } y^*_{21i} > 0 \\ 0 & \text{if } y^*_{21i} \le 0 \end{cases} \\
y_{22i} &= y^*_{22i}
\end{aligned}
$$

where $\mathrm{Cov}(u, \epsilon_1, \epsilon_2) = \eta$. You can estimate this model by using the following statements:

```
proc qlim data=a;
   model y1 = y21 y22 z1 z2 / censored(lb=0);
   model y21 = z1 z2 z3 z4  / discrete;
   model y22 = z1 z2 z3 z4;
run;
```

Probit Model with Binary, Censored, and Truncated Endogenous Explanatory Variables

Consider a probit model that contains binary, censored (below zero), and truncated (below zero) endogenous explanatory variables. The model is

$$
\begin{aligned}
y^*_{1i} &= \alpha_1 y_{21i} + \alpha_2 y_{22i} + \alpha_3 y_{23i} + u_i \\
y^*_{21i} &= \pi_{11} z_{1i} + \pi_{12} z_{2i} + \pi_{13} z_{3i} + \pi_{14} z_{4i} + \epsilon_{1i} \\
y^*_{22i} &= \pi_{21} z_{1i} + \pi_{22} z_{2i} + \pi_{23} z_{3i} + \pi_{24} z_{4i} + \epsilon_{2i} \\
y^*_{23i} &= \pi_{31} z_{1i} + \pi_{32} z_{2i} + \pi_{33} z_{3i} + \pi_{34} z_{4i} + \epsilon_{3i} \\
y_{1i} &= \begin{cases} 1 & \text{if } y^*_{1i} > 0 \\ 0 & \text{if } y^*_{1i} \le 0 \end{cases} \\
y_{21i} &= \begin{cases} 1 & \text{if } y^*_{21i} > 0 \\ 0 & \text{if } y^*_{21i} \le 0 \end{cases} \\
y_{22i} &= \begin{cases} y^*_{22i} & \text{if} y^*_{22i} > 0 \\ 0 & \text{if } y^*_{22i} \le 0 \end{cases} \\
y_{23i} &= y^*_{23i} \quad \text{if} \quad y^*_{23i} > 0
\end{aligned}
$$

where $z_1, \ldots, z_4$ are the instrumental variables that are independent of the errors. You can estimate this model by using the following statements:

```
proc qlim data=a;
   model y1  = y21 y22 y23 / discrete;
   model y21 = z1 z2 z3 z4 / discrete;
   model y22 = z1 z2 z3 z4 / censored(lb=0);
   model y23 = z1 z2 z3 z4 / truncated(lb=0);
run;
```

Note that the dependent variable y_1 should not occur in the models for the endogenous explanatory variables, because this causes inconsistent coefficient estimates. In other words, you should write the models for the endogenous explanatory variables as reduced form models. PROC QLIM does not handle simultaneous equations models.

Test for Endogeneity

PROC QLIM has two ways to test the null hypothesis that an endogenous explanatory variable (EEV) is in fact exogenous. In the case of a single EEV, the first testing method involves a likelihood ratio test of H_0 : _rho $= 0$. For example, consider the probit model with a binary endogenous explanatory variable that was considered earlier; y_2 is exogenous if the error term in the model for y_1^* is uncorrelated with the error term in the model for y_2^*. Therefore, testing to determine whether this correlation is 0 or not provides an endogeneity test for y_2. You can do this in PROC QLIM as follows:

```
proc qlim data=a;
   model y1 = y2 z1 z2 / discrete;
   model y2 = z1 z2 z3 z4 / discrete;
   test  _rho = 0 / LR;
run;
```

Failing to reject the null hypothesis favors the decision that $y2$ is exogenous in the model for $y1$.

When there are two or more EEVs, the test becomes the joint likelihood ratio test of whether corresponding correlations are 0 or not.

The second testing method is similar to the approach of Rivers and Vuong (1988). Considering the same model, you can write

$$u_i = \theta\epsilon_i + e_i$$

where $\theta = \eta/\sigma_\epsilon^2$ and e is independent of zs and ϵ. You can now write

$$y_{1i}^* = \alpha_1 y_{2i} + \beta_1 z_{1i} + \beta_2 z_{2i} + \theta\epsilon_i + e_i$$

Testing $H_0 : \theta = 0$ is the same as testing whether u_i is correlated with ϵ_i or testing whether y_{2i} is endogenous or not. Because ϵ_i are unobserved, you can replace them with the OLS residuals from the model for y_{2i}^* and apply a robust t test. Note that even though y_{2i} is binary (or censored), the test is still correct under H_0.

This approach can be summarized as a two-step procedure. In the first step, generated regressors—that is, the OLS residuals from the models for each of the EEVs—are obtained. In the second step, the structural model that includes the generated regressors as additional explanatory variables is estimated by the maximum likelihood method and the joint significance of these generated regressors is tested by the Wald test.

In PROC QLIM, you can apply the second method for the same test that was considered previously as follows:

```
proc qlim data=a;
   model y1 = y2 z1 z2 / discrete endotest(y2);
   model y2 = z1 z2 z3 z4 / discrete;
run;
```

Overidentification Test

In PROC QLIM you can test the validity of instrumental variables (IVs) by specifying the OVERID option in the ENDOGENOUS or MODEL statement. The OVERID test is a maximum likelihood version of the overidentifying restrictions test in the IV framework. If you have more IVs than are necessary for identification—that is, overidentifying IVs—you can use them to test the validity of your IVs. When you use the OVERID option to specify the overidentifying IVs, it applies the likelihood ratio test of the joint significance of these IVs, included as additional explanatory variables in the structural model that it estimates by the MLE jointly with the reduced form models. In effect, you test whether the overidentifying IVs are correlated with the error term in the structural model. You specify the reduced form models through the overidentifying IVs. The structural model is the model that includes the OVERID option. For example, consider the probit model that contains a continuous endogenous explanatory variable. You can consider $z3$ or $z4$ in the model for $y2$ as an overidentifying IV; therefore, you can specify the OVERID test as follows:

```
proc qlim data=a;
   model y1 = y2 z1 z2 / discrete overid(y2.z4);
   model y2 = z1 z2 z3 z4;
run;
```

In this case, PROC QLIM estimates the structural model $y1$, including the overidentifying IV $z4$ as an additional explanatory variable in this model, jointly with the reduced form model $y2$. Then it uses the likelihood ratio test to test the hypothesis that the overidentifying IV is insignificant. Rejecting this hypothesis raises doubts about the validity of the instruments $z3$ and $z4$.

Note that, as long as you have continuous endogenous explanatory variables, the test result is invariant to which overidentifying IVs you specify in the test.

Panel Data Analysis

You can use panel data to estimate the random effects in linear, censored response, truncated response, discrete choice, and stochastic frontier models.

Random-Effects Models for Panel Data

The general form of a nonlinear random-effects model is defined by the density for an observed random variable, y_{it}, as follows[1]

$$f(y_{it}|\mathbf{x}_{it},\mu_i) = g(y_{it},\mathbf{x}'_{it}\boldsymbol{\beta},\mu_i;\boldsymbol{\theta})$$

where $\boldsymbol{\theta}$ is a vector of ancillary parameters such as a scale parameter or an overdispersion parameter and μ_i, $i = 1,\ldots,N$, embodies the group-specific heterogeneity, which is unobservable and has the density

$$f(\mu_i) = h(\mu_i;\boldsymbol{\theta})$$

There are T_i observations for group i. For example, in the case of a random-effects Tobit model, y_{it} is specified as

$$y^*_{it} = \mathbf{x}'_{it}\boldsymbol{\beta} + \epsilon_{it}, \qquad t = 1,\ldots,T_i, \quad i = 1,\ldots,N$$

$$y_{it} = \begin{cases} y^*_{it} & \text{if} y^*_{it} > 0 \\ 0 & \text{if} y^*_{it} \le 0 \end{cases}$$

where

$$\epsilon_{it} = \mu_i + v_{it}$$

$$v_{it}|\mathbf{x}_i,\mu_i \sim N(0,\sigma^2)$$

$$\mu_i|\mathbf{x}_i \sim N(0,\sigma^2_\mu)$$

where $\mathbf{x}_i$ contains $\mathbf{x}_{it}$ for all t and $\boldsymbol{\theta}$ consists of σ and σ_μ. Therefore, for this model

$$f(y_{it}|\mathbf{x}_{it},\mu_i) = \{1-\Phi[(\mathbf{x}'_{it}\boldsymbol{\beta}+\mu_i)/\sigma]\}^{1[y_{it}=0]}\{(1/\sigma)\phi[(y_{it}-\mathbf{x}'_{it}\boldsymbol{\beta}-\mu_i)/\sigma]\}^{1[y_{it}>0]}$$

and

$$f(\mu_i) = \phi(\mu_i/\sigma_\mu)$$

where $\Phi(\cdot)$ is the cumulative density function and $\phi(\cdot)$ is the probability density function of the standard normal distribution and $1[\cdot]$ is the indicator function.

By construction, the T_i observations in group i are correlated and jointly distributed with a distribution that does not factor into the product of the marginal distributions. To be able to drive the joint distribution of the T_i+1 random variables, $f(y_{i1},y_{i2},\ldots,y_{iT_i},\mu_i|\mathbf{x}_i,\boldsymbol{\beta};\boldsymbol{\theta})$, the assumption that the T_i observations are independent conditional on μ_i is important. Under this assumption the joint distribution can be written as

$$\begin{aligned} f(y_{i1},y_{i2},\ldots,y_{iT_i},\mu_i|\mathbf{x}_i,\boldsymbol{\beta};\boldsymbol{\theta}) &= f(y_{i1},y_{i2},\ldots,y_{iT_i}|\mathbf{x}_i,\boldsymbol{\beta};\boldsymbol{\theta})f(\mu_i) \\ &= \prod_{t=1}^{T_i} g(y_{it},\mathbf{x}'_{it}\boldsymbol{\beta},\mu_i;\boldsymbol{\theta})h(\mu_i;\boldsymbol{\theta}) \end{aligned}$$

[1] The i subscript represents individuals, and the t subscript represents time.

In order to form the likelihood function for the observed data, the unobserved component, μ_i, must be integrated out. For individual i

$$L_i = f(y_{i1}, y_{i2}, \ldots, y_{iT_i} | \mathbf{x}_i, \boldsymbol{\beta}; \boldsymbol{\theta}) = \int_{\mu} \left[\prod_{t=1}^{T_i} g(y_{it}, \mathbf{x}_{it}'\boldsymbol{\beta}, \mu_i; \boldsymbol{\theta}) \right] h(\mu_i; \boldsymbol{\theta}) d\mu_i$$

Therefore, the log-likelihood function for the observed data becomes

$$\ln L = \sum_{i=1}^{N} \ln \left[\int_{\mu} \left(\prod_{t=1}^{T_i} g(y_{it}, \mathbf{x}_{it}'\boldsymbol{\beta}, \mu_i; \boldsymbol{\theta}) \right) h(\mu_i; \boldsymbol{\theta}) d\mu_i \right]$$

In most cases, the integral in the square brackets does not have a closed form. The following subsections describe three approaches to this integration.

Simulated Maximum Likelihood

You can use simulation to approximate the integral. First, note that

$$\int_{\mu} \left(\prod_{t=1}^{T_i} g(y_{it}, \mathbf{x}_{it}'\boldsymbol{\beta}, \mu_i; \boldsymbol{\theta}) \right) h(\mu_i; \boldsymbol{\theta}) d\mu_i = E[F(\mu_i; \boldsymbol{\theta})]$$

The function is smooth, continuous, and continuously differentiable. By the law of large numbers, if $(\mu_{i1}, \mu_{i2}, \ldots, \mu_{iR})$ is a sample of iid draws from $h(\mu_i; \boldsymbol{\theta})$, then

$$\operatorname{plim} \frac{1}{R} \sum_{r=1}^{R} F(\mu_{ir}; \boldsymbol{\theta}) = E[F(\mu_i; \boldsymbol{\theta})]$$

This operation is implemented by simulation that uses a random-number generator. PROC QLIM inserts the simulated integral in the log likelihood to obtain the simulated log likelihood

$$\ln L_{Simulated} = \sum_{i=1}^{N} \ln \left[\frac{1}{R} \sum_{r=1}^{R} \left(\prod_{t=1}^{T_i} g(y_{it}, \mathbf{x}_{it}'\boldsymbol{\beta}, \mu_{ir}; \boldsymbol{\theta}) \right) \right]$$

and maximizes the simulated log likelihood with respect to the parameters $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$.

Under certain assumptions (Greene 2001), the simulated likelihood estimator and the maximum likelihood estimator are equivalent. For this equivalence result to hold, the number of draws, R, must increase faster than the number of observations, N. For this reason, if the NDRAW option is not specified, then by default, it is tied to the sample size by using the rule $R = N^{1+\delta}$, where $\delta = 1/2$.

The use of independent random draws in simulation is conceptually straightforward, and the statistical properties of the simulated maximum likelihood estimator are easy to derive. However, simulation is a very computationally intensive technique. Moreover, the simulation method itself contributes to the variation of the simulated maximum likelihood estimator; see, for example, Geweke (1995). There are other ways to take draws that can provide greater accuracy by covering the domain of the integral more uniformly and by lowering the simulation variance (Train 2009 section 9.3). Quas–Monte Carlo methods (QMC), for example, are based on an integration technique that replaces the pseudorandom draws of Monte Carlo (MC) integration with a sequence of judiciously selected nonrandom points that provide more uniform coverage of the domain of the integral. Therefore, the advantage of QMC integration over MC integration is that for

some types of sequences, the accuracy is far greater, convergence is much faster, and the simulation variance is smaller. QMC methods are surveyed in Bhat (2001), Sloan and Woźniakowski (1998), and Morokoff and Caflisch (1995). Besides MC simulation, PROC QLIM offers the QMC integration method that uses Halton sequences.

QMC Method Using the Halton Sequence

Halton sequences (Halton 1960) provide a uniform coverage for each observations's integral, and they decrease the simulation variance by inducing a negative correlation over the draws for each observation. A Halton sequence is constructed deterministically in terms of a prime number as its base. For example, the following sequence is the Halton sequence for 2:

$$1/2, 1/4, 3/4, 1/8, 5/8, 3/8, 7/8, 1/16, 9/16, \ldots$$

For more information about how to generate a Halton sequence, see Train (2009) section 9.3.3.

If you use the QMC method, one long Halton sequence is created, and then part of the sequence (or the whole sequence, depending on whether you decide to discard the initial elements of the sequence[2]) is used in groups. Each group of consequent elements constitutes the "draws" for each cross-sectional observation. This way, each subsequence fills in the gaps for the previous subsequences, and the draws for one observation tend to be negatively correlated with those for the previous observation.

When the number of draws used for each observation rises, the coverage for each observation improves. This improvement in turn improves the accuracy; however, the negative covariance across observations diminishes. Because Halton draws are far more effective than random draws for simulation, a small number of Halton draws provide relatively good integration (Spanier and Maize 1991).

The Halton draws are for a uniform density. PROC QLIM evaluates the inverse cumulative standard normal density for each element of the Halton sequence to obtain draws from a standard normal density.

Approximation by Hermite Quadrature

This method is the Butler and Moffitt (1982) approach, which is based on models in which μ_i has a normal distribution. If μ_i is normally distributed with zero mean, then

$$\int_{\mu} \left(\prod_{t=1}^{T_i} g(y_{it}, \mathbf{x}_{it}'\boldsymbol{\beta}, \mu_i; \boldsymbol{\theta}) \right) h(\mu_i; \boldsymbol{\theta}) d\mu_i$$
$$= \frac{1}{\sigma_\mu \sqrt{2\pi}} \int_{-\infty}^{+\infty} \prod_{t=1}^{T_i} g(y_{it}, \mathbf{x}_{it}'\boldsymbol{\beta}, \mu_i; \boldsymbol{\theta}) \exp\left(\frac{-\mu_i^2}{2\sigma_\mu^2} \right) d\mu_i$$

Let $r_i = \mu_i/(\sigma_\mu\sqrt{2})$. Then $\mu_i = (\sigma_\mu\sqrt{2})r_i$ and $d\mu_i = (\sigma_\mu\sqrt{2})dr_i$. Making the change of variable and letting the error effects be additive produce

$$L_i = \frac{1}{\sqrt{\pi}} \int_{-\infty}^{+\infty} \exp(-r_i^2) \left[\prod_{t=1}^{T_i} g(y_{it}, \mathbf{x}_{it}'\boldsymbol{\beta} + (\sigma_\mu\sqrt{2})r_i; \boldsymbol{\theta}) \right] dr_i$$

[2] When sequences are created in multiple dimensions, the initial part of the series is usually eliminated because the initial terms of multiple Halton sequences are highly correlated. However, there is no such correlation for a single dimension.

This likelihood function is in a form that can be approximated accurately by using Gauss-Hermite quadrature, which eliminates the integration. Thus, the log-likelihood function can be approximated with

$$\ln L_h = \sum_{i=1}^{N} \ln \left[\frac{1}{\sqrt{\pi}} \sum_{h=1}^{H} w_h \prod_{t=1}^{T_i} g(y_{it}, \mathbf{x}_{it}'\boldsymbol{\beta} + (\sigma_\mu \sqrt{2}) r_i; \boldsymbol{\theta}) \right]$$

where w_h and r_h are the weights and nodes for the Hermite quadrature of degree H. PROC QLIM maximizes $\ln L_h$ when the Hermite quadrature option is specified (METHOD=HERMITE in the RANDOM statement).

Bayesian Analysis

To perform Bayesian analysis, you must specify a BAYES statement. Unless otherwise stated, all options in this section are options in the BAYES statement.

By default, PROC QLIM uses the random walk Metropolis algorithm to obtain posterior samples. For the implementation details of the Metropolis algorithm in PROC QLIM, such as the blocking of the parameters and tuning of the covariance matrices, see the sections "Blocking of Parameters" on page 2011 and "Tuning the Proposal Distribution" on page 2011.

The Bayes theorem states that

$$p(\theta|\mathbf{y}) \propto \pi(\theta) L(y|\theta)$$

where θ is a parameter or a vector of parameters and $\pi(\theta)$ is the product of the prior densities that are specified in the PRIOR statement. The term $L(y|\theta)$ is the likelihood associated with the MODEL statement.

Blocking of Parameters

In a multivariate parameter model, all the parameters are updated in one single block (by default or when you specify the SAMPLING=MULTIMETROPOLIS option). This could be inefficient, especially when parameters have vastly different scales. As an alternative, you could update the parameters one at the time (by specifying SAMPLING=UNIMETROPOLIS).

Tuning the Proposal Distribution

One key factor in achieving high efficiency of a Metropolis-based Markov chain is finding a good proposal distribution for each block of parameters. This process is called tuning. The tuning phase consists of a number of loops controlled by the options MINTUNE and MAXTUNE. The MINTUNE= option controls the minimum number of tuning loops and has a default value of 2. The MAXTUNE= option controls the maximum number of tuning loops and has a default value of 24. Each loop is iterated the number of times specified by the NTU= option, which has a default of 500. At the end of every loop, PROC QLIM examines the acceptance probability for each block. The acceptance probability is the percentage of NTU proposed values that have been accepted. If this probability does not fall within the acceptance tolerance range (see the following section), the proposal distribution is modified before the next tuning loop.

A good proposal distribution should resemble the actual posterior distribution of the parameters. Large sample theory states that the posterior distribution of the parameters approaches a multivariate normal distribution (see Gelman et al. 2004, Appendix B; Schervish 1995, Section 7.4). That is why a normal proposal distribution often works well in practice. The default proposal distribution in PROC QLIM is the normal distribution.

Scale Tuning

The acceptance rate is closely related to the sampling efficiency of a Metropolis chain. For a random walk Metropolis, a high acceptance rate means that most new samples occur right around the current data point. Their frequent acceptance means that the Markov chain is moving rather slowly and not exploring the parameter space fully. A low acceptance rate means that the proposed samples are often rejected; hence the chain is not moving much. An efficient Metropolis sampler has an acceptance rate that is neither too high nor too low. The scale c in the proposal distribution $q(\cdot|\cdot)$ effectively controls this acceptance probability. Roberts, Gelman, and Gilks (1997) show that if both the target and proposal densities are normal, the optimal acceptance probability for the Markov chain should be around 0.45 in a one-dimension problem and should asymptotically approach 0.234 in higher-dimension problems. The corresponding optimal scale is 2.38, which is the initial scale that is set for each block.

Because of the nature of stochastic simulations, it is impossible to fine-tune a set of variables so that the Metropolis chain has exactly the desired acceptance rate that you want. In addition, Roberts and Rosenthal (2001) empirically demonstrate that an acceptance rate between 0.15 and 0.5 is at least 80% efficient, so there is really no need to fine-tune the algorithms to reach an acceptance probability that is within a small tolerance of the optimal values. PROC QLIM works with a probability range, determined by TargetAcceptance ± 0.075. If the observed acceptance rate in a given tuning loop is less than the lower bound of the range, the scale is reduced; if the observed acceptance rate is greater than the upper bound of the range, the scale is increased. During the tuning phase, a scale parameter in the normal distribution is adjusted as a function of the observed acceptance rate and the target acceptance rate. PROC QLIM uses the following updating scheme: [3]

$$c_{\text{new}} = \frac{c_{\text{cur}} \cdot \Phi^{-1}(p_{\text{opt}}/2)}{\Phi^{-1}(p_{\text{cur}}/2)}$$

where c_{cur} is the current scale, p_{cur} is the current acceptance rate, and p_{opt} is the optimal acceptance probability.

Covariance Tuning

To tune a covariance matrix, PROC QLIM takes a weighted average of the old proposal covariance matrix and the recent observed covariance matrix, based on the number samples (as specified by the NTU= option) NTU samples in the current loop. The formula to update the covariance matrix is:

$$\text{COV}_{\text{new}} = 0.75\ \text{COV}_{\text{cur}} + 0.25\ \text{COV}_{\text{old}}$$

There are two ways to initialize the covariance matrix:

- The default is an identity matrix that is multiplied by the initial scale of 2.38 and divided by the square root of the number of estimated parameters in the model. A number of tuning phases might be required before the proposal distribution is tuned to its optimal stage, because the Markov chain needs to spend time to learn about the posterior covariance structure. If the posterior variances of your parameters vary by more than a few orders of magnitude, if the variances of your parameters are much different from 1, or if the posterior correlations are high, then the proposal tuning algorithm might have difficulty forming an acceptable proposal distribution.

[3] Roberts and associates demonstrate that the relationship between acceptance probability and scale in a random walk Metropolis scheme is $p = 2\Phi\left(-\sqrt{I}c/2\right)$, where c is the scale, p is the acceptance rate, Φ is the CDF of a standard normal, and $I \equiv E_f[(f'(x)/f(x))^2]$, $f(x)$ is the density function of samples (Roberts, Gelman, and Gilks 1997; Roberts and Rosenthal 2001). This relationship determines the updating scheme, with I replaced by the identity matrix to simplify calculation.

- Alternatively, you can use a numerical optimization routine, such as the quasi-Newton method, to find a starting covariance matrix. The optimization is performed on the joint posterior distribution, and the covariance matrix is a quadratic approximation at the posterior mode. In some cases this is a better and more efficient way of initializing the covariance matrix. However, there are cases, such as when the number of parameters is large, where the optimization could fail to find a matrix that is positive definite. In those cases, the tuning covariance matrix is reset to the identity matrix.

A by-product of the optimization routine is that it also finds the maximum a posteriori (MAP) estimates with respect to the posterior distribution. The MAP estimates are used as the initial values of the Markov chain.

For more information, see the section "INIT Statement" on page 1976.

Initial Values of the Markov Chains

You can assign initial values to any parameters. (For more information, see the INIT statement.) If you use the optimization option PROPCOV=, then PROC QLIM starts the tuning at the optimized values. This option overwrites the provided initial values. If you specify the RANDINIT option, the information that the INIT statement provides is overwritten.

Aggregation of Multiple Chains

When you want to exploit the possibility of running several MCMC instances at the same time (NTRDS=n>1), you face the problem of aggregating the chains. In ordinary applications, each MCMC instance can easily obtain stationary samples from the entire posterior distribution. In these applications, you can use the option AGGREGATION=UNWEIGHTED. This option piles up one chain on top of another and makes no particular adjustment. However, when the posterior distribution is characterized by multiple distinct posterior modes, some of the MCMC instances fail to obtain stationary samples from the entire posterior distribution. You can use the option AGGREGATION=WEIGHTED when the posterior samples from each MCMC instance approximate well only a part of the posterior distribution.

The main idea behind the option AGGREGATION=WEIGHTED is to consider the entire posterior distribution to be similar to a mixture distribution. When you are sampling with multiple threads, each MCMC instance samples from one of the mixture components. Then the samples from each mixture component are aggregated together using a resampling scheme in which weights are proportional to the nonnormalized posterior distribution.

Description of the Algorithm

The preliminary step of the aggregation that is implied by the option AGGREGATION=WEIGHTED is to run several (K) independent instances of the MCMC algorithm. Each instance searches for a set of stationary samples. Notice that the concept of stationarity is weaker: each instance might be able to explore not the entire posterior but only portions of it. In the next equation, each column represents the output from one MCMC instance,

$$\begin{pmatrix} x_{11} \\ x_{21} \\ \cdots \\ x_{n1} \end{pmatrix} \begin{pmatrix} x_{12} \\ x_{22} \\ \cdots \\ x_{n2} \end{pmatrix} \cdots \begin{pmatrix} x_{1K} \\ x_{2K} \\ \cdots \\ x_{nK} \end{pmatrix} \sim \text{globally/locally sampled from the posterior}$$

If the length of each chain is less than n, you can augment the corresponding chain by subsampling the chain itself. Each chain is then sorted with respect to the nonnormalized posterior density: $\pi(x_{[1].}) \leq \pi(x_{[2].}) \leq \cdots \pi(x_{[n].})$. Therefore,

$$\begin{pmatrix} x_{11} \\ x_{21} \\ \cdots \\ x_{n1} \end{pmatrix} \begin{pmatrix} x_{12} \\ x_{22} \\ \cdots \\ x_{n2} \end{pmatrix} \cdots \begin{pmatrix} x_{1K} \\ x_{2K} \\ \cdots \\ x_{nK} \end{pmatrix} \rightarrow \begin{pmatrix} x_{[1]1} \\ x_{[2]1} \\ \cdots \\ x_{[n]1} \end{pmatrix} \begin{pmatrix} x_{[1]2} \\ x_{[2]2} \\ \cdots \\ x_{[n]2} \end{pmatrix} \cdots \begin{pmatrix} x_{[1]K} \\ x_{[2]K} \\ \cdots \\ x_{[n]K} \end{pmatrix}$$

The final step is to use a multinomial sampler to resample each row i with weights proportional to the nonnormalized posterior densities:

$$\widetilde{x}_{(i-1)K+1}, \widetilde{x}_{(i-1)K+2}, \ldots, \widetilde{x}_{(i-1)K+K} \sim \text{Multinom}\left[x_{[i]1}, x_{[i]2}, \ldots, x_{[i]K}; \pi(x_{[i]1}), \pi(x_{[i]2}), \ldots, \pi(x_{[i]K})\right]$$

The resulting posterior sample

$$\widetilde{x}_1, \widetilde{x}_2, \ldots, \widetilde{x}_K, \ldots, \widetilde{x}_{(i-1)K+1}, \widetilde{x}_{(i-1)K+2}, \ldots, \widetilde{x}_{(i-1)K+K}, \ldots, \widetilde{x}_{(n-1)K+1}, \widetilde{x}_{(n-1)K+2}, \ldots, \widetilde{x}_{nK}$$

is a good approximation of the posterior distribution that is characterized by multiple modes.

Automated Initialization of MCMC

The MCMC methods can generate samples from the posterior distribution. The correct implementation of these methods often requires the stationarity analysis, the convergence analysis and the accuracy analysis of the posterior samples. These analyses usually imply the following:

- initialization of the proposal distribution
- initialization of the chains (starting values)
- determination of the burn-in
- determination of the length of the chains.

In more general terms, this determination is equivalent to deciding whether the samples are drawn from the posterior distribution (stationarity analysis), and whether the number of samples is large enough to accurately approximate the posterior distribution (accuracy analysis). You can use the AUTOMCMC option to automate and facilitate the stationary analysis and the accuracy analysis.

Description of the Algorithm

The algorithm consists of two phases. In the first phase, the stationarity phase, the algorithm tries to generate stationary samples from the posterior distribution. In the second phase, the accuracy phase, the algorithm searches for an accurate representation of the posterior distribution. The algorithm implements the following tools:

- Geweke test to check stationarity
- Heidelberger-Welch test to check stationarity and provide a proxy for the burn-in
- Heidelberger-Welch half-test to check the accuracy of the posterior mean

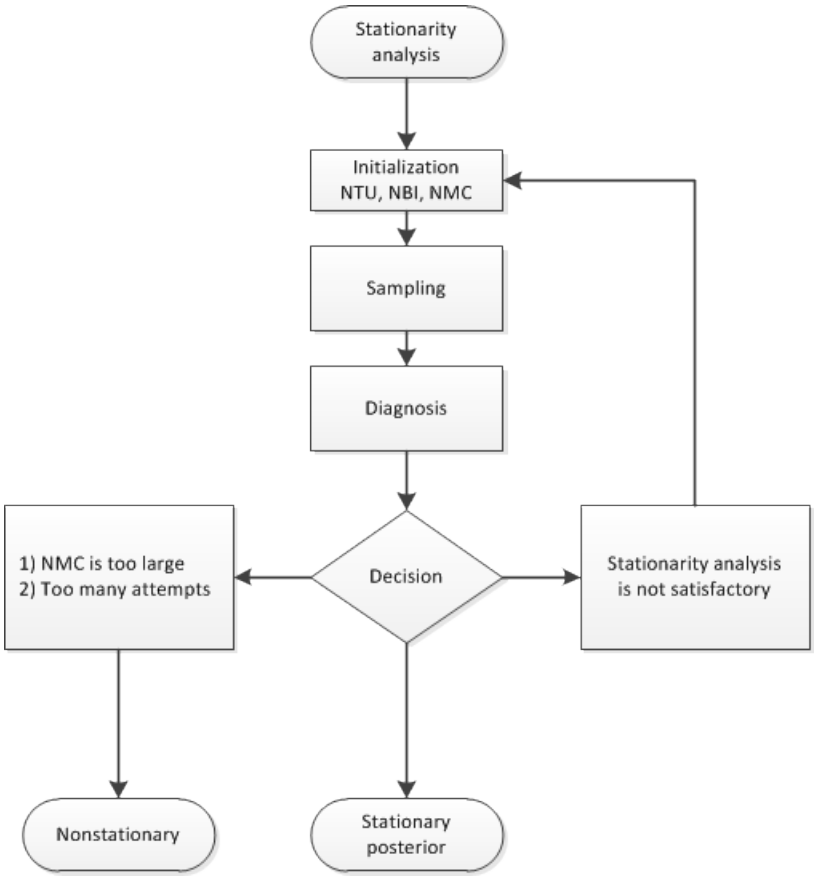
- Raftery-Lewis test to check the accuracy of a given percentile (indirectly proving a proxy for the number of required samples)
- effective sample size analysis to determine a proxy of the number of required samples

During the stationarity phase, the algorithm searches for stationarity. The number of attempts that the algorithm makes is determined by the option ATTEMPTS=*number*. During each attempt, a preliminary tuning stage chooses a proposal distribution for the MCMC sampler. At the end of the preliminary tuning phase, the algorithm analyzes tests for the stationarity of the samples. If the percentage of successful stationary tests is equal to or greater than the percentage that is indicated by the option TOL=*value*, then the posterior sample is considered to be stationary. If the sample cannot be considered stationary, then the algorithm attempts to achieve stationarity by changing some of the initialization parameters as follows:

- increasing the number of tuning samples (NTU)
- increasing the number of posterior samples (NMC)
- increasing the burn-in (NBI)

Figure 29.8 shows a flowchart of the algorithm as it searches for stationarity.

Figure 29.8 Flowchart of the AUTOMCMC Algorithm: Stationarity Analysis



You can initialize NMC=M, NBI=B, and NTU=T during the stationarity phase by specifying NMC, NBI, and NTU as options in the BAYES statement. You can also change the minimum stationarity acceptance ratio of successful stationarity tests that are needed to exit the stationarity phase. By default, TOL=0.95. For example:

```
proc qlim data=dataset;
   ...;
   bayes nmc=M nbi=B ntu=T automcmc=( stationarity=(tol=0.95) );
   ...;
run;
```

During the accuracy phase, the algorithm attempts to determine how many posterior samples are needed. The number of attempts is determined by the option ATTEMPTS=*number*. You can choose between two different approaches to study the accuracy:

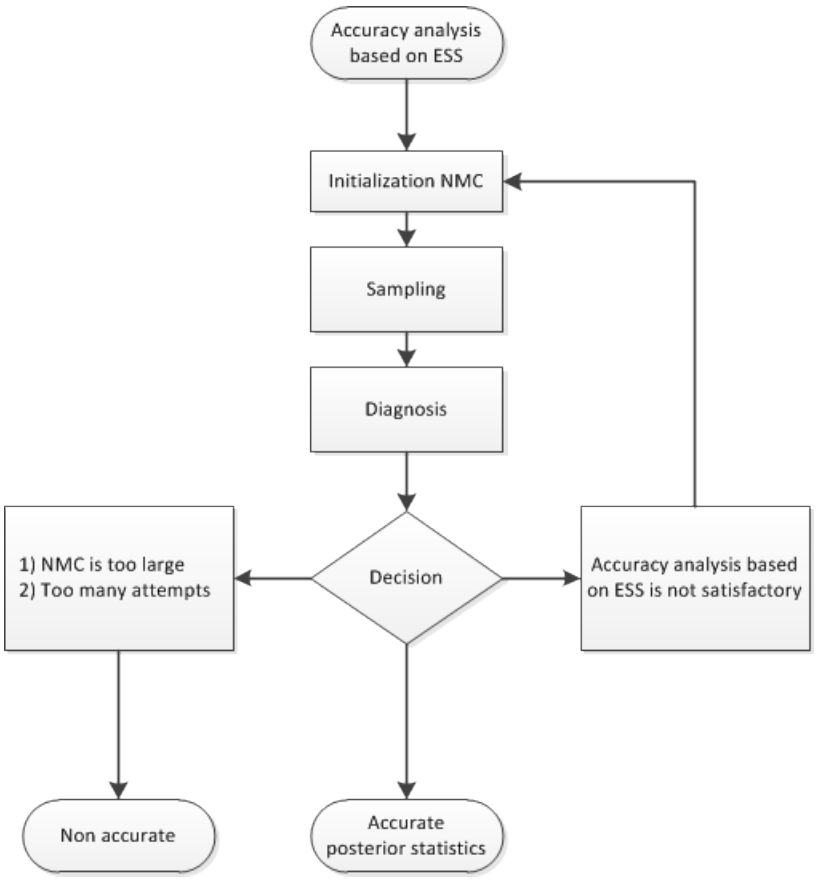
- accuracy analysis based on the effective sample size (ESS)
- accuracy analysis based on the Heidelberger-Welch half-test and the Raftery-Lewis test

If you choose the effective sample size approach, you must provide the minimum number of effective samples that are needed. You can also change the tolerance for the ESS accuracy analysis (by default, TOL=0.95). For example:

```
proc qlim data=dataset;
   ...;
   bayes automcmc=(targetess=N accuracy=(tol=0.95));
   ...;
run;
```

Figure 29.9 shows a flowchart of the algorithm based on the effective sample size approach to determine whether the samples provide an accurate representation of the posterior distribution.

Figure 29.9 Flowchart of the AUTOMCMC Algorithm: Accuracy Analysis Based on the ESS



If you choose the accuracy analysis based on the Heidelberger-Welch half-test and the Raftery-Lewis test (the default option), then you might want to choose a posterior quantile of interest for the Raftery-Lewis test (by default, 0.025). You can also change the tolerance for the accuracy analysis (by default, TOL=0.95). Notice that the Raftery-Lewis test produces a proxy of the number of posterior sample required. In each attempt, the current number of posterior samples is compared to this proxy. If the proxy is greater than the current nmc, then the algorithm reinitializes itself. To control this reinitialization, you can use the option RLLIMITS=(LB=lb UB=ub). In particular, there are three cases

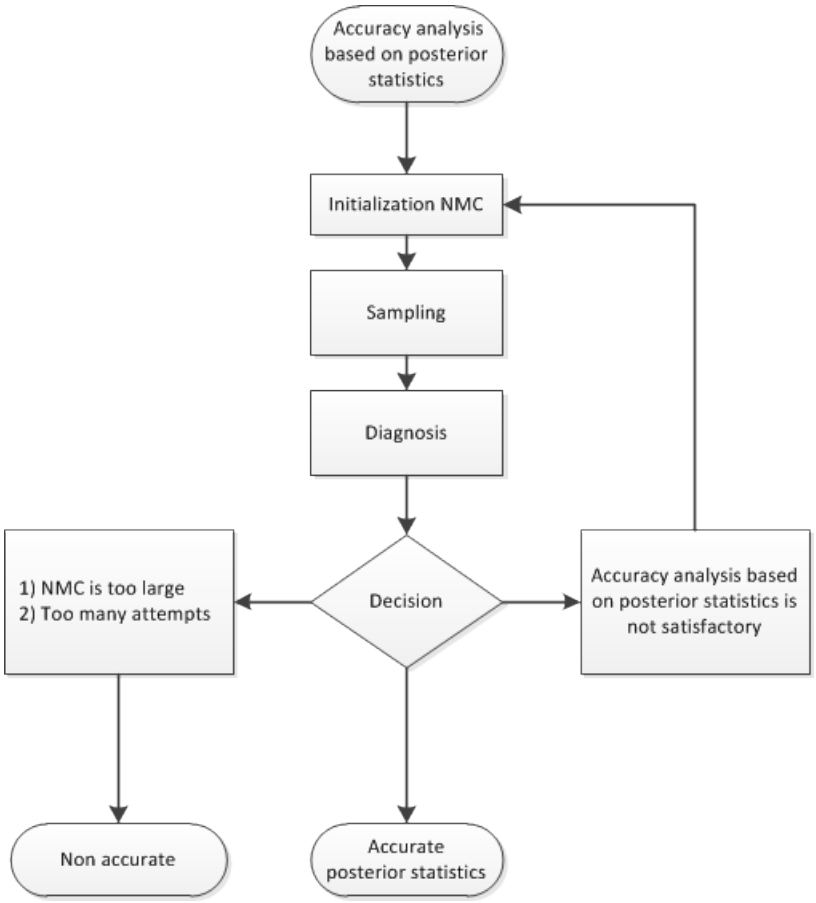
- if the proxy is greater than ub, then NMC is set equal to ub,
- if the proxy is less than lb, then NMC is set equal to lb,
- if lb is less than the proxy, which is less than ub, then NMC is set equal to the proxy.

For example:

```
proc qlim data=dataset;
   ...;
   bayes automcmc=( accuracy=(tol=0.95 targetstats=(rllimits=(lb=k1 ub=k2))) )
         raftery(q=0.025);
   ...;
run;
```

Figure 29.10 shows a flowchart of the algorithm based on the Heidelberger-Welch half-test and the Raftery-Lewis test approach to determine whether the posterior samples provide an accurate representation of the posterior distribution.

Figure 29.10 Flowchart of the AUTOMCMC Algorithm: Accuracy Analysis Based on the Heidelberger-Welch Half-test and the Raftery-Lewis Test



Prior Distributions

The PRIOR statement is used to specify the prior distribution of the model parameters. You must specify a list of parameters, a tilde ~, and then a distribution with its parameters. You can specify multiple PRIOR statements to define independent priors. Parameters that are associated with a regressor variable are referred to by the name of the corresponding regressor variable.

You can specify the special keyword _REGRESSORS to consider all the regressors of a model. If multiple prior statements affect the same parameter, the prior that is specified is used. For example, in a regression with three regressors (X1, X2, X3) the following statements imply that the prior on X1 is NORMAL(MEAN=0, VAR=1), the prior on X2 is GAMMA(SHAPE=3, SCALE=4), and the prior on X3 is UNIFORM(MIN=0, MAX=1):

```
...
prior _Regressors ~ uniform(min=0, max=1);
```

```
prior X1 X2 ~ gamma(shape=3, scale=4);
prior X1 ~ normal(mean=0, var=1);
...
```

If a parameter is not associated with a PRIOR statement or if some of the prior hyperparameters are missing, then the following default choices are considered:

Table 29.2 Default values for prior distributions.

PRIOR *distribution*	Hyperparameter$_1$	Hyperparameter$_2$	Min	Max	Parameters Default Choice
NORMAL	MEAN=0	VAR=1E6	$-\infty$	∞	Regression-Location-Threshold
IGAMMA	SHAPE=2.000001	SCALE=1	> 0	∞	Scale
SQIGAMMA	SHAPE=2.000001	SCALE=1	> 0	∞	Scale
GAMMA	SHAPE=1	SCALE=1	0	∞	
SQGAMMA	SHAPE=1	SCALE=1	0	∞	
UNIFORM			$-\infty$	∞	
UNIFORM			> -1	< 1	Cross-correlation
BETA	SHAPE1=1	SHAPE2=1	$-\infty$	∞	
T	LOCATION=0	DF=3	$-\infty$	∞	

See the section "Standard Distributions" on page 2024 for density specification.

Priors for Heteroscedastic Models

The choice of the prior distribution for a heteroscedastic model is particularly interesting. Based on the notation provided in section "HETERO Statement" on page 1975, you need to provide a prior for $\boldsymbol{\gamma}$. This prior is enough to induce different σ_i^2 into the analysis. The resulting inference is a compromise between two cases: the inference based on the entire sample and the inference based on a single unit $\mathbf{z}_i$. The degree of compromise is determined by $\pi(\boldsymbol{\gamma})$.

This type of modeling is similar to a method called "hierarchical Bayes," in which the prior is characterized by two levels: one for each individual $\pi(\sigma_i^2|\boldsymbol{\gamma})$ and one for the entire population $\pi(\boldsymbol{\gamma})$. In this scenario the degree of compromise between the information provided by a unit and the information provided by the entire sample is determined by the data.

The choice of the prior might not be straightforward, and it can heavily affect sampling performance. Depending on how the heteroscedastic effects are modeled, the default priors are

$$\text{if } \left[1+\exp(\mathbf{z}_i'\boldsymbol{\gamma})\right], \quad \pi(\gamma_j) = \textbf{normal}\left\{\text{mean} = \frac{1}{\bar{z}_j J}\left[\log\left(\frac{\varepsilon^4}{1+\varepsilon^2}\right)\right], \text{var} = \frac{1}{\bar{z}_j^2 J}\left[\log\left(\frac{1+\varepsilon^2}{\varepsilon^2}\right)\right]\right\}$$

$$\text{if } \left[\exp(\mathbf{z}_i'\boldsymbol{\gamma})\right], \quad \pi(\gamma_j) = \textbf{normal}\left\{\text{mean} = \frac{1}{\bar{z}_j J}\left[\log\left(\frac{1}{2}\right)\right], \text{var} = \frac{1}{\bar{z}_j^2 J}\left[\log(2)\right]\right\}$$

$$\text{if } \left(1+\mathbf{z}_i'\boldsymbol{\gamma}\right), \quad \pi(\gamma_j) = \textbf{normal}\left\{\text{mean} = 0, \text{var} = \frac{1}{\bar{z}_j^2 J}\right\}$$

$$\text{if } \left[1+(\mathbf{z}_i'\boldsymbol{\gamma})^2\right], \quad \pi(\gamma_j) = \textbf{normal}\left\{\text{mean} = \frac{(\varepsilon^2-1/2)^{1/4}}{\bar{z}_j J}, \text{var} = \frac{\varepsilon-(\varepsilon^2-1/2)^{1/2}}{\bar{z}_j^2 J}\right\}$$

where $\bar{z}_j = \frac{1}{n}\sum_{i=1}^{n} z_{ij}, \forall j$, and ε is a small number (by default, $\varepsilon = 0.1$ for the EXPONENTIAL link function and $\varepsilon = 0.71$ for the QUADRATIC link function).

The priors for the EXPONENTIAL and QUADRATIC link functions are not straightforward. To understand the choices, do the following:

1. Assume that
$$\mathbf{z}_i'\boldsymbol{\gamma} = z_{i1}\gamma_1 + \ldots + z_{iJ}\gamma_J \approx \bar{z}_1\gamma_1 + \ldots + \bar{z}_J\gamma_J, \qquad \forall i$$

2. Set the priors according to the link function type:

   - For the EXPONENTIAL link function, set
   $$\begin{aligned} \mathrm{E}\left[\exp(\mathbf{z}_i'\boldsymbol{\gamma})\right] &\approx \mathrm{E}\left[\exp(\bar{z}_1\gamma_1)\right] \times \ldots \times \mathrm{E}\left[\exp(\bar{z}_J\gamma_J)\right] = \varepsilon \\ \mathrm{V}\left[\exp(\mathbf{z}_i'\boldsymbol{\gamma})\right] &\approx \mathrm{E}\left[\exp(2\bar{z}_1\gamma_1)\right] \times \ldots \times \mathrm{E}\left[\exp(2\bar{z}_J\gamma_J)\right] - \varepsilon^2 = 1 \end{aligned}$$
   Assume a normal prior for $\pi(\gamma_j)$, and set
   $$\begin{aligned} \mathrm{E}\left[\exp(\bar{z}_j\gamma_j)\right] &= \varepsilon^{\frac{1}{J}}, \forall j \\ \mathrm{E}\left[\exp(2\bar{z}_j\gamma_j)\right] &= (1+\varepsilon^2)^{\frac{1}{J}}, \forall j \end{aligned}$$
   Based on the properties of the lognormal distribution, the prior hyperparameters for γ_j can be derived. Notice that J is the number of regressors that are used in the heterogeneous regression. If the intercept is excluded, then $\varepsilon = 1$.

   - For the QUADRATIC link function, set
   $$\begin{aligned} \mathrm{E}\left[(\mathbf{z}_i'\boldsymbol{\gamma})^2\right] &\approx \left[\mathrm{E}\left(\bar{z}_1\gamma_1 + \ldots + \bar{z}_J\gamma_J\right)\right]^2 + \mathrm{V}\left[\bar{z}_1\gamma_1 + \ldots + \bar{z}_J\gamma_J\right] = \varepsilon \\ \mathrm{V}\left[(\mathbf{z}_i'\boldsymbol{\gamma})^2\right] &\approx \mathrm{E}\left[(\bar{z}_1\gamma_1 + \ldots + \bar{z}_J\gamma_J)^4\right] - \varepsilon^2 = 1 \end{aligned}$$
   Assume a normal prior for $\pi(\gamma_j)$. Based on the properties of the normal distribution, the preceding expressions return
   $$\begin{aligned} \mathrm{E}\left[\bar{z}_1\gamma_1 + \ldots + \bar{z}_J\gamma_J\right] &= (\varepsilon^2 - 1/2)^{1/4} \\ \mathrm{V}\left[\bar{z}_1\gamma_1 + \ldots + \bar{z}_J\gamma_J\right] &= \varepsilon - (\varepsilon^2 - 1/2)^{1/2} \\ \varepsilon &> (1/2)^{1/2} \end{aligned}$$
   The prior hyperparameters for γ_j can be derived by setting
   $$\begin{aligned} \mathrm{E}\left[\bar{z}_j\gamma_j\right] &= \frac{(\varepsilon^2 - 1/2)^{1/4}}{J}, \forall j \\ \mathrm{V}\left[\bar{z}_j\gamma_j\right] &= \frac{\varepsilon - (\varepsilon^2 - 1/2)^{1/2}}{J}, \forall j \end{aligned}$$
   Notice that J is the number of regressors that are used in the heterogeneous regression. It is important to emphasize that the restriction $\varepsilon > (1/2)^{1/2} \approx 0.71$ is likely to introduce some distortion because ε cannot be any "small" number.

Automated MCMC

The main purpose is to provide the user with the opportunity of obtaining a good approximation of the posterior distribution without initializing the MCMC algorithm: initial values, proposal distributions, burn-in and number of samples.

The automated algorithm is composed of two phases: tuning and sampling. In the tuning phase, there are two main concerns: the choice of a good proposal distribution and the search for the stationary region of the posterior distribution. In the sampling phase, the algorithm will decide how many samples are necessary to obtain good approximations of the posterior mean and some quantiles of interest.

Stationarity Phase

During the stationarity phase, the algorithm tries to search for a good proposal distribution and, at the same time, to reach the stationary region of the posterior. The choice of the proposal distribution is based on the analysis of the acceptance rates. This is similar to what is done in PROC MCMC: for more details see Chapter 73.10, "Tuning the Proposal Distribution" (*SAS/STAT User's Guide*). For the stationarity analysis, the main idea is to run two tests, Geweke (Ge) and Heidleberger-Welch (HW), on the posterior chains at the end of each attempt. For more details, see Chapter 7.4, "Geweke Diagnostics" (*SAS/STAT User's Guide*), and Chapter 7.4, "Heidelberger and Welch Diagnostics" (*SAS/STAT User's Guide*). If the stationarity hypothesis is rejected, then the tuning samples are increased and the tests repeated in the next attempt. After 10 attempts, the stationarity phase will be ended regardless of the results. The tuning parameters for the first attempt are fixed:

1000	burn-in (nbi),
5000	tuning samples (ntu),
1000	MCMC samples (nmc).

For the remaining attempts, the tuning parameters will be adjusted dynamically. More specifically, each parameter will be assigned an acceptance ratio (AR) of the stationarity hypothesis:

$AR_i = 0$	if	both tests reject the stationarity hypothesis,
$AR_i = 0.5$	if	one tests rejects and the other does not,
$AR_i = 1$	if	both tests do not reject the stationarity hypothesis,

for $i = 1, \ldots, k$. For the Geweke test, the implemented significance level is 0.05. Then, an overall stationarity average (SA) for all parameters ratios is evaluated,

$$SA = \sum_{i=1}^{k} \frac{AR_i}{k}, \tag{29.1}$$

and the number of tuning samples is updated accordingly:

ntu = ntu + 2000	if	$SA < 70\%$,
ntu = ntu + 1000	if	$70\% \leq SA < 100\%$,
ntu = ntu	if	$SA = 100\%$.

The Heidelberger-Welch test also provides an indications of how much burn-in should be used. The algorithm requires this burn-in to be: nbi(HW) $= 0$. If that is not the case, the burn-in will updated accordingly,

nbi = nbi + nbi(HW), and a new attempt searching for stationarity will be implemented. This choice is motivated by the fact that the burn-in must be discarded in order to reach the stationary region of the posterior distribution.

The number of samples is updated at each attempt. However, in order to exit the stationarity phase, it will not be required nmc(RL) = 0. The default update is nmc = nmc + 1000. Depending on the outcome of the Raftery-Lewis diagnostics, if nmc $< \min \{\text{LB [nmc(RL)]}, \text{nmc(RL)}\}$, the number of sampling is further updated to nmc = LB [nmc(RL)]. By default, LB [nmc(RL)] = 10000. Finally, if the number of projected samples is not sufficient to perform a stable evaluation of the Raftery-Lewis test, the number of samples is updated to nmc = $\min$ [nmc(RL)]. For more details see "AUTOMCMC <=(*automcmc-options*)>" on page 1964 and Chapter 7.4, "Raftery and Lewis Diagnostics" (*SAS/STAT User's Guide*).

Accuracy Phase

The main idea of the accuracy phase is to make sure that the mean and a quantile of interest are evaluated accurately. This can be tested by implementing the half-width test by Heidelberger-Welch and by analyzing the Raftery-Lewis diagnostic tool. In addition, the requirements defined in the stationarity phase will also be checked: the Geweke and the Heidelberger-Welch tests must not reject the stationary hypothesis and the burn-in predicted by the Heidelberger-Welch test must be zero.

The accuracy phase is characterized by a maximum of 10 attempts. If the algorithm exceeds this limit, the accuracy phase will end and indications on how to improve sampling will be given. The search of accuracy can be performed using two different method. The first method (the default) is triggered by the option TARGETSTATS and it is based on the accuracy analysis of the mean and a percentile of interest. The secong method is triggered by the option TARGETESS and it targets a minimum number of effective samples. The accuracy phase will first update the burn-in with the information provided by the HW test: nbi = nbi + nbi(HW). Then, it determines the difference between the actual number of samples and the number of samples predicted by either the RL test or the ESS: Δ[nmc] = nmc(RL) − nmc, or Δ[nmc] = nmc(ESS) − nmc. The new number of samples will be updated accordingly:

$$\begin{aligned}
&\text{nmc} = \text{nmc} + \text{LB [nmc(RL)]} \quad \text{if} \quad 0 < \Delta[\text{nmc}] \leq \text{LB [nmc(RL)]},\\
&\text{nmc} = \text{nmc} + \Delta[\text{nmc}] \quad \text{if} \quad \text{LB [nmc(RL)]} < \Delta[\text{nmc}] \leq \text{UB [nmc(RL)]},\\
&\text{nmc} = \text{nmc} + \text{UB [nmc(RL)]} \quad \text{if} \quad \text{UB [nmc(RL)]} < \Delta[\text{nmc}].
\end{aligned}$$

By default, LB [nmc(RL)] = 10000 and UB [nmc(RL)] = 300000.

In addition, the accuracy search triggered by the option TARGETSTATS also implements the HW half-width test to checks whether the sample mean is accurate. If the mean of any parameters is not considered to be accurate and the number of samples has not been updated based on Δ[nmc], then the number of samples is increased:

$$\text{nmc} = \text{nmc} + 5000 \quad \text{if} \quad \Delta[\text{nmc}] \leq 0,$$

Marginal Likelihood

The Bayes theorem states that

$$p(\theta|\mathbf{y}) \propto \pi(\theta)L(y|\theta)$$

where θ is a vector of parameters and $\pi(\theta)$ is the product of the prior densities that are specified in the PRIOR statement. The term $L(y|\theta)$ is the likelihood that is associated with the MODEL statement. The function $\pi(\theta)L(y|\theta)$ is the nonnormalized posterior distribution over the parameter vector θ. The normalized posterior distribution (simply, the posterior distribution) is

$$p(\theta|\mathbf{y}) = \frac{\pi(\theta)L(y|\theta)}{\int_\theta \pi(\theta)L(y|\theta)d\theta}$$

The denominator $m(y) = \int_\theta \pi(\theta)L(y|\theta)d\theta$ (also called the "marginal likelihood") is a quantity of interest because it represents the probability of the data after the effect of the parameter vector has been averaged out. Because of its interpretation, the marginal likelihood can be used in various applications, including model averaging, variable selection, and model selection.

A natural estimate of the marginal likelihood is provided by the harmonic mean,

$$m(y) = \left\{ \frac{1}{n} \sum_{i=1}^{n} \frac{1}{L(y|\theta_i)} \right\}^{-1}$$

where θ_i is a sample draw from the posterior distribution. In practical applications, this estimator has proven to be unstable.

An alternative and more stable estimator can be obtained with an importance sampling scheme. The auxiliary distribution for the importance sampler can be chosen through the cross entropy theory (Chan and Eisenstat 2015). In particular, given a parametric family of distributions, the auxiliary density function is chosen to be the one closest, in terms of the Kullback-Leibler divergence, to the probability density that would give a zero variance estimate of the marginal likelihood. In practical terms, this is equivalent to the following algorithm:

1. Choose a parametric family, $f(., \beta)$, for the parameters of the model: $f(\theta|\beta)$.
2. Evaluate the maximum likelihood estimator of β by using the posterior samples $\theta_1, \ldots, \theta_n$ as data.
3. Use $f(\theta^*|\hat{\beta}_{mle})$ to generate the importance samples $\theta_1^*, \ldots, \theta_{n^*}^*$.
4. Estimate the marginal likelihood:

$$m(y) = \frac{1}{n^*} \sum_{j=1}^{n^*} \frac{L(y|\theta_j^*)\pi(\theta_j^*)}{f(\theta_j^*|\hat{\beta}_{mle})}$$

The parametric family for the auxiliary distribution is chosen to be Gaussian. The parameters that are subject to bounds are transformed accordingly

- If $-\infty < \theta < \infty$, then $p = \theta$.
- If $m \leq \theta < \infty$, then $q = \log(\theta - m)$.
- If $-\infty < \theta \leq M$, then $r = \log(M - \theta)$.
- If $m \leq \theta \leq M$, then $s = \log(\theta - m) - \log(M - \theta)$.

Assuming independence for the parameters that are subject to bounds, the auxiliary distribution to generate importance samples is

$$\begin{pmatrix} \mathbf{p} \\ \mathbf{q} \\ \mathbf{r} \\ \mathbf{s} \end{pmatrix} \sim \mathbf{N}\left[\begin{pmatrix} \mu_p \\ \mu_q \\ \mu_r \\ \mu_s \end{pmatrix}, \begin{pmatrix} \Sigma_p & 0 & 0 & 0 \\ 0 & \Sigma_q & 0 & 0 \\ 0 & 0 & \Sigma_r & 0 \\ 0 & 0 & 0 & \Sigma_r \end{pmatrix} \right]$$

where $\mathbf{p}$, $\mathbf{q}$, $\mathbf{r}$, and $\mathbf{s}$ are vectors that contain the transformations of the unbounded, bounded-below, bounded-above, and bounded-above-and-below parameters. Also, given the imposed independence structure, Σ_p can be a nondiagonal matrix, but Σ_q, Σ_r, and Σ_s are assumed to be diagonal matrices.

Standard Distributions

Table 29.3 through Table 29.10 show all the distribution density functions that PROC QLIM recognizes. You specify these distribution densities in the PRIOR statement.

Table 29.3 Beta Distribution

PRIOR statement	BETA(SHAPE1=a, SHAPE2=b, MIN=m, MAX=M) Note: Commonly $m = 0$ and $M = 1$.
Density	$\frac{(\theta-m)^{a-1}(M-\theta)^{b-1}}{B(a,b)(M-m)^{a+b-1}}$
Parameter restriction	$a > 0, \quad b > 0, \quad -\infty < m < M < \infty$
Range	$\begin{cases} [m, M] & \text{when } a = 1, b = 1 \\ [m, M) & \text{when } a = 1, b \neq 1 \\ (m, M] & \text{when } a \neq 1, b = 1 \\ (m, M) & \text{otherwise} \end{cases}$
Mean	$\frac{a}{a+b} \times (M - m) + m$
Variance	$\frac{ab}{(a+b)^2(a+b+1)} \times (M - m)^2$
Mode	$\begin{cases} \frac{a-1}{a+b-2} \times M + \frac{b-1}{a+b-2} \times m & a > 1, b > 1 \\ m \text{ and } M & a < 1, b < 1 \\ m & \begin{cases} a < 1, b \geq 1 \\ a = 1, b > 1 \end{cases} \\ M & \begin{cases} a \geq 1, b < 1 \\ a > 1, b = 1 \end{cases} \\ \text{not unique} & a = b = 1 \end{cases}$
Defaults	SHAPE1=SHAPE2=1, MIN $\rightarrow -\infty$, MAX $\rightarrow \infty$

Table 29.4 Gamma Distribution

PRIOR statement	GAMMA(SHAPE=a, SCALE=b)
Density	$\frac{1}{b^a\Gamma(a)}\theta^{a-1}e^{-\theta/b}$
Parameter restriction	$a > 0, b > 0$
Range	$[0, \infty)$
Mean	ab
Variance	ab^2
Mode	$(a-1)b$
Defaults	SHAPE=SCALE=1

Table 29.5 Square Root Gamma Distribution

PRIOR statement	SQGAMMA(SHAPE=a, SCALE=b)
Density	$\frac{2}{b^a\Gamma(a)}\theta^{2a-1}e^{-\theta^2/b}$
Parameter restriction	$a > 0, b > 0$
Range	$[0, \infty)$
Mean	$\frac{\Gamma(a+\frac{1}{2})}{\Gamma(a)}\sqrt{b}$
Variance	$\left\{a - \left[\frac{\Gamma(a+\frac{1}{2})}{\Gamma(a)}\right]^2\right\} b$
Mode	$\sqrt{(a-\frac{1}{2})b}, \qquad a \geq \frac{1}{2}$
Defaults	SHAPE=SCALE=1

See Stacy (1962) for more details.

Table 29.6 Inverse-Gamma Distribution

PRIOR statement	IGAMMA(SHAPE=a, SCALE=b)
Density	$\frac{b^a}{\Gamma(a)}\theta^{-(a+1)}e^{-b/\theta}$
Parameter restriction	$a > 0, b > 0$
Range	$0 < \theta < \infty$
Mean	$\frac{b}{a-1}, \qquad a > 1$
Variance	$\frac{b^2}{(a-1)^2(a-2)}, \qquad a > 2$
Mode	$\frac{b}{a+1}$
Defaults	SHAPE=2.000001, SCALE=1

Table 29.7 Square Root Inverse-Gamma Distribution

PRIOR statement	SQIGAMMA(SHAPE=a, SCALE=b)
Density	$\frac{2b^a}{\Gamma(a)}\theta^{-(2a+1)}e^{-b/\theta^2}$
Parameter restriction	$a > 0, b > 0$
Range	$0 < \theta < \infty$
Mean	$\frac{\Gamma(a-\frac{1}{2})}{\Gamma(a)}\sqrt{b}, \quad a > \frac{1}{2}$
Variance	$\left\{\frac{1}{a-1} - \left[\frac{\Gamma(a-\frac{1}{2})}{\Gamma(a)}\right]^2\right\} b, \quad a > 1$
Mode	$\sqrt{\frac{b}{a+\frac{1}{2}}}$
Defaults	SHAPE=2.000001, SCALE=1

See Stacy (1962) for more details.

Table 29.8 Normal Distribution

PRIOR statement	NORMAL(MEAN=μ, VAR=σ^2)
Density	$\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(\theta-\mu)^2}{2\sigma^2}\right)$
Parameter restriction	$\sigma^2 > 0$
Range	$-\infty < \theta < \infty$
Mean	μ
Variance	σ^2
Mode	μ
Defaults	MEAN=0, VAR=1000000

Table 29.9 *t* Distribution

PRIOR statement	T(LOCATION=μ, DF=ν)
Density	$\frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)\sqrt{\pi\nu}}\left[1 + \frac{(\theta-\mu)^2}{\nu}\right]^{-\frac{\nu+1}{2}}$
Parameter restriction	$\nu > 0$
Range	$-\infty < \theta < \infty$
Mean	μ, for $\nu > 1$
Variance	$\frac{\nu}{\nu-2}$, for $\nu > 2$
Mode	μ
Defaults	LOCATION=0, DF=3

Table 29.10 Uniform Distribution

PRIOR statement	UNIFORM(MIN=m, MAX=M)
Density	$\frac{1}{M-m}$
Parameter restriction	$-\infty < m < M < \infty$
Range	$\theta \in [m, M]$
Mean	$\frac{m+M}{2}$
Variance	$\frac{(M-m)^2}{12}$
Mode	Not unique
Defaults	MIN$\rightarrow -\infty$, MAX$\rightarrow \infty$

Output to SAS Data Set

XBeta, Predicted, Residual

Xbeta is the structural part on the right-hand side of the model. Predicted value is the predicted dependent variable value. For censored variables, if the predicted value is outside the boundaries, it is reported as the closest boundary. For discrete variables, it is the level whose boundaries Xbeta falls between. Residual is defined only for continuous variables and is defined as

$$Residual = Observed - Predicted$$

Error Standard Deviation

Error standard deviation is σ_i in the model. It varies only when the HETERO statement is used.

Marginal Effects

Marginal effect is defined as a contribution of one control variable to the response variable. For the binary choice model with two response categories, $\mu_0 = -\infty$, $\mu_1 = 0$, $\mu_2 = \infty$; and ordinal response model with M response categories, $\mu_0, \cdots, \mu_M$, define

$$R_{i,j} = \mu_j - \mathbf{x}_i'\boldsymbol{\beta}$$

The probability that the unobserved dependent variable is contained in the jth category can be written as

$$P[\mu_{j-1} < y_i^* \leq \mu_j] = F(R_{i,j}) - F(R_{i,j-1})$$

The marginal effect of changes in the regressors on the probability of $y_i = j$ is then

$$\frac{\partial Prob[y_i = j]}{\partial \mathbf{x}} = [f(\mu_{j-1} - \mathbf{x}_i'\boldsymbol{\beta}) - f(\mu_j - \mathbf{x}_i'\boldsymbol{\beta})]\boldsymbol{\beta}$$

where $f(x) = \frac{dF(x)}{dx}$. In particular,

$$f(x) = \frac{dF(x)}{dx} = \begin{cases} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} & \text{(probit)} \\ \frac{e^{-x}}{[1+e^{(-x)}]^2} & \text{(logit)} \end{cases}$$

The marginal effects in the Box-Cox regression model are

$$\frac{\partial E[y_i]}{\partial \mathbf{x}} = \boldsymbol{\beta} \frac{x^{\lambda_k - 1}}{y^{\lambda_0 - 1}}$$

The marginal effects in the truncated regression model are

$$\frac{\partial E[y_i | L_i < y_i^* < R_i]}{\partial \mathbf{x}} = \boldsymbol{\beta} \left[1 - \frac{(\phi(a_i) - \phi(b_i))^2}{(\Phi(b_i) - \Phi(a_i))^2} + \frac{a_i \phi(a_i) - b_i \phi(b_i)}{\Phi(b_i) - \Phi(a_i)} \right]$$

where $a_i = \frac{L_i - \mathbf{x}_i' \boldsymbol{\beta}}{\sigma_i}$ and $b_i = \frac{R_i - \mathbf{x}_i' \boldsymbol{\beta}}{\sigma_i}$.

The marginal effects in the censored regression model are

$$\frac{\partial E[y | \mathbf{x}_i]}{\partial \mathbf{x}} = \boldsymbol{\beta} \times Prob[L_i < y_i^* < R_i]$$

Inverse Mills Ratio, Expected and Conditionally Expected Values

Expected and conditionally expected values are computed only for continuous variables. The inverse Mills ratio is computed for censored or truncated continuous, binary discrete, and selection endogenous variables.

Let L_i and R_i be the lower boundary and upper boundary, respectively, for the y_i. Define $a_i = \frac{L_i - \mathbf{x}_i' \boldsymbol{\beta}}{\sigma_i}$ and $b_i = \frac{R_i - \mathbf{x}_i' \boldsymbol{\beta}}{\sigma_i}$. Then the inverse Mills ratio is defined as

$$\lambda = \frac{(\phi(a_i) - \phi(b_i))}{(\Phi(b_i) - \Phi(a_i))}$$

for a continuous variable and defined as

$$\lambda = \frac{\phi(\mathbf{x}_i' \boldsymbol{\beta})}{\Phi(\mathbf{x}_i' \boldsymbol{\beta})}$$

for a binary discrete variable.

The expected value is the unconditional expectation of the dependent variable. For a censored variable, it is

$$E[y_i] = \Phi(a_i) L_i + (\mathbf{x}_i' \boldsymbol{\beta} + \lambda \sigma_i)(\Phi(b_i) - \Phi(a_i)) + (1 - \Phi(b_i)) R_i$$

For a left-censored variable ($R_i = \infty$), this formula is

$$E[y_i] = \Phi(a_i) L_i + (\mathbf{x}_i' \boldsymbol{\beta} + \lambda \sigma_i)(1 - \Phi(a_i))$$

where $\lambda = \frac{\phi(a_i)}{1 - \Phi(a_i)}$.

For a right-censored variable ($L_i = -\infty$), this formula is

$$E[y_i] = (\mathbf{x}_i'\boldsymbol{\beta} + \lambda\sigma_i)\Phi(b_i) + (1 - \Phi(b_i))R_i$$

where $\lambda = -\frac{\phi(b_i)}{\Phi(b_i)}$.

For a noncensored variable, this formula is

$$E[y_i] = \mathbf{x}_i'\boldsymbol{\beta}$$

The conditional expected value is the expectation given that the variable is inside the boundaries:

$$E[y_i | L_i < y_i < R_i] = \mathbf{x}_i'\boldsymbol{\beta} + \lambda\sigma_i$$

Probability

Probability applies only to discrete responses. It is the marginal probability that the discrete response is taking the value of the observation. If the PROBALL option is specified, then the probability for all of the possible responses of the discrete variables is computed.

Technical Efficiency

Technical efficiency for each producer is computed only for stochastic frontier models.

In general, the stochastic production frontier can be written as

$$y_i = f(x_i;\beta)\exp\{v_i\}TE_i$$

where y_i denotes producer i's actual output, $f(\cdot)$ is the deterministic part of production frontier, $\exp\{v_i\}$ is a producer-specific error term, and TE_i is the technical efficiency coefficient, which can be written as

$$TE_i = \frac{y_i}{f(x_i;\beta)\exp\{v_i\}}.$$

In the case of a Cobb-Douglas production function, $TE_i = \exp\{-u_i\}$. See the section "Stochastic Frontier Production and Cost Models" on page 1991.

Cost frontier can be written in general as

$$E_i = c(y_i, w_i;\beta)\exp\{v_i\}/CE_i$$

where w_i denotes producer i's input prices, $c(\cdot)$ is the deterministic part of cost frontier, $\exp\{v_i\}$ is a producer-specific error term, and CE_i is the cost efficiency coefficient, which can be written as

$$CE_i = \frac{c(x_i, w_i;\beta)\exp\{v_i\}}{E_i}$$

In the case of a Cobb-Douglas cost function, $CE_i = \exp\{-u_i\}$. See the section "Stochastic Frontier Production and Cost Models" on page 1991. Hence, both technical and cost efficiency coefficients are the same. The estimates of technical efficiency are provided in the following subsections.

Normal-Half Normal Model

Define $\mu_* = -\epsilon\sigma_u^2/\sigma^2$ and $\sigma_*^2 = \sigma_u^2\sigma_v^2/\sigma^2$. Then, as it is shown by Jondrow et al. (1982), conditional density is as follows:

$$f(u|\epsilon) = \frac{f(u,\epsilon)}{f(\epsilon)} = \frac{1}{\sqrt{2\pi}\sigma_*}\exp\left\{-\frac{(u-\mu_*)^2}{2\sigma_*^2}\right\} \Big/ \left[1-\Phi\left(-\frac{\mu_*}{\sigma_*}\right)\right]$$

Hence, $f(u|\epsilon)$ is the density for $N^+(\mu_*, \sigma_*^2)$.

Using this result, it follows that the estimate of technical efficiency (Battese and Coelli 1988) is

$$TE1_i = E(\exp\{-u_i\}|\epsilon_i) = \left[\frac{1-\Phi(\sigma_* - \mu_{*i}/\sigma_*)}{1-\Phi(-\mu_{*i}/\sigma_*)}\right]\exp\left\{-\mu_{*i}+\frac{1}{2}\sigma_*^2\right\}$$

The second version of the estimate (Jondrow et al. 1982) is

$$TE2_i = \exp\{-E(u_i|\epsilon_i)\}$$

where

$$E(u_i|\epsilon_i) = \mu_{*i} + \sigma_*\left[\frac{\phi(-\mu_{*i}/\sigma_*)}{1-\Phi(-\mu_{*i}/\sigma_*)}\right] = \sigma_*\left[\frac{\phi(\epsilon_i\lambda/\sigma)}{1-\Phi(\epsilon_i\lambda/\sigma)} - \left(\frac{\epsilon_i\lambda}{\sigma}\right)\right]$$

Normal-Exponential Model

Define $A = -\tilde{\mu}/\sigma_v$ and $\tilde{\mu} = -\epsilon - \sigma_v^2/\sigma_u$. Then, as it is shown by Kumbhakar and Lovell (2000), conditional density is as follows:

$$f(u|\epsilon) = \frac{1}{\sqrt{2\pi}\sigma_v\Phi(-\tilde{\mu}/\sigma_v)}\exp\left\{-\frac{(u-\tilde{\mu})^2}{2\sigma^2}\right\}$$

Hence, $f(u|\epsilon)$ is the density for $N^+(\tilde{\mu}, \sigma_v^2)$.

Using this result, it follows that the estimate of technical efficiency is

$$TE1_i = E(\exp\{-u_i\}|\epsilon_i) = \left[\frac{1-\Phi(\sigma_v - \tilde{\mu}_i/\sigma_v)}{1-\Phi(-\tilde{\mu}_i/\sigma_v)}\right]\exp\left\{-\tilde{\mu}_i+\frac{1}{2}\sigma_v^2\right\}$$

The second version of the estimate is

$$TE2_i = \exp\{-E(u_i|\epsilon_i)\}$$

where

$$E(u_i|\epsilon_i) = \tilde{\mu}_i + \sigma_v\left[\frac{\phi(-\tilde{\mu}_i/\sigma_v)}{1-\Phi(-\tilde{\mu}_i/\sigma_v)}\right] = \sigma_v\left[\frac{\phi(A)}{\Phi(-A)} - A\right]$$

Normal-Truncated Normal Model

Define $\tilde{\mu} = (-\sigma_u^2\epsilon_i + \mu\sigma_v^2)/\sigma^2$ and $\sigma_*^2 = \sigma_u^2\sigma_v^2/\sigma^2$. Then, as it is shown by Kumbhakar and Lovell (2000), conditional density is as follows:

$$f(u|\epsilon) = \frac{1}{\sqrt{2\pi}\sigma_*[1-\Phi(-\tilde{\mu}/\sigma_*)]}\exp\left\{-\frac{(u-\tilde{\mu})^2}{2\sigma_*^2}\right\}$$

Hence, $f(u|\epsilon)$ is the density for $N^+(\tilde{\mu}, \sigma_*^2)$.

Using this result, it follows that the estimate of technical efficiency is

$$TE1_i = E(\exp\{-u_i\}|\epsilon_i) = \frac{1-\Phi(\sigma_* - \tilde{\mu}_i/\sigma_*)}{1-\Phi(-\tilde{\mu}_i/\sigma_*)} \exp\left\{-\tilde{\mu}_i + \frac{1}{2}\sigma_*^2\right\}$$

The second version of the estimate is

$$TE2_i = \exp\{-E(u_i|\epsilon_i)\}$$

where

$$E(u_i|\epsilon_i) = \tilde{\mu}_i + \sigma_* \left[\frac{\phi(\tilde{\mu}_i/\sigma_*)}{1-\Phi(-\tilde{\mu}_i/\sigma_*)}\right]$$

OUTEST= Data Set

The OUTEST= data set contains all the parameters estimated in a MODEL statement. The OUTEST= option can be used when the PROC QLIM call contains one MODEL statement:

```
proc qlim data=a outest=e;
   model y = x1 x2 x3;
   endogenous y ~ censored(lb=0);
run;
```

Each parameter contains the estimate for the corresponding parameter in the corresponding model. In addition, the OUTEST= data set contains the following variables:

NAME	the name of the independent variable
TYPE	type of observation. PARM indicates the row of coefficients; STD indicates the row of standard deviations of the corresponding coefficients.
STATUS	convergence status for optimization

The rest of the columns correspond to the explanatory variables.

The OUTEST= data set contains one observation for the MODEL statement, giving the parameter estimates for that model. If the COVOUT option is specified, the OUTEST= data set includes additional observations for the MODEL statement, giving the rows of the covariance matrix of parameter estimates. For covariance observations, the value of the _TYPE_ variable is COV, and the _NAME_ variable identifies the parameter associated with that row of the covariance matrix. If the CORROUT option is specified, the OUTEST= data set includes additional observations for the MODEL statement, giving the rows of the correlation matrix of parameter estimates. For correlation observations, the value of the _TYPE_ variable is CORR, and the _NAME_ variable identifies the parameter associated with that row of the correlation matrix.

Naming

Naming of Parameters

When there is only one equation in the estimation, parameters are named in the same way as in other SAS procedures such as REG, PROBIT, and so on. The constant in the regression equation is called Intercept. The coefficients on independent variables are named by the independent variables. The standard deviation of the errors is called _Sigma. If there are Box-Cox transformations, the coefficients are named _Lambda*i*, where *i* increments from 1, or as specified by the user. The limits for the discrete dependent variable are named _Limit*i*. If the LIMIT=varying option is specified, then _Limit*i* starts from 1. If the LIMIT=varying option is not specified, then _Limit1 is set to 0 and the limit parameters start from $i = 2$. If the HETERO statement is included, the coefficients of the independent variables in the hetero equation are called _H.*x*, where *x* is the name of the independent variable. If the parameter name includes interaction terms, it needs to be enclosed in quotation marks followed by *N*. The following example restricts the parameter that includes the interaction term to be greater than zero:

```
proc qlim data=a;
   model y = x1|x2;
   endogenous y ~ discrete;
   restrict "x1*x2"N>0;
run;
```

When there are multiple equations in the estimation, the parameters in the main equation are named in the format of *y*.*x*, where *y* is the name of the dependent variable and *x* is the name of the independent variable. The standard deviation of the errors is called _Sigma.*y*. The correlation of the errors is called _Rho for bivariate model. For the model with three variables it is _Rho.*y1*.*y2*, _Rho.*y1*.*y3*, _Rho.*y2*.*y3*. The construction of correlation names for multivariate models is analogous. Box-Cox parameters are called _Lambda*i*.*y* and limit variables are called _Limit*i*.*y*. Parameters in the HETERO statement are named as _H.*y*.*x*. In the OUTEST= data set, all variables are changed from '.' to '_'.

Naming of Output Variables

The following table shows the option in the OUTPUT statement, with the corresponding variable names and their explanation.

Option	Name	Explanation
PREDICTED	P_y	Predicted value of y
RESIDUAL	RESID_y	Residual of y, (y-PredictedY)
XBETA	XBETA_y	Structure part ($\mathbf{x}'\boldsymbol{\beta}$) of y equation
ERRSTD	ERRSTD_y	Standard deviation of error term
PROB	PROB_y	Probability that y is taking the observed value in this observation (discrete y only)
PROBALL	PROBi_y	Probability that y is taking the ith value (discrete y only)
MILLS	MILLS_y	Inverse Mills ratio for y
EXPECTED	EXPCT_y	Unconditional expected value of y
CONDITIONAL	CEXPCT_y	Conditional expected value of y, condition on the truncation.
MARGINAL	MEFF_x	Marginal effect of x on y ($\frac{\partial y}{\partial x}$) with single equation
	MEFF_y_x	Marginal effect of x on y ($\frac{\partial y}{\partial x}$) with multiple equations
	MEFF_Pi_x	Marginal effect of x on y ($\frac{\partial Prob(y=i)}{\partial x}$) with single equation and discrete y
	MEFF_Pi_y_x	Marginal effect of x on y ($\frac{\partial Prob(y=i)}{\partial x}$) with multiple equations and discrete y
TE1	TE1	Technical efficiency estimate for each producer proposed by Battese and Coelli (1988)
TE2	TE2	Technical efficiency estimate for each producer proposed by Jondrow et al. (1982)

If you prefer to name the output variables differently, you can use the RENAME option in the data set. For example, the following statements rename the residual of y as *Resid*:

```
proc qlim data=one;
   model y = x1-x10 / censored;
   output out=outds(rename=(resid_y=resid)) residual;
run;
```

ODS Table Names

PROC QLIM assigns a name to each table it creates. You can use these names to denote the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in the Table 29.11.

Table 29.11 ODS Tables Produced in PROC QLIM by the MODEL Statement and TEST Statement

ODS Table Name	Description	Option
ODS Tables Created by the MODEL Statement and TEST Statement		
ResponseProfile	Response profile	default
ClassLevels	Class levels	default
FitSummary	Summary of nonlinear estimation	default
GoodnessOfFit	Pseudo-R-square measures	default
ConvergenceStatus	Convergence status	default
ParameterEstimates	Parameter estimates	default
SummaryContResponse	Summary of continuous response	default
CovB	Covariance of parameter estimates	COVB
CorrB	Correlation of parameter estimates	CORRB
FitSummaryHeckman1	Heckman First Step Model Fit Summary	HECKIT
FitSummaryHeckman2	Heckman Second Model Fit Summary	HECKIT
LinCon	Linear constraints	ITPRINT
InputOptions	Input options	ITPRINT
ProblemDescription	Problem description	ITPRINT
IterStart	Optimization start summary	ITPRINT
IterHist	Iteration history	ITPRINT
IterStop	Optimization results	ITPRINT
ConvergenceStatus	Convergence status	ITPRINT
ParameterEstimatesStart	Optimization start	ITPRINT
ParameterEstimatesResults	Resulting parameters	ITPRINT
LinConSol	Linear constraints evaluated at solution	ITPRINT
VariableSelection	Variable selection summary	SELECTVAR
ODS Tables Created by the TEST Statement		
TestResults	Test results	default
ODS Tables Created by the BAYES Statement		
AutoMcmcSummary	Automatic MCMC summary	DIAGNOSTICS=AUTOSUM
AutoCorr	Autocorrelation statistics for each parameter	default
Corr	Correlation matrix of the posterior samples	STATS=COR
Cov	Covariance matrix of the posterior samples	STATS=COV
ESS	Effective sample size for each parameter	Default
MCSE	Monte Carlo standard error for each parameter	Default
Geweke	Geweke diagnostics for each parameter	Default
Heidelberger	Heidelberger-Welch diagnostics for each parameter	DIAGNOSTICS=HEIDEL
LogMarginLike	Marginal likelihood	MARGINLIKE
PostIntervals	Equal-tail and HPD intervals for each parameter	Default
PosteriorSample	Posterior samples	(ODS output data set only)
PostSummaries	Posterior summaries	default
PriorSample	Prior samples used for prior predictive analysis	(ODS output data set only)

Table 29.11 *(continued)*

ODS Table Name	Description	Option
PriorSummaries	Prior summaries	STATS=PRIOR
Raftery	Raftery-Lewis diagnostics for each parameter	DIAGNOSTICS=RAFTER

ODS Graphics

You can reference every graph that is produced through ODS Graphics with a name. The names of the graphs that PROC QLIM generates are listed in Table 29.12 for the frequentist approach and in Table 29.13 for the Bayesian approach.

Table 29.12 Graphs Produced by PROC QLIM without a BAYES Statement

ODS Graph Name	Plot Description	Statement & Option
Frequentist Output Plots		
ResidPlot	Frequentist analysis of residuals	PLOTS=RESIDUAL
XbetaPlot	Frequentist analysis of xbeta	PLOTS=XBETA
PredPlot	Frequentist analysis of Predictions	PLOTS=PREDICTED
MarginalPlot	Frequentist analysis of marginal effects	PLOTS=MARGINAL
ErrStdPlot	Frequentist analysis of the error standard deviation (meaningful only with a HETERO statement)	PLOTS=ERRSTD
MillsPlot	Frequentist analysis of Mills ratio	PLOTS=MILLS
ExpctPlot	Frequentist analysis of expected values for continuous endogenous variables	PLOTS=EXPECTED
TE1Plot	Frequentist analysis of technical efficiency (only in stochastic frontier model) suggested by Battese and Coelli (1988)	PLOTS=TE1
TE2Plot	Frequentist analysis of technical efficiency (only in stochastic frontier model) suggested by Jondrow et al. (1982)	PLOTS=TE2
CExpctPlot	Frequentist analysis of conditional expected values for continuous endogenous variables	PLOTS=CONDITIONAL
ProbPlot	Frequentist analysis of probability of discrete endogenous variables that take the current observed responses	PLOTS=PROB
ProbAllPlot	Frequentist analysis of probability of discrete endogenous variables for all responses	PLOTS=PROBALL
ProfLikPlot	Profile log-likelihood plot	PLOTS=PROFLIK

Table 29.13 Graphs Produced by PROC QLIM When a BAYES Statement Is Included

ODS Graph Name	Plot Description	Statement and Option
Bayesian Diagnostic Plots		
ADPanel	Autocorrelation function and density panel	PLOTS=(AUTOCORR DENSITY)
AutocorrPanel	Autocorrelation function panel	PLOTS=AUTOCORR
AutocorrPlot	Autocorrelation function plot	PLOTS(UNPACK)=AUTOCORR
DensityPanel	Density panel	PLOTS=DENSITY
DensityPlot	Density plot	PLOTS(UNPACK)=DENSITY
ProfLikPlot	Profile log-likelihood plot	PLOTS=PROFLIK
TAPanel	Trace and autocorrelation function panel	PLOTS=(TRACE AUTOCORR)
TADPanel	Trace, density, and autocorrelation function panel	PLOTS=(TRACE AUTOCORR DENSITY) PLOTS=BAYESDIAG
TDPanel	Trace and density panel	PLOTS=(TRACE DENSITY)
TracePanel	Trace panel	PLOTS=TRACE
TracePlot	Trace plot	PLOTS(UNPACK)=TRACE
Bayesian Summary Plots		
BayesSumPlot	Prior/posterior densities and MLE	PLOTS=BAYESSUM
Bayesian Output Plots		
PredictiveByObsNumPlot	Predictive analysis by observation number	PLOTS(PRIOR)=BAYESPRED
PredictivePlot	Predictive analysis by regressor	PLOTS(PRIOR)=BAYESPRED

Examples: QLIM Procedure

Example 29.1: Ordered Data Modeling

Cameron and Trivedi (1986, 1998) studied the number of doctor visits from the Australian Health Survey 1977-78. In the following data set, the dependent variable, DVISITS, contains the number of doctor visits in the past 2 weeks (0, 1, or more than 2). The explanatory variables are: SEX indicates if the patient is female; AGE is the age in years divided by 100; INCOME is the annual income ($10,000); LEVYPLUS indicates if the patient has private health insurance; FREEPOOR indicates free government health insurance due to low income; FREEREPA indicates free government health insurance for other reasons; ILLNESS is the number of illnesses in the past 2 weeks; ACTDAYS is the number of days the illness caused reduced activity; HSCORE is a questionnaire score; CHCOND1 indicates a chronic condition that does not limit activity; and CHCOND2 indicates a chronic condition that limits activity.

```
data docvisit;
   input sex age agesq income levyplus freepoor freerepa
         illness actdays hscore chcond1 chcond2 dvisits;
   y = (dvisits > 0);
   if ( dvisits > 8 ) then dvisits = 8;
datalines;
1 0.19 0.0361 0.55  1  0  0  1  4  1  0  0  1
1 0.19 0.0361 0.45  1  0  0  1  2  1  0  0  1

   ... more lines ...

1 0.37 0.1369 0.25  0  0  1  1  0  1  0  0  0
1 0.52 0.2704 0.65  0  0  0  0  0  0  0  0  0
0 0.72 0.5184 0.25  0  0  1  0  0  0  0  0  0
;
```

The dependent variable, dvisits, has nine ordered values. The following SAS statements estimate the ordinal probit model:

```
/*-- Ordered Discrete Responses --*/
proc qlim data=docvisit;
   model dvisits = sex age agesq income levyplus
                   freepoor freerepa illness actdays hscore
                   chcond1 chcond2 / discrete;
run;
```

The output of the QLIM procedure for ordered data modeling is shown in Output 29.1.1.

Output 29.1.1 Ordered Data Modeling

Binary Data

The QLIM Procedure

Discrete Response Profile of dvisits

Index	Value	Total Frequency
1	0	4141
2	1	782
3	2	174
4	3	30
5	4	24
6	5	9
7	6	12
8	7	12
9	8	6

Output 29.1.1 *continued*

Model Fit Summary	
Number of Endogenous Variables	1
Endogenous Variable	dvisits
Number of Observations	5190
Log Likelihood	-3138
Maximum Absolute Gradient	0.0003675
Number of Iterations	82
Optimization Method	Quasi-Newton
AIC	6316
Schwarz Criterion	6447

Goodness-of-Fit Measures

Measure	Value	Formula
Likelihood Ratio (R)	789.73	2 * (LogL - LogL0)
Upper Bound of R (U)	7065.9	- 2 * LogL0
Aldrich-Nelson	0.1321	R / (R+N)
Cragg-Uhler 1	0.1412	1 - exp(-R/N)
Cragg-Uhler 2	0.1898	(1-exp(-R/N)) / (1-exp(-U/N))
Estrella	0.149	1 - (1-R/U)^(U/N)
Adjusted Estrella	0.1416	1 - ((LogL-K)/LogL0)^(-2/N*LogL0)
McFadden's LRI	0.1118	R / U
Veall-Zimmermann	0.2291	(R * (U+N)) / (U * (R+N))
McKelvey-Zavoina	0.2036	

N = # of observations, K = # of regressors

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	-1.378705	0.147413	-9.35	<.0001
sex	1	0.131885	0.043785	3.01	0.0026
age	1	-0.534190	0.815907	-0.65	0.5126
agesq	1	0.857308	0.898364	0.95	0.3399
income	1	-0.062211	0.068017	-0.91	0.3604
levyplus	1	0.137030	0.053262	2.57	0.0101
freepoor	1	-0.346045	0.129638	-2.67	0.0076
freerepa	1	0.178382	0.074348	2.40	0.0164
illness	1	0.150485	0.015747	9.56	<.0001
actdays	1	0.100575	0.005850	17.19	<.0001
hscore	1	0.031862	0.009201	3.46	0.0005
chcond1	1	0.061601	0.049024	1.26	0.2089
chcond2	1	0.135321	0.067711	2.00	0.0457
_Limit2	1	0.938884	0.031219	30.07	<.0001
_Limit3	1	1.514288	0.049329	30.70	<.0001
_Limit4	1	1.711660	0.058151	29.43	<.0001
_Limit5	1	1.952860	0.072014	27.12	<.0001
_Limit6	1	2.087422	0.081655	25.56	<.0001
_Limit7	1	2.333786	0.101760	22.93	<.0001
_Limit8	1	2.789796	0.156189	17.86	<.0001

By default, ordinal probit/logit models are estimated assuming that the first threshold or limit parameter (μ_1) is 0. However, this parameter can also be estimated when the LIMIT1=VARYING option is specified. The probability that y_i^* belongs to the jth category is defined as

$$P[\mu_{j-1} < y_i^* < \mu_j] = F(\mu_j - \mathbf{x}_i'\boldsymbol{\beta}) - F(\mu_{j-1} - \mathbf{x}_i'\boldsymbol{\beta})$$

where $F(\cdot)$ is the logistic or standard normal CDF, $\mu_0 = -\infty$ and $\mu_9 = \infty$. Output 29.1.2 lists ordinal probit estimates computed in the following program. Note that the intercept term is suppressed for model identification when μ_1 is estimated.

```
/*-- Ordered Probit --*/
proc qlim data=docvisit;
   model dvisits = sex age agesq income levyplus
                   freepoor freerepa illness actdays hscore
                   chcond1 chcond2 / discrete(d=normal) limit1=varying;
run;
```

Output 29.1.2 Ordinal Probit Parameter Estimates with LIMIT1=VARYING

Binary Data

The QLIM Procedure

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
sex	1	0.131885	0.043785	3.01	0.0026
age	1	-0.534181	0.815915	-0.65	0.5127
agesq	1	0.857298	0.898371	0.95	0.3399
income	1	-0.062211	0.068017	-0.91	0.3604
levyplus	1	0.137031	0.053262	2.57	0.0101
freepoor	1	-0.346045	0.129638	-2.67	0.0076
freerepa	1	0.178382	0.074348	2.40	0.0164
illness	1	0.150485	0.015747	9.56	<.0001
actdays	1	0.100575	0.005850	17.19	<.0001
hscore	1	0.031862	0.009201	3.46	0.0005
chcond1	1	0.061602	0.049024	1.26	0.2089
chcond2	1	0.135322	0.067711	2.00	0.0457
_Limit1	1	1.378706	0.147415	9.35	<.0001
_Limit2	1	2.317590	0.150206	15.43	<.0001
_Limit3	1	2.892994	0.155198	18.64	<.0001
_Limit4	1	3.090367	0.158263	19.53	<.0001
_Limit5	1	3.331566	0.164065	20.31	<.0001
_Limit6	1	3.466128	0.168799	20.53	<.0001
_Limit7	1	3.712493	0.179756	20.65	<.0001
_Limit8	1	4.168502	0.215738	19.32	<.0001

Example 29.2: Tobit Analysis

The following statements show a subset of the Mroz (1987) data set. In these data, Hours is the number of hours the wife worked outside the household in a given year, Yrs_Ed is the years of education, and Yrs_Exp

is the years of work experience. A Tobit model will be fit to the hours worked with years of education and experience as covariates.

By the nature of the data it is clear that there are a number of women who committed some positive number of hours to outside work ($y_i > 0$ is observed). There are also a number of women who did not work at all ($y_i = 0$ is observed). This gives us the following model:

$$y_i^* = \mathbf{x}_i' \boldsymbol{\beta} + \epsilon_i$$

$$y_i = \begin{cases} y_i^* & \text{if } y_i^* > 0 \\ 0 & \text{if } y_i^* \leq 0 \end{cases}$$

where $\epsilon_i \sim iidN(0, \sigma^2)$. The set of explanatory variables is denoted by $\mathbf{x}_i$.

```
title1 'Estimating a Tobit model';

data subset;
   input Hours Yrs_Ed Yrs_Exp @@;
   if Hours eq 0 then Lower=.;
      else               Lower=Hours;
datalines;
0 8 9 0 8 12 0 9 10 0 10 15 0 11 4 0 11 6
1000 12 1 1960 12 29 0 13 3 2100 13 36
3686 14 11 1920 14 38 0 15 14 1728 16 3
1568 16 19 1316 17 7 0 17 15
;

/*-- Tobit Model --*/
proc qlim data=subset;
   model hours = yrs_ed yrs_exp;
   endogenous hours ~ censored(lb=0);
run;
```

The output of the QLIM procedure is shown in Output 29.2.1.

Output 29.2.1 Tobit Analysis Results

Estimating a Tobit model

The QLIM Procedure

Model Fit Summary	
Number of Endogenous Variables	1
Endogenous Variable	Hours
Number of Observations	17
Log Likelihood	-74.93700
Maximum Absolute Gradient	1.18953E-6
Number of Iterations	23
Optimization Method	Quasi-Newton
AIC	157.87400
Schwarz Criterion	161.20685

Output 29.2.1 *continued*

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	-5598.295129	27.692220	-202.16	<.0001
Yrs_Ed	1	373.123254	53.988877	6.91	<.0001
Yrs_Exp	1	63.336247	36.551299	1.73	0.0831
_Sigma	1	1582.859635	390.076480	4.06	<.0001

In the "Parameter Estimates" table there are four rows. The first three of these rows correspond to the vector estimate of the regression coefficients $\boldsymbol{\beta}$. The last one is called _Sigma, which corresponds to the estimate of the error variance σ.

Example 29.3: Bivariate Probit Analysis

This example shows how to estimate a bivariate probit model. Note the INIT statement in the following program, which sets the initial values for some parameters in the optimization:

```
data a;
   keep y1 y2 x1 x2;
   do i = 1 to 500;
      x1 = rannor( 19283 );
      x2 = rannor( 19283 );
      u1 = rannor( 19283 );
      u2 = rannor( 19283 );
      y1l = 1 + 2 * x1 + 3 * x2 + u1;
      y2l = 3 + 4 * x1 - 2 * x2 + u1*.2 + u2;
      if ( y1l > 0 ) then y1 = 1;
      else                y1 = 0;
      if ( y2l > 0 ) then y2 = 1;
      else                y2 = 0;
      output;
   end;
run;

/*-- Bivariate Probit --*/
proc qlim data=a method=qn;
   init y1.x1 2.8, y1.x2 2.1, _rho .1;
   model y1 = x1 x2;
   model y2 = x1 x2;
   endogenous y1 y2 ~ discrete;
run;
```

The output of the QLIM procedure is shown in Output 29.3.1.

Output 29.3.1 Bivariate Probit Analysis Results

Estimating a Tobit model

The QLIM Procedure

Model Fit Summary	
Number of Endogenous Variables	2
Endogenous Variable	y1 y2
Number of Observations	500
Log Likelihood	-134.90796
Maximum Absolute Gradient	3.23363E-7
Number of Iterations	17
Optimization Method	Quasi-Newton
AIC	283.81592
Schwarz Criterion	313.31817

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
y1.Intercept	1	1.003639	0.153678	6.53	<.0001
y1.x1	1	2.244374	0.256062	8.76	<.0001
y1.x2	1	3.273441	0.341581	9.58	<.0001
y2.Intercept	1	3.621164	0.457173	7.92	<.0001
y2.x1	1	4.551525	0.576547	7.89	<.0001
y2.x2	1	-2.442769	0.332295	-7.35	<.0001
_Rho	1	0.144097	0.336459	0.43	0.6685

Example 29.4: Sample Selection Model

This example illustrates the use of PROC QLIM for sample selection models. The data set is the same one from Mroz (1987). The goal is to estimate a wage offer function for married women, accounting for potential selection bias. Of the 753 women, the wage is observed for 428 working women. The labor force participation equation estimated in the introductory example is used for selection. The wage equation uses log wage (lwage) as the dependent variable. The explanatory variables in the wage equation are the woman's years of schooling (educ), wife's labor experience (exper), and square of experience (expersq). The program is as follows:

```
/*-- Sample Selection --*/
proc qlim data=mroz;
   model inlf = nwifeinc educ exper expersq
                age kidslt6 kidsge6 /discrete;
   model lwage = educ exper expersq / select(inlf=1);
run;
```

The output of the QLIM procedure is shown in Output 29.4.1.

Output 29.4.1 Sample Selection

Binary Data

The QLIM Procedure

Model Fit Summary	
Number of Endogenous Variables	2
Endogenous Variable	inlf lwage
Number of Observations	753
Log Likelihood	-832.88509
Maximum Absolute Gradient	0.00502
Number of Iterations	78
Optimization Method	Quasi-Newton
AIC	1694
Schwarz Criterion	1759

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
lwage.Intercept	1	-0.552716	0.260371	-2.12	0.0338
lwage.educ	1	0.108351	0.014861	7.29	<.0001
lwage.exper	1	0.042837	0.014878	2.88	0.0040
lwage.expersq	1	-0.000837	0.000417	-2.01	0.0449
_Sigma.lwage	1	0.663397	0.022706	29.22	<.0001
inlf.Intercept	1	0.266459	0.508954	0.52	0.6006
inlf.nwifeinc	1	-0.012132	0.004877	-2.49	0.0129
inlf.educ	1	0.131341	0.025383	5.17	<.0001
inlf.exper	1	0.123282	0.018728	6.58	<.0001
inlf.expersq	1	-0.001886	0.000601	-3.14	0.0017
inlf.age	1	-0.052829	0.008479	-6.23	<.0001
inlf.kidslt6	1	-0.867398	0.118647	-7.31	<.0001
inlf.kidsge6	1	0.035872	0.043476	0.83	0.4093
_Rho	1	0.026617	0.147073	0.18	0.8564

Note the correlation estimate is insignificant. This indicates that selection bias is not a big problem in the estimation of wage equation.

Example 29.5: Sample Selection Model with Truncation and Censoring

In this example the data are generated such that the selection variable is discrete and the variable Y is truncated from below by zero. The program follows, with the results shown in Output 29.5.1:

```
data trunc;
   keep z y x1 x2;
   do i = 1 to 500;
      x1 = rannor( 19283 );
      x2 = rannor( 19283 );
      u1 = rannor( 19283 );
      u2 = rannor( 19283 );
```

```
      zl = 1 + 2 * x1 + 3 * x2 + u1;
      y = 3 + 4 * x1 - 2 * x2 + u1*.2 + u2;
      if ( zl > 0 ) then z = 1;
      else z = 0;
      if y>=0 then output;
   end;
run;

/*-- Sample Selection with Truncation --*/
proc qlim data=trunc method=qn;
   model z = x1 x2 / discrete;
   model y = x1 x2 / select(z=1) truncated(lb=0);
run;
```

Output 29.5.1 Sample Selection with Truncation

Binary Data

The QLIM Procedure

Model Fit Summary	
Number of Endogenous Variables	2
Endogenous Variable	z y
Number of Observations	379
Log Likelihood	-344.10722
Maximum Absolute Gradient	4.95535E-6
Number of Iterations	17
Optimization Method	Quasi-Newton
AIC	704.21444
Schwarz Criterion	735.71473

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
y.Intercept	1	3.014158	0.128548	23.45	<.0001
y.x1	1	3.995671	0.099599	40.12	<.0001
y.x2	1	-1.972697	0.096385	-20.47	<.0001
_Sigma.y	1	0.923428	0.047233	19.55	<.0001
z.Intercept	1	0.949444	0.190265	4.99	<.0001
z.x1	1	2.163928	0.288384	7.50	<.0001
z.x2	1	3.134213	0.379251	8.26	<.0001
_Rho	1	0.494356	0.176542	2.80	0.0051

In the following statements the data are generated such that the selection variable is discrete and the variable *Y* is censored from below by zero. The results are shown in Output 29.5.2.

```
data cens;
   keep z y x1 x2;
   do i = 1 to 500;
      x1 = rannor( 19283 );
      x2 = rannor( 19283 );
      u1 = rannor( 19283 );
      u2 = rannor( 19283 );
      zl = 1 + 2 * x1 + 3 * x2 + u1;
      yl = 3 + 4 * x1 - 2 * x2 + u1*.2 + u2;
      if ( zl > 0 ) then z = 1;
```

```
      else               z = 0;
      if ( yl > 0 ) then y = yl;
      else               y = 0;
      output;
   end;
run;

/*-- Sample Selection with Censoring --*/
proc qlim data=cens method=qn;
   model z = x1 x2 / discrete;
   model y = x1 x2 / select(z=1) censored(lb=0);
run;
```

Output 29.5.2 Sample Selection with Censoring

Binary Data

The QLIM Procedure

Model Fit Summary	
Number of Endogenous Variables	2
Endogenous Variable	z y
Number of Observations	500
Log Likelihood	-399.78508
Maximum Absolute Gradient	2.30443E-6
Number of Iterations	19
Optimization Method	Quasi-Newton
AIC	815.57015
Schwarz Criterion	849.28702

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
y.Intercept	1	3.074276	0.111617	27.54	<.0001
y.x1	1	3.963619	0.085796	46.20	<.0001
y.x2	1	-2.023548	0.088714	-22.81	<.0001
_Sigma.y	1	0.920860	0.043278	21.28	<.0001
z.Intercept	1	1.013610	0.154081	6.58	<.0001
z.x1	1	2.256922	0.255999	8.82	<.0001
z.x2	1	3.302692	0.342168	9.65	<.0001
_Rho	1	0.350776	0.197093	1.78	0.0751

Example 29.6: Types of Tobit Models

The following five examples show how to estimate different types of Tobit models (see "Types of Tobit Models" on page 1989). Output 29.6.1 through Output 29.6.5 show the results of the corresponding programs.

Type 1 Tobit

```
data a1;
   keep y x;
   do i = 1 to 500;
      x = rannor( 19283 );
```

```
      u = rannor( 19283 );
      yl = 1 + 2 * x + u;
      if ( yl > 0 ) then y = yl;
      else               y = 0;
      output;
   end;
run;

/*-- Type 1 Tobit --*/
proc qlim data=a1 method=qn;
   model y = x;
   endogenous y ~ censored(lb=0);
run;
```

Output 29.6.1 Type 1 Tobit

Binary Data

The QLIM Procedure

Model Fit Summary	
Number of Endogenous Variables	1
Endogenous Variable	y
Number of Observations	500
Log Likelihood	-554.17696
Maximum Absolute Gradient	4.65556E-7
Number of Iterations	9
Optimization Method	Quasi-Newton
AIC	1114
Schwarz Criterion	1127

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	0.942734	0.056784	16.60	<.0001
x	1	2.049571	0.066979	30.60	<.0001
_Sigma	1	1.016571	0.039035	26.04	<.0001

Type 2 Tobit

```
data a2;
   keep y1 y2 x1 x2;
   do i = 1 to 500;
      x1 = rannor( 19283 );
      x2 = rannor( 19283 );
      u1 = rannor( 19283 );
      u2 = rannor( 19283 );
      y1l = 1 + 2 * x1 + 3 * x2 + u1;
      y2l = 3 + 4 * x1 - 2 * x2 + u1*.2 + u2;
      if ( y1l > 0 ) then y1 = 1;
      else                y1 = 0;
      if ( y1l > 0 ) then y2 = y2l;
      else                y2 = 0;
```

```
      output;
   end;
run;

/*-- Type 2 Tobit --*/
proc qlim data=a2 method=qn;
   model y1 = x1 x2 / discrete;
   model y2 = x1 x2 / select(y1=1);
run;
```

Output 29.6.2 Type 2 Tobit

Binary Data

The QLIM Procedure

Model Fit Summary	
Number of Endogenous Variables	2
Endogenous Variable	y1 y2
Number of Observations	500
Log Likelihood	-476.12328
Maximum Absolute Gradient	8.30075E-7
Number of Iterations	17
Optimization Method	Quasi-Newton
AIC	968.24655
Schwarz Criterion	1002

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
y2.Intercept	1	3.066992	0.106903	28.69	<.0001
y2.x1	1	4.004874	0.072043	55.59	<.0001
y2.x2	1	-2.079352	0.087544	-23.75	<.0001
_Sigma.y2	1	0.940559	0.039321	23.92	<.0001
y1.Intercept	1	1.017140	0.154975	6.56	<.0001
y1.x1	1	2.253080	0.256097	8.80	<.0001
y1.x2	1	3.305140	0.343695	9.62	<.0001
_Rho	1	0.292992	0.210073	1.39	0.1631

Type 3 Tobit

```
data a3;
   keep y1 y2 x1 x2;
   do i = 1 to 500;
      x1 = rannor( 19283 );
      x2 = rannor( 19283 );
      u1 = rannor( 19283 );
      u2 = rannor( 19283 );
      y1l = 1 + 2 * x1 + 3 * x2 + u1;
      y2l = 3 + 4 * x1 - 2 * x2 + u1*.2 + u2;
      if ( y1l > 0 ) then y1 = y1l;
      else               y1 = 0;
      if ( y1l > 0 ) then y2 = y2l;
```

```
      else                y2 = 0;
      output;
   end;
run;

/*-- Type 3 Tobit --*/
proc qlim data=a3 method=qn;
   model y1 = x1 x2 / censored(lb=0);
   model y2 = x1 x2 / select(y1>0);
run;
```

Output 29.6.3 Type 3 Tobit

Binary Data

The QLIM Procedure

Model Fit Summary	
Number of Endogenous Variables	2
Endogenous Variable	y1 y2
Number of Observations	500
Log Likelihood	-838.94087
Maximum Absolute Gradient	9.71691E-6
Number of Iterations	16
Optimization Method	Quasi-Newton
AIC	1696
Schwarz Criterion	1734

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
y2.Intercept	1	3.081206	0.080121	38.46	<.0001
y2.x1	1	3.998361	0.063734	62.73	<.0001
y2.x2	1	-2.088280	0.072876	-28.66	<.0001
_Sigma.y2	1	0.939799	0.039047	24.07	<.0001
y1.Intercept	1	0.981975	0.067351	14.58	<.0001
y1.x1	1	2.032675	0.059363	34.24	<.0001
y1.x2	1	2.976609	0.065584	45.39	<.0001
_Sigma.y1	1	0.969968	0.039795	24.37	<.0001
_Rho	1	0.226281	0.057672	3.92	<.0001

Type 4 Tobit

```
data a4;
   keep y1 y2 y3 x1 x2;
   do i = 1 to 500;
      x1 = rannor( 19283 );
      x2 = rannor( 19283 );
      u1 = rannor( 19283 );
      u2 = rannor( 19283 );
      u3 = rannor( 19283 );
      y1l = 1 + 2 * x1 + 3 * x2 + u1;
      y2l = 3 + 4 * x1 - 2 * x2 + u1*.2 + u2;
      y3l = 0 - 1 * x1 + 1 * x2 + u1*.1 - u2*.5 + u3*.5;
      if ( y1l > 0 ) then y1 = y1l;
```

```
      else                y1 = 0;
      if ( y1l > 0 ) then y2 = y2l;
      else                y2 = 0;
      if ( y1l <= 0 ) then y3 = y3l;
      else                y3 = 0;
      output;
   end;
run;

/*-- Type 4 Tobit --*/
proc qlim data=a4 method=qn;
   model y1 = x1 x2 / censored(lb=0);
   model y2 = x1 x2 / select(y1>0);
   model y3 = x1 x2 / select(y1<=0);
run;
```

Output 29.6.4 Type 4 Tobit

Binary Data

The QLIM Procedure

Model Fit Summary	
Number of Endogenous Variables	3
Endogenous Variable	y1 y2 y3
Number of Observations	500
Log Likelihood	-1128
Maximum Absolute Gradient	0.0000161
Number of Iterations	21
Optimization Method	Quasi-Newton
AIC	2285
Schwarz Criterion	2344

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
y2.Intercept	1	2.894656	0.076079	38.05	<.0001
y2.x1	1	4.072704	0.062675	64.98	<.0001
y2.x2	1	-1.901163	0.076874	-24.73	<.0001
_Sigma.y2	1	0.981655	0.039564	24.81	<.0001
y3.Intercept	1	0.064594	0.179441	0.36	0.7189
y3.x1	1	-0.938384	0.096570	-9.72	<.0001
y3.x2	1	1.035798	0.123104	8.41	<.0001
_Sigma.y3	1	0.743124	0.038240	19.43	<.0001
y1.Intercept	1	0.987370	0.067861	14.55	<.0001
y1.x1	1	2.050408	0.060819	33.71	<.0001
y1.x2	1	2.982190	0.072552	41.10	<.0001
_Sigma.y1	1	1.032473	0.040971	25.20	<.0001
_Rho.y1.y2	1	0.291587	0.053436	5.46	<.0001
_Rho.y1.y3	1	-0.031665	0.260057	-0.12	0.9031

Type 5 Tobit

```
data a5;
   keep y1 y2 y3 x1 x2;
   do i = 1 to 500;
      x1 = rannor( 19283 );
      x2 = rannor( 19283 );
      u1 = rannor( 19283 );
      u2 = rannor( 19283 );
      u3 = rannor( 19283 );
      y1l = 1 + 2 * x1 + 3 * x2 + u1;
      y2l = 3 + 4 * x1 - 2 * x2 + u1*.2 + u2;
      y3l = 0 - 1 * x1 + 1 * x2 + u1*.1 - u2*.5 + u3*.5;
      if ( y1l > 0 ) then y1 = 1;
      else               y1 = 0;
      if ( y1l > 0 ) then y2 = y2l;
      else               y2 = 0;
      if ( y1l <= 0 ) then y3 = y3l;
      else               y3 = 0;
      output;
   end;
run;

/*-- Type 5 Tobit --*/
proc qlim data=a5 method=qn;
   model y1 = x1 x2 / discrete;
   model y2 = x1 x2 / select(y1>0);
   model y3 = x1 x2 / select(y1<=0);
run;
```

Output 29.6.5 Type 5 Tobit

Binary Data

The QLIM Procedure

Model Fit Summary	
Number of Endogenous Variables	3
Endogenous Variable	y1 y2 y3
Number of Observations	500
Log Likelihood	-734.50612
Maximum Absolute Gradient	3.57134E-7
Number of Iterations	20
Optimization Method	Quasi-Newton
AIC	1495
Schwarz Criterion	1550

Output 29.6.5 *continued*

Parameter Estimates					
Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
y2.Intercept	1	2.887523	0.095193	30.33	<.0001
y2.x1	1	4.078926	0.069623	58.59	<.0001
y2.x2	1	-1.898898	0.086578	-21.93	<.0001
_Sigma.y2	1	0.983059	0.039987	24.58	<.0001
y3.Intercept	1	0.071764	0.171522	0.42	0.6757
y3.x1	1	-0.935299	0.092843	-10.07	<.0001
y3.x2	1	1.039954	0.120697	8.62	<.0001
_Sigma.y3	1	0.743083	0.038225	19.44	<.0001
y1.Intercept	1	1.067578	0.142789	7.48	<.0001
y1.x1	1	2.068376	0.226020	9.15	<.0001
y1.x2	1	3.157385	0.314743	10.03	<.0001
_Rho.y1.y2	1	0.312369	0.177010	1.76	0.0776
_Rho.y1.y3	1	-0.018225	0.234886	-0.08	0.9382

Example 29.7: Stochastic Frontier Models

This example illustrates the estimation of stochastic frontier production and cost models.

First, a production function model is estimated. The data for this example were collected by Christensen Associates; they represent a sample of 125 observations on inputs and output for 10 airlines between 1970 and 1984. The explanatory variables (inputs) are fuel (LF), materials (LM), equipment (LE), labor (LL), and property (LP), and (LQ) is an index that represents passengers, charter, mail, and freight transported.

The following statements create the dataset:

```
title1 'Stochastic Frontier Production Model';
data airlines;
   input TS FIRM NI LQ LF LM LE LL LP;
datalines;
1 1 15 -0.0484 0.2473 0.2335 0.2294 0.2246 0.2124
1 1 15 -0.0133 0.2603 0.2492 0.241 0.2216 0.1069
2 1 15 0.088 0.2666 0.3273 0.3365 0.2039 0.0865
3 1 15 0.1619 0.3019 0.4573 0.3532 0.2346 0.0242

   ... more lines ...
```

The following statements estimate a stochastic frontier exponential production model that uses Christensen Associates data:

```
/*-- Stochastic Frontier Production Model --*/
proc qlim data=airlines;
   model LQ=LF LM LE LL LP;
   endogenous LQ ~ frontier (type=exponential production);
run;
```

Figure 29.7.1 shows the results from this production model.

Output 29.7.1 Stochastic Frontier Production Model

Stochastic Frontier Production Model

The QLIM Procedure

Model Fit Summary	
Number of Endogenous Variables	1
Endogenous Variable	LQ
Number of Observations	125
Log Likelihood	83.27815
Maximum Absolute Gradient	9.92882E-7
Number of Iterations	25
Optimization Method	Quasi-Newton
AIC	-150.55630
Schwarz Criterion	-127.92979
Sigma	0.12445
Lambda	0.55766

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	-0.085048	0.024528	-3.47	0.0005
LF	1	-0.115802	0.124178	-0.93	0.3511
LM	1	0.756253	0.078755	9.60	<.0001
LE	1	0.424916	0.081893	5.19	<.0001
LL	1	-0.136421	0.089702	-1.52	0.1283
LP	1	0.098967	0.042776	2.31	0.0207
_Sigma_v	1	0.108688	0.010063	10.80	<.0001
_Sigma_u	1	0.060611	0.017603	3.44	0.0006

Similarly, the stochastic frontier production function can be estimated with (type=half) or (type=truncated) options that represent half-normal and truncated normal production models.

In the next step, stochastic frontier cost function is estimated. The data for the cost model are provided by Christensen and Greene (1976). The data describe costs and production inputs of 145 U.S. electricity producers in 1955. The model being estimated follows the nonhomogeneous version of the Cobb-Douglas cost function:

$$\log\left(\frac{\text{Cost}}{\text{FPrice}}\right) = \beta_0 + \beta_1 \log\left(\frac{\text{KPrice}}{\text{FPrice}}\right) + \beta_2 \log\left(\frac{\text{LPrice}}{\text{FPrice}}\right) + \beta_3 \log(\text{Output}) + \beta_4 \frac{1}{2} \log(\text{Output})^2 + \epsilon$$

All dollar values are normalized by fuel price. The quadratic log of the output is added to capture nonlinearities due to scale effects in cost functions. New variables, log_C_PF, log_PK_PF, log_PL_PF, log_y, and log_y_sq, are created to reflect transformations. The following statements create the data set and transformed variables:

```
title1 'Stochastic Frontier Cost Model';
data electricity;
   input Firm Year Cost Output LPrice LShare KPrice KShare FPrice FShare;
datalines;
 1  1955  .0820  2.0  2.090  .3164  183.000  .4521  17.9000  .2315
 2  1955  .6610  3.0  2.050  .2073  174.000  .6676  35.1000  .1251
 3  1955  .9900  4.0  2.050  .2349  171.000  .5799  35.1000  .1852
 4  1955  .3150  4.0  1.830  .1152  166.000  .7857  32.2000  .0990

   ... more lines ...


/* Data transformations */
data electricity;
   set electricity;
   label Firm="firm index"
         Year="1955 for all observations"
         Cost="Total cost"
         Output="Total output"
         LPrice="Wage rate"
         LShare="Cost share for labor"
         KPrice="Capital price index"
         KShare="Cost share for capital"
         FPrice="Fuel price"
         FShare"Cost share for fuel";
   log_C_PF=log(Cost/FPrice);
   log_PK_PF=log(KPrice/FPrice);
   log_PL_PF=log(LPrice/FPrice);
   log_y=log(Output);
   log_y_sq=log_y**2/2;
run;
```

The following statements estimate a stochastic frontier exponential cost model that uses Christensen and Greene (1976) data:

```
/*-- Stochastic Frontier Cost Model --*/
proc qlim data=electricity;
   model log_C_PF = log_PK_PF log_PL_PF log_y log_y_sq;
   endogenous log_C_PF ~ frontier (type=exponential cost);
run;
```

Output 29.7.2 shows the results.

Output 29.7.2 Exponential Distribution

Stochastic Frontier Cost Model

The QLIM Procedure

Model Fit Summary	
Number of Endogenous Variables	1
Endogenous Variable	log_C_PF
Number of Observations	159
Log Likelihood	-23.30430
Maximum Absolute Gradient	3.0458E-6
Number of Iterations	21
Optimization Method	Quasi-Newton
AIC	60.60860
Schwarz Criterion	82.09093
Sigma	0.30750
Lambda	1.71345

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	-4.983211	0.543328	-9.17	<.0001
log_PK_PF	1	0.090242	0.109202	0.83	0.4086
log_PL_PF	1	0.504299	0.118263	4.26	<.0001
log_y	1	0.427182	0.066680	6.41	<.0001
log_y_sq	1	0.066120	0.010079	6.56	<.0001
_Sigma_v	1	0.154998	0.020271	7.65	<.0001
_Sigma_u	1	0.265581	0.033614	7.90	<.0001

Similarly, the stochastic frontier cost model can be estimated with (type=half) or (type=truncated) options that represent half-normal and truncated normal errors.

The following statements illustrate the half-normal option:

```
/*-- Stochastic Frontier Cost Model --*/
proc qlim data=electricity;
   model log_C_PF = log_PK_PF log_PL_PF log_y log_y_sq;
   endogenous log_C_PF ~ frontier (type=half cost);
run;
```

Output 29.7.3 shows the result.

Output 29.7.3 Half-Normal Distribution

Stochastic Frontier Cost Model

The QLIM Procedure

Model Fit Summary	
Number of Endogenous Variables	1
Endogenous Variable	log_C_PF
Number of Observations	159
Log Likelihood	-34.95304
Maximum Absolute Gradient	0.0001150
Number of Iterations	22
Optimization Method	Quasi-Newton
AIC	83.90607
Schwarz Criterion	105.38840
Sigma	0.42761
Lambda	1.80031

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	-4.434634	0.690197	-6.43	<.0001
log_PK_PF	1	0.069624	0.136250	0.51	0.6093
log_PL_PF	1	0.474578	0.146812	3.23	0.0012
log_y	1	0.256874	0.080777	3.18	0.0015
log_y_sq	1	0.088051	0.011817	7.45	<.0001
_Sigma_v	1	0.207637	0.039222	5.29	<.0001
_Sigma_u	1	0.373810	0.073605	5.08	<.0001

The following statements illustrate the truncated normal option:

```
/*-- Stochastic Frontier Cost Model --*/
proc qlim data=electricity;
   model log_C_PF = log_PK_PF log_PL_PF log_y log_y_sq;
   endogenous log_C_PF ~ frontier (type=truncated cost);
run;
```

Output 29.7.4 shows the results.

Output 29.7.4 Truncated Normal Distribution

Stochastic Frontier Cost Model

The QLIM Procedure

Model Fit Summary	
Number of Endogenous Variables	1
Endogenous Variable	log_C_PF
Number of Observations	159
Log Likelihood	-36.87279
Maximum Absolute Gradient	271.78546
Number of Iterations	9
Optimization Method	Quasi-Newton
AIC	89.74557
Schwarz Criterion	114.29681
Sigma	0.30309
Lambda	1.04294E-7

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	-3.772132	0.338518	-11.14	<.0001
log_PK_PF	1	-0.030841	0.143593	-0.21	0.8299
log_PL_PF	1	0.574626	0.155390	3.70	0.0002
log_y	1	0.133254	0.058093	2.29	0.0218
log_y_sq	1	0.103028	0.009912	10.39	<.0001
_Sigma_v	1	0.303087	0.016898	17.94	<.0001
_Sigma_u	1	3.1610136E-8	.	.	.
_Mu	1	0.531720	0.338538	1.57	0.1163

If no (Production) or (Cost) option is specified, the stochastic frontier production model is estimated by default.

Example 29.8: Bayesian Modeling

This example illustrates how to use the QLIM procedure to perform Bayesian analysis. The generated data mimic a hypothetical scenario in which you study the number of tickets sold for a sports event given the probability of the hosting team winning and the price of the tickets. The following statements create the dataset:

```
title1 'Bayesian Analysis';

ods graphics on;

data test;
   do i=1 to 200;
      e1 = rannor(8726)*2000;
      WinChance = ranuni(8772);
```

```
      Price = 10+ranexp(8773)*4;
      y = 48000 + 5000*WinChance - 100 * price +  e1;
      if y>50000 then TicketSales = 50000;
      if y<=50000 then TicketSales = y;
      output;
   end;
   keep WinChance price  y TicketSales;
run;
```

The following statements perform Bayesian analysis of a Tobit model:

```
proc qlim data=test plots(prior)=all;
   model TicketSales = WinChance price;
   endogenous TicketSales ~ censored(lb=0 ub= 50000);
   prior intercept~normal(mean=48000);
   prior WinChance~normal(mean=5000);
   prior Price~normal(mean=-100);
   bayes NBI=10000 NMC=30000 THIN=1 ntrds=1 DIAG=ALL STATS=ALL seed=2;
run;
```

Output 29.8.1 shows the results from the maximum likelihood estimation and the Bayesian analysis with diffuse prior of this Tobit model.

Output 29.8.1 Bayesian Tobit Model

Bayesian Analysis

The QLIM Procedure

Parameter Estimates

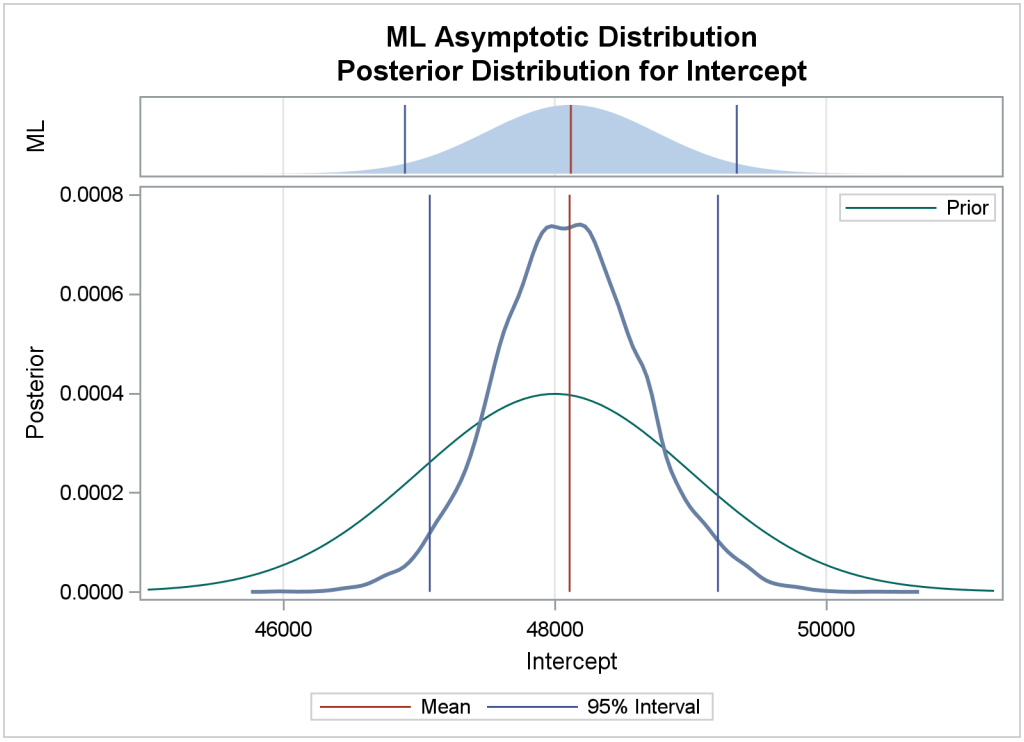
Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	48119	623.565045	77.17	<.0001
WinChance	1	5242.083501	559.151222	9.38	<.0001
Price	1	-106.731665	40.660795	-2.62	0.0087
_Sigma	1	1939.607206	134.348772	14.44	<.0001

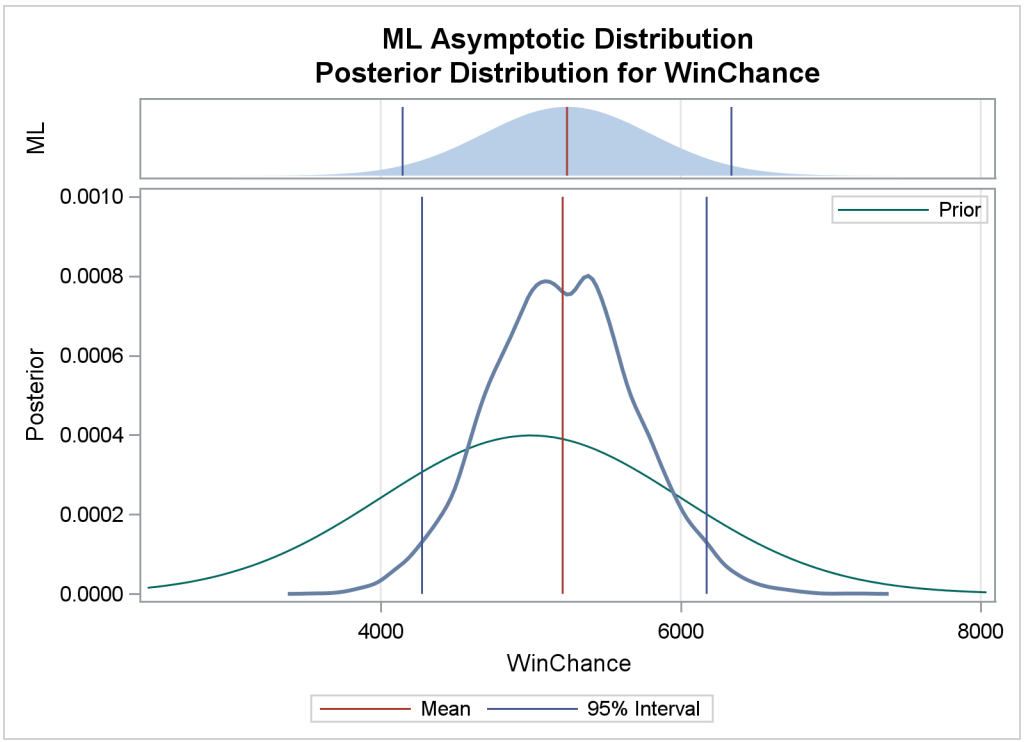
Posterior Summaries

Parameter	N	Mean	Standard Deviation	Percentiles 25%	50%	75%
Intercept	30000	48109.4	535.0	47750.5	48102.6	48460.1
WinChance	30000	5212.9	483.4	4878.8	5205.2	5533.0
Price	30000	-104.7	36.5224	-128.6	-104.2	-79.4191
_Sigma	30000	1950.9	132.9	1858.4	1945.0	2034.0

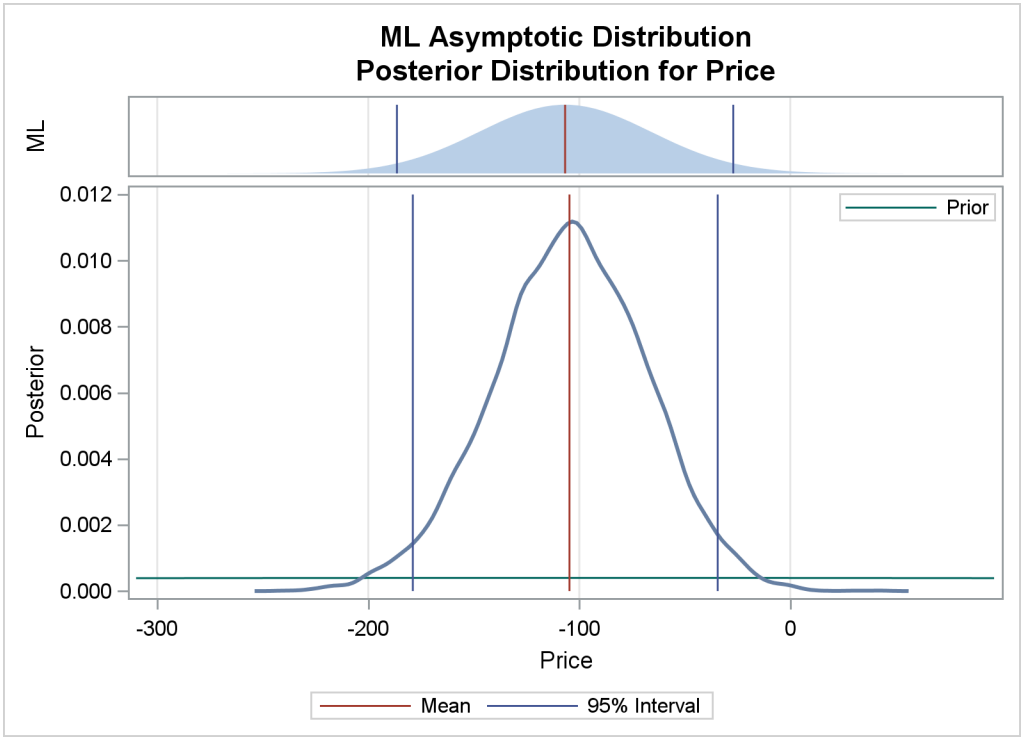
Output 29.8.2 depicts a graphical representation of MLE, prior, and posterior distributions.

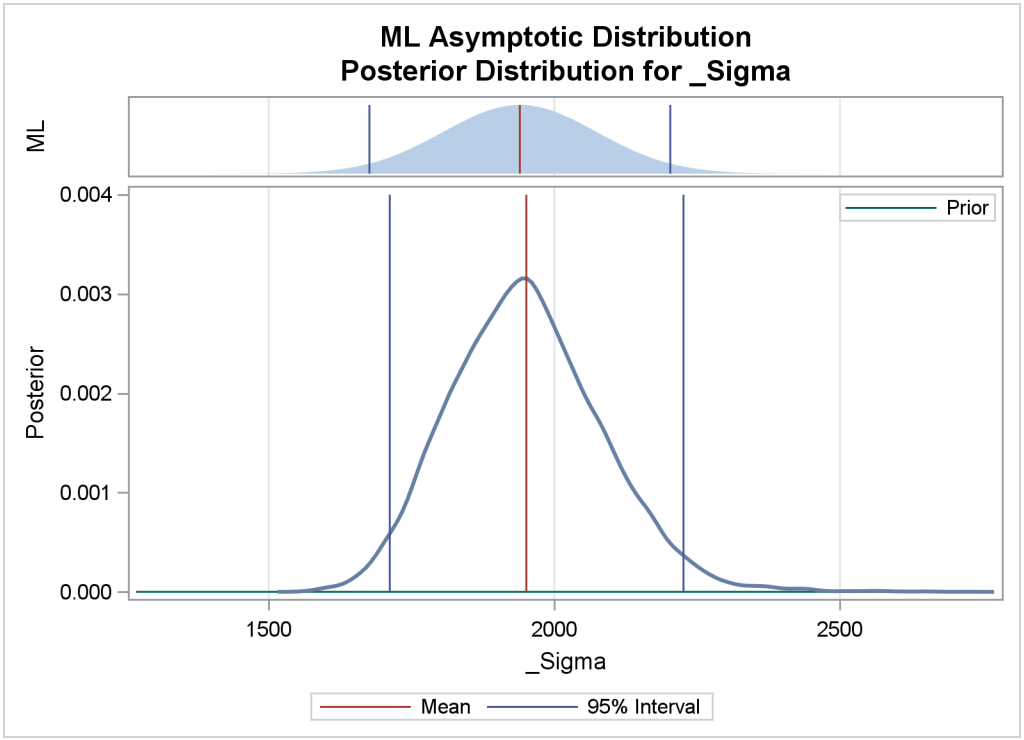
Output 29.8.2 Predictive Analysis by Observation Number





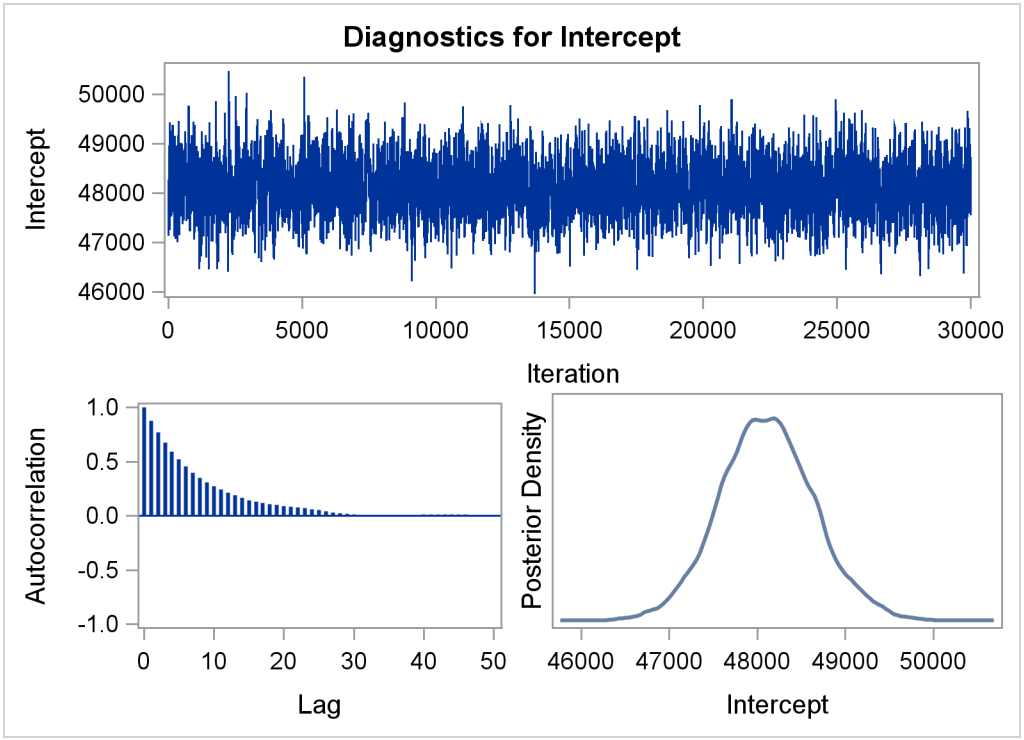
Output 29.8.2 *continued*

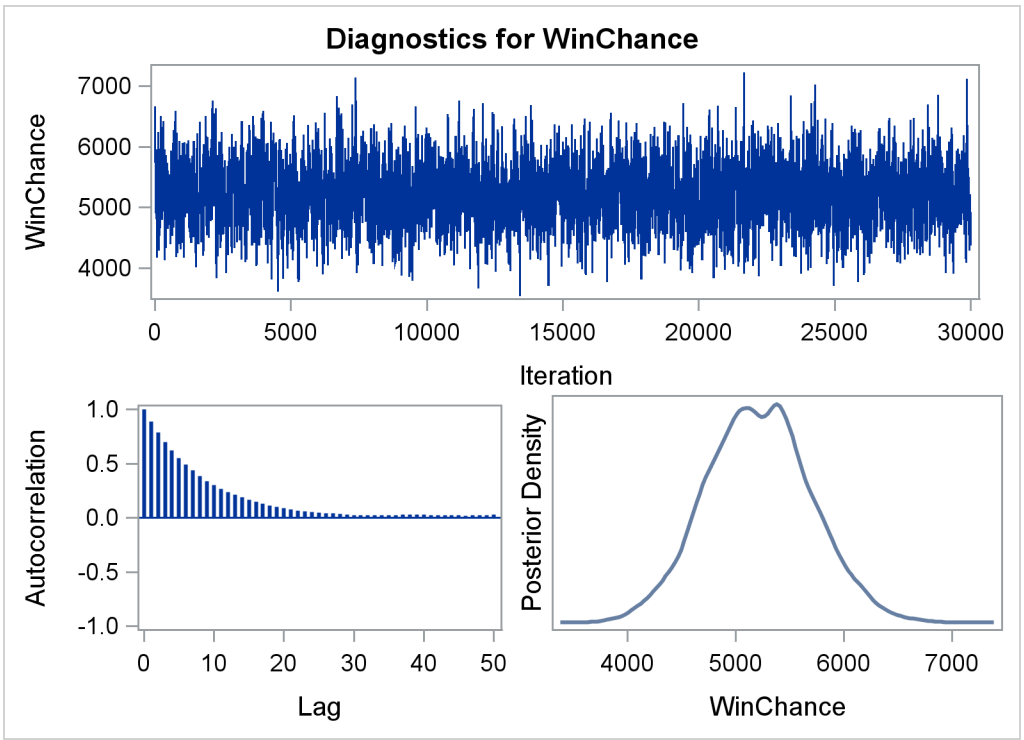




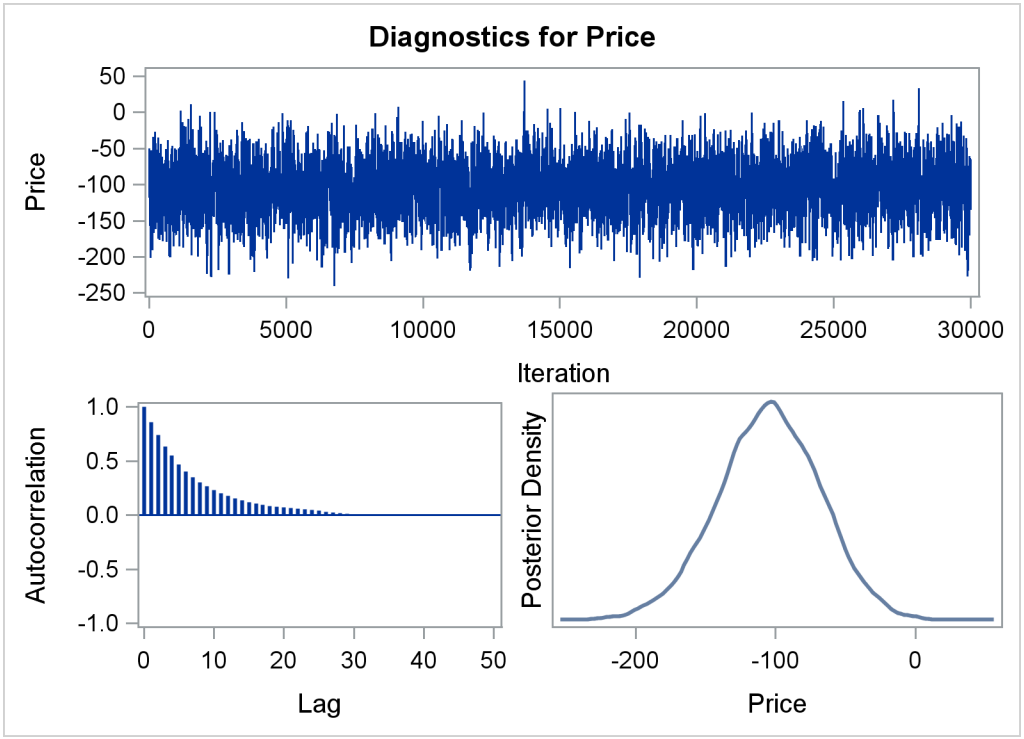
The validity of the MCMC sampling phase can be monitored with Output 29.8.3.

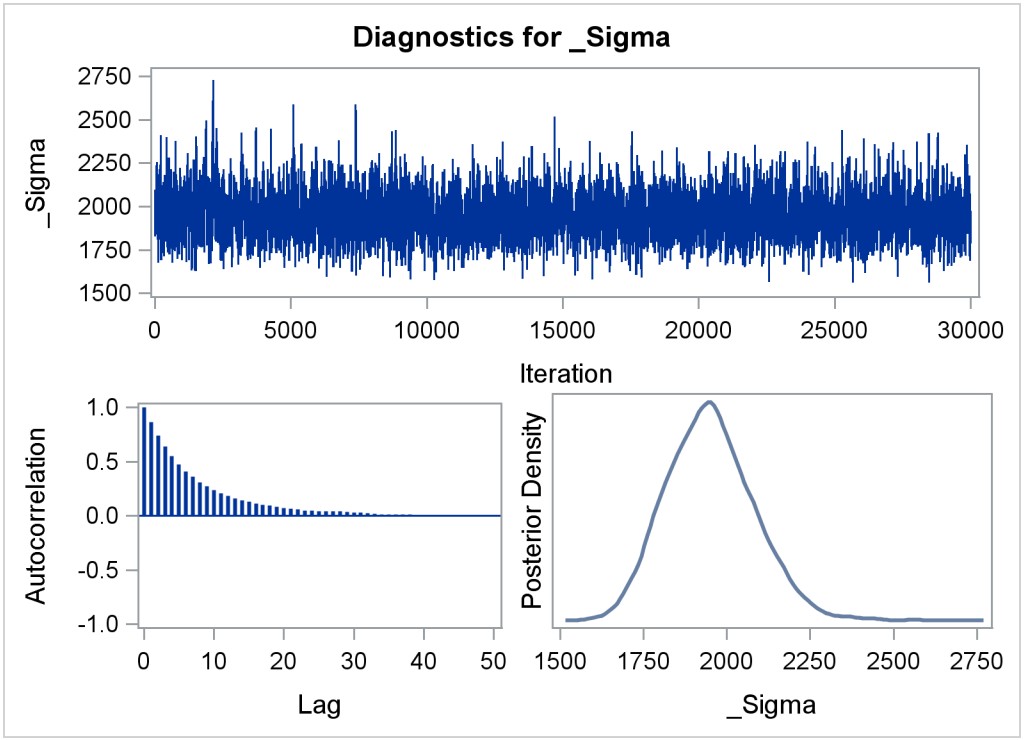
Output 29.8.3 Predictive Analysis by Observation Number





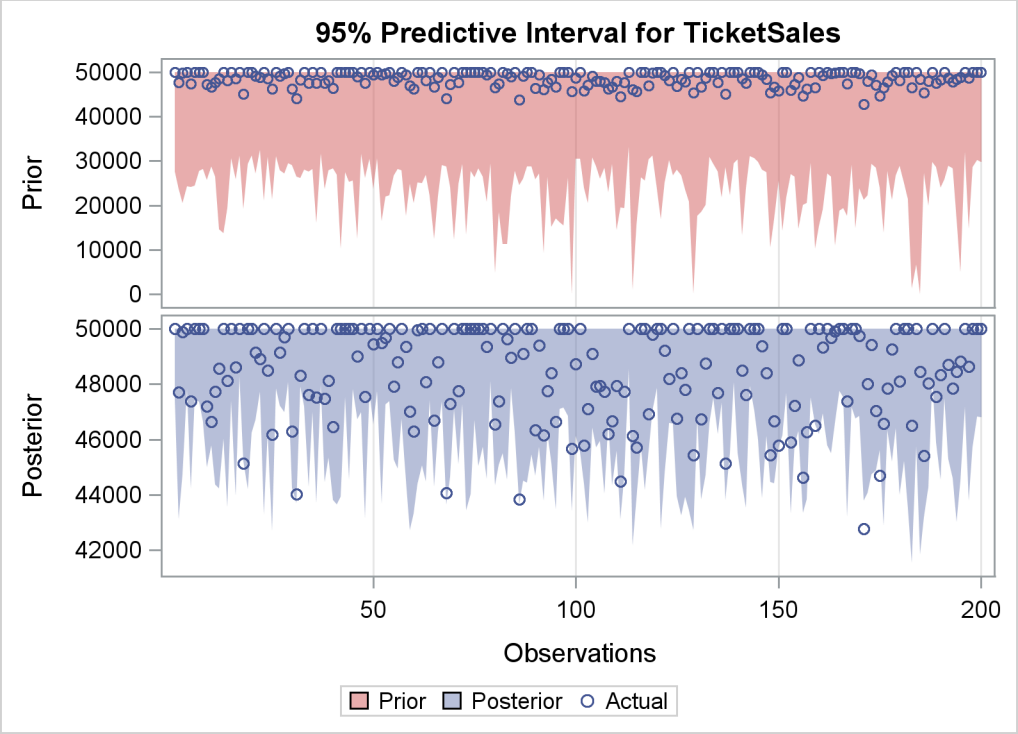
Output 29.8.3 *continued*





Finally the prior and the posterior predictive analyses are represented in Output 29.8.4

Output 29.8.4 Predictive Analysis by Observation Number



References

Abramowitz, M., and Stegun, I. A., eds. (1970). *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.* 9th printing. New York: Dover.

Aigner, C., Lovell, C. A. K., and Schmidt, P. (1977). "Formulation and Estimation of Stochastic Frontier Production Function Models." *Journal of Econometrics* 6:21–37.

Aitchison, J., and Silvey, S. (1957). "The Generalization of Probit Analysis to the Case of Multiple Responses." *Biometrika* 44:131–140.

Amemiya, T. (1978a). "The Estimation of a Simultaneous Equation Generalized Probit Model." *Econometrica* 46:1193–1205.

Amemiya, T. (1978b). "On a Two-Step Estimate of a Multivariate Logit Model." *Journal of Econometrics* 8:13–21.

Amemiya, T. (1981). "Qualitative Response Models: A Survey." *Journal of Economic Literature* 19:483–536.

Amemiya, T. (1984). "Tobit Models: A Survey." *Journal of Econometrics* 24:3–61.

Amemiya, T. (1985). *Advanced Econometrics*. Cambridge, MA: Harvard University Press.

Battese, G. E., and Coelli, T. J. (1988). "Prediction of Firm-Level Technical Efficiencies with a Generalized Frontier Production Function and Panel Data." *Journal of Econometrics* 38:387–399.

Ben-Akiva, M., and Lerman, S. R. (1985). *Discrete Choice Analysis: Theory and Application to Travel Demand*. Cambridge, MA: MIT Press.

Bera, A. K., Jarque, C. M., and Lee, L.-F. (1984). "Testing the Normality Assumption in Limited Dependent Variable Models." *International Economic Review* 25:563–578.

Berger, J. O. (1985). *Statistical Decision Theory and Bayesian Analysis*. 2nd ed. New York: Springer-Verlag.

Bhat, C. R. (2001). "Quasi-random Maximum Simulated Likelihood Estimation of the Mixed Multinomial Logit Model." *Transportation Research, Part B* 35:677–693.

Bloom, D. E., and Killingsworth, M. R. (1985). "Correcting for Truncation Bias Caused by a Latent Truncation Variable." *Journal of Econometrics* 27:131–135.

Box, G. E. P., and Cox, D. R. (1964). "An Analysis of Transformations." *Journal of the Royal Statistical Society, Series B* 26:211–234.

Butler, J. S., and Moffitt, R. (1982). "A Computationally Efficient Quadrature Procedure for the One-Factor Multinomial Probit Model." *Econometrica* 50:761–764.

Cameron, A. C., and Trivedi, P. K. (1986). "Econometric Models Based on Count Data: Comparisons and Applications of Some Estimators and Tests." *Journal of Applied Econometrics* 1:29–53.

Cameron, A. C., and Trivedi, P. K. (1998). *Regression Analysis of Count Data*. Cambridge: Cambridge University Press.

Chan, J. C. C., and Eisenstat, E. (2015). "Marginal Likelihood Estimation with the Cross-Entropy Method." *Econometric Reviews* 34:256–285.

Christensen, L. R., and Greene, W. H. (1976). "Economies of Scale in U.S. Electric Power Generation." *Journal of Political Economy* 84:655–676.

Coelli, T. J., Prasada Rao, D. S., and Battese, G. E. (1998). *An Introduction to Efficiency and Productivity Analysis*. London: Kluwer Academic.

Copley, P. A., Doucet, M. S., and Gaver, K. M. (1994). "A Simultaneous Equations Analysis of Quality Control Review Outcomes and Engagement Fees for Audits of Recipients of Federal Financial Assistance." *Accounting Review* 69:244–256.

Cox, D. R. (1970). *Analysis of Binary Data*. London: Metheun.

Cox, D. R. (1972). "Regression Models and Life Tables." *Journal of the Royal Statistical Society, Series B* 20:187–220. With discussion.

Cox, D. R. (1975). "Partial Likelihood." *Biometrika* 62:269–276.

Deis, D. R., and Hill, R. C. (1998). "An Application of the Bootstrap Method to the Simultaneous Equations Model of the Demand and Supply of Audit Services." *Contemporary Accounting Research* 15:83–99.

Estrella, A. (1998). "A New Measure of Fit for Equations with Dichotomous Dependent Variables." *Journal of Business and Economic Statistics* 16:198–205.

Gallant, A. R. (1987). *Nonlinear Statistical Models*. New York: John Wiley & Sons.

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (2004). *Bayesian Data Analysis*. 2nd ed. London: Chapman & Hall.

Genz, A. (1992). "Numerical Computation of Multivariate Normal Probabilities." *Journal of Computational and Graphical Statistics* 1:141–150.

Geweke, J. (1995). *Monte Carlo Simulation and Numerical Integration*. Staff Research Report 192, Federal Reserve Bank of Minneapolis.

Godfrey, L. G. (1988). *Misspecification Tests in Econometrics*. Cambridge: Cambridge University Press.

Gourieroux, C., Monfort, A., Renault, E., and Trognon, A. (1987). "Generalized Residuals." *Journal of Econometrics* 34:5–32.

Greene, W. H. (1997). *Econometric Analysis*. 3rd ed. Upper Saddle River, NJ: Prentice-Hall.

Greene, W. H. (2001). "Fixed and Random Effects in Nonlinear Models." Department of Economics, Leonard N. Stern School of Business, New York University.

Gregory, A. W., and Veall, M. R. (1985). "On Formulating Wald Tests for Nonlinear Restrictions." *Econometrica* 53:1465–1468.

Hajivassiliou, V. A. (1993). "Simulation Estimation Methods for Limited Dependent Variable Models." In *Econometrics*, edited by G. S. Maddala, C. R. Rao, and H. D. Vinod. Vol. 11 of Handbook of Statistics, 519–543. New York: Elsevier Science.

Hajivassiliou, V. A., and McFadden, D. L. (1998). "The Method of Simulated Scores for the Estimation of LDV Models." *Econometrica* 66:863–896.

Hajivassiliou, V. A., McFadden, D. L., and Ruud, P. A. (1996). "Simulation of Multivariate Normal Rectangle Probabilities and Their Derivatives: Theoretical and Computational Results." *Journal of Econometrics* 72:85–134.

Halton, J. H. (1960). "On the Efficiency of Certain Quasi-random Sequences of Points in Evaluating Multi-dimensional Integrals." *Numerische Mathematik* 2:84–90.

Heckman, J. J. (1978). "Dummy Endogenous Variables in a Simultaneous Equation System." *Econometrica* 46:931–959.

Hinkley, D. V. (1975). "On Power Transformations to Symmetry." *Biometrika* 62:101–111.

Jondrow, J., Lovell, C. A. K., Materov, I. S., and Schmidt, P. (1982). "On the Estimation of Technical Efficiency in the Stochastic Frontier Production Function Model." *Journal of Econometrics* 19:233–238.

Kim, M., and Hill, R. C. (1993). "The Box-Cox Transformation-of-Variables in Regression." *Empirical Economics* 18:307–319.

Kumbhakar, S. C., and Lovell, C. A. K. (2000). *Stochastic Frontier Analysis*. New York: Cambridge University Press.

Lee, L.-F. (1981). "Simultaneous Equations Models with Discrete and Censored Dependent Variables." In *Structural Analysis of Discrete Data with Econometric Applications*, edited by C. F. Manski, and D. McFadden, 346–364. Cambridge, MA: MIT Press.

Long, J. S. (1997). *Regression Models for Categorical and Limited Dependent Variables*. Thousand Oaks, CA: Sage Publications.

McFadden, D. (1974). "Conditional Logit Analysis of Qualitative Choice Behavior." In *Frontiers in Econometrics*, edited by P. Zarembka, 105–142. New York: Academic Press.

McFadden, D. (1981). "Econometric Models of Probabilistic Choice." In *Structural Analysis of Discrete Data with Econometric Applications*, edited by C. F. Manski, and D. McFadden, 2–50. Cambridge, MA: MIT Press.

McKelvey, R. D., and Zavoina, W. (1975). "A Statistical Model for the Analysis of Ordinal Level Dependent Variables." *Journal of Mathematical Sociology* 4:103–120.

Meeusen, W., and van den Broeck, J. (1977). "Efficiency Estimation from Cobb-Douglas Production Functions with Composed Error." *International Economic Review* 18:435–444.

Morokoff, W. J., and Caflisch, R. E. (1995). "Quasi-Monte Carlo Integration." *Journal of Computational Physics* 122:218–230.

Mroz, T. A. (1987). "The Sensitivity of an Empirical Model of Married Women's Work to Economic and Statistical Assumptions." *Econometrica* 55:765–799.

Mroz, T. A. (1999). "Discrete Factor Approximations in Simultaneous Equation Models: Estimating the Impact of a Dummy Endogenous Variable on a Continuous Outcome." *Journal of Econometrics* 92:233–274.

Nawata, K. (1994). "Estimation of Sample Selection Bias Models by the Maximum Likelihood Estimator and Heckman's Two-Step Estimator." *Economics Letters* 45:33–40.

Parks, R. W. (1967). "Efficient Estimation of a System of Regression Equations When Disturbances Are Both Serially and Contemporaneously Correlated." *Journal of the American Statistical Association* 62:500–509.

Phillips, C. B., and Park, J. Y. (1988). "On Formulating Wald Tests of Nonlinear Restrictions." *Econometrica* 56:1065–1083.

Powers, D. A., and Xie, Y. (2000). *Statistical Methods for Categorical Data Analysis*. San Diego: Academic Press.

Rivers, D., and Vuong, Q. H. (1988). "Limited Information Estimators and Exogeneity Tests for Simultaneous Probit Models." *Journal of Econometrics* 39:347–366.

Roberts, G. O., Gelman, A., and Gilks, W. R. (1997). "Weak Convergence and Optimal Scaling of Random Walk Metropolis Algorithms." *Annals of Applied Probability* 7:110–120.

Roberts, G. O., and Rosenthal, J. S. (2001). "Optimal Scaling for Various Metropolis-Hastings Algorithms." *Statistical Science* 16:351–367.

Schervish, M. J. (1995). *Theory of Statistics*. New York: Springer-Verlag.

Sloan, I. H., and Woźniakowski, H. (1998). "When Are Quasi-Monte Carlo Algorithms Efficient for High Dimensional Integrals?" *Journal of Complexity* 14:1–33.

Spanier, J., and Maize, E. (1991). "Quasi-random Methods for Estimating Integrals Using Relatively Small Samples." *SIAM Review* 36:18–44.

Stacy, E. W. (1962). "A Generalization of the Gamma Distribution." *Annals of Mathematical Statistics* 33:1187–1192.

Train, K. E. (2009). *Discrete Choice Methods with Simulation*. 2nd ed. Cambridge: Cambridge University Press.

Wooldridge, J. M. (2002). *Econometric Analysis of Cross Section and Panel Data*. Cambridge, MA: MIT Press.

Wooldridge, J. M. (2010). *Econometric Analysis of Cross Section and Panel Data*. 2nd ed. Cambridge, MA: MIT Press.

Chapter 30
The SEVERITY Procedure

Contents

Overview: SEVERITY Procedure **2068**
Getting Started: SEVERITY Procedure **2070**
A Simple Example of Fitting Predefined Distributions 2070
An Example with Left-Truncation and Right-Censoring 2075
An Example of Modeling Regression Effects 2080
Syntax: SEVERITY Procedure **2085**
Functional Summary 2085
PROC SEVERITY Statement 2087
BY Statement 2096
CLASS Statement 2096
DIST Statement 2098
LOSS Statement 2100
NLOPTIONS Statement 2103
OUTSCORELIB Statement 2103
SCALEMODEL Statement 2105
WEIGHT Statement 2107
Programming Statements 2107
Details: SEVERITY Procedure **2108**
Predefined Distributions 2108
Censoring and Truncation 2118
Parameter Estimation Method 2120
Parameter Initialization 2122
Estimating Regression Effects 2123
Levelization of Classification Variables 2128
Specification and Parameterization of Model Effects 2131
Empirical Distribution Function Estimation Methods 2138
Statistics of Fit 2144
Defining a Severity Distribution Model with the FCMP Procedure 2148
Predefined Utility Functions 2160
Scoring Functions 2165
Custom Objective Functions 2173
Multithreaded Computation 2176
Input Data Sets 2176
Output Data Sets 2178
Displayed Output 2182
ODS Graphics 2184

Examples: SEVERITY Procedure **2188**
Example 30.1: Defining a Model for Gaussian Distribution 2188
Example 30.2: Defining a Model for the Gaussian Distribution with a Scale Parameter 2192
Example 30.3: Defining a Model for Mixed-Tail Distributions 2199
Example 30.4: Estimating Parameters Using Cramér-von Mises Estimator 2208
Example 30.5: Fitting a Scaled Tweedie Model with Regressors 2210
Example 30.6: Fitting Distributions to Interval-Censored Data 2214
Example 30.7: Defining a Finite Mixture Model That Has a Scale Parameter 2217
Example 30.8: Predicting Mean and Value-at-Risk by Using Scoring Functions 2224
Example 30.9: Scale Regression with Rich Regression Effects 2229
References **2232**

Overview: SEVERITY Procedure

The SEVERITY procedure estimates parameters of any arbitrary continuous probability distribution that is used to model the magnitude (severity) of a continuous-valued event of interest. Some examples of such events are loss amounts paid by an insurance company and demand of a product as depicted by its sales. PROC SEVERITY is especially useful when the severity of an event does not follow typical distributions (such as the normal distribution) that are often assumed by standard statistical methods.

PROC SEVERITY provides a default set of probability distribution models that includes the Burr, exponential, gamma, generalized Pareto, inverse Gaussian (Wald), lognormal, Pareto, Tweedie, and Weibull distributions. In the simplest form, you can estimate the parameters of any of these distributions by using a list of severity values that are recorded in a SAS data set. You can optionally group the values by a set of BY variables. PROC SEVERITY computes the estimates of the model parameters, their standard errors, and their covariance structure by using the maximum likelihood method for each of the BY groups.

PROC SEVERITY can fit multiple distributions at the same time and choose the best distribution according to a selection criterion that you specify. You can use seven different statistics of fit as selection criteria. They are log likelihood, Akaike's information criterion (AIC), corrected Akaike's information criterion (AICC), Schwarz Bayesian information criterion (BIC), Kolmogorov-Smirnov statistic (KS), Anderson-Darling statistic (AD), and Cramér-von Mises statistic (CvM).

You can request the procedure to output the status of the estimation process, the parameter estimates and their standard errors, the estimated covariance structure of the parameters, the statistics of fit, estimated cumulative distribution function (CDF) for each of the specified distributions, and the empirical distribution function (EDF) estimate (which is used to compute the KS, AD, and CvM statistics of fit).

A high-performance version of PROC SEVERITY is available as the HPSEVERITY procedure in the SAS High-Performance Econometrics product. The following key features make PROC SEVERITY and PROC HPSEVERITY unique among SAS procedures that can estimate continuous probability distributions:

- Both procedures enable you to fit a distribution model when the severity values are truncated or censored or both. You can specify any combination of the following types of censoring and truncation

effects: left-censoring, right-censoring, left-truncation, or right-truncation. This is especially useful in applications with an insurance-type model where a severity (loss) is reported and recorded only if it is greater than the deductible amount (left-truncation) and where a severity value greater than or equal to the policy limit is recorded at the limit (right-censoring). Another useful application is that of interval-censored data, where you know both the lower limit (right-censoring) and upper limit (left-censoring) on the severity, but you do not know the exact value.

PROC SEVERITY also enables you to specify a *probability of observability* for the left-truncated data, which is a probability of observing values greater than the left-truncation threshold. This additional information can be useful in certain applications to more correctly model the distribution of the severity of events.

Both procedures use an appropriate estimator of the empirical distribution function (EDF). EDF is required to compute the KS, AD, and CvM statistics-of-fit. The procedures also provide the EDF estimates to your custom parameter initialization method. When you specify truncation or censoring, the EDF is estimated by using either Kaplan-Meier's product-limit estimator or Turnbull's estimator. The former is used by default when you specify only one form of censoring effect (right-censoring or left-censoring), whereas the latter is used by default when you specify both left-censoring and right-censoring effects. Both procedures compute the standard errors for all EDF estimators.

- Both procedures enable you to define any arbitrary continuous parametric distribution model and to estimate its parameters. You just need to define the key components of the distribution, such as its probability density function (PDF) and cumulative distribution function (CDF), as a set of functions and subroutines written with the FCMP procedure, which is part of Base SAS software. As long as the functions and subroutines follow certain rules, the SEVERITY and HPSEVERITY procedures can fit the distribution model defined by them.

- Both procedures can model the influence of exogenous or regressor variables on a probability distribution, as long as the distribution has a scale parameter. A linear combination of regression effects is assumed to affect the scale parameter via an exponential link function.

  If a distribution does not have a scale parameter, then either it needs to have another parameter that can be derived from a scale parameter by using a supported transformation or it needs to be reparameterized to have a scale parameter. If neither of these is possible, then regression effects cannot be modeled.

  You can easily construct many types of regression effects by using various operators on a set of classification and continuous variables. You can specify classification variables in the CLASS statement.

- Both procedures enable you to specify your own objective function to be optimized for estimating the parameters of a model. You can write SAS programming statements to specify the contribution of each observation to the objective function. You can use keyword functions such as _PDF_ and _CDF_ to generalize the objective function to any distribution. If you do not specify your own objective function, then the parameters of a model are estimated by maximizing the likelihood function of the data.

- Both procedures enable you to create scoring functions that offer a convenient way to evaluate any distribution function, such as PDF, CDF, QUANTILE, or your custom distribution function, for a fitted model on new observations.

- Both procedures use multithreading to significantly reduce the time it takes to fit a distribution model.

Getting Started: SEVERITY Procedure

This section outlines the use of the SEVERITY procedure to fit continuous probability distribution models. Three examples illustrate different features of the procedure.

A Simple Example of Fitting Predefined Distributions

The simplest way to use PROC SEVERITY is to fit all the predefined distributions to a set of values and let the procedure identify the best fitting distribution.

Consider a lognormal distribution, whose probability density function (PDF) f and cumulative distribution function (CDF) F are as follows, respectively, where Φ denotes the CDF of the standard normal distribution:

$$f(x;\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\log(x)-\mu}{\sigma}\right)^2} \quad \text{and} \quad F(x;\mu,\sigma) = \Phi\left(\frac{\log(x)-\mu}{\sigma}\right)$$

The following DATA step statements simulate a sample from a lognormal distribution with population parameters $\mu = 1.5$ and $\sigma = 0.25$, and store the sample in the variable Y of a data set Work.Test_sev1:

```
/*------------- Simple Lognormal Example -------------*/
data test_sev1(keep=y label='Simple Lognormal Sample');
   call streaminit(45678);
   label y='Response Variable';
   Mu = 1.5;
   Sigma = 0.25;
   do n = 1 to 100;
      y = exp(Mu) * rand('LOGNORMAL')**Sigma;
      output;
   end;
run;
```

The following statements fit all the predefined distribution models to the values of Y and identify the best distribution according to the corrected Akaike's information criterion (AICC):

```
proc severity data=test_sev1 crit=aicc;
   loss y;
   dist _predefined_;
run;
```

The PROC SEVERITY statement specifies the input data set along with the model selection criterion, the LOSS statement specifies the variable to be modeled, and the DIST statement with the _PREDEFINED_ keyword specifies that all the predefined distribution models be fitted.

Some of the default output displayed by this step is shown in Figure 30.1 through Figure 30.5. First, information about the input data set is displayed followed by the "Model Selection" table, as shown in Figure 30.1. The model selection table displays the convergence status, the value of the selection criterion, and the selection status for each of the candidate models. The Converged column indicates whether the estimation process for a given distribution model has converged, might have converged, or failed. The Selected column indicates whether a given distribution has the best fit for the data according to the selection criterion. For this example, the lognormal distribution model is selected, because it has the lowest value for the selection criterion.

Figure 30.1 Data Set Information and Model Selection Table

The SEVERITY Procedure

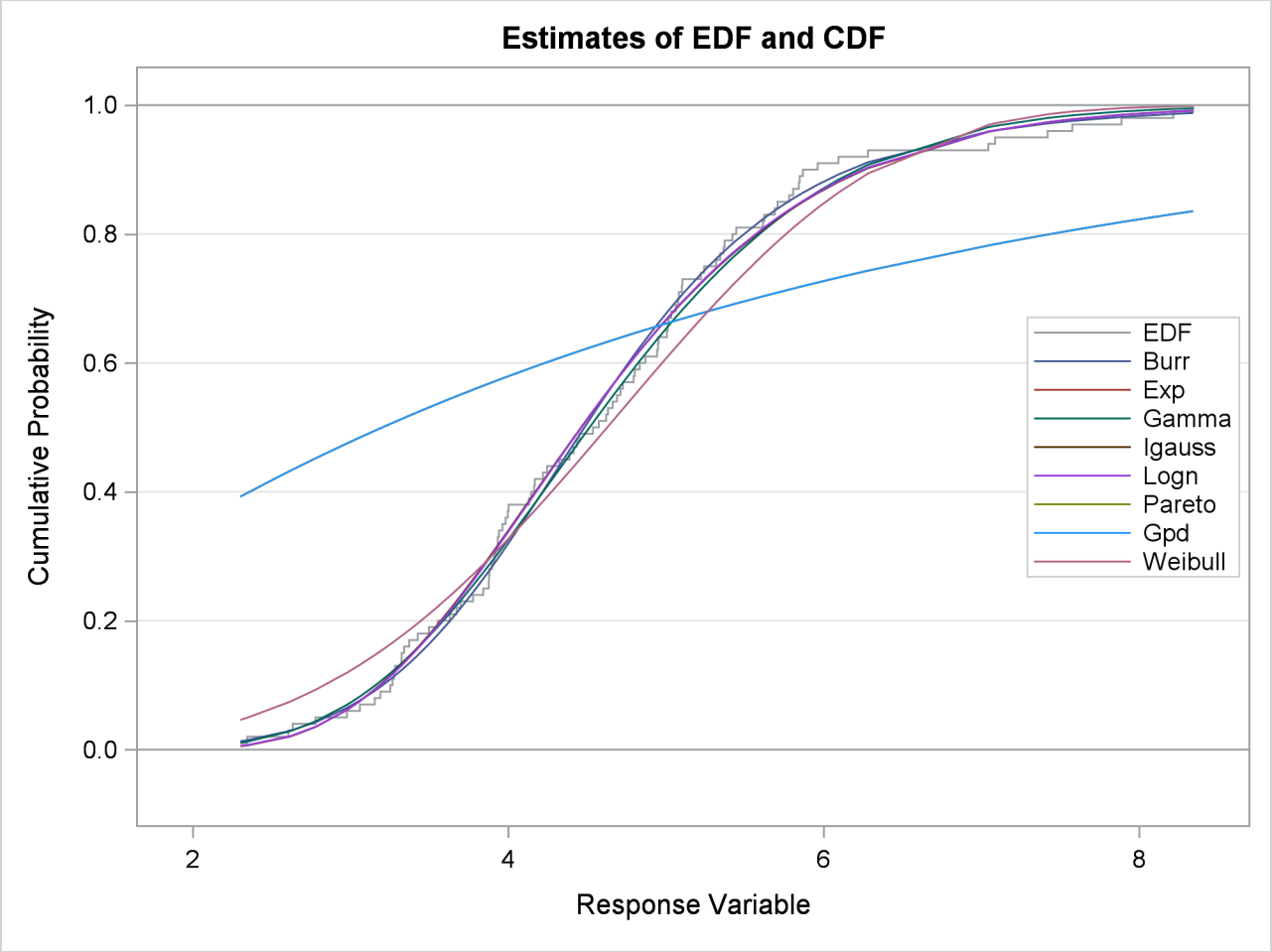
Input Data Set	
Name	WORK.TEST_SEV1
Label	Simple Lognormal Sample

Model Selection

Distribution	Converged	AICC	Selected
Burr	Yes	322.50845	No
Exp	Yes	508.12287	No
Gamma	Yes	320.50264	No
Igauss	Yes	319.61652	No
Logn	Yes	319.56579	Yes
Pareto	Yes	510.28172	No
Gpd	Yes	510.20576	No
Weibull	Yes	334.82373	No

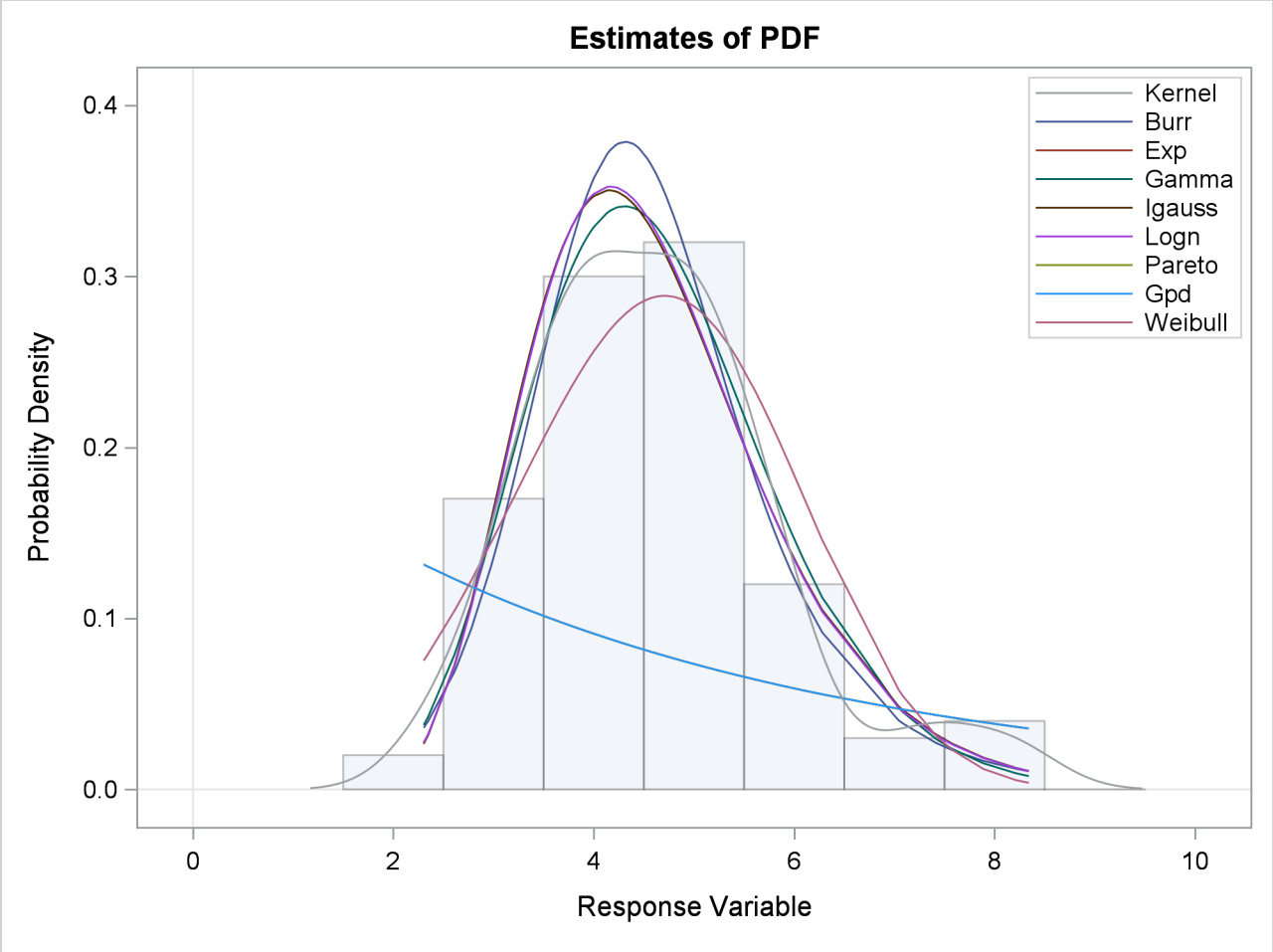
Next, two comparative plots are prepared. These plots enable you to visually verify how the models differ from each other and from the nonparametric estimates. The plot in Figure 30.2 displays the cumulative distribution function (CDF) estimates of all the models and the estimates of the empirical distribution function (EDF). The CDF plot indicates that the Exp (exponential), Pareto, and Gpd (generalized Pareto) distributions are a poor fit as compared to the EDF estimate. The Weibull distribution is also a poor fit, although not as poor as exponential, Pareto, and Gpd. The other four distributions seem to be quite close to each other and to the EDF estimate.

Figure 30.2 Comparison of EDF and CDF Estimates of the Fitted Models



The plot in Figure 30.3 displays the probability density function (PDF) estimates of all the models and the nonparametric kernel and histogram estimates. The PDF plot enables better visual comparison between the Burr, Gamma, Igauss (inverse Gaussian), and Logn (lognormal) models. The Burr and Gamma differ significantly from the Igauss and Logn distributions in the central portion of the range of Y values, while the latter two fit the data almost identically. This provides a visual confirmation of the information in the "Model Selection" table of Figure 30.1, which indicates that the AICC values of Igauss and Logn distributions are very close.

Figure 30.3 Comparison of PDF Estimates of the Fitted Models



The comparative plots are followed by the estimation information for each of the candidate models. The information for the lognormal model, which is the best fitting model, is shown in Figure 30.4. The first table displays a summary of the distribution. The second table displays the convergence status. This is followed by a summary of the optimization process which indicates the technique used, the number of iterations, the number of times the objective function was evaluated, and the log likelihood attained at the end of the optimization. Since the model with lognormal distribution has converged, PROC SEVERITY displays its statistics of fit and parameter estimates. The estimates of *Mu*=1.49605 and *Sigma*=0.26243 are quite close to the population parameters of *Mu*=1.5 and *Sigma*=0.25 from which the sample was generated. The *p*-value for each estimate indicates the rejection of the null hypothesis that the estimate is 0, implying that both the estimates are significantly different from 0.

Figure 30.4 Estimation Details for the Lognormal Model

The SEVERITY Procedure
Logn Distribution

Distribution Information	
Name	Logn
Description	Lognormal Distribution
Distribution Parameters	2

Convergence Status
Convergence criterion (GCONV=1E-8) satisfied.

Optimization Summary	
Optimization Technique	Trust Region
Iterations	2
Function Calls	8
Log Likelihood	-157.72104

Fit Statistics	
-2 Log Likelihood	315.44208
AIC	319.44208
AICC	319.56579
BIC	324.65242
Kolmogorov-Smirnov	0.50641
Anderson-Darling	0.31240
Cramer-von Mises	0.04353

Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|
Mu	1.49605	0.02651	56.43	<.0001
Sigma	0.26243	0.01874	14.00	<.0001

The parameter estimates of the Burr distribution are shown in Figure 30.5. These estimates are used in the next example.

Figure 30.5 Parameter Estimates for the Burr Model

Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|
Theta	4.62348	0.46181	10.01	<.0001
Alpha	1.15706	0.47493	2.44	0.0167
Gamma	6.41227	0.99039	6.47	<.0001

An Example with Left-Truncation and Right-Censoring

PROC SEVERITY enables you to specify that the response variable values are left-truncated or right-censored. The following DATA step expands the data set of the previous example to simulate a scenario that is typically encountered by an automobile insurance company. The values of the variable Y represent the loss values on claims that are reported to an auto insurance company. The variable THRESHOLD records the deductible on the insurance policy. If the actual value of Y is less than or equal to the deductible, then it is unobservable and does not get recorded. In other words, THRESHOLD specifies the left-truncation of Y. LIMIT records the policy limit. If the value of Y is equal to or greater than the recorded value, then the observation is right-censored.

```
/*----- Lognormal Model with left-truncation and censoring -----*/
data test_sev2(keep=y threshold limit
        label='A Lognormal Sample With Censoring and Truncation');
   set test_sev1;
   label y='Censored & Truncated Response';
   if _n_ = 1 then call streaminit(45679);

   /* make about 20% of the observations left-truncated */
   if (rand('UNIFORM') < 0.2) then
      threshold = y * (1 - rand('UNIFORM'));
   else
      threshold = .;
   /* make about 15% of the observations right-censored */
   iscens = (rand('UNIFORM') < 0.15);
   if (iscens) then
      limit = y;
   else
      limit = .;
run;
```

The following statements use the AICC criterion to analyze which of the four predefined distributions (lognormal, Burr, gamma, and Weibull) has the best fit for the data:

```
proc severity data=test_sev2 crit=aicc
       print=all plots=(cdfperdist pp qq);
   loss y / lt=threshold rc=limit;

   dist logn burr gamma weibull;
run;
```

The LOSS statement specifies the left-truncation and right-censoring variables. The DIST statement specifies the candidate distributions. The PRINT= option in the PROC SEVERITY statement requests that all the displayed output be prepared. The PLOTS= option in the PROC SEVERITY statement requests that the CDF plot, P-P plot, and Q-Q plot be prepared for each candidate distribution in addition to the default plots.

Some of the key results prepared by PROC SEVERITY are shown in Figure 30.6 through Figure 30.13. In addition to the estimates of the range, mean, and standard deviation of Y, the "Descriptive Statistics for y" table shown in Figure 30.6 also indicates the number of observations that are left-truncated or right-censored. The "Model Selection" table in Figure 30.6 shows that models with all the candidate distributions have converged and that the Logn (lognormal) model has the best fit for the data according to the AICC criterion.

Figure 30.6 Summary Results for the Truncated and Censored Data

The SEVERITY Procedure

Input Data Set	
Name	WORK.TEST_SEV2
Label	A Lognormal Sample With Censoring and Truncation

Descriptive Statistics for y	
Observations	100
Observations Used for Estimation	100
Minimum	2.30264
Maximum	8.34116
Mean	4.62007
Standard Deviation	1.23627
Left Truncated Observations	23
Right Censored Observations	14

Model Selection			
Distribution	**Converged**	**AICC**	**Selected**
Logn	Yes	298.92672	Yes
Burr	Yes	302.66229	No
Gamma	Yes	299.45293	No
Weibull	Yes	309.26779	No

PROC SEVERITY also prepares a table that shows all the fit statistics for all the candidate models. It is useful to see which model would be the best fit according to each of the criteria. The "All Fit Statistics" table prepared for this example is shown in Figure 30.7. It indicates that the lognormal model is chosen by all the criteria.

Figure 30.7 Comparing All Statistics of Fit for the Truncated and Censored Data

All Fit Statistics												
Distribution	**-2 Log Likelihood**		**AIC**		**AICC**		**BIC**		**KS**		**AD**	
Logn	294.80301	*	298.80301	*	298.92672	*	304.01335	*	0.51824	*	0.34736	*
Burr	296.41229		302.41229		302.66229		310.22780		0.66984		0.36712	
Gamma	295.32921		299.32921		299.45293		304.53955		0.62511		0.42921	
Weibull	305.14408		309.14408		309.26779		314.35442		0.93307		1.40699	

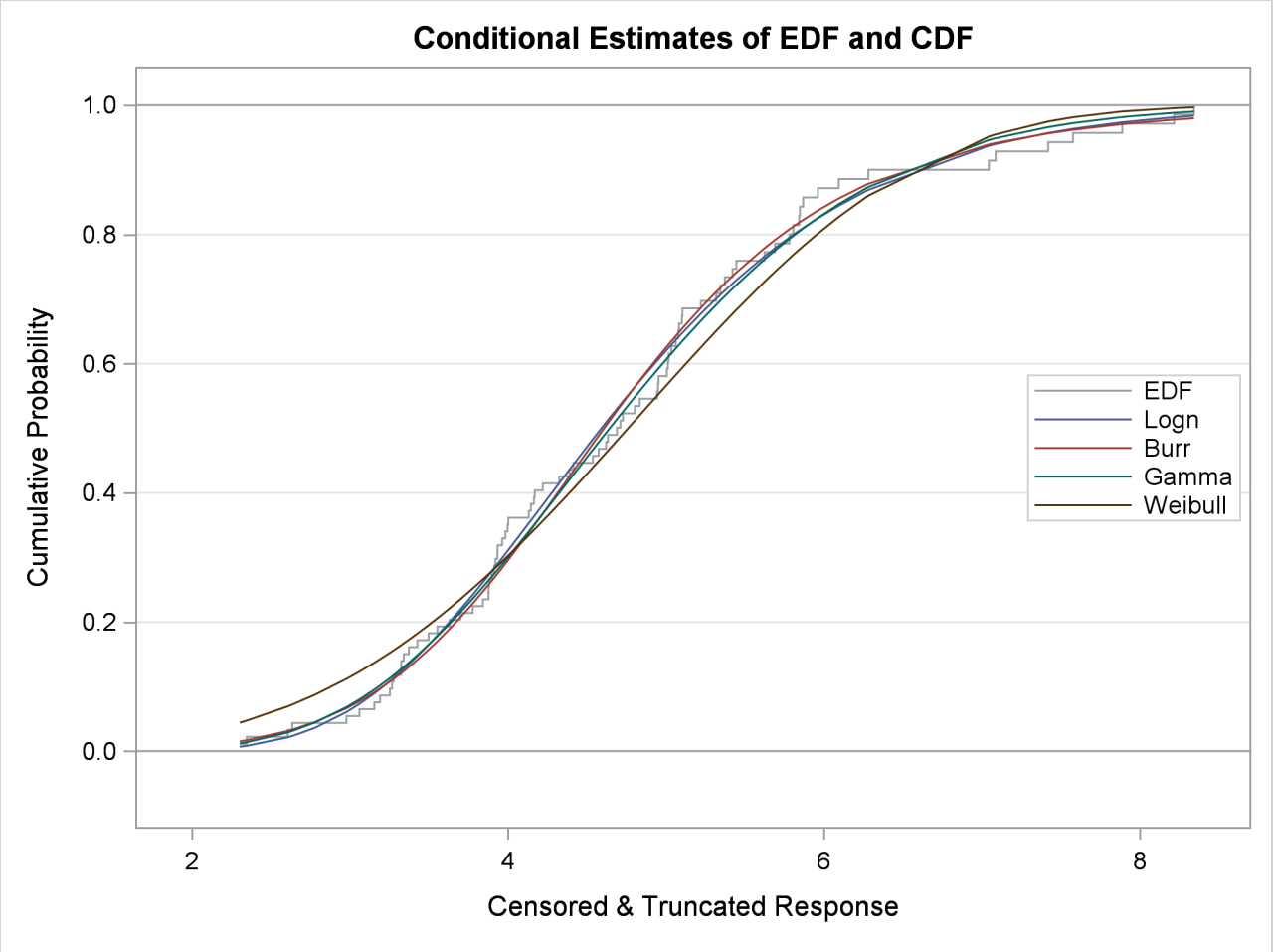
Note: The asterisk (*) marks the best model according to each column's criterion.

All Fit Statistics		
Distribution		**CvM**
Logn	0.05159	*
Burr	0.05726	
Gamma	0.05526	
Weibull	0.17465	

Note: The asterisk (*) marks the best model according to each column's criterion.

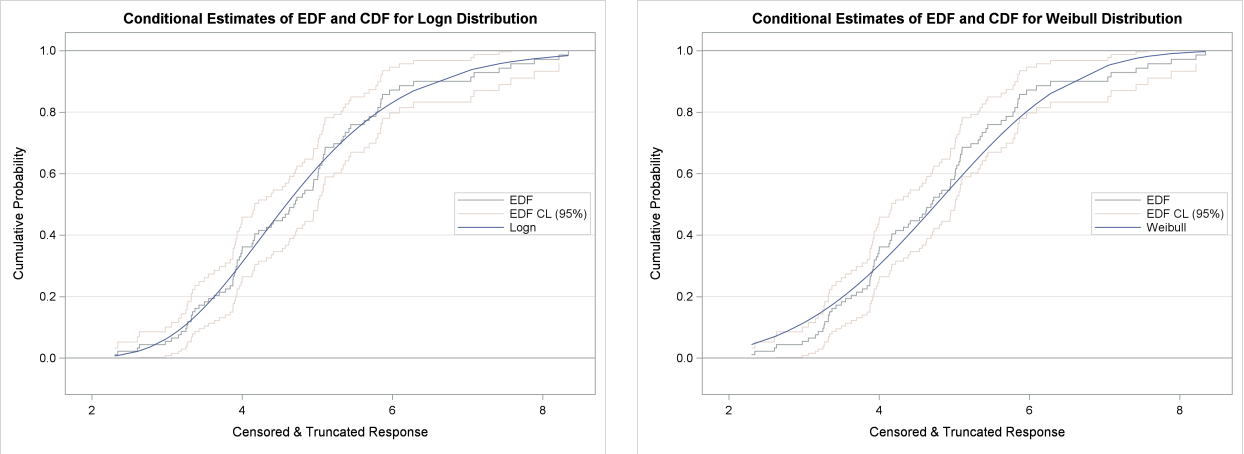
The plot that compares EDF and CDF estimates is shown in Figure 30.8. When you specify left-truncation, both the EDF and CDF estimates are conditional on the response variable being greater than the smallest left-truncation threshold in the sample.

Figure 30.8 EDF and CDF Estimates for the Truncated and Censored Data



When you specify the PLOTS=CDFPERDIST option, PROC SEVERITY prepares a plot that compares the nonparametric EDF estimates with the parametric CDF estimates for each distribution. These plots for lognormal and Weibull distributions are shown in Figure 30.9. These plots also contain the lower and upper confidence limits of EDF for the specified confidence level. Because no confidence level is specified in the EDFALPHA= option in the PROC SEVERITY statement, a default confidence level of 95% is used, which is equivalent to specifying EDFALPHA=0.05. If the CDF estimates lie entirely within the EDF confidence interval, then you can be 95% confident that the parametric and nonparametric estimates are in agreement.

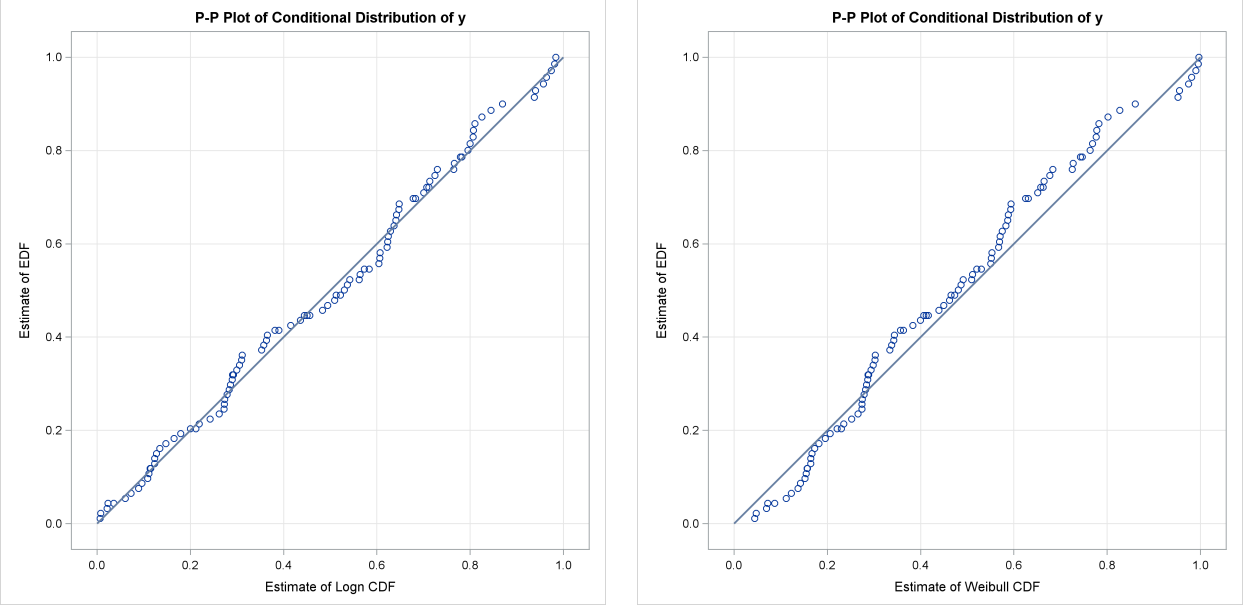
Figure 30.9 Comparing EDF and CDF Estimates for Lognormal and Weibull Models Fitted to Truncated and Censored Data



There are two additional ways to compare nonparametric (empirical) and parametric estimates for each model that has not failed to converge:

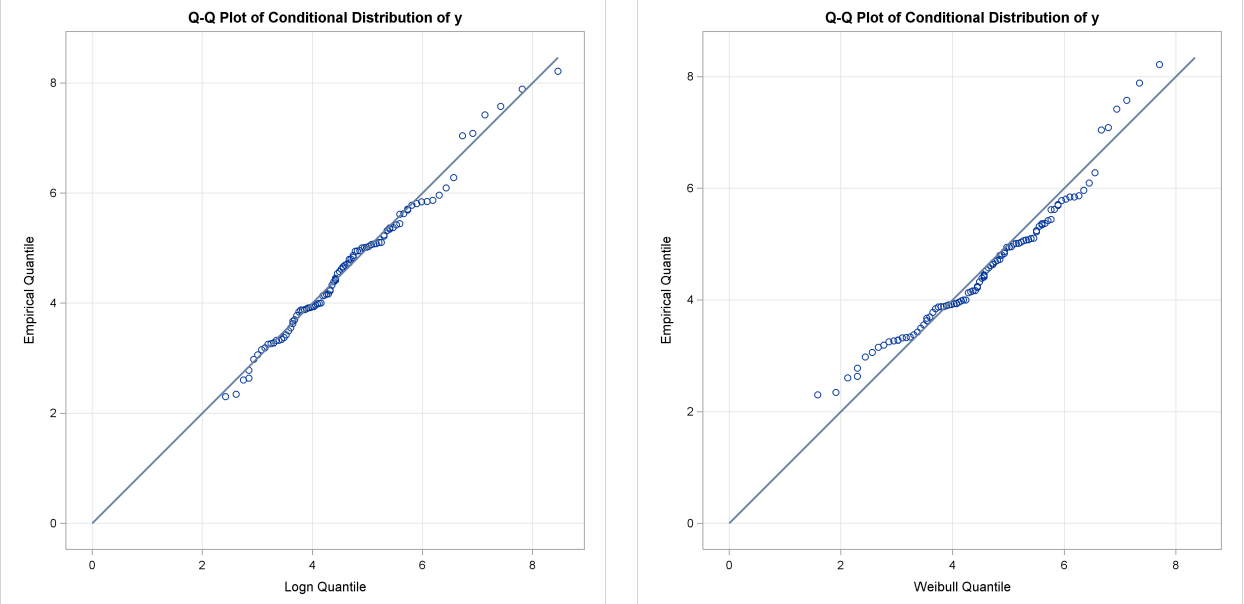
- A P-P plot is a scatter plot of the EDF and the CDF estimates. The model for which the points are scattered closer to the unit-slope reference line is a better fit. The P-P plot for the lognormal distribution is shown in Figure 30.10. It indicates that the EDF and the CDF match very closely. In contrast, the P-P plot for the Weibull distribution, also shown in Figure 30.10, indicates a poor fit.

Figure 30.10 P-P Plots for Lognormal and Weibull Models Fitted to Truncated and Censored Data



- A Q-Q plot is a scatter plot of empirical quantiles and the quantiles of a parametric distribution. Like the P-P plot, points scattered closer to the unit-slope reference line indicate a better fit. The Q-Q plots of lognormal and Weibull distributions are shown in Figure 30.11, which confirm the conclusions arrived at by comparing the P-P plots.

Figure 30.11 Q-Q Plots for Lognormal and Weibull Models Fitted to Truncated and Censored Data



Specifying Initial Values for Parameters

All the predefined distributions have parameter initialization functions built into them. For the current example, Figure 30.12 shows the initial values that are obtained by the predefined method for the Burr distribution. It also shows the summary of the optimization process and the final parameter estimates.

Figure 30.12 Burr Model Summary for the Truncated and Censored Data

Initial Parameter Values and Bounds

Parameter	Initial Value	Lower Bound	Upper Bound
Theta	4.78102	1.05367E-8	Infty
Alpha	2.00000	1.05367E-8	Infty
Gamma	2.00000	1.05367E-8	Infty

Optimization Summary

Optimization Technique	Trust Region
Iterations	8
Function Calls	23
Log Likelihood	-148.20614

Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|
Theta	4.76980	0.62492	7.63	<.0001
Alpha	1.16363	0.58859	1.98	0.0509
Gamma	5.94081	1.05004	5.66	<.0001

You can specify a different set of initial values if estimates are available from fitting the distribution to similar data. For this example, the parameters of the Burr distribution can be initialized with the final parameter estimates of the Burr distribution that were obtained in the first example (shown in Figure 30.5). One of the ways in which you can specify the initial values is as follows:

```
/*------ Specifying initial values using INIT= option -------*/
proc severity data=test_sev2 crit=aicc print=all plots=none;
   loss y / lt=threshold rc=limit;

   dist burr(init=(theta=4.62348 alpha=1.15706 gamma=6.41227));
run;
```

The names of the parameters that are specified in the INIT option must match the parameter names in the definition of the distribution. The results obtained with these initial values are shown in Figure 30.13. These results indicate that new set of initial values causes the optimizer to reach the same solution with fewer iterations and function evaluations as compared to the default initialization.

Figure 30.13 Burr Model Optimization Summary for the Truncated and Censored Data

The SEVERITY Procedure
Burr Distribution

Optimization Summary	
Optimization Technique	Trust Region
Iterations	5
Function Calls	16
Log Likelihood	-148.20614

Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|
Theta	4.76980	0.62492	7.63	<.0001
Alpha	1.16363	0.58859	1.98	0.0509
Gamma	5.94081	1.05004	5.66	<.0001

An Example of Modeling Regression Effects

Consider a scenario in which the magnitude of the response variable might be affected by some regressor (exogenous or independent) variables. The SEVERITY procedure enables you to model the effect of such variables on the distribution of the response variable via an exponential link function. In particular, if you have k random regressor variables denoted by x_j $(j = 1, \ldots, k)$, then the distribution of the response variable Y is assumed to have the form

$$Y \sim \exp(\sum_{j=1}^{k} \beta_j x_j) \cdot \mathcal{F}(\Theta)$$

where $\mathcal{F}$ denotes the distribution of Y with parameters Θ and $\beta_j (j = 1, \ldots, k)$ denote the regression parameters (coefficients). For the effective distribution of Y to be a valid distribution from the same

parametric family as $\mathcal{F}$, it is necessary for $\mathcal{F}$ to have a scale parameter. The effective distribution of Y can be written as

$$Y \sim \mathcal{F}(\theta, \Omega)$$

where θ denotes the scale parameter and Ω denotes the set of nonscale parameters. The scale θ is affected by the regressors as

$$\theta = \theta_0 \cdot \exp(\sum_{j=1}^{k} \beta_j x_j)$$

where θ_0 denotes a *base* value of the scale parameter.

Given this form of the model, PROC SEVERITY allows a distribution to be a candidate for modeling regression effects only if it has an untransformed or a log-transformed scale parameter.

All the predefined distributions, except the lognormal distribution, have a direct scale parameter (that is, a parameter that is a scale parameter without any transformation). For the lognormal distribution, the parameter μ is a log-transformed scale parameter. This can be verified by replacing μ with a parameter $\theta = e^{\mu}$, which results in the following expressions for the PDF f and the CDF F in terms of θ and σ, respectively, where Φ denotes the CDF of the standard normal distribution:

$$f(x;\theta,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\log(x)-\log(\theta)}{\sigma}\right)^2} \quad \text{and} \quad F(x;\theta,\sigma) = \Phi\left(\frac{\log(x)-\log(\theta)}{\sigma}\right)$$

With this parameterization, the PDF satisfies the $f(x;\theta,\sigma) = \frac{1}{\theta} f(\frac{x}{\theta};1,\sigma)$ condition and the CDF satisfies the $F(x;\theta,\sigma) = F(\frac{x}{\theta};1,\sigma)$ condition. This makes θ a scale parameter. Hence, $\mu = \log(\theta)$ is a log-transformed scale parameter and the lognormal distribution is eligible for modeling regression effects.

The following DATA step simulates a lognormal sample whose scale is decided by the values of the three regressors X1, X2, and X3 as follows:

$$\mu = \log(\theta) = 1 + 0.75\ \mathrm{X1} - \mathrm{X2} + 0.25\ \mathrm{X3}$$

```
/*----------- Lognormal Model with Regressors ------------*/
data test_sev3(keep=y x1-x3
               label='A Lognormal Sample Affected by Regressors');
   array x{*} x1-x3;
   array b{4} _TEMPORARY_ (1 0.75 -1 0.25);
   call streaminit(45678);
   label y='Response Influenced by Regressors';
   Sigma = 0.25;
   do n = 1 to 100;
      Mu = b(1); /* log of base value of scale */
      do i = 1 to dim(x);
         x(i) = rand('UNIFORM');
         Mu = Mu + b(i+1) * x(i);
      end;
      y = exp(Mu) * rand('LOGNORMAL')**Sigma;
      output;
   end;
run;
```

The following PROC SEVERITY step fits the lognormal, Burr, and gamma distribution models to this data. The regressors are specified in the SCALEMODEL statement. The DFMIXTURE= option in the SCALEMODEL statement specifies the method of computing the CDF estimates that are used to compute the EDF-based statistics of fit.

```
proc severity data=test_sev3 crit=aicc print=all;
   loss y;
   scalemodel x1-x3 / dfmixture=full;

   dist logn burr gamma;
run;
```

Some of the key results prepared by PROC SEVERITY are shown in Figure 30.14 through Figure 30.18. The descriptive statistics of all the variables are shown in Figure 30.14.

Figure 30.14 Summary Results for the Regression Example

The SEVERITY Procedure

Input Data Set	
Name	WORK.TEST_SEV3
Label	A Lognormal Sample Affected by Regressors

Descriptive Statistics for y	
Observations	100
Observations Used for Estimation	100
Minimum	1.17863
Maximum	6.65269
Mean	2.99859
Standard Deviation	1.12845

Descriptive Statistics for Regressors

Variable	N	Minimum	Maximum	Mean	Standard Deviation
x1	100	0.0005115	0.97971	0.51689	0.28206
x2	100	0.01883	0.99937	0.47345	0.28885
x3	100	0.00255	0.97558	0.48301	0.29709

The comparison of the fit statistics of all the models is shown in Figure 30.15. It indicates that the lognormal model is the best model according to each of the likelihood-based statistics, whereas the gamma model is the best model according to two of the three EDF-based statistics.

Figure 30.15 Comparison of Statistics of Fit for the Regression Example

All Fit Statistics

Distribution	-2 Log Likelihood		AIC		AICC		BIC		KS		AD	
Logn	187.49609	*	197.49609	*	198.13439	*	210.52194	*	0.68991	*	0.74299	
Burr	190.69154		202.69154		203.59476		218.32256		0.72348		0.73064	
Gamma	188.91483		198.91483		199.55313		211.94069		0.69101		0.72219	*

Note: The asterisk (*) marks the best model according to each column's criterion.

All Fit Statistics

Distribution	CvM	
Logn	0.11044	
Burr	0.11332	
Gamma	0.10546	*

Note: The asterisk (*) marks the best model according to each column's criterion.

The distribution information and the convergence results of the lognormal model are shown in Figure 30.16. The iteration history gives you a summary of how the optimizer is traversing the surface of the log-likelihood function in its attempt to reach the optimum. Both the change in the log likelihood and the maximum gradient of the objective function with respect to any of the parameters typically approach 0 if the optimizer converges.

Figure 30.16 Convergence Results for the Lognormal Model with Regressors

The SEVERITY Procedure
Logn Distribution

Distribution Information

Name	Logn
Description	Lognormal Distribution
Distribution Parameters	2
Regression Parameters	3

Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

Optimization Iteration History

Iter	Function Calls	-Log Likelihood	Change	Maximum Gradient
0	2	93.75285		6.16002
1	4	93.74805	-0.0048055	0.11031
2	6	93.74805	-1.5017E-6	0.00003376
3	10	93.74805	-1.421E-13	3.2667E-12

Optimization Summary

Optimization Technique	Trust Region
Iterations	3
Function Calls	10
Log Likelihood	-93.74805

The final parameter estimates of the lognormal model are shown in Figure 30.17. All the estimates are significantly different from 0. The estimate that is reported for the parameter *Mu* is the base value for the log-transformed scale parameter μ. Let $x_i (1 \leq i \leq 3)$ denote the observed value for regressor X*i*. If the lognormal distribution is chosen to model *Y*, then the effective value of the parameter μ varies with the observed values of regressors as

$$\mu = 1.04047 + 0.65221\, x_1 - 0.91116\, x_2 + 0.16243\, x_3$$

These estimated coefficients are reasonably close to the population parameters (that is, within one or two standard errors).

Figure 30.17 Parameter Estimates for the Lognormal Model with Regressors

Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|
Mu	1.04047	0.07614	13.66	<.0001
Sigma	0.22177	0.01609	13.78	<.0001
x1	0.65221	0.08167	7.99	<.0001
x2	-0.91116	0.07946	-11.47	<.0001
x3	0.16243	0.07782	2.09	0.0395

The estimates of the gamma distribution model, which is the best model according to a majority of the EDF-based statistics, are shown in Figure 30.18. The estimate that is reported for the parameter *Theta* is the base value for the scale parameter θ. If the gamma distribution is chosen to model *Y*, then the effective value of the scale parameter is $\theta = 0.14293 \exp(0.64562\, x_1 - 0.89831\, x_2 + 0.14901\, x_3)$.

Figure 30.18 Parameter Estimates for the Gamma Model with Regressors

Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|
Theta	0.14293	0.02329	6.14	<.0001
Alpha	20.37726	2.93277	6.95	<.0001
x1	0.64562	0.08224	7.85	<.0001
x2	-0.89831	0.07962	-11.28	<.0001
x3	0.14901	0.07870	1.89	0.0613

Syntax: SEVERITY Procedure

The following statements are available in the SEVERITY procedure:

PROC SEVERITY *options* ;
BY *variable-list* ;
LOSS < *response-variable* > < / *censoring-truncation-options* > ;
WEIGHT *weight-variable* ;
CLASS *variable* < *(options)* > . . . < *variable* < *(options)* > > < / *global-options* > ;
SCALEMODEL *regression-effect-list* < / *scalemodel-options* > ;
DIST *distribution-name-or-keyword* < *(distribution-option)* < *distribution-name-or-keyword* < *(distribution-option)* > > . . . > < / *preprocess-options* > ;
OUTSCORELIB < *OUTLIB=* > *fcmp-library-name options* ;
NLOPTIONS *options* ;
Programming statements ;

Functional Summary

Table 30.1 summarizes the statements and options that control the SEVERITY procedure.

Table 30.1 SEVERITY Functional Summary

Description	Statement	Option
Statements		
Specifies BY-group processing	BY	
Specifies the response variable to model along with censoring and truncation effects	LOSS	
Specifies the weight variable	WEIGHT	
Specifies the classification variables	CLASS	
Specifies the regression effects to model	SCALEMODEL	
Specifies distributions to fit	DIST	
Specifies the library to write scoring functions to	OUTSCORELIB	
Specifies optimization options	NLOPTIONS	
Specifies programming statements that define an objective function	Programming statements	
Input and Output Options		
Specifies that the OUTEST= data set contain covariance estimates	PROC SEVERITY	COVOUT
Specifies the input data set	PROC SEVERITY	DATA=
Specifies the input data set for parameter estimates	PROC SEVERITY	INEST=
Specifies the input item store for parameter initialization	PROC SEVERITY	INSTORE=
Limits the length of effect names	PROC SEVERITY	NAMELEN=
Specifies the output data set for CDF estimates	PROC SEVERITY	OUTCDF=
Specifies the output data set for parameter estimates	PROC SEVERITY	OUTEST=

Table 30.1 *continued*

Description	Statement	Option
Specifies the output data set for model information	PROC SEVERITY	OUTMODELINFO=
Specifies the output data set for statistics of fit	PROC SEVERITY	OUTSTAT=
Specifies the output item store for context and estimation results	PROC SEVERITY	OUTSTORE=
Data Interpretation Options		
Specifies left-censoring	LOSS	LEFTCENSORED=
Specifies left-truncation	LOSS	LEFTTRUNCATED=
Specifies the probability of observability	LOSS	PROBOBSERVED=
Specifies right-censoring	LOSS	RIGHTCENSORED=
Specifies right-truncation	LOSS	RIGHTTRUNCATED=
Model Estimation Options		
Specifies the model selection criterion	PROC SEVERITY	CRITERION=
Specifies the method for computing mixture distribution	SCALEMODEL	DFMIXTURE=
Specifies initial values for model parameters	DIST	INIT=
Specifies the objective function symbol	PROC SEVERITY	OBJECTIVE=
Specifies the offset variable in the scale regression model	SCALEMODEL	OFFSET=
Specifies the denominator for computing covariance estimates	PROC SEVERITY	VARDEF=
Empirical Distribution Function (EDF) Estimation Options		
Specifies the confidence level for reporting the confidence interval for EDF estimates	PROC SEVERITY	EDFALPHA=
Specifies the nonparametric method of CDF estimation	PROC SEVERITY	EMPIRICALCDF=
Specifies the sample to be used for computing the EDF estimates	PROC SEVERITY	INITSAMPLE
EMPIRICALCDF=MODIFIEDKM Options		
Specifies the α value for the lower bound on risk set size	PROC SEVERITY	ALPHA=
Specifies the c value for the lower bound on risk set size	PROC SEVERITY	C=
Specifies the absolute lower bound on risk set size	PROC SEVERITY	RSLB=
EMPIRICALCDF=TURNBULL Options		
Specifies that the final EDF estimates be maximum likelihood estimates	PROC SEVERITY	ENSUREMLE
Specifies the relative convergence criterion	PROC SEVERITY	EPS=
Specifies the maximum number of iterations	PROC SEVERITY	MAXITER=

Table 30.1 *continued*

Description	Statement	Option
Specifies the threshold below which an EDF estimate is deemed to be 0	PROC SEVERITY	ZEROPROB=
Scoring Function Generation Options		
Specifies that scoring functions of all models be written to one package	OUTSCORELIB	COMMONPACKAGE
Specifies the output data set for BY-group identifiers	OUTSCORELIB	OUTBYID=
Specifies the output library for scoring functions	OUTSCORELIB	OUTLIB=
Displayed Output and Plotting Options		
Specifies that distributions be listed to the log without estimating any models that use them	DIST	LISTONLY
Limits or suppresses the display of class levels	PROC SEVERITY	NOCLPRINT
Suppresses all displayed and graphical output	PROC SEVERITY	NOPRINT
Specifies which graphical output to prepare	PROC SEVERITY	PLOTS=
Specifies which output to display	PROC SEVERITY	PRINT=
Specifies that distributions be validated without estimating any models that use them	DIST	VALIDATEONLY

PROC SEVERITY Statement

PROC SEVERITY *options* **;**

The PROC SEVERITY statement invokes the procedure. You can specify two types of *options* in the PROC SEVERITY statement. One set of *options* controls input and output. The other set of *options* controls the model estimation and selection process.

The following *options* control the input data sets used by PROC SEVERITY and various forms of output generated by PROC SEVERITY. The *options* are listed in alphabetical order.

COVOUT

specifies that the OUTEST= data set contain the estimate of the covariance structure of the parameters. This option has no effect if you do not specify the OUTEST= option. For more information about how the covariance is reported in the OUTEST= data set, see the section "OUTEST= Data Set" on page 2179.

DATA=*SAS-data-set*

names the input data set. If you do not specify the DATA= option, then the most recently created SAS data set is used.

EDFALPHA=*confidence-level*

specifies the confidence level in the (0,1) range that is used for computing the confidence intervals for the EDF estimates. The lower and upper confidence limits that correspond to this level are reported in the OUTCDF= data set, if specified, and are displayed in the plot that is created when you specify the PLOTS=CDFPERDIST option.

If you do not specify the EDFALPHA= option, then PROC SEVERITY uses a default value of 0.05.

INEST=*SAS-data-set*

names the input data set that contains the initial values of the parameter estimates to start the optimization process. The initial values that you specify in the INIT= option in the DIST statement take precedence over any initial values that you specify in the INEST= data set. For more information about the variables in this data set, see the section "INEST= Data Set" on page 2177.

If you specify the SCALEMODEL statement, then PROC SEVERITY reads the INEST= data set only if the SCALEMODEL statement contains singleton continuous effects. For more generic regression effects, you should save the estimates by specifying the OUTSTORE= item store in a step and then use the INSTORE= option to read those estimates. The INSTORE= option is the newer and more flexible method of specifying initial values for distribution and regression parameters.

INITSAMPLE (*initsample-option***)**

INITSAMPLE (*initsample-option . . . initsample-option***)**

specifies that a sample of the input data be used for initializing the distribution parameters. If you specify more than one *initsample-option*, then separate them with spaces.

When you do not specify initial values for the distribution parameters, PROC SEVERITY needs to compute the empirical distribution function (EDF) estimates as part of the default method for parameter initialization. The EDF estimation process can be expensive, especially when you specify censoring or truncation effects for the loss variable. Furthermore, it is not amenable to parallelism due to the sequential nature of the algorithm for truncation effects. You can use the INITSAMPLE option to specify that only a fraction of the input data be used in order to reduce the time taken to compute the EDF estimates. PROC SEVERITY uses the uniform random sampling method to select the sample, the size and randomness of which is controlled by the following *initsample-options*:

FRACTION=*number*

specifies the fraction, between 0 and 1, of the input data to be used for sampling.

SEED=*number*

specifies the seed to be used for the uniform random number generator. This option enables you to select the same sample from the same input data across different runs of PROC SEVERITY, which can be useful for replicating the results across different runs. If you do not specify the seed value, PROC SEVERITY generates a seed that is based on the system clock.

SIZE=*number*

specifies the size of the sample. If the data are distributed across different nodes, then this size applies to the sample that is prepared at each node. For example, let the input data set of size 100,000 observations be distributed across 10 nodes such that each node has 10,000 observations. If you specify SIZE=1000, then each node computes a local EDF estimate by using a sample of size 1,000 selected randomly from its 10,000 observations. If you specify both of the SIZE= and FRACTION= options, then the value that you specify in the SIZE= option is used and the FRACTION= option is ignored.

If you do not specify the INITSAMPLE option, then PROC SEVERITY computes the EDF estimates by using all valid observations in the DATA= data set, or by using all valid observations in the current BY group if you specify a BY statement.

INSTORE=*store-name* (Experimental)
names the item store that contains the context and results of the severity model estimation process. An item store has a binary file format that cannot be modified. You must specify an item store that you have created in another PROC SEVERITY step by using the OUTSTORE= option.

The *store-name* is a usual one- or two-level SAS name, as for SAS data sets. If you specify a one-level name, then PROC SEVERITY reads the item store from the WORK library. If you specify a two-level name of the form *libname.membername*, then PROC SEVERITY reads the item store from the *libname* library.

This option is more flexible than the INEST= option, because it can read estimates of any type of scale regression model; the INEST= option can read only scale regression models that contain singleton continuous effects.

For more information about how the input item store is used for parameter initialization, see the sections "Parameter Initialization" on page 2122 and "Parameter Initialization for Regression Models" on page 2124.

NAMELEN=*number*
specifies the length to which long regression effect names are shortened. The default and minimum value is 20.

This option does not apply to the names of singleton continuous effects if you have not specified any CLASS variables.

NOCLPRINT<=*number***>**
suppresses the display of the "Class Level Information" table if you do not specify *number*. If you specify *number*, the values of the classification variables are displayed for only those variables whose number of levels is less than *number*. Specifying a *number* helps to reduce the size of the "Class Level Information" table if some classification variables have a large number of levels. This option has no effect if you do not specify the CLASS statement.

NOPRINT
turns off all displayed and graphical output. If you specify this option, then any value that you specify for the PRINT= and PLOTS= options is ignored.

OUTCDF=*SAS-data-set*
names the output data set to contain estimates of the cumulative distribution function (CDF) value at each of the observations. The information is output for each specified model whose parameter estimation process converges. The data set also contains the estimates of the empirical distribution function (EDF). For more information about the variables in this data set, see the section "OUTCDF= Data Set" on page 2178.

OUTEST=*SAS-data-set*
names the output data set to contain estimates of the parameter values and their standard errors for each model whose parameter estimation process converges. For more information about the variables in this data set, see the section "OUTEST= Data Set" on page 2179.

If you specify the SCALEMODEL statement such that it contains at least one effect that is not a singleton continuous effect, then the OUTEST= data set that this option creates cannot be used as an INEST= data set in a subsequent PROC SEVERITY step. In such cases, it is recommended that you use the newer OUTSTORE= option to save the estimates and specify those estimates in a subsequent PROC SEVERITY step by using the INSTORE= option.

OUTMODELINFO=*SAS-data-set*
names the output data set to contain the information about each candidate distribution. For more information about the variables in this data set, see the section "OUTMODELINFO= Data Set" on page 2180.

OUTSTAT=*SAS-data-set*
names the output data set to contain the values of statistics of fit for each model whose parameter estimation process converges. For more information about the variables in this data set, see the section "OUTSTAT= Data Set" on page 2181.

OUTSTORE=*store-name* (Experimental)
names the item store to contain the context and results of the severity model estimation process. The resulting item store has a binary file format that cannot be modified. You can specify this item store in a subsequent PROC SEVERITY step by using the INSTORE= option.

The *store-name* is a usual one- or two-level SAS name, as for SAS data sets. If you specify a one-level name, then the item store resides in the WORK library and is deleted at the end of the SAS session. Because item stores are meant to be consumed by a subsequent PROC SEVERITY step for parameter initialization, typical usage specifies a two-level name of the form *libname.membername*.

This option is more useful than the OUTEST= option, especially when you specify a scale regression model that contains interaction effects or effects that have CLASS variables. You can initialize such scale regression models in a subsequent PROC SEVERITY step only by specifying the item store that this option creates as an INSTORE= item store in that step.

PLOTS <(*global-plot-options***)> <=***plot-request-option***>**
PLOTS <(*global-plot-options***)> <=(***plot-request-option . . . plot-request-option***)>**
specifies the desired graphical output. If you specify more than one *global-plot-option*, then separate them with spaces and enclose them in parentheses. If you specify more than one *plot-request-option*, then separate them with spaces and enclose them in parentheses.

You can specify the following *global-plot-options*:

HISTOGRAM
plots the histogram of the response variable on the PDF plots.

KERNEL
plots the kernel estimate of the probability density of the response variable on the PDF plots.

ONLY
turns off the default graphical output and creates only the requested plots.

You can specify the following *plot-request-options*:

ALL
creates all the graphical output.

CDF
creates a plot that compares the cumulative distribution function (CDF) estimates of all the candidate distribution models to the empirical distribution function (EDF) estimate. The plot does not contain CDF estimates for models whose parameter estimation process does not converge.

CDFPERDIST
creates a plot of the CDF estimates of each candidate distribution model. A plot is not created for models whose parameter estimation process does not converge.

NONE
creates none of the graphical output. If you specify this option, then it overrides all the other *plot-request-options*. The default graphical output is also suppressed.

PDF
creates a plot that compares the probability density function (PDF) estimates of all the candidate distribution models. The plot does not contain PDF estimates for models whose parameter estimation process does not converge.

PDFPERDIST
creates a plot of the PDF estimates of each candidate distribution model. A plot is not created for models whose parameter estimation process does not converge.

PP
creates the probability-probability plot (known as the P-P plot), which compares the CDF estimate of each candidate distribution model to the empirical distribution function (EDF). The data that are shown in this plot are used for computing the EDF-based statistics of fit.

QQ
creates the quantile-quantile plot (known as the Q-Q plot), which compares the empirical quantiles to the quantiles of each candidate distribution model.

If you do not specify the PLOTS= option or if you do not specify the ONLY *global-plot-option*, then the default graphical output is equivalent to specifying PLOTS(HISTOGRAM KERNEL)=(CDF PDF).

PRINT <(*global-display-option***)> <=***display-option***>**

PRINT <(*global-display-option***)> <= (***display-option . . . display-option***) >**
specifies the desired displayed output. If you specify more than one *display-option*, then separate them with spaces and enclose them in parentheses.

You can specify the following *global-display-option*:

ONLY
turns off the default displayed output and displays only the requested output.

You can specify the following *display-options*:

ALL
displays all the output.

ALLFITSTATS
displays the comparison of all the statistics of fit for all the models in one table. The table does not include the models whose parameter estimation process does not converge.

CONVSTATUS
displays the convergence status of the parameter estimation process.

DESCSTATS
displays the descriptive statistics for the response variable. If you specify the SCALEMODEL statement, then this option also displays the descriptive statistics for the regression effects that do not contain a CLASS variable.

DISTINFO
displays the information about each specified distribution. For each distribution, the information includes the name, description, validity status, and number of distribution parameters.

ESTIMATES | PARMEST
displays the final estimates of parameters. The estimates are not displayed for models whose parameter estimation process does not converge.

INITIALVALUES
displays the initial values and bounds used for estimating each model.

NLOHISTORY
displays the iteration history of the nonlinear optimization process used for estimating the parameters.

NLOSUMMARY
displays the summary of the nonlinear optimization process used for estimating the parameters.

NONE
displays none of the output. If you specify this option, then it overrides all other display options. The default displayed output is also suppressed.

SELECTION | SELECT
displays the model selection table.

STATISTICS | FITSTATS
displays the statistics of fit for each model. The statistics of fit are not displayed for models whose parameter estimation process does not converge.

If you do not specify the PRINT= option or if you do not specify the ONLY *global-display-option*, then the default displayed output is equivalent to specifying PRINT=(SELECTION CONVSTATUS NLOSUMMARY STATISTICS ESTIMATES).

VARDEF=DF | N
specifies the denominator to use for computing the covariance estimates. You can specify one of the following values:

DF
specifies that the number of nonmissing observations minus the model degrees of freedom (number of parameters) be used.

N
specifies that the number of nonmissing observations be used.

For more information about the covariance estimation, see the section "Estimating Covariance and Standard Errors" on page 2121.

The following *options* control the model estimation and selection process:

CRITERION | CRITERIA | CRIT=*criterion-option*
specifies the model selection criterion.

If you specify two or more candidate models for estimation, then the one with the best value for the selection criterion is chosen as the best model. If you specify the OUTSTAT= data set, then the best model's observation has a value of 1 for the _SELECTED_ variable.

You can specify one of the following *criterion-options*:

AD
specifies the Anderson-Darling (AD) statistic value, which is computed by using the empirical distribution function (EDF) estimate, as the selection criterion. A lower value is deemed better.

AIC
specifies Akaike's information criterion (AIC) as the selection criterion. A lower value is deemed better.

AICC
specifies the finite-sample corrected Akaike's information criterion (AICC) as the selection criterion. A lower value is deemed better.

BIC
specifies the Schwarz Bayesian information criterion (BIC) as the selection criterion. A lower value is deemed better.

CUSTOM
specifies the custom objective function as the selection criterion. You can specify this only if you also specify the OBJECTIVE= option. A lower value is deemed better.

CVM
specifies the Cramér-von Mises (CvM) statistic value, which is computed by using the empirical distribution function (EDF) estimate, as the selection criterion. A lower value is deemed better.

KS
specifies the Kolmogorov-Smirnov (KS) statistic value, which is computed by using the empirical distribution function (EDF) estimate, as the selection criterion. A lower value is deemed better.

LOGLIKELIHOOD | LL
specifies $-2 * \log(L)$ as the selection criterion, where L is the likelihood of the data. A lower value is deemed better. This is the default.

For more information about these *criterion-options*, see the section "Statistics of Fit" on page 2144.

EMPIRICALCDF | EDF=*method*
specifies the method to use for computing the nonparametric or empirical estimate of the cumulative distribution function of the data. You can specify one of the following values for *method*:

AUTOMATIC | AUTO
specifies that the method be chosen automatically based on the data specification. This option is the default.

If you do not specify any censoring or truncation, then the standard empirical estimation method (STANDARD) is chosen. If you specify both right-censoring and left-censoring, then Turnbull's estimation method (TURNBULL) is chosen. For all other combinations of censoring and truncation, the Kaplan-Meier method (KAPLANMEIER) is chosen.

KAPLANMEIER | KM
specifies that the product limit estimator proposed by Kaplan and Meier (1958) be used. Specification of this method has no effect when you specify both right-censoring and left-censoring.

MODIFIEDKM | MKM <(*options***)>**
specifies that the modified product limit estimator be used. Specification of this method has no effect when you specify both right-censoring and left-censoring.

This method allows Kaplan-Meier's product limit estimates to be more robust by ignoring the contributions to the estimate due to small risk-set sizes. The risk set is the set of observations at the risk of failing, where an observation is said to fail if it has not been processed yet and might experience censoring or truncation. You can specify the minimum risk-set size that makes it eligible to be included in the estimation either as an absolute lower bound on the size (RSLB= option) or a relative lower bound determined by the formula cn^{α} proposed by Lai and Ying (1991). You can specify the values of c and α by using the C= and ALPHA= options, respectively. By default, the relative lower bound is used with values of $c = 1$ and $\alpha = 0.5$. However, you can modify the default by using the following *options*:

ALPHA | A=*number*
specifies the value to use for α when the lower bound on the risk set size is defined as cn^{α}. This value must satisfy $0 < \alpha < 1$.

C=*number*
specifies the value to use for c when the lower bound on the risk set size is defined as cn^{α}. This value must satisfy $c > 0$.

RSLB=*number*
specifies the absolute lower bound on the risk set size to be included in the estimate.

STANDARD | STD
specifies that the standard empirical estimation method be used. If you specify both right-censoring and left-censoring, then the specification of this method has no effect. If you specify

any other combination of censoring or truncation effects, then this method ignores such effects, and can thus result in estimates that are more biased than those obtained with other methods that are more suitable for censored or truncated data.

TURNBULL | EM <(*options*)>
specifies that the Turnbull's method be used. This method is used when you specify both right-censoring and left-censoring. An iterative expectation-maximization (EM) algorithm proposed by Turnbull (1976) is used to compute the empirical estimates. If you also specify truncation, then the modification suggested by Frydman (1994) is used. You can modify the default behavior of the EM algorithm by using the following *options*:

ENSUREMLE
specifies that the final EDF estimates be maximum likelihood estimates. The Kuhn-Tucker conditions are computed for the likelihood maximization problem and checked to ensure that EM algorithm converges to maximum likelihood estimates. The method generalizes the method proposed by Gentleman and Geyer (1994) by taking into account any truncation information that you might specify.

EPS=*number*
specifies the maximum relative error to be allowed between estimates of two consecutive iterations. This criterion is used to check the convergence of the algorithm. If you do not specify this option, then PROC SEVERITY uses a default value of 1.0E–8.

MAXITER=*number*
specifies the maximum number of iterations to attempt to find the empirical estimates. If you do not specify this option, then PROC SEVERITY uses a default value of 500.

ZEROPROB=*number*
specifies the threshold below which an empirical estimate of the probability is considered zero. This option is used to decide if the final estimate is a maximum likelihood estimate. This option does not have an effect if you do not specify the ENSUREMLE option. If you specify the ENSUREMLE option, but do not specify this option, then PROC SEVERITY uses a default value of 1.0E–8.

For more information about each of the methods, see the section "Empirical Distribution Function Estimation Methods" on page 2138.

OBJECTIVE=*symbol-name*
names the symbol that represents the objective function in the SAS programming statements that you specify. For each model to be estimated, PROC SEVERITY executes the programming statements to compute the value of this symbol for each observation. The values are added across all observations to obtain the value of the objective function. The optimization algorithm estimates the model parameters such that the objective function value is *minimized*. A separate optimization problem is solved for each candidate distribution. If you specify a BY statement, then a separate optimization problem is solved for each candidate distribution within each BY group.

For more information about writing SAS programming statements to define your own objective function, see the section "Custom Objective Functions" on page 2173.

BY Statement

BY *variable-list* **;**

A BY statement can be used in the SEVERITY procedure to process the input data set in groups of observations defined by the BY variables.

If you specify the BY statement, then PROC SEVERITY expects the input data set to be sorted in the order of the BY variables unless you specify the NOTSORTED option.

CLASS Statement

CLASS *variable* < *(options)* > . . . < *variable* < *(options)* > > < / *global-options* > **;**

The CLASS statement names the classification variables to be used in the scale regression model. These variables enter the analysis not through their values, but through levels to which the unique values are mapped. For more information about these mappings, see the section "Levelization of Classification Variables" on page 2128.

If you specify a CLASS statement, then it must precede the SCALEMODEL statement.

You can specify options either as individual variable *options* or as *global-options*. You can specify *options* for each variable by enclosing the options in parentheses after the variable name. You can also specify *global-options* for the CLASS statement by placing them after a slash (/). *Global-options* are applied to all the variables that you specify in the CLASS statement. If you specify more than one CLASS statement, the *global-options* that are specified in any one CLASS statement apply to all CLASS statements. However, individual CLASS variable *options* override the *global-options*.

You can specify the following values for either an *option* or a *global-option*:

DESCENDING

DESC

reverses the sort order of the classification variable. If you specify both the DESCENDING and ORDER= options, the SEVERITY procedure orders the levels of classification variables according to the ORDER= option and then reverses that order.

ORDER=DATA | FORMATTED | INTERNAL

ORDER=FREQ | FREQDATA | FREQFORMATTED | FREQINTERNAL

specifies the sort order for the levels of classification variables. This order is used by the parameterization method to create the parameters in the model. By default, ORDER=FORMATTED. For ORDER=FORMATTED and ORDER=INTERNAL, the sort order is machine-dependent. When ORDER=FORMATTED is in effect for numeric variables for which you have supplied no explicit format, the levels are ordered by their internal values.

The following table shows how the SEVERITY procedure interprets values of the ORDER= option.

Value of ORDER=	Levels Sorted By
DATA	Order of appearance in the input data set
FORMATTED	External formatted values, except for numeric variables that have no explicit format, which are sorted by their unformatted (internal) values
FREQ	Descending frequency count (levels that have more observations come earlier in the order)
FREQDATA	Order of descending frequency count, and within counts by order of appearance in the input data set when counts are tied
FREQFORMATTED	Order of descending frequency count, and within counts by formatted value when counts are tied
FREQINTERNAL	Order of descending frequency count, and within counts by unformatted (internal) value when counts are tied
INTERNAL	Unformatted value

For more information about sort order, see the chapter about the SORT procedure in *Base SAS Procedures Guide* and the discussion of BY-group processing in *SAS Language Reference: Concepts*.

REF=*'level'* | *keyword*

REFERENCE=*'level'* | *keyword*

specifies the reference level that is used when you specify PARAM=REFERENCE. For an individual (but not a global) variable REF= *option*, you can specify the *level* of the variable to use as the reference level. Specify the formatted value of the variable if a format is assigned. For a REF= *option* or *global-option*, you can use one of the following *keywords*.

FIRST designates the first-ordered level as reference.

LAST designates the last-ordered level as reference.

By default, REF=LAST.

If you choose a reference level for any CLASS variable, all variables are parameterized in the reference parameterization for computational efficiency. In other words, the SEVERITY procedure applies a single parameterization method to all classification variables.

Suppose that the variable temp has three levels (`'hot'`, `'warm'`, and `'cold'`) and that the variable gender has two levels (`'M'` and `'F'`). The following statements fit a scale regression model:

```
proc severity;
   loss y;
   class gender(ref='F') temp;
   scalemodel gender*temp gender;
run;
```

Both CLASS variables are in reference parameterization in this model. The reference levels are `'F'` for the variable gender and `'warm'` for the variable temp, because the statements are equivalent to the following statements:

```
proc severity;
   loss y;
   class gender(ref='F') temp(ref=last);
   scalemodel gender*temp gender;
run;
```

You can specify the following *global-options*:

MISSING
treats missing values (".", ".A", . . . , ".Z" for numeric variables and blanks for character variables) as valid values for the CLASS variable.

If you do not specify the MISSING option, observations that have missing values for CLASS variables are removed from the analysis, even if the CLASS variables are not used in the model formulation.

PARAM=*keyword*
specifies the parameterization method for the classification variable or variables. You can specify the following *keywords*:

GLM specifies a less-than-full-rank reference cell coding.

REFERENCE specifies a reference cell encoding. You can choose the reference value by specifying an option for a specific *variable* or set of *variables* in the CLASS statement, or you can designate the first- or last-ordered value by specifying a *global-option*. By default, REFERENCE=LAST.

The GLM parameterization is the default. For more information about how parameterization of classification variables affects the construction and interpretation of model effects, see the section "Specification and Parameterization of Model Effects" on page 2131.

TRUNCATE< =*n* **>**
specifies the truncation width of formatted values of CLASS variables when the optional *n* is specified.

If *n* is not specified, the TRUNCATE option requests that classification levels be determined by using no more than the first 16 characters of the formatted values of CLASS variables.

DIST Statement

DIST *distribution-name-or-keyword* < *(distribution-option)* < *distribution-name-or-keyword* < *(distribution-option)* > > . . . > < / *preprocess-options* > **;**

The DIST statement specifies candidate distributions to be estimated by the SEVERITY procedure. You can specify multiple DIST statements, and each statement can contain one or more distribution specifications.

For your convenience, PROC SEVERITY provides the following 10 different predefined distributions (the name in the parentheses is the name to use in the DIST statement): Burr (BURR), exponential (EXP), gamma (GAMMA), generalized Pareto (GPD), inverse Gaussian or Wald (IGAUSS), lognormal (LOGN), Pareto (PARETO), Tweedie (TWEEDIE), scaled Tweedie (STWEEDIE), and Weibull (WEIBULL). These are described in detail in the section "Predefined Distributions" on page 2108.

You can specify any of the predefined distributions or any distribution that you have defined. If a distribution that you specify is not a predefined distribution, then you must submit the CMPLIB= system option with appropriate libraries before you submit the PROC SEVERITY step to enable the procedure to find the functions associated with your distribution. The predefined distributions are defined in the Sashelp.Svrtdist library. However, you are not required to specify this library in the CMPLIB= system option. For more information about defining your own distributions, see the section "Defining a Severity Distribution Model with the FCMP Procedure" on page 2148.

As a convenience, you can also use a shortcut keyword to indicate a list of distributions. You can specify one or more of the following keywords:

ALL
: specifies all the predefined distributions and the distributions that you have defined in the libraries that you specify in the CMPLIB= system option. In addition to the eight predefined distributions included by the _PREDEFINED_ keyword, this list also includes the Tweedie and scaled Tweedie distributions that are defined in the Sashelp.Svrtdist library.

PREDEFINED
: specifies the list of eight predefined distributions: BURR, EXP, GAMMA, GPD, IGAUSS, LOGN, PARETO, and WEIBULL. Although the TWEEDIE and STWEEDIE distributions are available in the Sashelp.Svrtdist library along with these eight distributions, they are not included by this keyword. If you want to fit the TWEEDIE and STWEEDIE distributions, then you must specify them explicitly or use the _ALL_ keyword.

USER
: specifies the list of all the distributions that you have defined in the libraries that you specify in the CMPLIB= system option. This list does not include the distributions defined in the Sashelp.Svrtdist library, even if you specify the Sashelp.Svrtdist library in the CMPLIB= option.

The use of these keywords, especially _ALL_, can result in a large list of distributions, which might take a longer time to estimate. A warning is printed to the SAS log if the number of total distribution models to estimate exceeds 10.

If you specify the OUTCDF= option or request a CDF plot and you do not specify any DIST statement, then PROC SEVERITY does not fit any distributions and produces the empirical estimates of the cumulative distribution function.

The following *distribution-option* values can be used in the DIST statement for a distribution name that is not a shortcut keyword:

INIT=*(name=value . . . name=value)*
: specifies the initial values to be used for the distribution parameters to start the parameter estimation process. You must specify the values by parameter names and the parameter names must match the names used in the model definition. For example, let a model M's definition contain a M_PDF function with following signature:

```
function M_PDF(x, alpha, beta);
```

: For this model, the names **alpha** and **beta** must be used for the INIT option. The names are case-insensitive. If you do not specify initial values for some parameters in the INIT statement, then a default value of 0.001 is assumed for those parameters. If you specify an incorrect parameter, PROC SEVERITY prints a warning to the SAS log and does not fit the model. All specified values must be nonmissing.

If you are modeling regression effects, then the initial value of the first distribution parameter (**alpha** in the preceding example) should be the initial *base* value of the scale parameter or log-transformed scale parameter. For more information, see the section "Estimating Regression Effects" on page 2123.

The use of INIT= option is one of the three methods available for initializing the parameters. For more information, see the section "Parameter Initialization" on page 2122. If none of the initialization methods is used, then PROC SEVERITY initializes all parameters to 0.001.

You can specify the following *preprocess-options* in the DIST statement:

LISTONLY
specifies that the list of all candidate distributions be printed to the SAS log without doing any further processing on them. This option is especially useful when you use a shortcut keyword to include a list of distributions. It enables you to find out which distributions are included by the keyword.

VALIDATEONLY
specifies that all candidate distributions be checked for validity without doing any further processing on them. If a distribution is invalid, the reason for invalidity is written to the SAS log. If all distributions are valid, then the distribution information is written to the SAS log. The information includes name, description, validity status (valid or invalid), and number of distribution parameters. The information is not written to the SAS log if you specify an OUTMODELINFO= data set or the PRINT=DISTINFO or PRINT=ALL option in the PROC SEVERITY statement. This option is especially useful when you specify your own distributions or when you specify the _USER_ or _ALL_ keywords in the DIST statement. It enables you to check whether your custom distribution definitions satisfy PROC SEVERITY's requirements for the specified modeling task. It is recommended that you specify the SCALEMODEL statement if you intend to fit a model with regression effects, because the SCALEMODEL statement instructs PROC SEVERITY to perform additional checks to validate whether regression effects can be modeled on each candidate distribution.

LOSS Statement

LOSS < *response-variable-name* > < / *censoring-truncation-options* > **;**

The LOSS statement specifies the name of the response or loss variable whose distribution needs to be modeled. You can also specify additional options to indicate any truncation or censoring of the response. The specification of response variable is optional if you specify at least one type of censoring. You must specify a response variable if you do not specify any censoring. If you specify more than one LOSS statement, then the first statement is used.

All the analysis variables that you specify in this statement must be present in the input data set that you specify by using the DATA= option in the PROC SEVERITY statement. The response variable is expected to have nonmissing values. If the variable has a missing value in an observation, then a warning is written to the SAS log and that observation is ignored.

The following *censoring-truncation-options* can be used in the LOSS statement:

LEFTCENSORED | LC=*variable-name*

LEFTCENSORED | LC=*number*

specifies the left-censoring variable or a global left-censoring limit.

You can use the *variable-name* argument to specify a data set variable that contains the left-censoring limit. If the value of this variable is missing, then PROC SEVERITY assumes that such observations are not left-censored.

Alternatively, you can use the *number* argument to specify a left-censoring limit value that applies to all the observations in the data set. This limit must be a nonzero positive number.

By the definition of left-censoring, an exact value of the response is not known when it is less than or equal to the left-censoring limit. If you specify the response variable and the value of that variable is less than or equal to the value of the left-censoring limit for some observations, then PROC SEVERITY treats such observations as left-censored and the value of the response variable is ignored. If you specify the response variable and the value of that variable is greater than the value of the left-censoring limit for some observations, then PROC SEVERITY assumes that such observations are not left-censored and the value of the left-censoring limit is ignored.

If you specify both right-censoring and left-censoring limits, then the left-censoring limit must be greater than or equal to the right-censoring limit. If both limits are identical, then the observation is assumed to be uncensored.

For more information about left-censoring, see the section "Censoring and Truncation" on page 2118.

LEFTTRUNCATED | LT=*variable-name* < *(left-truncation-option)* >

LEFTTRUNCATED | LT=*number* < *(left-truncation-option)* >

specifies the left-truncation variable or a global left-truncation threshold.

You can use the *variable-name* argument to specify a data set variable that contains the left-truncation threshold. If the value of this variable is missing or 0 for some observations, then PROC SEVERITY assumes that such observations are not left-truncated.

Alternatively, you can use the *number* argument to specify a left-truncation threshold that applies to all the observations in the data set. This threshold must be a nonzero positive number.

It is assumed that the response variable contains the observed values. By the definition of left-truncation, you can observe only a value that is greater than the left-truncation threshold. If a response variable value is less than or equal to the left-truncation threshold, a warning is printed to the SAS log, and the observation is ignored. For more information about left-truncation, see the section "Censoring and Truncation" on page 2118.

You can specify the following *left-truncation-option* for an alternative interpretation of the left-truncation threshold:

PROBOBSERVED | POBS=*number*

specifies the probability of observability, which is defined as the probability that the underlying severity event is observed (and recorded) for the specified left-threshold value.

The specified *number* must lie in the (0.0, 1.0] interval. A value of 1.0 is equivalent to specifying that there is no left-truncation, because it means that no severity events can occur with a value less than or equal to the threshold. If you specify value of 1.0, PROC SEVERITY prints a warning to the SAS log and proceeds by assuming that LEFTTRUNCATED= option is not specified.

For more information, see the section "Probability of Observability" on page 2119.

RIGHTCENSORED | RC=*variable-name*

RIGHTCENSORED | RC=*number*

specifies the right-censoring variable or a global right-censoring limit.

You can use the *variable-name* argument to specify a data set variable that contains the right-censoring limit. If the value of this variable is missing, then PROC SEVERITY assumes that such observations are not right-censored.

Alternatively, you can use the *number* argument to specify a right-censoring limit value that applies to all the observations in the data set. This limit must be a nonzero positive number.

By the definition of right-censoring, an exact value of the response is not known when it is greater than or equal to the right-censoring limit. If you specify the response variable and the value of that variable is greater than or equal to the value of the right-censoring limit for some observations, then PROC SEVERITY treats such observations as right-censored and the value of the response variable is ignored. If you specify the response variable and the value of that variable is less than the value of the right-censoring limit for some observations, then PROC SEVERITY assumes that such observations are not right-censored and the value of the right-censoring limit is ignored.

If you specify both right-censoring and left-censoring limits, then the left-censoring limit must be greater than or equal to the right-censoring limit. If both limits are identical, then the observation is assumed to be uncensored.

For more information about right-censoring, see the section "Censoring and Truncation" on page 2118.

RIGHTTRUNCATED | RT=*variable-name*

RIGHTTRUNCATED | RT=*number*

specifies the right-truncation variable or a global right-truncation threshold.

You can use the *variable-name* argument to specify a data set variable that contains the right-truncation threshold. If the value of this variable is missing for some observations, then PROC SEVERITY assumes that such observations are not right-truncated.

Alternatively, you can use the *number* argument to specify a right-truncation threshold that applies to all the observations in the data set. This threshold must be a nonzero positive number.

It is assumed that the response variable contains the observed values. By the definition of right-truncation, you can observe only a value that is less than or equal to the right-truncation threshold. If a response variable value is greater than the right-truncation threshold, a warning is printed to the SAS log, and the observation is ignored. For more information about right-truncation, see the section "Censoring and Truncation" on page 2118.

NLOPTIONS Statement

NLOPTIONS *options* **;**

The SEVERITY procedure uses the nonlinear optimization (NLO) subsystem to perform the nonlinear optimization of the likelihood function to obtain the estimates of distribution and regression parameters. You can use the NLOPTIONS statement to control different aspects of this optimization process. For most problems, the default settings of the optimization process are adequate. However, in some cases it might be useful to change the optimization technique or to change the maximum number of iterations. The following statement uses the MAXITER= option to set the maximum number of iterations to 200 and uses the TECH= option to change the optimization technique to the double-dogleg optimization (DBLDOG) rather than the default technique, the trust region optimization (TRUREG), that is used in the SEVERITY procedure:

```
nloptions tech=dbldog maxiter=200;
```

A discussion of the full range of *options* that can be used in the NLOPTIONS statement is given in Chapter 7, "Nonlinear Optimization Methods." The SEVERITY procedure supports all those options except the options that are related to displaying the optimization information. You can use the PRINT= option in the PROC SEVERITY statement to request the optimization summary and iteration history. If you specify more than one NLOPTIONS statement, then the first statement is used.

OUTSCORELIB Statement

OUTSCORELIB < **OUTLIB**=> *fcmp-library-name options* **;**

The OUTSCORELIB statement specifies the library to write scoring functions to. Scoring functions enable you to easily compute a distribution function on the fitted parameters of the distribution without going through a potentially complex process of extracting the fitted parameter estimates from other output such as the OUTEST= data set that is created by PROC SEVERITY.

If you specify the SCALEMODEL statement and if you specify interaction or classification effects, then PROC SEVERITY ignores the OUTSCORELIB statement and does not generate scoring functions. In other words, if you specify the SCALEMODEL statement, then PROC SEVERITY generates scoring functions if you specify only singleton continuous effects in the SCALEMODEL statement.

You must specify the following option as the first option in the statement:

OUTLIB=*fcmp-library-name*
names the FCMP library to contain the scoring functions. PROC SEVERITY writes the scoring functions to the FCMP library named *fcmp-library-name*. If a library or data set named *fcmp-library-name* already exists, PROC SEVERITY deletes it before proceeding.

This option is similar to the OUTLIB= option that you would specify in a PROC FCMP statement, except that *fcmp-library-name* must be a two-level name whereas the OUTLIB= option in the PROC FCMP statement requires a three-level name. The third level of a three-level name specifies the package to which the functions belong. You do not need to specify the package name in the *fcmp-library-name*, because PROC SEVERITY automatically creates the package for you. By default, a separate package is created for each distribution that has not failed to converge. Each package is named for a distribution.

For example, if you define and fit a distribution named *mydist*, and if *mydist* does not fail to converge, then PROC SEVERITY creates a package named *mydist* in the OUTLIB= library that you specify. Further, let the definition of the *mydist* distribution contain three distribution functions, *mydist_PDF(x,Parm1,Parm2)*, *mydist_LOGCDF(x,Parm1,Parm2)*, and *mydist_XYZ(x,Parm1,Parm2)*. If you specify the OUTSCORELIB statement

```
outscorelib outlib=sasuser.scorefunc;
```

then the Sasuser.Scorefunc library contains the following three functions in a package named *mydist*: *SEV_PDF(x)*, *SEV_LOGCDF(x)*, and *SEV_XYZ(x)*.

The key feature of scoring functions is that they do not require the parameter arguments (*Parm1* and *Parm2* in this example). The fitted parameter estimates are encoded inside the scoring function so that you can compute or score the value of each function for a given value of the loss variable without having to know or extract the parameter estimates through some other means.

For convenience, you can omit the OUTLIB= portion of the specification and just specify the name, as in the following example:

```
outscorelib sasuser.scorefunc;
```

When the SEVERITY procedure runs successfully, the *fcmp-library-name* is appended to the CMPLIB system option, so you can immediately start using the scoring functions in a DATA step or PROC FCMP step.

You can specify the following *options* in the OUTSCORELIB statement:

COMMONPACKAGE

ONEPACKAGE

requests that only one common package be created to contain all the scoring functions.

If you specify this option, then all the scoring functions are created in a package called *sevfit*. For each distribution function that has the name *distribution_suffix*, the name of the corresponding scoring function is formed as *SEV_suffix_distribution*. For example, the scoring function of the distribution function 'MYDIST_BAR' is named 'SEV_BAR_MYDIST'.

If you do not specify this option, then all scoring functions for a distribution are created in a package that has the same name as the distribution, and for each distribution function that has the name *distribution_suffix*, the name of the corresponding scoring function is formed as *SEV_suffix*. For example, the scoring function of the distribution function 'MYDIST_BAR' is named 'SEV_BAR'.

OUTBYID=*SAS-data-set*

names the output data set to contain the unique identifier for each BY group. This unique identifier is used as part of the name of the package or scoring function for each distribution. This is a required option when you specify a BY statement in PROC SEVERITY.

The OUTBYID= data set contains one observation per BY group and a variable named _ID_ in addition to the BY variables that you specify in the BY statement. The _ID_ variable contains the unique identifier for each BY group. The identifier of the BY group is the decimal representation of the sequence number of the BY group. The first BY group has an identifier of 1, the second BY group has an identifier of 2, the tenth BY group has an identifier of 10, and so on.

If you do not specify the COMMONPACKAGE option in the OUTSCORELIB statement, then for each distribution, PROC SEVERITY creates as many packages as the number of BY groups. The unique BY-group identifier is used as a suffix for the package name. For example, if your DATA= data set has three BY groups and if you specify the OUTSCORELIB statement

```
outscorelib outlib=sasuser.byscorefunc outbyid=sasuser.byid;
```

then for the distribution 'MYDIST', the Sasuser.Byscorefunc library contains the three packages 'MYDIST1', 'MYDIST2', and 'MYDIST3', and each package contains one scoring function named 'SEV_BAR' for each distribution function named 'MYDIST_BAR'.

If you specify the COMMONPACKAGE option in the OUTSCORELIB statement, PROC SEVERITY creates as many versions of the distribution function as the number of BY groups. The unique BY-group identifier is used as a suffix for the function name. Extending the previous example, if you specify the OUTSCORELIB statement with the COMMONPACKAGE option,

```
outscorelib outlib=sasuser.byscorefunc outbyid=sasuser.byid commonpackage;
```

then for the distribution function 'MYDIST_BAR' of the distribution 'MYDIST', the Sasuser.Byscorefunc library contains the following three scoring functions: 'SEV_BAR_MYDIST1', 'SEV_BAR_MYDIST2', and 'SEV_BAR_MYDIST3'. All the scoring functions are created in one common package named *sevfit*.

For both the preceding examples, the Sasuser.Byid data set contains three observations, one for each BY group. The value of the _ID_ variable is 1 for the first BY group, 2 for the second BY group, and 3 for the third BY group.

For more information about scoring functions, see the section "Scoring Functions" on page 2165.

SCALEMODEL Statement

SCALEMODEL *regression-effect-list* < / *scalemodel-options* > **;**

The SCALEMODEL statement specifies regression effects. A regression effect is formed from one or more regressor variables according to effect construction rules. Each regression effect forms one element of $\mathbf{X}$ in the linear model structure $\mathbf{X}\boldsymbol{\beta}$ that affects the scale parameter of the distribution. The SCALEMODEL statement in conjunction with the CLASS statement supports a rich set of effects. Effects are specified by a special notation that uses regressor variable names and operators. There are two types of regressor variables: classification (or CLASS) variables and continuous variables. Classification variables can be either numeric or character and are specified in a CLASS statement. To include CLASS variables in regression effects, you must specify the CLASS statement so that it appears before the SCALEMODEL statement. A regressor variable that is not declared in the CLASS statement is assumed to be continuous. For more information about effect construction rules, see the section "Specification and Parameterization of Model Effects" on page 2131.

All the regressor variables must be present in the input data set that you specify by using the DATA= option in the PROC SEVERITY statement. The scale parameter of each candidate distribution is linked to the linear predictor $\mathbf{X}\boldsymbol{\beta}$ that includes an intercept. If a distribution does not have a scale parameter, then a model based on that distribution is not estimated. If you specify more than one SCALEMODEL statement, then the first statement is used.

The regressor variables are expected to have nonmissing values. If any of the variables has a missing value in an observation, then a warning is written to the SAS log and that observation is ignored.

For more information about modeling regression effects, see the section "Estimating Regression Effects" on page 2123.

You can specify the following *scalemodel-options* in the SCALEMODEL statement:

DFMIXTURE=*method-name* < *(method-options)* >
specifies the method for computing representative estimates of the cumulative distribution function (CDF) and the probability density function (PDF).

When you specify regression effects, the scale of the distribution depends on the values of the regressors. For a given distribution family, each observation in the input data set implies a different scaled version of the distribution. To compute estimates of CDF and PDF that are comparable across different distribution families, PROC SEVERITY needs to construct a single representative distribution from all such distributions. You can specify one of the following *method-name* values to specify the method that is used to construct the representative distribution. For more information about each of the methods, see the section "CDF and PDF Estimates with Regression Effects" on page 2126.

FULL
specifies that the representative distribution be the mixture of N distributions such that each distribution has a scale value that is implied by each of the N observations that are used for estimation. This method is the slowest.

MEAN
specifies that the representative distribution be the one-point mixture of the distribution whose scale value is computed by using the mean of the N values of the linear predictor that are implied by the N observations that are used for estimation. If you do not specify the DFMIXTURE= option, then this method is used by default. This is also the fastest method.

QUANTILE < (K=*q***) >**
specifies that the representative distribution be the mixture of a fixed number of distributions whose scale values are computed by using the quantiles from the sample of N values of the linear predictor that are implied by the N observations that are used for estimation.

You can use the K= option to specify the number of distributions in the mixture. If you specify K=q, then the mixture contains $(q-1)$ distributions such that each distribution has as its scale one of the $(q-1)$-quantiles.

If you do not specify the K= option, then PROC SEVERITY uses the default of 2, which implies the use of a one-point mixture with a distribution whose scale value is the median of all scale values.

RANDOM < (*random-method-options***) >**
specifies that the representative distribution be the mixture of a fixed number of distributions whose scale values are computed by using the values of the linear predictor that are implied by a randomly chosen subset of the set of all observations that are used for estimation. The same subset of observations is used for each distribution family.

You can specify the following *random-method-options* to specify how the subset is chosen:

K=*r*
specifies the number of distributions to include in the mixture. If you do not specify this option, then PROC SEVERITY uses the default of 15.

SEED=*number*
specifies the seed that is used to generate the uniform random sample of observation indices. If you do not specify this option, then PROC SEVERITY generates a seed internally that is based on the current value of the system clock.

OFFSET=*offset-variable-name*
specifies the name of the offset variable in the scale regression model. An offset variable is a regressor variable whose regression coefficient is known to be 1. For more information, see the section "Offset Variable" on page 2124.

WEIGHT Statement

WEIGHT *variable-name* **;**

The WEIGHT statement specifies the name of a variable whose values represent the weight of each observation. PROC SEVERITY associates a weight of w to each observation, where w is the value of the WEIGHT variable for the observation. If the weight value is missing or less than or equal to 0, then the observation is ignored and a warning is written to the SAS log. When you do not specify the WEIGHT statement, each observation is assigned a weight of 1. If you specify more than one WEIGHT statement, then the last statement is used.

The weights are normalized so that they add up to the actual sample size. In particular, the weight of each observation is multiplied by $\frac{N}{\sum_{i=1}^{N} w_i}$, where N is the sample size.

Programming Statements

You can use a series of programming statements that use variables in the input data set that you specify in the DATA= option in the PROC SEVERITY statement to assign a value to an objective function symbol. You must specify the objective function symbol by using the OBJECTIVE= option in the PROC SEVERITY statement. If you do not specify the OBJECTIVE= option in the PROC SEVERITY statement, then the programming statements are ignored and models are estimated using the maximum likelihood method.

You can use most DATA step statements and functions in your program. Any additional functions, restrictions, and differences are listed in the section "Custom Objective Functions" on page 2173.

Details: SEVERITY Procedure

Predefined Distributions

PROC SEVERITY assumes the following model for the response variable Y

$$Y \sim \mathcal{F}(\Theta)$$

where $\mathcal{F}$ is a continuous probability distribution with parameters Θ. The model hypothesizes that the observed response is generated from a stochastic process that is governed by the distribution $\mathcal{F}$. This model is usually referred to as the error model. Given a representative input sample of response variable values, PROC SEVERITY estimates the model parameters for any distribution $\mathcal{F}$ and computes the statistics of fit for each model. This enables you to find the distribution that is most likely to generate the observed sample.

A set of predefined distributions is provided with the SEVERITY procedure. A summary of the distributions is provided in Table 30.2. For each distribution, the table lists the name of the distribution that should be used in the DIST statement, the parameters of the distribution along with their bounds, and the mathematical expressions for the probability density function (PDF) and cumulative distribution function (CDF) of the distribution.

All the predefined distributions, except LOGN and TWEEDIE, are parameterized such that their first parameter is the scale parameter. For LOGN, the first parameter μ is a log-transformed scale parameter. TWEEDIE does not have a scale parameter. The presence of scale parameter or a log-transformed scale parameter enables you to use all of the predefined distributions, except TWEEDIE, as a candidate for estimating regression effects.

A distribution model is associated with each predefined distribution. You can also define your own distribution model, which is a set of functions and subroutines that you define by using the FCMP procedure. For more information, see the section "Defining a Severity Distribution Model with the FCMP Procedure" on page 2148.

Table 30.2 Predefined SEVERITY Distributions

Name	Distribution	Parameters	PDF (f) and CDF (F)
BURR	Burr	$\theta > 0, \alpha > 0,$ $\gamma > 0$	$f(x) = \frac{\alpha\gamma z^{\gamma}}{x(1+z^{\gamma})^{(\alpha+1)}}$ $F(x) = 1 - \left(\frac{1}{1+z^{\gamma}}\right)^{\alpha}$
EXP	Exponential	$\theta > 0$	$f(x) = \frac{1}{\theta}e^{-z}$ $F(x) = 1 - e^{-z}$
GAMMA	Gamma	$\theta > 0, \alpha > 0$	$f(x) = \frac{z^{\alpha}e^{-z}}{x\Gamma(\alpha)}$ $F(x) = \frac{\gamma(\alpha,z)}{\Gamma(\alpha)}$
GPD	Generalized Pareto	$\theta > 0, \xi > 0$	$f(x) = \frac{1}{\theta}(1+\xi z)^{-1-1/\xi}$ $F(x) = 1 - (1+\xi z)^{-1/\xi}$
IGAUSS	Inverse Gaussian (Wald)	$\theta > 0, \alpha > 0$	$f(x) = \frac{1}{\theta}\sqrt{\frac{\alpha}{2\pi z^3}}\, e^{\frac{-\alpha(z-1)^2}{2z}}$ $F(x) = \Phi\left((z-1)\sqrt{\frac{\alpha}{z}}\right) +$ $\Phi\left(-(z+1)\sqrt{\frac{\alpha}{z}}\right)\, e^{2\alpha}$
LOGN	Lognormal	μ (no bounds), $\sigma > 0$	$f(x) = \frac{1}{x\sigma\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{\log(x)-\mu}{\sigma}\right)^2}$ $F(x) = \Phi\left(\frac{\log(x)-\mu}{\sigma}\right)$
PARETO	Pareto	$\theta > 0, \alpha > 0$	$f(x) = \frac{\alpha\theta^{\alpha}}{(x+\theta)^{\alpha+1}}$ $F(x) = 1 - \left(\frac{\theta}{x+\theta}\right)^{\alpha}$
TWEEDIE	Tweedie[6]	$p > 1, \mu > 0,$ $\phi > 0$	$f(x) = a(x,\phi)\exp\left[\frac{1}{\phi}\left(\frac{x\mu^{1-p}}{1-p} - \kappa(\mu,p)\right)\right]$ $F(x) = \int_0^x f(t)dt$
STWEEDIE	Scaled Tweedie[6]	$\theta > 0, \lambda > 0,$ $1 < p < 2$	$f(x) = a(x,\theta,\lambda,p)\exp\left(-\frac{x}{\theta} - \lambda\right)$ $F(x) = \int_0^x f(t)dt$
WEIBULL	Weibull	$\theta > 0, \tau > 0$	$f(x) = \frac{1}{x}\tau z^{\tau}e^{-z^{\tau}}$ $F(x) = 1 - e^{-z^{\tau}}$

Notes:
1. $z = x/\theta$, wherever z is used.
2. θ denotes the scale parameter for all the distributions. For LOGN, $\log(\theta) = \mu$.
3. Parameters are listed in the order in which they are defined in the distribution model.
4. $\gamma(a,b) = \int_0^b t^{a-1}e^{-t}dt$ is the lower incomplete gamma function.
5. $\Phi(y) = \frac{1}{2}\left(1 + \text{erf}\left(\frac{y}{\sqrt{2}}\right)\right)$ is the standard normal CDF.
6. For more information, see the section "Tweedie Distributions" on page 2110.

Tweedie Distributions

Tweedie distributions are a special case of the exponential dispersion family (Jørgensen 1987) with a property that the variance of the distribution is equal to $\phi\mu^p$, where μ is the mean of the distribution, ϕ is a dispersion parameter, and p is an index parameter as discovered by Tweedie (1984). The distribution is defined for all values of p except for values of p in the open interval $(0, 1)$. Many important known distributions are a special case of Tweedie distributions including normal (p=0), Poisson (p=1), gamma (p=2), and the inverse Gaussian (p=3). Apart from these special cases, the probability density function (PDF) of the Tweedie distribution does not have an analytic expression. For $p > 1$, it has the form (Dunn and Smyth 2005),

$$f(x; \mu, \phi, p) = a(x, \phi) \exp\left[\frac{1}{\phi}\left(\frac{x\mu^{1-p}}{1-p} - \kappa(\mu, p)\right)\right]$$

where $\kappa(\mu, p) = \mu^{2-p}/(2-p)$ for $p \neq 2$ and $\kappa(\mu, p) = \log(\mu)$ for $p = 2$. The function $a(x, \phi)$ does not have an analytical expression. It is typically evaluated using series expansion methods described in Dunn and Smyth (2005).

For $1 < p < 2$, the Tweedie distribution is a compound Poisson-gamma mixture distribution, which is the distribution of S defined as

$$S = \sum_{i=1}^{N} X_i$$

where $N \sim \text{Poisson}(\lambda)$ and $X_i \sim \text{gamma}(\alpha, \theta)$ are independent and identically distributed gamma random variables with shape parameter α and scale parameter θ. At $X = 0$, the density is a probability mass that is governed by the Poisson distribution, and for values of $X > 0$, it is a mixture of gamma variates with Poisson mixing probability. The parameters λ, α, and θ are related to the natural parameters μ, ϕ, and p of the Tweedie distribution as

$$\lambda = \frac{\mu^{2-p}}{\phi(2-p)}$$
$$\alpha = \frac{2-p}{p-1}$$
$$\theta = \phi(p-1)\mu^{p-1}$$

The mean of a Tweedie distribution is positive for $p > 1$.

Two predefined versions of the Tweedie distribution are provided with the SEVERITY procedure. The first version, named TWEEDIE and defined for $p > 1$, has the natural parameterization with parameters μ, ϕ, and p. The second version, named STWEEDIE and defined for $1 < p < 2$, is the version with a scale parameter. It corresponds to the compound Poisson-gamma distribution with gamma scale parameter θ, Poisson mean parameter λ, and the index parameter p. The index parameter decides the shape parameter α of the gamma distribution as

$$\alpha = \frac{2-p}{p-1}$$

The parameters θ and λ of the STWEEDIE distribution are related to the parameters μ and ϕ of the TWEEDIE distribution as

$$\mu = \lambda\theta\alpha$$
$$\phi = \frac{(\lambda\theta\alpha)^{2-p}}{\lambda(2-p)} = \frac{\theta}{(p-1)(\lambda\theta\alpha)^{p-1}}$$

You can fit either version when there are no regression variables. Each version has its own merits. If you fit the TWEEDIE version, you have the direct estimate of the overall mean of the distribution. If you are interested in the most practical range of the index parameter $1 < p < 2$, then you can fit the STWEEDIE version, which provides you direct estimates of the Poisson and gamma components that comprise the distribution (an estimate of the gamma shape parameter α is easily obtained from the estimate of p).

If you want to estimate the effect of exogenous (regression) variables on the distribution, then you must use the STWEEDIE version, because PROC SEVERITY requires a distribution to have a scale parameter in order to estimate regression effects. For more information, see the section "Estimating Regression Effects" on page 2123. The gamma scale parameter θ is the scale parameter of the STWEEDIE distribution. If you are interested in determining the effect of regression variables on the mean of the distribution, you can do so by first fitting the STWEEDIE distribution to determine the effect of the regression variables on the scale parameter θ. Then, you can easily estimate how the mean of the distribution μ is affected by the regression variables using the relationship $\mu = c\theta$, where $c = \lambda\alpha = \lambda(2-p)/(p-1)$. The estimates of the regression parameters remain the same, whereas the estimate of the intercept parameter is adjusted by the estimates of the λ and p parameters.

Parameter Initialization for Predefined Distributions

The parameters are initialized by using the method of moments for all the distributions, except for the gamma and the Weibull distributions. For the gamma distribution, approximate maximum likelihood estimates are used. For the Weibull distribution, the method of percentile matching is used.

Given n observations of the severity value y_i $(1 \le i \le n)$, the estimate of kth raw moment is denoted by m_k and computed as

$$m_k = \frac{1}{n}\sum_{i=1}^{n} y_i^k$$

The $100p$th percentile is denoted by π_p $(0 \le p \le 1)$. By definition, π_p satisfies

$$F(\pi_p-) \le p \le F(\pi_p)$$

where $F(\pi_p-) = \lim_{h\downarrow 0} F(\pi_p - h)$. PROC SEVERITY uses the following practical method of computing π_p. Let $\hat{F}_n(y)$ denote the empirical distribution function (EDF) estimate at a severity value y. Let y_p^- and y_p^+ denote two consecutive values in the ascending sequence of y values such that $\hat{F}_n(y_p^-) < p$ and $\hat{F}_n(y_p^+) \ge p$. Then, the estimate $\hat{\pi}_p$ is computed as

$$\hat{\pi}_p = y_p^- + \frac{p - \hat{F}_n(y_p^-)}{\hat{F}_n(y_p^+) - \hat{F}_n(y_p^-)}(y_p^+ - y_p^-)$$

Let ϵ denote the smallest double-precision floating-point number such that $1 + \epsilon > 1$. This machine precision constant can be obtained by using the CONSTANT function in Base SAS software.

The details of how parameters are initialized for each predefined distribution are as follows:

BURR The parameters are initialized by using the method of moments. The kth raw moment of the Burr distribution is:

$$E[X^k] = \frac{\theta^k \Gamma(1 + k/\gamma)\Gamma(\alpha - k/\gamma)}{\Gamma(\alpha)}, \quad -\gamma < k < \alpha\gamma$$

Three moment equations $E[X^k] = m_k$ $(k = 1, 2, 3)$ need to be solved for initializing the three parameters of the distribution. In order to get an approximate closed form solution, the second shape parameter $\hat{\gamma}$ is initialized to a value of 2. If $2m_3 - 3m_1m_2 > 0$, then simplifying and solving the moment equations yields the following feasible set of initial values:

$$\hat{\theta} = \sqrt{\frac{m_2 m_3}{2m_3 - 3m_1 m_2}}, \quad \hat{\alpha} = 1 + \frac{m_3}{2m_3 - 3m_1 m_2}, \quad \hat{\gamma} = 2$$

If $2m_3 - 3m_1m_2 < \epsilon$, then the parameters are initialized as follows:

$$\hat{\theta} = \sqrt{m_2}, \quad \hat{\alpha} = 2, \quad \hat{\gamma} = 2$$

EXP The parameters are initialized by using the method of moments. The kth raw moment of the exponential distribution is:

$$E[X^k] = \theta^k \Gamma(k+1), \quad k > -1$$

Solving $E[X] = m_1$ yields the initial value of $\hat{\theta} = m_1$.

GAMMA The parameter α is initialized by using its *approximate* maximum likelihood (ML) estimate. For a set of n independent and identically distributed observations y_i $(1 \le i \le n)$ drawn from a gamma distribution, the log likelihood l is defined as follows:

$$\begin{aligned} l &= \sum_{i=1}^{n} \log\left(y_i^{\alpha-1} \frac{e^{-y_i/\theta}}{\theta^\alpha \Gamma(\alpha)} \right) \\ &= (\alpha - 1) \sum_{i=1}^{n} \log(y_i) - \frac{1}{\theta} \sum_{i=1}^{n} y_i - n\alpha \log(\theta) - n \log(\Gamma(\alpha)) \end{aligned}$$

Using a shorter notation of $\sum$ to denote $\sum_{i=1}^{n}$ and solving the equation $\partial l / \partial \theta = 0$ yields the following ML estimate of θ:

$$\hat{\theta} = \frac{\sum y_i}{n\alpha} = \frac{m_1}{\alpha}$$

Substituting this estimate in the expression of l and simplifying gives

$$l = (\alpha - 1) \sum \log(y_i) - n\alpha - n\alpha \log(m_1) + n\alpha \log(\alpha) - n \log(\Gamma(\alpha))$$

Let d be defined as follows:

$$d = \log(m_1) - \frac{1}{n} \sum \log(y_i)$$

Solving the equation $\partial l / \partial \alpha = 0$ yields the following expression in terms of the digamma function, $\psi(\alpha)$:

$$\log(\alpha) - \psi(\alpha) = d$$

The digamma function can be approximated as follows:

$$\hat{\psi}(\alpha) \approx \log(\alpha) - \frac{1}{\alpha}\left(0.5 + \frac{1}{12\alpha + 2}\right)$$

This approximation is within 1.4% of the true value for all the values of $\alpha > 0$ except when α is arbitrarily close to the positive root of the digamma function (which is approximately 1.461632). Even for the values of α that are close to the positive root, the absolute error between true and approximate values is still acceptable ($|\hat{\psi}(\alpha) - \psi(\alpha)| < 0.005$ for $\alpha > 1.07$). Solving the equation that arises from this approximation yields the following estimate of α:

$$\hat{\alpha} = \frac{3 - d + \sqrt{(d-3)^2 + 24d}}{12d}$$

If this approximate ML estimate is infeasible, then the method of moments is used. The kth raw moment of the gamma distribution is:

$$E[X^k] = \theta^k \frac{\Gamma(\alpha + k)}{\Gamma(\alpha)}, \quad k > -\alpha$$

Solving $E[X] = m_1$ and $E[X^2] = m_2$ yields the following initial value for α:

$$\hat{\alpha} = \frac{m_1^2}{m_2 - m_1^2}$$

If $m_2 - m_1^2 < \epsilon$ (almost zero sample variance), then α is initialized as follows:

$$\hat{\alpha} = 1$$

After computing the estimate of α, the estimate of θ is computed as follows:

$$\hat{\theta} = \frac{m_1}{\hat{\alpha}}$$

Both the maximum likelihood method and the method of moments arrive at the same relationship between $\hat{\alpha}$ and $\hat{\theta}$.

GPD

The parameters are initialized by using the method of moments. Notice that for $\xi > 0$, the CDF of the generalized Pareto distribution (GPD) is:

$$\begin{aligned} F(x) &= 1 - \left(1 + \frac{\xi x}{\theta}\right)^{-1/\xi} \\ &= 1 - \left(\frac{\theta/\xi}{x + \theta/\xi}\right)^{1/\xi} \end{aligned}$$

This is equivalent to a Pareto distribution with scale parameter $\theta_1 = \theta/\xi$ and shape parameter $\alpha = 1/\xi$. Using this relationship, the parameter initialization method used for the PARETO distribution is used to get the following initial values for the parameters of the GPD distribution:

$$\hat{\theta} = \frac{m_1 m_2}{2(m_2 - m_1^2)}, \quad \hat{\xi} = \frac{m_2 - 2m_1^2}{2(m_2 - m_1^2)}$$

If $m_2 - m_1^2 < \epsilon$ (almost zero sample variance) or $m_2 - 2m_1^2 < \epsilon$, then the parameters are initialized as follows:

$$\hat{\theta} = \frac{m_1}{2}, \quad \hat{\xi} = \frac{1}{2}$$

IGAUSS The parameters are initialized by using the method of moments. The standard parameterization of the inverse Gaussian distribution (also known as the Wald distribution), in terms of the location parameter μ and shape parameter λ, is as follows (Klugman, Panjer, and Willmot 1998, p. 583):

$$f(x) = \sqrt{\frac{\lambda}{2\pi x^3}}\ \exp\left(\frac{-\lambda(x-\mu)^2}{2\mu^2 x}\right)$$
$$F(x) = \Phi\left(\left(\frac{x}{\mu}-1\right)\sqrt{\frac{\lambda}{x}}\right) + \Phi\left(-\left(\frac{x}{\mu}+1\right)\sqrt{\frac{\lambda}{x}}\right)\ \exp\left(\frac{2\lambda}{\mu}\right)$$

For this parameterization, it is known that the mean is $E[X] = \mu$ and the variance is $Var[X] = \mu^3/\lambda$, which yields the second raw moment as $E[X^2] = \mu^2(1+\mu/\lambda)$ (computed by using $E[X^2] = Var[X] + (E[X])^2$).

The predefined IGAUSS distribution in PROC SEVERITY uses the following alternate parameterization to allow the distribution to have a scale parameter, θ:

$$f(x) = \sqrt{\frac{\alpha\theta}{2\pi x^3}}\ \exp\left(\frac{-\alpha(x-\theta)^2}{2x\theta}\right)$$
$$F(x) = \Phi\left(\left(\frac{x}{\theta}-1\right)\sqrt{\frac{\alpha\theta}{x}}\right) + \Phi\left(-\left(\frac{x}{\theta}+1\right)\sqrt{\frac{\alpha\theta}{x}}\right)\ \exp\left(2\alpha\right)$$

The parameters θ (scale) and α (shape) of this alternate form are related to the parameters μ and λ of the preceding form such that $\theta = \mu$ and $\alpha = \lambda/\mu$. Using this relationship, the first and second raw moments of the IGAUSS distribution are:

$$E[X] = \theta$$
$$E[X^2] = \theta^2\left(1 + \frac{1}{\alpha}\right)$$

Solving $E[X] = m_1$ and $E[X^2] = m_2$ yields the following initial values:

$$\hat{\theta} = m_1, \quad \hat{\alpha} = \frac{m_1^2}{m_2 - m_1^2}$$

If $m_2 - m_1^2 < \epsilon$ (almost zero sample variance), then the parameters are initialized as follows:

$$\hat{\theta} = m_1, \quad \hat{\alpha} = 1$$

LOGN The parameters are initialized by using the method of moments. The kth raw moment of the lognormal distribution is:

$$E[X^k] = \exp\left(k\mu + \frac{k^2\sigma^2}{2}\right)$$

Solving $E[X] = m_1$ and $E[X^2] = m_2$ yields the following initial values:

$$\hat{\mu} = 2\log(m1) - \frac{\log(m2)}{2}, \quad \hat{\sigma} = \sqrt{\log(m2) - 2\log(m1)}$$

PARETO The parameters are initialized by using the method of moments. The kth raw moment of the Pareto distribution is:

$$E[X^k] = \frac{\theta^k \Gamma(k+1)\Gamma(\alpha - k)}{\Gamma(\alpha)}, -1 < k < \alpha$$

Solving $E[X] = m_1$ and $E[X^2] = m_2$ yields the following initial values:

$$\hat{\theta} = \frac{m_1 m_2}{m_2 - 2m_1^2}, \quad \hat{\alpha} = \frac{2(m_2 - m_1^2)}{m_2 - 2m_1^2}$$

If $m_2 - m_1^2 < \epsilon$ (almost zero sample variance) or $m_2 - 2m_1^2 < \epsilon$, then the parameters are initialized as follows:

$$\hat{\theta} = m_1, \quad \hat{\alpha} = 2$$

TWEEDIE The parameter p is initialized by assuming that the sample is generated from a gamma distribution with shape parameter α and by computing $\hat{p} = \frac{\hat{\alpha}+2}{\hat{\alpha}+1}$. The initial value $\hat{\alpha}$ is obtained from using the method previously described for the GAMMA distribution. The parameter μ is the mean of the distribution. Hence, it is initialized to the sample mean as

$$\hat{\mu} = m_1$$

Variance of a Tweedie distribution is equal to $\phi\mu^p$. Thus, the sample variance is used to initialize the value of ϕ as

$$\hat{\phi} = \frac{m_2 - m_1^2}{\hat{\mu}^{\hat{p}}}$$

STWEEDIE STWEEDIE is a compound Poisson-gamma mixture distribution with mean $\mu = \lambda\theta\alpha$, where α is the shape parameter of the gamma random variables in the mixture and the parameter p is determined solely by α. First, the parameter p is initialized by assuming that the sample is generated from a gamma distribution with shape parameter α and by computing $\hat{p} = \frac{\hat{\alpha}+2}{\hat{\alpha}+1}$. The initial value $\hat{\alpha}$ is obtained from using the method previously described for the GAMMA distribution. As done for initializing the parameters of the TWEEDIE distribution, the sample mean and variance are used to compute the values $\hat{\mu}$ and $\hat{\phi}$ as

$$\hat{\mu} = m_1$$
$$\hat{\phi} = \frac{m_2 - m_1^2}{\hat{\mu}^{\hat{p}}}$$

Based on the relationship between the parameters of TWEEDIE and STWEEDIE distributions described in the section "Tweedie Distributions" on page 2110, values of θ and λ are initialized as

$$\hat{\theta} = \hat{\phi}(\hat{p} - 1)\hat{\mu}^{p-1}$$
$$\hat{\lambda} = \frac{\hat{\mu}}{\hat{\theta}\hat{\alpha}}$$

WEIBULL The parameters are initialized by using the percentile matching method. Let $q1$ and $q3$ denote the estimates of the 25th and 75th percentiles, respectively. Using the formula for the CDF of Weibull distribution, they can be written as

$$1 - \exp(-(q1/\theta)^{\tau}) = 0.25$$
$$1 - \exp(-(q3/\theta)^{\tau}) = 0.75$$

Simplifying and solving these two equations yields the following initial values:

$$\hat{\theta} = \exp\left(\frac{r\log(q1) - \log(q3)}{r-1}\right), \quad \hat{\tau} = \frac{\log(\log(4))}{\log(q3) - \log(\hat{\theta})}$$

where $r = \log(\log(4))/\log(\log(4/3))$. These initial values agree with those suggested in Klugman, Panjer, and Willmot (1998).

A summary of the initial values of all the parameters for all the predefined distributions is given in Table 30.3. The table also provides the names of the parameters to use in the INIT= option in the DIST statement if you want to provide a different initial value.

Table 30.3 Parameter Initialization for Predefined Distributions

Distribution	Parameter	Name for INIT option	Default Initial Value
BURR	θ	theta	$\sqrt{\frac{m_2 m_3}{2m_3 - 3m_1 m_2}}$
	α	alpha	$1 + \frac{m_3}{2m_3 - 3m_1 m_2}$
	γ	gamma	2
EXP	θ	theta	m_1
GAMMA	θ	theta	m_1/α
	α	alpha	$\frac{3-d+\sqrt{(d-3)^2+24d}}{12d}$
GPD	θ	theta	$m_1 m_2/(2(m_2 - m_1^2))$
	ξ	xi	$(m_2 - 2m_1^2)/(2(m_2 - m_1^2))$
IGAUSS	θ	theta	m_1
	α	alpha	$m_1^2/(m_2 - m_1^2)$
LOGN	μ	mu	$2\log(m1) - \log(m2)/2$
	σ	sigma	$\sqrt{\log(m2) - 2\log(m1)}$
PARETO	θ	theta	$m_1 m_2/(m_2 - 2m_1^2)$
	α	alpha	$2(m_2 - m_1^2)/(m_2 - 2m_1^2)$
TWEEDIE	μ	mu	m_1
	ϕ	phi	$(m_2 - m_1^2)/m_1^p$
	p	p	$(\alpha + 2)/(\alpha + 1)$ where $\alpha = \frac{3-d+\sqrt{(d-3)^2+24d}}{12d}$
STWEEDIE	θ	theta	$(m_2 - m_1^2)(p-1)/m_1$
	λ	lambda	$m_1^2/(\alpha(m_2 - m_1^2)(p-1))$
	p	p	$(\alpha + 2)/(\alpha + 1)$ where $\alpha = \frac{3-d+\sqrt{(d-3)^2+24d}}{12d}$
WEIBULL	θ	theta	$\exp\left(\frac{r\log(q1) - \log(q3)}{r-1}\right)$
	τ	tau	$\log(\log(4))/(\log(q3) - \log(\hat{\theta}))$

Notes:

- m_k denotes the kth raw moment
- $d = \log(m_1) - (\sum \log(y_i))/n$
- $q1$ and $q3$ denote the 25th and 75th percentiles, respectively
- $r = \log(\log(4))/\log(\log(4/3))$

Censoring and Truncation

One of the key features of PROC SEVERITY is that it enables you to specify whether the severity event's magnitude is observable and if it is observable, then whether the exact value of the magnitude is known. If an event is unobservable when the magnitude is in certain intervals, then it is referred to as a truncation effect. If the exact magnitude of the event is not known, but it is known to have a value in a certain interval, then it is referred to as a censoring effect.

PROC SEVERITY allows a severity event to be subject to any combination of the following four censoring and truncation effects:

- Left-truncation: An event is said to be left-truncated if it is observed only when $Y > T^l$, where Y denotes the random variable for the magnitude and T^l denotes a random variable for the truncation threshold. You can specify left-truncation using the LEFTTRUNCATED= option in the LOSS statement.
- Right-truncation: An event is said to be right-truncated if it is observed only when $Y \leq T^r$, where Y denotes the random variable for the magnitude and T^r denotes a random variable for the truncation threshold. You can specify right-truncation using the RIGHTTRUNCATED= option in the LOSS statement.
- Left-censoring: An event is said to be left-censored if it is known that the magnitude is $Y \leq C^l$, but the exact value of Y is not known. C^l is a random variable for the censoring limit. You can specify left-censoring using the LEFTCENSORED= option in the LOSS statement.
- Right-censoring: An event is said to be right-censored if it is known that the magnitude is $Y > C^r$, but the exact value of Y is not known. C^r is a random variable for the censoring limit. You can specify right-censoring using the RIGHTCENSORED= option in the LOSS statement.

For each effect, you can specify a different threshold or limit for each observation or specify a single threshold or limit that applies to all the observations.

If all the four types of effects are present on an event, then the following relationship holds: $T^l < C^r \leq C^l \leq T^r$. PROC SEVERITY checks these relationships and write a warning to the SAS log if any is violated.

If you specify the response variable in the LOSS statement, then PROC SEVERITY also checks whether each observation satisfies the definitions of the specified censoring and truncation effects. If you specify left-truncation, then PROC SEVERITY ignores observations where $Y \leq T^l$, because such observations are not observable by definition. Similarly, if you specify right-truncation, then PROC SEVERITY ignores observations where $Y > T^r$. If you specify left-censoring, then PROC SEVERITY treats an observation with $Y > C^l$ as uncensored and ignores the value of C^l. The observations with $Y \leq C^l$ are considered as left-censored, and the value of Y is ignored. If you specify right-censoring, then PROC SEVERITY treats an observation with $Y \leq C^r$ as uncensored and ignores the value of C^r. The observations with $Y > C^r$ are considered as right-censored, and the value of Y is ignored. If you specify both left-censoring and right-censoring, it is referred to as interval-censoring. If $C^r < C^l$ is satisfied for an observation, then it is considered as interval-censored and the value of the response variable is ignored. If $C^r = C^l$ for an observation, then PROC SEVERITY assumes that observation to be uncensored. If all the observations in a data set are censored in some form, then the specification of the response variable in the LOSS statement is optional, because the actual value of the response variable is not required for the purposes of estimating a model.

Specification of censoring and truncation affects the likelihood of the data (see the section "Likelihood Function" on page 2120) and how the empirical distribution function (EDF) is estimated (see the section "Empirical Distribution Function Estimation Methods" on page 2138).

Probability of Observability

For left-truncated data, PROC SEVERITY also enables you to provide additional information in the form of *probability of observability* by using the PROBOBSERVED= option. It is defined as the probability that the underlying severity event gets observed (and recorded) for the specified left-truncation threshold value. For example, if you specify a value of 0.75, then for every 75 observations recorded above a specified threshold, 25 more events have happened with a severity value less than or equal to the specified threshold. Although the exact severity value of those 25 events is not known, PROC SEVERITY can use the information about the number of those events.

In particular, for each left-truncated observation, PROC SEVERITY assumes a presence of $(1 - p)/p$ additional observations with $y_i = t_i$. These additional observations are then used for computing the likelihood (see the section "Probability of Observability and Likelihood" on page 2121) and an unconditional estimate of the empirical distribution function (see the section "EDF Estimates and Truncation" on page 2142).

Truncation and Conditional CDF Estimates

If you specify left-truncation without the probability of observability or if you specify right-truncation, then the EDF estimates that are computed by all methods except the STANDARD method are conditional on the truncation information. See the section "EDF Estimates and Truncation" on page 2142 for more information. In such cases, PROC SEVERITY uses conditional estimates of the CDF whenever they are used for computational or visual comparison with the EDF estimates.

Let $t^l_{\min} = \min_i\{t^l_i\}$ be the smallest value of the left-truncation threshold (t^l_i is the left-truncation threshold for observation i) and $t^r_{\max} = \max_i\{t^r_i\}$ be the largest value of the right-truncation threshold (t^r_i is the right-truncation threshold for observation i). If $\hat{F}(y)$ denotes the unconditional estimate of the CDF at y, then the conditional estimate $\hat{F}^c(y)$ is computed as follows:

- If you do not specify the probability of observability, then the EDF estimates are conditional on the left-truncation information. If an observation is both left-truncated and right-truncated, then

$$\hat{F}^c(y) = \frac{\hat{F}(y) - \hat{F}(t^l_{\min})}{\hat{F}(t^r_{\max}) - \hat{F}(t^l_{\min})}$$

  If an observation is left-truncated but not right-truncated, then

$$\hat{F}^c(y) = \frac{\hat{F}(y) - \hat{F}(t^l_{\min})}{1 - \hat{F}(t^l_{\min})}$$

  If an observation is right-truncated but not left-truncated, then

$$\hat{F}^c(y) = \frac{\hat{F}(y)}{\hat{F}(t^r_{\max})}$$

- If you specify the probability of observability, then EDF estimates are not conditional on the left-truncation information. If an observation is not right-truncated, then the conditional estimate is the

same as the unconditional estimate. If an observation is right-truncated, then the conditional estimate is computed as

$$\hat{F}^c(y) = \frac{\hat{F}(y)}{\hat{F}(t^r_{\max})}$$

If you specify regression effects, then $\hat{F}(y)$, $\hat{F}(t^l_{\min})$, and $\hat{F}(t^r_{\max})$ are all computed from a mixture distribution, as described in the section "CDF and PDF Estimates with Regression Effects" on page 2126.

Parameter Estimation Method

If you do not specify a custom objective function by specifying programming statements and the OBJECTIVE= option in the PROC SEVERITY statement, then PROC SEVERITY uses the maximum likelihood (ML) method to estimate the parameters of each model. A nonlinear optimization process is used to maximize the log of the likelihood function. If you specify a custom objective function, then PROC SEVERITY uses a nonlinear optimization algorithm to estimate the parameters of each model that minimize the value of your specified objective function. For more information, see the section "Custom Objective Functions" on page 2173.

Likelihood Function

Let $f_\Theta(x)$ and $F_\Theta(x)$ denote the PDF and CDF, respectively, evaluated at x for a set of parameter values Θ. Let Y denote the random response variable, and let y denote its value recorded in an observation in the input data set. Let T^l and T^r denote the random variables for the left-truncation and right-truncation threshold, respectively, and let t^l and t^r denote their values for an observation, respectively. If there is no left-truncation, then $t^l = \tau^l$, where τ^l is the smallest value in the support of the distribution; so $F(t^l) = 0$. If there is no right-truncation, then $t^r = \tau_h$, where τ_h is the largest value in the support of the distribution; so $F(t^r) = 1$. Let C^l and C^r denote the random variables for the left-censoring and right-censoring limit, respectively, and let c^l and c^r denote their values for an observation, respectively. If there is no left-censoring, then $c^l = \tau_h$; so $F(c^l) = 1$. If there is no right-censoring, then $c^r = \tau^l$; so $F(c^r) = 0$.

The set of input observations can be categorized into the following four subsets within each BY group:

- E is the set of uncensored and untruncated observations. The likelihood of an observation in E is

$$l_E = \Pr(Y = y) = f_\Theta(y)$$

- E_t is the set of uncensored observations that are truncated. The likelihood of an observation in E_t is

$$l_{E_t} = \Pr(Y = y | t^l < Y \le t^r) = \frac{f_\Theta(y)}{F_\Theta(t^r) - F_\Theta(t^l)}$$

- C is the set of censored observations that are not truncated. The likelihood of an observation C is

$$l_C = \Pr(c^r < Y \le c^l) = F_\Theta(c^l) - F_\Theta(c^r)$$

- C_t is the set of censored observations that are truncated. The likelihood of an observation C_t is

$$l_{C_t} = \Pr(c^r < Y \le c^l | t^l < Y \le t^r) = \frac{F_\Theta(c^l) - F_\Theta(c^r)}{F_\Theta(t^r) - F_\Theta(t^l)}$$

Note that $(E \cup E_t) \cap (C \cup C_t) = \emptyset$. Also, the sets E_t and C_t are empty when you do not specify truncation, and the sets C and C_t are empty when you do not specify censoring.

Given this, the likelihood of the data L is as follows:

$$L = \prod_{E} f_\Theta(y) \prod_{E_t} \frac{f_\Theta(y)}{F_\Theta(t^r) - F_\Theta(t^l)} \prod_{C} F_\Theta(c^l) - F_\Theta(c^r) \prod_{C_t} \frac{F_\Theta(c^l) - F_\Theta(c^r)}{F_\Theta(t^r) - F_\Theta(t^l)}$$

The maximum likelihood procedure used by PROC SEVERITY finds an optimal set of parameter values $\hat{\Theta}$ that maximizes $\log(L)$ subject to the boundary constraints on parameter values. For a distribution *dist*, you can specify such boundary constraints by using the *dist*_LOWERBOUNDS and *dist*_UPPERBOUNDS subroutines. For more information, see the section "Defining a Severity Distribution Model with the FCMP Procedure" on page 2148. Some aspects of the optimization process can be controlled by using the NLOPTIONS statement.

Probability of Observability and Likelihood

If you specify the probability of observability for the left-truncation, then PROC SEVERITY uses a modified likelihood function for each truncated observation. If the probability of observability is $p \in (0.0, 1.0]$, then for each left-truncated observation with truncation threshold t^l, there exist $(1-p)/p$ observations with a response variable value less than or equal to t^l. Each such observation has a probability of $\Pr(Y \le t^l) = F_\Theta(t^l)$. The right-truncation and censoring information does not apply to these added observations. Thus, following the notation of the section "Likelihood Function" on page 2120, the likelihood of the data is as follows:

$$\begin{aligned} L = & \prod_{E} f_\Theta(y) \prod_{E_t, t^l = \tau^l} \frac{f_\Theta(y)}{F_\Theta(t^r)} \prod_{E_t, t^l > \tau^l} \frac{f_\Theta(y)}{F_\Theta(t^r)} F_\Theta(t^l)^{\frac{1-p}{p}} \\ & \prod_{C} F_\Theta(c^l) - F_\Theta(c^r) \prod_{C_t, t^l = \tau^l} \frac{F_\Theta(c^l) - F_\Theta(c^r)}{F_\Theta(t^r)} \prod_{C_t, t^l > \tau^l} \frac{F_\Theta(c^l) - F_\Theta(c^r)}{F_\Theta(t^r)} F_\Theta(t^l)^{\frac{1-p}{p}} \end{aligned}$$

Note that the likelihood of the observations that are not left-truncated (observations in sets E and C, and observations in sets E_t and C_t for which $t^l = \tau^l$) is not affected.

If you specify a custom objective function, then PROC SEVERITY accounts for the probability of observability only while computing the empirical distribution function estimate. The parameter estimates are affected only by your custom objective function.

Estimating Covariance and Standard Errors

PROC SEVERITY computes an estimate of the covariance matrix of the parameters by using the asymptotic theory of the maximum likelihood estimators (MLE). If N denotes the number of observations used for estimating a parameter vector $\boldsymbol{\theta}$, then the theory states that as $N \to \infty$, the distribution of $\hat{\boldsymbol{\theta}}$, the estimate of $\boldsymbol{\theta}$, converges to a normal distribution with mean $\boldsymbol{\theta}$ and covariance $\hat{\mathbf{C}}$ such that $\mathbf{I}(\boldsymbol{\theta}) \cdot \hat{\mathbf{C}} \to 1$, where $\mathbf{I}(\boldsymbol{\theta}) = -E\left[\nabla^2 \log(L(\boldsymbol{\theta}))\right]$ is the information matrix for the likelihood of the data, $L(\boldsymbol{\theta})$. The covariance estimate is obtained by using the inverse of the information matrix.

In particular, if $\mathbf{G} = \nabla^2(-\log(L(\boldsymbol{\theta})))$ denotes the Hessian matrix of the negative of log likelihood, then the covariance estimate is computed as

$$\hat{\mathbf{C}} = \frac{N}{d}\mathbf{G}^{-1}$$

where d is a denominator that is determined by the VARDEF= option. If VARDEF=N, then $d = N$, which yields the asymptotic covariance estimate. If VARDEF=DF, then $d = N - k$, where k is number of parameters (the model's degrees of freedom). The VARDEF=DF option is the default, because it attempts to correct the potential bias introduced by the finite sample.

The standard error s_i of the parameter θ_i is computed as the square root of the ith diagonal element of the estimated covariance matrix; that is, $s_i = \sqrt{\hat{C}_{ii}}$.

If you specify a custom objective function, then the covariance matrix of the parameters is still computed by inverting the information matrix, except that the Hessian matrix $\mathbf{G}$ is computed as $\mathbf{G} = \nabla^2 \log(U(\boldsymbol{\theta}))$, where U denotes your custom objective function that is minimized by the optimizer.

Covariance and standard error estimates might not be available if the Hessian matrix is found to be singular at the end of the optimization process. This can especially happen if the optimization process stops without converging.

Parameter Initialization

PROC SEVERITY enables you to initialize parameters of a model in different ways. A model can have two kinds of parameters: distribution parameters and regression parameters.

The distribution parameters can be initialized by using one of the following three methods:

INIT= option	You can use the INIT= option in the DIST statement.
INEST= or INSTORE= option	You can use either the INEST= data set or the INSTORE= item store, but not both.
PARMINIT subroutine	You can define a *dist*_PARMINIT subroutine in the distribution model. For more information, see the section "Defining a Severity Distribution Model with the FCMP Procedure" on page 2148.

Note that only one of the initialization methods is used. You cannot combine them. They are used in the following order:

- The method that uses the INIT= option takes the highest precedence. If you use the INIT= option to provide an initial value for at least one parameter, then other initialization methods (INEST=, INSTORE=, or PARMINIT) are not used. If you specify initial values for some but not all the parameters by using the INIT= option, then the uninitialized parameters are initialized to the default value of 0.001.

  If you use this option and if you specify the regression effects, then the value of the first distribution parameter must be related to the initial value for the *base* value of the scale or log-transformed scale parameter. For more information, see the section "Estimating Regression Effects" on page 2123.

- The method that uses the INEST= data set or INSTORE= item store takes second precedence. If the INEST= data set or INSTORE= item store contains a nonmissing value for even one distribution parameter, then the PARMINIT method is not used and any uninitialized parameters are initialized to the default value of 0.001.
- If none of the distribution parameters are initialized by using the INIT= option, the INEST= data set, or the INSTORE= item store, but the distribution model defines a PARMINIT subroutine, then PROC SEVERITY invokes that subroutine with appropriate inputs to initialize the parameters. If the PARMINIT subroutine returns missing values for some parameters, then those parameters are initialized to the default value of 0.001.
- If none of the initialization methods are used, each distribution parameter is initialized to the default value of 0.001.

For more information about regression models and initialization of regression parameters, see the section "Estimating Regression Effects" on page 2123.

Estimating Regression Effects

The SEVERITY procedure enables you to estimate the influence of regression (exogenous) effects while fitting a distribution if the distribution has a scale parameter or a log-transformed scale parameter.

Let x_j, $j = 1, \ldots, k$, denote the k regression effects. Let β_j denote the regression parameter that corresponds to the effect x_j. If you do not specify regression effects, then the model for the response variable Y is of the form

$$Y \sim \mathcal{F}(\Theta)$$

where $\mathcal{F}$ is the distribution of Y with parameters Θ. This model is usually referred to as the error model. The regression effects are modeled by extending the error model to the following form:

$$Y \sim \exp(\sum_{j=1}^{k} \beta_j x_j) \cdot \mathcal{F}(\Theta)$$

Under this model, the distribution of Y is valid and belongs to the same parametric family as $\mathcal{F}$ if and only if $\mathcal{F}$ has a scale parameter. Let θ denote the scale parameter and Ω denote the set of nonscale distribution parameters of $\mathcal{F}$. Then the model can be rewritten as

$$Y \sim \mathcal{F}(\theta, \Omega)$$

such that θ is modeled by the regression effects as

$$\theta = \theta_0 \cdot \exp(\sum_{j=1}^{k} \beta_j x_j)$$

where θ_0 is the *base* value of the scale parameter. Thus, the scale regression model consists of the following parameters: θ_0, Ω, and $\beta_j (j = 1, \ldots, k)$.

Given this form of the model, distributions without a scale parameter cannot be considered when regression effects are to be modeled. If a distribution does not have a direct scale parameter, then PROC SEVERITY accepts it only if it has a log-transformed scale parameter — that is, if it has a parameter $p = \log(\theta)$.

Offset Variable

You can specify that an offset variable be included in the scale regression model by specifying it in the OFFSET= option of the SCALEMODEL statement. The offset variable is a regressor whose regression coefficient is known to be 1. If x_o denotes the offset variable, then the scale regression model becomes

$$\theta = \theta_0 \cdot \exp(x_o + \sum_{j=1}^{k} \beta_j x_j)$$

The regression coefficient of the offset variable is fixed at 1 and not estimated, so it is not reported in the ParameterEstimates ODS table. However, if you specify the OUTEST= data set, then the regression coefficient is added as a variable to that data set. The value of the offset variable in OUTEST= data set is equal to 1 for the estimates row (_TYPE_='EST') and is equal to a special missing value (.F) for the standard error (_TYPE_='STDERR') and covariance (_TYPE_='COV') rows.

An offset variable is useful to model the scale parameter per unit of some measure of exposure. For example, in the automobile insurance context, measure of exposure can be the number of car-years insured or the total number of miles driven by a fleet of cars at a rental car company. For worker's compensation insurance, if you want to model the expected loss per enterprise, then you can use the number of employees or total employee salary as the measure of exposure. For epidemiological data, measure of exposure can be the number of people who are exposed to a certain pathogen when you are modeling the loss associated with an epidemic. In general, if e denotes the value of the exposure measure and if you specify $x_o = \log(e)$ as the offset variable, then you are modeling the influence of other regression effects (x_j) on the size of the scale of the distribution *per unit of exposure*.

Another use for an offset variable is when you have a priori knowledge of the influence of some exogenous variables that cannot be included in the SCALEMODEL statement. You can model the combined influence of such variables as an offset variable in order to correct for the omitted variable bias.

Parameter Initialization for Regression Models

The regression parameters are initialized either by using the values that you specify or by the default method.

- If you provide initial values for the regression parameters, then you must provide valid, nonmissing initial values for θ_0 and β_j parameters for all j.

  You can specify the initial value for θ_0 by using either the INEST= data set, the INSTORE= item store, or the INIT= option in the DIST statement. If the distribution has a direct scale parameter (no transformation), then the initial value for the first parameter of the distribution is used as an initial value for θ_0. If the distribution has a log-transformed scale parameter, then the initial value for the first parameter of the distribution is used as an initial value for $\log(\theta_0)$.

  You can use only the INEST= data set or the INSTORE= item store, but not both, to specify the initial values for β_j. The requirements for each option are as follows:

  – If you use the INEST= data set, then it must contain nonmissing initial values for all the regressors that you specify in the SCALEMODEL statement. The only missing value that is allowed is the special missing value .R, which indicates that the regressor is linearly dependent on other regressors. If you specify .R for a regressor for one distribution in a BY group, you must specify it the same way for all the distributions in that BY group.

Note that you cannot specify INEST= data set if the regression model contains effects that have CLASS variables or interaction effects.

– The parameter estimates in the INSTORE= item store are used to initialize the parameters of a model if the item store contains a model specification that matches the model specification in the current PROC SEVERITY step according to the following rules:
  * The distribution name and the number and names of the distribution parameters must match.
  * The model in the item store must include a scale regression model whose regression parameters match as follows:
    · If the regression model in the item store does not contain any redundant parameters, then at least one regression parameter must match. Initial values of the parameters that match are set equal to the estimates that are read from the item store, and initial values of the other regression parameters are set equal to the default value of 0.001.
    · If the regression model in the item store contains any redundant parameters, then all the regression parameters must match, and the initial values of all parameters are set equal to the estimates that are read from the item store.

  Note that a regression parameter is defined by the variables that form the underlying regression effect and by the levels of the CLASS variables if the effect contains any CLASS variables.

- If you do not specify valid initial values for θ_0 or β_j parameters for all j, then PROC SEVERITY initializes those parameters by using the following method:

  Let a random variable Y be distributed as $\mathcal{F}(\theta, \Omega)$, where θ is the scale parameter. By the definition of the scale parameter, a random variable $W = Y/\theta$ is distributed as $\mathcal{G}(\Omega)$ such that $\mathcal{G}(\Omega) = \mathcal{F}(1, \Omega)$. Given a random error term e that is generated from a distribution $\mathcal{G}(\Omega)$, a value y from the distribution of Y can be generated as

  $$y = \theta \cdot e$$

  Taking the logarithm of both sides and using the relationship of θ with the regression effects yields:

  $$\log(y) = \log(\theta_0) + \sum_{j=1}^{k} \beta_j x_j + \log(e)$$

  PROC SEVERITY makes use of the preceding relationship to initialize parameters of a regression model with distribution *dist* as follows:

  1. The following linear regression problem is solved to obtain initial estimates of β_0 and β_j:

     $$\log(y) = \beta_0 + \sum_{j=1}^{k} \beta_j x_j$$

     The estimates of $\beta_j (j = 1, \ldots, k)$ in the solution of this regression problem are used to initialize the respective regression parameters of the model. The estimate of β_0 is later used to initialize the value of θ_0.

     The results of this regression are also used to detect whether any regression parameters are linearly dependent on the other regression parameters. If any such parameters are found, then a warning is written to the SAS log and the corresponding parameter is eliminated from further analysis. The estimates for linearly dependent parameters are denoted by a special missing value of .R in the OUTEST= data set and in any displayed output.

2. Let s_0 denote the initial value of the scale parameter.

   If the distribution model of *dist* does not contain the *dist*_PARMINIT subroutine, then s_0 and all the nonscale distribution parameters are initialized to the default value of 0.001.

   However, it is strongly recommended that each distribution's model contain the *dist*_PARMINIT subroutine. For more information, see the section "Defining a Severity Distribution Model with the FCMP Procedure" on page 2148. If that subroutine is defined, then s_0 is initialized as follows:

   Each input value y_i of the response variable is transformed to its scale-normalized version w_i as

   $$w_i = \frac{y_i}{\exp(\beta_0 + \sum_{j=1}^{k} \beta_j x_{ij})}$$

   where x_{ij} denotes the value of *j*th regression effect in the *i*th input observation. These w_i values are used to compute the input arguments for the *dist*_PARMINIT subroutine. The values that are computed by the subroutine for nonscale parameters are used as their respective initial values. If the distribution has an untransformed scale parameter, then s_0 is set to the value of the scale parameter that is computed by the subroutine. If the distribution has a log-transformed scale parameter P, then s_0 is computed as $s_0 = \exp(l_0)$, where l_0 is the value of P computed by the subroutine.

3. The value of θ_0 is initialized as

   $$\theta_0 = s_0 \cdot \exp(\beta_0)$$

Reporting Estimates of Regression Parameters

When you request estimates to be written to the output (either ODS displayed output or in the OUTEST= data set), the estimate of the base value of the first distribution parameter is reported. If the first parameter is the log-transformed scale parameter, then the estimate of $\log(\theta_0)$ is reported; otherwise, the estimate of θ_0 is reported. The transform of the first parameter of a distribution *dist* is controlled by the *dist*_SCALETRANSFORM function that is defined for it.

CDF and PDF Estimates with Regression Effects

When regression effects are estimated, the estimate of the scale parameter depends on the values of the regressors and the estimates of the regression parameters. This dependency results in a potentially different distribution for each observation. To make estimates of the cumulative distribution function (CDF) and probability density function (PDF) comparable across distributions and comparable to the empirical distribution function (EDF), PROC SEVERITY computes and reports the CDF and PDF estimates from a representative distribution. The *representative distribution* is a mixture of a certain number of distributions, where each distribution differs only in the value of the scale parameter. You can specify the number of distributions in the mixture and how their scale values are chosen by using the DFMIXTURE= option in the SCALEMODEL statement.

Let N denote the number of observations that are used for estimation, K denote the number of components in the mixture distribution, s_k denote the scale parameter of the kth mixture component, and d_k denote the weight associated with kth mixture component.

Let $f(y; s_k, \hat{\Omega})$ and $F(y; s_k, \hat{\Omega})$ denote the PDF and CDF, respectively, of the kth component distribution, where $\hat{\Omega}$ denotes the set of estimates of all parameters of the distribution other than the scale parameter. Then, the PDF and CDF estimates, $f^*(y)$ and $F^*(y)$, respectively, of the mixture distribution at y are computed as

$$f^*(y) = \frac{1}{D} \sum_{k=1}^{K} d_k f(y; s_k, \hat{\Omega})$$

$$F^*(y) = \frac{1}{D} \sum_{k=1}^{K} d_k F(y; s_k, \hat{\Omega})$$

where D is the normalization factor ($D = \sum_{k=1}^{K} d_k$).

PROC SEVERITY uses the $F^*(y)$ values to compute the EDF-based statistics of fit and to create the OUTCDF= data set and the CDF plots. The PDF estimates that is plots in PDF plots are the $f^*(y)$ values.

The scale values s_k for the K mixture components are derived from the set $\{\hat{\lambda}_i\}$ ($i = 1, \ldots, N$) of N linear predictor values, where $\hat{\lambda}_i$ denotes the estimate of the linear predictor due to observation i. It is computed as

$$\hat{\lambda}_i = \log(\hat{\theta}_0) + \sum_{j=1}^{k} \hat{\beta}_j x_{ij}$$

where $\hat{\theta}_0$ is an estimate of the base value of the scale parameter, $\hat{\beta}_j$ are the estimates of regression coefficients, and x_{ij} is the value of jth regression effect in observation i.

Let w_i denote the weight of observation i. If you specify the WEIGHT statement, then the weight is equal to the value of the specified weight variable for the corresponding observation in the DATA= data set; otherwise, the weight is set to 1.

You can specify one of the following *method-names* in the DFMIXTURE= option in the SCALEMODEL statement to specify the method of choosing K and the corresponding s_k and d_k values:

FULL — In this method, there are as many mixture components as the number of observations that are used for estimation. In other words, $K = N$, $s_k = \hat{\theta}_k$, and $d_k = w_k$ ($k = 1, \ldots, N$). This is the slowest method, because it requires $O(N)$ computations to compute the mixture CDF $F^*(y_i)$ or the mixture PDF $f^*(y_i)$ of one observation. For N observations, the computational complexity in terms of number of CDF or PDF evaluations is $O(N^2)$. Even for moderately large values of N, the time that is taken to compute the mixture CDF and PDF can significantly exceed the time that is taken to estimate the model parameters. So it is recommended that you use the FULL method only for small data sets.

MEAN — In this method, the mixture contains only one distribution, whose scale value is determined by the mean of the linear predictor values that are implied by all the observations. In other words, s_1 is computed as

$$s_1 = \exp\left(\frac{1}{N} \sum_{i=1}^{N} \hat{\lambda}_i\right)$$

The component's weight d_1 is set to 1.

This method is the fastest because it requires only one CDF or PDF evaluation per observation. The computational complexity is $O(N)$ for N observations.

If you do not specify the DFMIXTURE= option in the SCALEMODEL statement, then this is the default method.

QUANTILE In this method, a certain number of quantiles are chosen from the set of all linear predictor values. If you specify a value of q for the K= option when specifying this method, then $K = q - 1$ and s_k $(k = 1, \ldots, K)$ is computed as $s_k = \exp(\hat{\lambda}_k)$, where $\hat{\lambda}_k$ is the kth q-quantile from the set $\{\hat{\lambda}_i\}$ $(i = 1, \ldots, N)$. The weight of each of the components (d_k) is assumed to be 1 for this method.

The default value of q is 2, which implies a one-point mixture that has a distribution whose scale value is equal to the median scale value.

For this method, PROC SEVERITY needs to sort the N linear predictor values in the set $\{\hat{\lambda}_i\}$; the sorting requires $O(N \log(N))$ computations. Then, computing the mixture estimate of one observation requires $(q - 1)$ CDF or PDF evaluations. Hence, the computational complexity of this method is $O(qN) + O(N \log(N))$ for computing a mixture CDF or PDF of N observations. For $q << N$, the QUANTILE method is significantly faster than the FULL method.

RANDOM In this method, a uniform random sample of observations is chosen, and the mixture contains the distributions that are implied by those observations. If you specify a value of r for the K= option when specifying this method, then the size of the sample is r. Hence, $K = r$. If l_j denotes the index of jth observation in the sample $(j = 1, \ldots, r)$, such that $1 \leq l_j \leq N$, then the scale of kth component distribution in the mixture is $s_k = \exp(\hat{\lambda}_{l_k})$. The weight of each of the components (d_k) is assumed to be 1 for this method.

You can also specify the seed to be used for generating the random sample by using the SEED= option for this method. The same sample of observations is used for all models.

Computing a mixture estimate of one observation requires r CDF or PDF evaluations. Hence, the computational complexity of this method is $O(rN)$ for computing a mixture CDF or PDF of N observations. For $r << N$, the RANDOM method is significantly faster than the FULL method.

Levelization of Classification Variables

A classification variable enters the statistical analysis or model not through its values but through its levels. The process of associating values of a variable with levels is called *levelization*.

During the process of levelization, observations that share the same value are assigned to the same level. The manner in which values are grouped can be affected by the inclusion of formats. You can determine the sort order of the levels by specifying the ORDER= option in the CLASS statement. You can also control the sort order separately for each variable in the CLASS statement.

Consider the data on nine observations in Table 30.4. The variable A is integer-valued, and the variable X is a continuous variable that has a missing value for the fourth observation. The fourth and fifth columns of Table 30.4 apply two different formats to the variable X.

Table 30.4 Example Data for Levelization

Obs	A	X	FORMAT X 3.0	FORMAT X 3.1
1	2	1.09	1	1.1
2	2	1.13	1	1.1
3	2	1.27	1	1.3
4	3	.	.	.
5	3	2.26	2	2.3
6	3	2.48	2	2.5
7	4	3.34	3	3.3
8	4	3.34	3	3.3
9	4	3.14	3	3.1

By default, levelization of the variables groups the observations by the formatted value of the variable, except for numerical variables for which no explicit format is provided. Those numerical variables are sorted by their internal value. The levelization of the four columns in Table 30.4 leads to the level assignment in Table 30.5.

Table 30.5 Values and Levels

	A		X		FORMAT X 3.0		FORMAT X 3.1	
Obs	Value	Level	Value	Level	Value	Level	Value	Level
1	2	1	1.09	1	1	1	1.1	1
2	2	1	1.13	2	1	1	1.1	1
3	2	1	1.27	3	1	1	1.3	2
4	3	2	.	.	.	.	.	.
5	3	2	2.26	4	2	2	2.3	3
6	3	2	2.48	5	2	2	2.5	4
7	4	3	3.34	7	3	3	3.3	6
8	4	3	3.34	7	3	3	3.3	6
9	4	3	3.14	6	3	3	3.1	5

You can specify the sort order for the levels of CLASS variables in the ORDER= option in the CLASS statement.

When ORDER=FORMATTED (which is the default) is in effect for numeric variables for which you have supplied no explicit format, the levels are ordered by their internal values. To order numeric class levels that have no explicit format by their BEST12. formatted values, you can specify the BEST12. format explicitly for the CLASS variables.

Table 30.6 shows how values of the ORDER= option are interpreted.

Table 30.6 Interpretation of Values of ORDER= Option

Value of ORDER=	Levels Sorted By
DATA	Order of appearance in the input data set
FORMATTED	External formatted value, except for numeric variables that have no explicit format, which are sorted by their unformatted (internal) value
FREQ	Descending frequency count (levels that have the most observations come first in the order)
INTERNAL	Unformatted value
FREQDATA	Order of descending frequency count, and within counts by order of appearance in the input data set when counts are tied
FREQFORMATTED	Order of descending frequency count, and within counts by formatted value when counts are tied
FREQINTERNAL	Order of descending frequency count, and within counts by unformatted (internal) value when counts are tied

For FORMATTED, FREQFORMATTED, FREQINTERNAL, and INTERNAL values, the sort order is machine-dependent. For more information about sort order, see the chapter about the SORT procedure in the *Base SAS Procedures Guide* and the discussion of BY-group processing in *SAS Language Reference: Concepts*.

When you specify the MISSING option in the CLASS statement, the missing values ('.' for a numeric variable and blanks for a character variable) are included in the levelization and are assigned a level. Table 30.7 displays the results of levelizing the values in Table 30.4 when the MISSING option is in effect.

Table 30.7 Values and Levels with the MISSING Option

	A		X		FORMAT x 3.0		FORMAT x 3.1	
Obs	Value	Level	Value	Level	Value	Level	Value	Level
1	2	1	1.09	2	1	2	1.1	2
2	2	1	1.13	3	1	2	1.1	2
3	2	1	1.27	4	1	2	1.3	3
4	3	2	.	1	.	1	.	1
5	3	2	2.26	5	2	3	2.3	4
6	3	2	2.48	6	2	3	2.5	5
7	4	3	3.34	8	3	4	3.3	7
8	4	3	3.34	8	3	4	3.3	7
9	4	3	3.14	7	3	4	3.1	6

When you do not specify the MISSING option, it is important to understand the implications of missing values for your statistical analysis. When PROC SEVERITY levelizes the CLASS variables, any observations for which a CLASS variable has a missing value are excluded from the analysis. This is true regardless of whether the variable is used to form the statistical model. For example, consider the case in which some observations contain missing values for variable A but the records for these observations are otherwise complete with respect to all other variables in the model. The analysis results that come from the following statements do not include any observations for which variable A contains missing values, even though A is not specified in the SCALEMODEL statement:

```
class A B;
scalemodel B x B*x;
```

You can request PROC SEVERITY to print the "Descriptive Statistics" table, which shows the number of observations that are read from the data set and the number of observations that are used in the analysis. Pay careful attention to this table—especially when your data set contains missing values—to ensure that no observations are unintentionally excluded from the analysis.

Specification and Parameterization of Model Effects

PROC SEVERITY supports formation of regression effects in the SCALEMODEL statement. A *regression effect* is formed from one or more regressor variables according to effect construction rules (*parameterization*). Each regression effect forms one element of $\mathbf{X}$ in the linear model structure $\mathbf{X}\boldsymbol{\beta}$ that affects the scale parameter. The SCALEMODEL statement in conjunction with the CLASS statement supports a rich set of effects. In order to correctly interpret the results, you need to understand the specification and parameterization of effects that are discussed in this section.

Effects are specified by a special notation that uses variable names and operators. There are two types of regressor variables: classification (or CLASS) variables and continuous variables. *Classification variables* can be either numeric or character and are specified in a CLASS statement. For more information, see the section "Levelization of Classification Variables" on page 2128. A regressor variable that is not declared in the CLASS statement is assumed to be *continuous*.

Two primary operators (crossing and nesting) are used for combining the variables, and several additional operators are used to simplify effect specification. Operators are discussed in the section "Effect Operators" on page 2131.

If you specify the CLASS statement, then PROC SEVERITY supports a general linear model (GLM) parameterization and a reference parameterization for the classification variables. The GLM parameterization is the default. For more information, see the sections "GLM Parameterization of Classification Variables and Effects" on page 2134 and "Reference Parameterization" on page 2137.

Effect Operators

Table 30.8 summarizes the operators that are available for selecting and constructing effects. These operators are discussed in the following sections.

Table 30.8 Available Effect Operators

Operator	Example	Description
Interaction	A*B	Crosses the levels of the effects
Nesting	A(B)	Nests A levels within B levels
Bar operator	A | B | C	Specifies all interactions
At sign operator	A | B | C@2	Reduces interactions in bar effects
Dash operator	A1-A10	Specifies sequentially numbered variables
Colon operator	A:	Specifies variables that have a common prefix
Double dash operator	A- -C	Specifies sequential variables in data set order

Bar and At Sign Operators

You can shorten the specification of a large factorial model by using the bar operator. For example, two ways of writing the model for a full three-way factorial model follow:

```
scalemodel A B C   A*B A*C B*C   A*B*C;

scalemodel A|B|C;
```

When you use the bar (|), the right and left sides become effects, and the cross of them becomes an effect. Multiple bars are permitted. The expressions are expanded from left to right, using rules 2–4 from Searle (1971, p. 390).

- Multiple bars are evaluated from left to right. For example, A | B | C is evaluated as follows:

$$\begin{aligned} \text{A | B | C} &\rightarrow \{\text{ A | B }\}\text{ | C} \\ &\rightarrow \{\text{ A B A*B }\}\text{ | C} \\ &\rightarrow \text{A B A*B C A*C B*C A*B*C} \end{aligned}$$

- Crossed and nested groups of variables are combined. For example, A(B) | C(D) generates A*C(B D), among other terms.
- Duplicate variables are removed. For example, A(C) | B(C) generates A*B(C C), among other terms, and the extra C is removed.
- Effects are discarded if a variable occurs on both the crossed and nested parts of an effect. For example, A(B) | B(D E) generates A*B(B D E), but this effect is eliminated immediately.

You can also specify the maximum number of variables involved in any effect that results from bar evaluation by specifying that maximum number, preceded by an at sign (@), at the end of the bar effect. For example, the following specification selects only those effects that contain two or fewer variables:

```
scalemodel A|B|C@2;
```

The preceding example is equivalent to the following SCALEMODEL statement:

```
scalemodel A B C   A*B A*C B*C;
```

More examples of using the bar and at sign operators follow:

A | C(B)	is equivalent to	A C(B) A*C(B)
A(B) | C(B)	is equivalent to	A(B) C(B) A*C(B)
A(B) | B(D E)	is equivalent to	A(B) B(D E)
A | B(A) | C	is equivalent to	A B(A) C A*C B*C(A)
A | B(A) | C@2	is equivalent to	A B(A) C A*C
A | B | C | D@2	is equivalent to	A B A*B C A*C B*C D A*D B*D C*D
A*B(C*D)	is equivalent to	A*B(C D)

NOTE: The preceding examples assume the following CLASS statement specification:

```
class A B C D;
```

Colon, Dash, and Double Dash Operators

You can simplify the specification of a large model when some of your variables have a common prefix by using the colon (:) operator and the dash (-) operator. The colon operator selects all variables that have a particular prefix, and the dash operator enables you to list variables that are numbered sequentially. For example, if your data set contains the variables X1 through X9, the following SCALEMODEL statements are equivalent:

```
scalemodel X1 X2 X3 X4 X5 X6 X7 X8 X9;

scalemodel X1-X9;

scalemodel X:;
```

If your data set contains only the three covariates X1, X2, and X9, then the colon operator selects all three variables:

```
scalemodel X:;
```

However, the following specification returns an error because X3 through X8 are not in the data set:

```
scalemodel X1-X9;
```

The double dash (--) operator enables you to select variables that are stored sequentially in the SAS data set, whether or not they have a common prefix. You can use the CONTENTS procedure (see *Base SAS Procedures Guide*) to determine your variable ordering. For example, if you replace the dash in the preceding SCALEMODEL statement with a double dash, as follows, then all three variables are selected:

```
scalemodel X1--X9;
```

If your data set contains the variables A, B, and C, then you can use the double dash operator to select these variables by specifying the following:

```
scalemodel A--C;
```

GLM Parameterization of Classification Variables and Effects

Table 30.9 shows the types of effects that are available in the SEVERITY procedure; they are discussed in more detail in the following sections. Let A, B, and C represent classification variables, and let X and Z represent continuous variables.

Table 30.9 Available Types of Effects

Effect	Example	Description
Singleton continuous	X Z	Continuous variables
Polynomial continuous	X*Z	Interaction of continuous variables
Main	A B	CLASS variables
Interaction	A*B	Crossing of CLASS variables
Nested	A(B)	Main effect A nested within CLASS effect B
Continuous-by-class	X*A	Crossing of continuous and CLASS variables
Continuous-nesting-class	X(A)	Continuous variable X nested within CLASS variable A
General	X*Z*A(B)	Combinations of different types of effects

Continuous Effects

Continuous variables or polynomial terms that involve them can be included in the model as continuous effects. An effect that contains a single continuous variable is referred to as a *singleton continuous* effect, and an effect that contains an interaction of only continuous variables is referred to as a *polynomial continuous* effect. The actual values of such terms are included as columns of the relevant model matrices. You can use the bar operator along with a continuous variable to generate polynomial effects. For example, X | X | X expands to X X*X X*X*X, which is a cubic model.

Main Effects

If a classification variable has m levels, the GLM parameterization generates m columns for its main effect in the model matrix. Each column is an indicator variable for a given level. The order of the columns is the sort order of the values of their levels and can be controlled by the ORDER= option in the CLASS statement.

Table 30.10 is an example where β_0 denotes the intercept and A and B are classification variables that have two and three levels, respectively.

Table 30.10 Example of Main Effects

Data		I	A		B		
A	B	β_0	A1	A2	B1	B2	B3
1	1	1	1	0	1	0	0
1	2	1	1	0	0	1	0
1	3	1	1	0	0	0	1
2	1	1	0	1	1	0	0
2	2	1	0	1	0	1	0
2	3	1	0	1	0	0	1

There are usually more columns for these effects than there are degrees of freedom to estimate them. In other words, the GLM parameterization of main effects is *singular*.

Interaction Effects

Often a regression model includes interaction (crossed) effects to account for how the effect of a variable changes along with the values of other variables. In an interaction, the terms are first reordered to correspond to the order of the variables in the CLASS statement. Thus, B*A becomes A*B if A precedes B in the CLASS statement. Then, the GLM parameterization generates columns for all combinations of levels that occur in the data. The order of the columns is such that the rightmost variables in the interaction change faster than the leftmost variables, as illustrated in Table 30.11.

Table 30.11 Example of Interaction Effects

Data		**I**	A		B			A*B					
A	B	β_0	A1	A2	B1	B2	B3	A1B1	A1B2	A1B3	A2B1	A2B2	A2B3
1	1	1	1	0	1	0	0	1	0	0	0	0	0
1	2	1	1	0	0	1	0	0	1	0	0	0	0
1	3	1	1	0	0	0	1	0	0	1	0	0	0
2	1	1	0	1	1	0	0	0	0	0	1	0	0
2	2	1	0	1	0	1	0	0	0	0	0	1	0
2	3	1	0	1	0	0	1	0	0	0	0	0	1

In the matrix in Table 30.11, main-effects columns are not linearly independent of crossed-effects columns. In fact, the column space for the crossed effects contains the space of the main effect.

When your regression model contains many interaction effects, you might be able to code them more parsimoniously by using the bar operator (|). The bar operator generates all possible interaction effects. For example, A | B | C expands to A B A*B C A*C B*C A*B*C. To eliminate higher-order interaction effects, use the at sign (@) in conjunction with the bar operator. For example, A | B | C | D@2 expands to A B A*B C A*C B*C D A*D B*D C*D.

Nested Effects

Nested effects are generated in the same manner as crossed effects. Hence, the design columns that are generated by the following two statements are the same (but the ordering of the columns is different):

```
scalemodel A B(A);

scalemodel A A*B;
```

The nesting operator in PROC SEVERITY is more of a notational convenience than an operation that is distinct from crossing. Nested effects are usually characterized by the property that the nested variables do not appear as main effects. The order of the variables within nesting parentheses is made to correspond to the order of these variables in the CLASS statement. The order of the columns is such that variables outside the parentheses index faster than those inside the parentheses, and the rightmost nested variables index faster than the leftmost variables, as illustrated in Table 30.12.

Table 30.12 Example of Nested Effects

Data		**I**	A		B(A)					
A	B	β_0	A1	A2	B1A1	B2A1	B3A1	B1A2	B2A2	B3A2
1	1	1	1	0	1	0	0	0	0	0
1	2	1	1	0	0	1	0	0	0	0
1	3	1	1	0	0	0	1	0	0	0
2	1	1	0	1	0	0	0	1	0	0

Table 30.12 *continued*

Data		I	A		B(A)					
2	2	1	0	1	0	0	0	0	1	0
2	3	1	0	1	0	0	0	0	0	1

Continuous-Nesting-Class Effects

When a continuous variable nests or crosses with a classification variable, the design columns are constructed by multiplying the continuous values into the design columns for the classification effect, as illustrated in Table 30.13.

Table 30.13 Example of Continuous-Nesting-Class Effects

Data		I	A		X(A)	
X	A	β_0	A1	A2	X(A1)	X(A2)
21	1	1	1	0	21	0
24	1	1	1	0	24	0
22	1	1	1	0	22	0
28	2	1	0	1	0	28
19	2	1	0	1	0	19
23	2	1	0	1	0	23

Continuous-by-Class Effects

Continuous-by-class effects generate the same design columns as continuous-nesting-class effects. Table 30.14 shows the construction of the X*A effect. The two columns for this effect are the same as the columns for the X(A) effect in Table 30.13.

Table 30.14 Example of Continuous-by-Class Effects

Data		I	X	A		X*A	
X	A	β_0	X	A1	A2	X*A1	X*A2
21	1	1	21	1	0	21	0
24	1	1	24	1	0	24	0
22	1	1	22	1	0	22	0
28	2	1	28	0	1	0	28
19	2	1	19	0	1	0	19
23	2	1	23	0	1	0	23

General Effects

An example that combines all the effects is X1*X2*A*B*C(D E). The continuous list comes first, followed by the crossed list, followed by the nested list in parentheses. PROC SEVERITY might rename effects to correspond to ordering rules. For example, B*A(E D) might be renamed A*B(D E) to satisfy the following:

- Classification variables that occur outside parentheses (crossed effects) are sorted in the order in which they appear in the CLASS statement.

- Variables within parentheses (nested effects) are sorted in the order in which they appear in the CLASS statement.

The sequencing of the parameters that are generated by an effect is determined by the variables whose levels are indexed faster:

- Variables in the crossed list index faster than variables in the nested list.
- Within a crossed or nested list, variables to the right index faster than variables to the left.

For example, suppose a model includes four effects—A, B, C, and D—each of which has two levels, 1 and 2. Assume the CLASS statement is

```
class A B C D;
```

Then the order of the parameters for the effect B*A(C D), which is renamed A*B(C D), is

$$\begin{array}{llll} A_1B_1C_1D_1 \rightarrow & A_1B_2C_1D_1 \rightarrow & A_2B_1C_1D_1 \rightarrow & A_2B_2C_1D_1 \rightarrow \\ A_1B_1C_1D_2 \rightarrow & A_1B_2C_1D_2 \rightarrow & A_2B_1C_1D_2 \rightarrow & A_2B_2C_1D_2 \rightarrow \\ A_1B_1C_2D_1 \rightarrow & A_1B_2C_2D_1 \rightarrow & A_2B_1C_2D_1 \rightarrow & A_2B_2C_2D_1 \rightarrow \\ A_1B_1C_2D_2 \rightarrow & A_1B_2C_2D_2 \rightarrow & A_2B_1C_2D_2 \rightarrow & A_2B_2C_2D_2 \end{array}$$

Note that first the crossed effects B and A are sorted in the order in which they appear in the CLASS statement so that A precedes B in the parameter list. Then, for each combination of the nested effects in turn, combinations of A and B appear. The B effect changes fastest because it is rightmost in the cross list. Then A changes next fastest, and D changes next fastest after that. The C effect changes most slowly because it is leftmost in the nested list.

Reference Parameterization

Classification variables can be represented in the reference parameterization. Consider the classification variable A that has four values, 1, 2, 5, and 7. The reference parameterization generates three columns (one less than the number of variable levels). The columns indicate group membership of the nonreference levels. For the reference level, the three dummy variables have a value of 0. If the reference level is 7 (REF='7'), the design columns for variable A are as shown in Table 30.15.

Table 30.15 Reference Coding

	Design Matrix		
A	**A1**	**A2**	**A5**
1	1	0	0
2	0	1	0
5	0	0	1
7	0	0	0

Parameter estimates of CLASS main effects that use the reference coding scheme estimate the difference in the effect of each nonreference level compared to the effect of the reference level.

Empirical Distribution Function Estimation Methods

The empirical distribution function (EDF) is a nonparametric estimate of the cumulative distribution function (CDF) of the distribution. PROC SEVERITY computes EDF estimates for two purposes: to send the estimates to a distribution's PARMINIT subroutine in order to initialize the distribution parameters, and to compute the EDF-based statistics of fit.

To reduce the time that it takes to compute the EDF estimates, you can use the INITSAMPLE option to specify that only a fraction of the input data be used. If you do not specify the INITSAMPLE option, then PROC SEVERITY computes the EDF estimates by using all valid observations in the DATA= data set, or by using all valid observations in the current BY group if you specify a BY statement.

This section describes the methods that are used for computing EDF estimates.

Notation

Let there be a set of N observations, each containing a quintuplet of values $(y_i, t_i^l, t_i^r, c_i^r, c_i^l), i = 1, \ldots, N$, where y_i is the value of the response variable, t_i^l is the value of the left-truncation threshold, t_i^r is the value of the right-truncation threshold, c_i^r is the value of the right-censoring limit, and c_i^l is the value of the left-censoring limit.

If an observation is not left-truncated, then $t_i^l = \tau^l$, where τ^l is the smallest value in the support of the distribution; so $F(t_i^l) = 0$. If an observation is not right-truncated, then $t_i^r = \tau_h$, where τ_h is the largest value in the support of the distribution; so $F(t_i^r) = 1$. If an observation is not right-censored, then $c_i^r = \tau^l$; so $F(c_i^r) = 0$. If an observation is not left-censored, then $c_i^l = \tau_h$; so $F(c_i^l) = 1$.

Let w_i denote the weight associated with ith observation. If you specify the WEIGHT statement, then w_i is the normalized value of the weight variable; otherwise, it is set to 1. The weights are normalized such that they sum up to N.

An indicator function $I[e]$ takes a value of 1 or 0 if the expression e is true or false, respectively.

Estimation Methods

If the response variable is subject to both left-censoring and right-censoring effects, then PROC SEVERITY uses the Turnbull's method. This section describes methods other than Turnbull's method. For Turnbull's method, see the next section "Turnbull's EDF Estimation Method" on page 2140.

The method descriptions assume that all observations are either uncensored or right-censored; that is, each observation is of the form $(y_i, t_i^l, t_i^r, \tau^l, \tau_h)$ or $(y_i, t_i^l, t_i^r, c_i^r, \tau_h)$.

If all observations are either uncensored or left-censored, then each observation is of the form $(y_i, t_i^l, t_i^r, \tau_l, c_i^l)$. It is converted to an observation $(-y_i, -t_i^r, -t_i^l, -c_i^l, \tau_h)$; that is, the signs of all the response variable values are reversed, the new left-truncation threshold is equal to the negative of the original right-truncation threshold, the new right-truncation threshold is equal to the negative of the original left-truncation threshold, and the negative of the original left-censoring limit becomes the new right-censoring limit. With this transformation, each observation is either uncensored or right-censored. The methods described for handling uncensored or right-censored data are now applicable. After the EDF estimates are computed, the observations are transformed back to the original form and EDF estimates are adjusted such $F_n(y_i) = 1 - F_n(-y_i-)$, where $F_n(-y_i-)$ denotes the EDF estimate of the value slightly less than the transformed value $-y_i$.

Further, a set of uncensored or right-censored observations can be converted to a set of observations of the form $(y_i, t_i^l, t_i^r, \delta_i)$, where δ_i is the indicator of right-censoring. $\delta_i = 0$ indicates a right-censored observation, in which case y_i is assumed to record the right-censoring limit c_i^r. $\delta_i = 1$ indicates an uncensored observation, and y_i records the exact observed value. In other words, $\delta_i = I[Y \leq C^r]$ and $y_i = \min(y_i, c_i^r)$.

Given this notation, the EDF is estimated as

$$F_n(y) = \begin{cases} 0 & \text{if } y < y^{(1)} \\ \hat{F}_n(y^{(k)}) & \text{if } y^{(k)} \leq y < y^{(k+1)}, k = 1, \ldots, N-1 \\ \hat{F}_n(y^{(N)}) & \text{if } y^{(N)} \leq y \end{cases}$$

where $y^{(k)}$ denotes the kth order statistic of the set $\{y_i\}$ and $\hat{F}_n(y^{(k)})$ is the estimate computed at that value. The definition of $\hat{F}_n$ depends on the estimation method. You can specify a particular method or let PROC SEVERITY choose an appropriate method by using the EMPIRICALCDF= option in the PROC SEVERITY statement. Each method computes $\hat{F}_n$ as follows:

STANDARD

This method is the standard way of computing EDF. The EDF estimate at observation i is computed as follows:

$$\hat{F}_n(y_i) = \frac{1}{N} \sum_{j=1}^{N} w_j \cdot I[y_j \leq y_i]$$

If you do not specify any censoring or truncation information, then this method is chosen. If you explicitly specify this method, then PROC SEVERITY ignores any censoring and truncation information that you specify in the LOSS statement.

The standard error of $\hat{F}_n(y_i)$ is computed by using the normal approximation method:

$$\hat{\sigma}_n(y_i) = \sqrt{\hat{F}_n(y_i)(1 - \hat{F}_n(y_i))/N}$$

KAPLANMEIER

The Kaplan-Meier (KM) estimator, also known as the product-limit estimator, was first introduced by Kaplan and Meier (1958) for censored data. Lynden-Bell (1971) derived a similar estimator for left-truncated data. PROC SEVERITY uses the definition that combines both censoring and truncation information (Klein and Moeschberger 1997; Lai and Ying 1991).

The EDF estimate at observation i is computed as

$$\hat{F}_n(y_i) = 1 - \prod_{\tau \leq y_i} \left(1 - \frac{n(\tau)}{R_n(\tau)}\right)$$

where $n(\tau)$ and $R_n(\tau)$ are defined as follows:

- $n(\tau) = \sum_{k=1}^{N} w_k \cdot I[y_k = \tau \text{ and } \tau \leq t_k^r \text{ and } \delta_k = 1]$, which is the number of uncensored observations ($\delta_k = 1$) for which the response variable value is equal to τ and τ is observable according to the right-truncation threshold of that observation ($\tau \leq t_k^r$).

- $R_n(\tau) = \sum_{k=1}^{N} w_k \cdot I[y_k \geq \tau > t_k^l]$, which is the size (cardinality) of the risk set at τ. The term *risk set* has its origins in survival analysis; it contains the events that are at risk of failure at a given time, τ. In other words, it contains the events that have survived up to time τ and might fail at or after τ. For PROC SEVERITY, *time* is equivalent to the magnitude of the event and *failure* is equivalent to an uncensored and observable event, where observable means it satisfies the truncation thresholds.

This method is chosen when you specify at least one form of censoring or truncation.

The standard error of $\hat{F}_n(y_i)$ is computed by using Greenwood's formula (Greenwood 1926):

$$\hat{\sigma}_n(y_i) = \sqrt{(1 - \hat{F}_n(y_i))^2 \cdot \sum_{\tau \leq y_i} \left(\frac{n(\tau)}{R_n(\tau)(R_n(\tau) - n(\tau))} \right)}$$

MODIFIEDKM

The product-limit estimator used by the KAPLANMEIER method does not work well if the risk set size becomes very small. For right-censored data, the size can become small towards the right tail. For left-truncated data, the size can become small at the left tail and can remain so for the entire range of data. This was demonstrated by Lai and Ying (1991). They proposed a modification to the estimator that ignores the effects due to small risk set sizes.

The EDF estimate at observation i is computed as

$$\hat{F}_n(y_i) = 1 - \prod_{\tau \leq y_i} \left(1 - \frac{n(\tau)}{R_n(\tau)} \cdot I[R_n(\tau) \geq cN^{\alpha}] \right)$$

where the definitions of $n(\tau)$ and $R_n(\tau)$ are identical to those used for the KAPLAN-MEIER method described previously.

You can specify the values of c and α by using the C= and ALPHA= options. If you do not specify a value for c, the default value used is $c = 1$. If you do not specify a value for α, the default value used is $\alpha = 0.5$.

As an alternative, you can also specify an absolute lower bound, say L, on the risk set size by using the RSLB= option, in which case $I[R_n(\tau) \geq cN^{\alpha}]$ is replaced by $I[R_n(\tau) \geq L]$ in the definition.

The standard error of $\hat{F}_n(y_i)$ is computed by using Greenwood's formula (Greenwood 1926):

$$\hat{\sigma}_n(y_i) = \sqrt{(1 - \hat{F}_n(y_i))^2 \cdot \sum_{\tau \leq y_i} \left(\frac{n(\tau)}{R_n(\tau)(R_n(\tau) - n(\tau))} \cdot I[R_n(\tau) \geq cN^{\alpha}] \right)}$$

Turnbull's EDF Estimation Method

If the response variable is subject to both left-censoring and right-censoring effects, then the SEVERITY procedure uses a method proposed by Turnbull (1976) to compute the nonparametric estimates of the cumulative distribution function. The original Turnbull's method is modified using the suggestions made by Frydman (1994) when truncation effects are present.

Let the input data consist of N observations in the form of quintuplets of values $(y_i, t_i^l, t_i^r, c_i^r, c_i^l), i = 1, \ldots, N$ with notation described in the section "Notation" on page 2138. For each observation, let $A_i = (c_i^r, c_i^l]$ be the censoring interval; that is, the response variable value is known to lie in the interval A_i, but the exact value is not known. If an observation is uncensored, then $A_i = (y_i - \epsilon, y_i]$ for any arbitrarily small value of $\epsilon > 0$. If an observation is censored, then the value y_i is ignored. Similarly, for each observation, let $B_i = (t_i^l, t_i^r]$ be the truncation interval; that is, the observation is drawn from a truncated (conditional) distribution $F(y, B_i) = P(Y \leq y | Y \in B_i)$.

Two sets, L and R, are formed using A_i and B_i as follows:

$$L = \{c_i^r, 1 \leq i \leq N\} \cup \{t_i^r, 1 \leq i \leq N\}$$
$$R = \{c_i^l, 1 \leq i \leq N\} \cup \{t_i^l, 1 \leq i \leq N\}$$

The sets L and R represent the left endpoints and right endpoints, respectively. A set of disjoint intervals $C_j = [q_j, p_j], 1 \leq j \leq M$ is formed such that $q_j \in L$ and $p_j \in R$ and $q_j \leq p_j$ and $p_j < q_{j+1}$. The value of M is dependent on the nature of censoring and truncation intervals in the input data. Turnbull (1976) showed that the maximum likelihood estimate (MLE) of the EDF can increase only inside intervals C_j. In other words, the MLE estimate is constant in the interval (p_j, q_{j+1}). The likelihood is independent of the behavior of F_n inside any of the intervals C_j. Let s_j denote the increase in F_n inside an interval C_j. Then, the EDF estimate is as follows:

$$F_n(y) = \begin{cases} 0 & \text{if } y < q_1 \\ \sum_{k=1}^{j} s_k & \text{if } p_j < y < q_{j+1}, 1 \leq j \leq M-1 \\ 1 & \text{if } y > p_M \end{cases}$$

PROC SEVERITY computes the estimates $F_n(p_j+) = F_n(q_{j+1}-) = \sum_{k=1}^{j} s_k$ at points p_j and q_{j+1} and computes $F_n(q_1-) = 0$ at point q_1, where $F_n(x+)$ denotes the limiting estimate at a point that is infinitesimally larger than x when approaching x from values larger than x and where $F_n(x-)$ denotes the limiting estimate at a point that is infinitesimally smaller than x when approaching x from values smaller than x.

PROC SEVERITY uses the expectation-maximization (EM) algorithm proposed by Turnbull (1976), who referred to the algorithm as the self-consistency algorithm. By default, the algorithm runs until one of the following criteria is met:

- Relative-error criterion: The maximum relative error between the two consecutive estimates of s_j falls below a threshold ϵ. If l indicates an index of the current iteration, then this can be formally stated as

$$\arg \max_{1 \leq j \leq M} \left\{ \frac{|s_j^l - s_j^{l-1}|}{s_j^{l-1}} \right\} \leq \epsilon$$

  You can control the value of ϵ by specifying the EPS= suboption of the EDF=TURNBULL option in the PROC SEVERITY statement. The default value is 1.0E–8.

- Maximum-iteration criterion: The number of iterations exceeds an upper limit that you specify for the MAXITER= suboption of the EDF=TURNBULL option in the PROC SEVERITY statement. The default number of maximum iterations is 500.

The self-consistent estimates obtained in this manner might not be maximum likelihood estimates. Gentleman and Geyer (1994) suggested the use of the Kuhn-Tucker conditions for the maximum likelihood problem to ensure that the estimates are MLE. If you specify the ENSUREMLE suboption of the EDF=TURNBULL option in the PROC SEVERITY statement, then PROC SEVERITY computes the Kuhn-Tucker conditions at the end of each iteration to determine whether the estimates $\{s_j\}$ are MLE. If you do not specify any truncation effects, then the Kuhn-Tucker conditions derived by Gentleman and Geyer (1994) are used. If you specify any truncation effects, then PROC SEVERITY uses modified Kuhn-Tucker conditions that account for the truncation effects. An integral part of checking the conditions is to determine whether an estimate s_j is zero or whether an estimate of the Lagrange multiplier or the reduced gradient associated with the estimate s_j is zero. PROC SEVERITY declares these values to be zero if they are less than or equal to a threshold δ. You can control the value of δ by specifying the ZEROPROB= suboption of the EDF=TURNBULL option in the PROC SEVERITY statement. The default value is 1.0E–8. The algorithm continues until the Kuhn-Tucker conditions are satisfied or the number of iterations exceeds the upper limit. The relative-error criterion stated previously is not used when you specify the ENSUREMLE option.

The standard errors for Turnbull's EDF estimates are computed by using the asymptotic theory of the maximum likelihood estimators (MLE), even though the final estimates might not be MLE. Turnbull's estimator essentially attempts to maximize the likelihood L, which depends on the parameters s_j $(j = 1 \ldots M)$. Let $\boldsymbol{s} = \{s_j\}$ denote the set of these parameters. If $\mathbf{G}(\boldsymbol{s}) = \nabla^2(-\log(L(\boldsymbol{s})))$ denotes the Hessian matrix of the negative of log likelihood, then the variance-covariance matrix of $\boldsymbol{s}$ is estimated as $\hat{\mathbf{C}}(\boldsymbol{s}) = \mathbf{G}^{-1}(\boldsymbol{s})$. Given this matrix, the standard error of $F_n(y)$ is computed as

$$\sigma_n(y) = \sqrt{\sum_{k=1}^{j}\left(\hat{C}_{kk} + 2\cdot\sum_{l=1}^{k-1}\hat{C}_{kl}\right)}, \text{ if } p_j < y < q_{j+1}, 1 \le j \le M-1$$

The standard error is undefined outside of these intervals.

EDF Estimates and Truncation

If you specify truncation, then the estimate $\hat{F}_n(y)$ that is computed by any method other than the STANDARD method is a *conditional* estimate. In other words, $\hat{F}_n(y) = \Pr(Y \le y | \tau_G < Y \le \tau_H)$, where G and H denote the (unknown) distribution functions of the left-truncation threshold variable T^l and the right-truncation threshold variable T^r, respectively, τ_G denotes the smallest left-truncation threshold with a nonzero cumulative probability, and τ_H denotes the largest right-truncation threshold with a nonzero cumulative probability. Formally, $\tau_G = \inf\{s : G(s) > 0\}$ and $\tau_H = \sup\{s : H(s) > 0\}$. For computational purposes, PROC SEVERITY estimates τ_G and τ_H by $t^l_{\min}$ and $t^r_{\max}$, respectively, defined as

$$t^l_{\min} = \min\{t^l_k : 1 \le k \le N\}$$
$$t^r_{\max} = \max\{t^r_k : 1 \le k \le N\}$$

These estimates of $t^l_{\min}$ and $t^r_{\max}$ are used to compute the conditional estimates of the CDF as described in the section "Truncation and Conditional CDF Estimates" on page 2119.

If you specify left-truncation *with* the probability of observability p, then PROC SEVERITY uses the additional information provided by p to compute an estimate of the EDF that is not conditional on the left-truncation information. In particular, for each left-truncated observation i with response variable value y_i and truncation threshold t^l_i, an observation j is added with *weight* $w_j = (1-p)/p$ and $y_j = t^l_j$. Each added observation is assumed to be uncensored and untruncated. Then, your specified EDF method is used

by assuming no left-truncation. The EDF estimate that is obtained using this method is not conditional on the left-truncation information. For the KAPLANMEIER and MODIFIEDKM methods with uncensored or right-censored data, definitions of $n(\tau)$ and $R_n(\tau)$ are modified to account for the added observations. If N^a denotes the total number of observations including the added observations, then $n(\tau)$ is defined as $n(\tau) = \sum_{k=1}^{N^a} w_k I[y_k = \tau \text{ and } \tau \leq t_k^r \text{ and } \delta_k = 1]$, and $R_n(\tau)$ is defined as $R_n(\tau) = \sum_{k=1}^{N^a} w_k I[y_k \geq \tau]$. In the definition of $R_n(\tau)$, the left-truncation information is not used, because it was used along with p to add the observations.

If the original data are a combination of left- and right-censored data, then Turnbull's method is applied to the appended set that contains no left-truncated observations.

Supplying EDF Estimates to Functions and Subroutines

The parameter initialization subroutines in distribution models and some predefined utility functions require EDF estimates. For more information, see the sections "Defining a Severity Distribution Model with the FCMP Procedure" on page 2148 and "Predefined Utility Functions" on page 2160.

PROC SEVERITY supplies the EDF estimates to these subroutines and functions by using two arrays, x and F, the dimension of each array, and a type of the EDF estimates. The type identifies how the EDF estimates are computed and stored. They are as follows:

Type 1 specifies that EDF estimates are computed using the STANDARD method; that is, the data that are used for estimation are neither censored nor truncated.

Type 2 specifies that EDF estimates are computed using either the KAPLANMEIER or the MODIFIEDKM method; that is, the data that are used for estimation are subject to truncation and one type of censoring (left or right, but not both).

Type 3 specifies that EDF estimates are computed using the TURNBULL method; that is, the data that are used for estimation are subject to both left- and right-censoring. The data might or might not be truncated.

For Types 1 and 2, the EDF estimates are stored in arrays x and F of dimension N such that the following holds:

$$F_n(y) = \begin{cases} 0 & \text{if } y < x[1] \\ F[k] & \text{if } x[k] \leq y < x[k+1], k = 1, \ldots, N-1 \\ F[N] & \text{if } x[N] \leq y \end{cases}$$

where $[k]$ denotes kth element of the array ($[1]$ denotes the first element of the array).

For Type 3, the EDF estimates are stored in arrays x and F of dimension N such that the following holds:

$$F_n(y) = \begin{cases} 0 & \text{if } y < x[1] \\ \text{undefined} & \text{if } x[2k-1] \leq y < x[2k], k = 1, \ldots, (N-1)/2 \\ F[2k] = F[2k+1] & \text{if } x[2k] \leq y < x[2k+1], k = 1, \ldots, (N-1)/2 \\ F[N] & \text{if } x[N] \leq y \end{cases}$$

Although the behavior of EDF is theoretically undefined for the interval $[x[2k-1], x[2k])$, for computational purposes, all predefined functions and subroutines assume that the EDF increases linearly from $F[2k-1]$ to $F[2k]$ in that interval if $x[2k-1] < x[2k]$. If $x[2k-1] = x[2k]$, which can happen when the EDF is estimated from a combination of uncensored and interval-censored data, the predefined functions and subroutines assume that $F_n(x[2k-1]) = F_n(x[2k]) = F[2k]$.

Statistics of Fit

PROC SEVERITY computes and reports various statistics of fit to indicate how well the estimated model fits the data. The statistics belong to two categories: likelihood-based statistics and EDF-based statistics. Neg2LogLike, AIC, AICC, and BIC are likelihood-based statistics, and KS, AD, and CvM are EDF-based statistics. The following subsections provide definitions of each.

Likelihood-Based Statistics of Fit

Let $y_i, i = 1, \ldots, N$, denote the response variable values. Let L be the likelihood as defined in the section "Likelihood Function" on page 2120. Let p denote the number of model parameters that are estimated. Note that $p = p_d + (k - k_r)$, where p_d is the number of distribution parameters, k is the number of all regression parameters, and k_r is the number of regression parameters that are found to be linearly dependent (redundant) on other regression parameters. Given this notation, the likelihood-based statistics are defined as follows:

Neg2LogLike The log likelihood is reported as

$$\text{Neg2LogLike} = -2\log(L)$$

The multiplying factor -2 makes it easy to compare it to the other likelihood-based statistics. A model that has a smaller value of Neg2LogLike is deemed better.

AIC Akaike's information criterion (AIC) is defined as

$$\text{AIC} = -2\log(L) + 2p$$

A model that has a smaller AIC value is deemed better.

AICC The corrected Akaike's information criterion (AICC) is defined as

$$\text{AICC} = -2\log(L) + \frac{2Np}{N - p - 1}$$

A model that has a smaller AICC value is deemed better. It corrects the finite-sample bias that AIC has when N is small compared to p. AICC is related to AIC as

$$\text{AICC} = \text{AIC} + \frac{2p(p+1)}{N - p - 1}$$

As N becomes large compared to p, AICC converges to AIC. AICC is usually recommended over AIC as a model selection criterion.

BIC The Schwarz Bayesian information criterion (BIC) is defined as

$$\text{BIC} = -2\log(L) + p\log(N)$$

A model that has a smaller BIC value is deemed better.

EDF-Based Statistics

This class of statistics is based on the difference between the estimate of the cumulative distribution function (CDF) and the estimate of the empirical distribution function (EDF). A model that has a smaller value of the chosen EDF-based statistic is deemed better.

Let $y_i, i = 1, \ldots, N$ denote the sample of N values of the response variable. Let $r_i = \sum_{j=1}^{N} I[y_j \leq y_i]$ denote the number of observations with a value less than or equal to y_i, where I is an indicator function. Let $F_n(y_i)$ denote the EDF estimate that is computed by using the method that you specify in the EMPIRICAL-CDF= option. Let $Z_i = \hat{F}(y_i)$ denote the estimate of the CDF. Let $F_n(Z_i)$ denote the EDF estimate of Z_i values that are computed using the same method that is used to compute the EDF of y_i values. Using the probability integral transformation, if $F(y)$ is the true distribution of the random variable Y, then the random variable $Z = F(y)$ is uniformly distributed between 0 and 1 (D'Agostino and Stephens 1986, Ch. 4). Thus, comparing $F_n(y_i)$ with $\hat{F}(y_i)$ is equivalent to comparing $F_n(Z_i)$ with $\hat{F}(Z_i) = Z_i$ (uniform distribution).

Note the following two points regarding which CDF estimates are used for computing the test statistics:

- If you specify regression effects, then the CDF estimates Z_i that are used for computing the EDF test statistics are from a mixture distribution. See the section "CDF and PDF Estimates with Regression Effects" on page 2126 for more information.
- If the EDF estimates are conditional because of the truncation information, then each unconditional estimate Z_i is converted to a conditional estimate using the method described in the section "Truncation and Conditional CDF Estimates" on page 2119.

In the following, it is assumed that Z_i denotes an appropriate estimate of the CDF if you specify any truncation or regression effects. Given this, the EDF-based statistics of fit are defined as follows:

KS The Kolmogorov-Smirnov (KS) statistic computes the largest vertical distance between the CDF and the EDF. It is formally defined as follows:

$$\text{KS} = \sup_y |F_n(y) - F(y)|$$

If the STANDARD method is used to compute the EDF, then the following formula is used:

$$D^+ = \max_i \left(\frac{r_i}{N} - Z_i\right)$$
$$D^- = \max_i \left(Z_i - \frac{r_{i-1}}{N}\right)$$
$$\text{KS} = \sqrt{N}\, \max(D^+, D^-) + \frac{0.19}{\sqrt{N}}$$

Note that r_0 is assumed to be 0.

If the method used to compute the EDF is any method other than the STANDARD method, then the following formula is used:

$$D^+ = \max_i (F_n(Z_i) - Z_i), \text{ if } F_n(Z_i) \geq Z_i$$
$$D^- = \max_i (Z_i - F_n(Z_i)), \text{ if } F_n(Z_i) < Z_i$$
$$\text{KS} = \sqrt{N}\, \max(D^+, D^-) + \frac{0.19}{\sqrt{N}}$$

AD The Anderson-Darling (AD) statistic is a quadratic EDF statistic that is proportional to the expected value of the weighted squared difference between the EDF and CDF. It is formally defined as follows:

$$\text{AD} = N \int_{-\infty}^{\infty} \frac{(F_n(y) - F(y))^2}{F(y)(1 - F(y))} dF(y)$$

If the STANDARD method is used to compute the EDF, then the following formula is used:

$$\text{AD} = -N - \frac{1}{N} \sum_{i=1}^{N} [(2r_i - 1)\log(Z_i) + (2N + 1 - 2r_i)\log(1 - Z_i)]$$

If the method used to compute the EDF is any method other than the STANDARD method, then the statistic can be computed by using the following two pieces of information:

- If the EDF estimates are computed using the KAPLANMEIER or MODIFIEDKM methods, then EDF is a step function such that the estimate $F_n(z)$ is a constant equal to $F_n(Z_{i-1})$ in interval $[Z_{i-1}, Z_i]$. If the EDF estimates are computed using the TURNBULL method, then there are two types of intervals: one in which the EDF curve is constant and the other in which the EDF curve is theoretically undefined. For computational purposes, it is assumed that the EDF curve is linear for the latter type of the interval. For each method, the EDF estimate $F_n(y)$ at y can be written as

  $$F_n(z) = F_n(Z_{i-1}) + S_i(z - Z_{i-1}), \text{ for } z \in [Z_{i-1}, Z_i]$$

  where S_i is the slope of the line defined as

  $$S_i = \frac{F_n(Z_i) - F_n(Z_{i-1})}{Z_i - Z_{i-1}}$$

  For the KAPLANMEIER or MODIFIEDKM method, $S_i = 0$ in each interval.
- Using the probability integral transform $z = F(y)$, the formula simplifies to

  $$\text{AD} = N \int_{-\infty}^{\infty} \frac{(F_n(z) - z)^2}{z(1 - z)} dz$$

The computation formula can then be derived from the following approximation:

$$\begin{aligned}
\text{AD} &= N \sum_{i=1}^{K+1} \int_{Z_{i-1}}^{Z_i} \frac{(F_n(z) - z)^2}{z(1 - z)} dz \\
&= N \sum_{i=1}^{K+1} \int_{Z_{i-1}}^{Z_i} \frac{(F_n(Z_{i-1}) + S_i(z - Z_{i-1}) - z)^2}{z(1 - z)} dz \\
&= N \sum_{i=1}^{K+1} \int_{Z_{i-1}}^{Z_i} \frac{(P_i - Q_i z)^2}{z(1 - z)} dz
\end{aligned}$$

where $P_i = F_n(Z_{i-1}) - S_i Z_{i-1}$, $Q_i = 1 - S_i$, and K is the number of points at which the EDF estimate are computed. For the TURNBULL method, $K = 2k$ for some k.

Assuming $Z_0 = 0$, $Z_{K+1} = 1$, $F_n(0) = 0$, and $F_n(Z_K) = 1$ yields the following computation formula:

$$\begin{aligned}\text{AD} = &- N(Z_1 + \log(1 - Z_1) + \log(Z_K) + (1 - Z_K)) \\ &+ N \sum_{i=2}^{K} \left[P_i^2 A_i - (Q_i - P_i)^2 B_i - Q_i^2 C_i \right]\end{aligned}$$

where $A_i = \log(Z_i) - \log(Z_{i-1})$, $B_i = \log(1 - Z_i) - \log(1 - Z_{i-1})$, and $C_i = Z_i - Z_{i-1}$.

If EDF estimates are computed using the KAPLANMEIER or MODIFIEDKM method, then $P_i = F_n(Z_{i-1})$ and $Q_i = 1$, which simplifies the formula as

$$\begin{aligned}\text{AD} = &- N(1 + \log(1 - Z_1) + \log(Z_K)) \\ &+ N \sum_{i=2}^{K} \left[F_n(Z_{i-1})^2 A_i - (1 - F_n(Z_{i-1}))^2 B_i \right]\end{aligned}$$

CvM The Cramér-von Mises (CvM) statistic is a quadratic EDF statistic that is proportional to the expected value of the squared difference between the EDF and CDF. It is formally defined as follows:

$$\text{CvM} = N \int_{-\infty}^{\infty} (F_n(y) - F(y))^2 dF(y)$$

If the STANDARD method is used to compute the EDF, then the following formula is used:

$$\text{CvM} = \frac{1}{12N} + \sum_{i=1}^{N} \left(Z_i - \frac{(2r_i - 1)}{2N} \right)^2$$

If the method used to compute the EDF is any method other than the STANDARD method, then the statistic can be computed by using the following two pieces of information:

- As described previously for the AD statistic, the EDF estimates are assumed to be piecewise linear such that the estimate $F_n(y)$ at y is

  $$F_n(z) = F_n(Z_{i-1}) + S_i(z - Z_{i-1}), \text{ for } z \in [Z_{i-1}, Z_i]$$

  where S_i is the slope of the line defined as

  $$S_i = \frac{F_n(Z_i) - F_n(Z_{i-1})}{Z_i - Z_{i-1}}$$

  For the KAPLANMEIER or MODIFIEDKM method, $S_i = 0$ in each interval.

- Using the probability integral transform $z = F(y)$, the formula simplifies to:

  $$\text{CvM} = N \int_{-\infty}^{\infty} (F_n(z) - z)^2 dz$$

The computation formula can then be derived from the following approximation:

$$\begin{aligned}\text{CvM} &= N \sum_{i=1}^{K+1} \int_{Z_{i-1}}^{Z_i} (F_n(z) - z)^2 dz \\ &= N \sum_{i=1}^{K+1} \int_{Z_{i-1}}^{Z_i} (F_n(Z_{i-1}) + S_i(z - Z_{i-1}) - z)^2 dz \\ &= N \sum_{i=1}^{K+1} \int_{Z_{i-1}}^{Z_i} (P_i - Q_i z)^2 dz\end{aligned}$$

where $P_i = F_n(Z_{i-1}) - S_i Z_{i-1}$, $Q_i = 1 - S_i$, and K is the number of points at which the EDF estimate are computed. For the TURNBULL method, $K = 2k$ for some k.

Assuming $Z_0 = 0$, $Z_{K+1} = 1$, and $F_n(0) = 0$ yields the following computation formula:

$$\text{CvM} = N\frac{Z_1^3}{3} + N \sum_{i=2}^{K+1} \left[P_i^2 A_i - P_i Q_i B_i - \frac{Q_i^2}{3} C_i \right]$$

where $A_i = Z_i - Z_{i-1}$, $B_i = Z_i^2 - Z_{i-1}^2$, and $C_i = Z_i^3 - Z_{i-1}^3$.

If EDF estimates are computed using the KAPLANMEIER or MODIFIEDKM method, then $P_i = F_n(Z_{i-1})$ and $Q_i = 1$, which simplifies the formula as

$$\text{CvM} = \frac{N}{3} + N \sum_{i=2}^{K+1} \left[F_n(Z_{i-1})^2 (Z_i - Z_{i-1}) - F_n(Z_{i-1})(Z_i^2 - Z_{i-1}^2) \right]$$

which is similar to the formula proposed by Koziol and Green (1976).

Defining a Severity Distribution Model with the FCMP Procedure

A *severity distribution model* consists of a set of functions and subroutines that are defined using the FCMP procedure. The FCMP procedure is part of Base SAS software. Each function or subroutine must be named as *<distribution-name>_<keyword>*, where *distribution-name* is the identifying short name of the distribution and *keyword* identifies one of the functions or subroutines. The total length of the name should not exceed 32. Each function or subroutine must have a specific signature, which consists of the number of arguments, sequence and types of arguments, and return value type. The summary of all the recognized function and subroutine names and their expected behavior is given in Table 30.16.

Consider the following points when you define a distribution model:

- When you define a function or subroutine requiring parameter arguments, the names and order of those arguments must be the same. Arguments other than the parameter arguments can have any name, but they must satisfy the requirements on their type and order.
- When the SEVERITY procedure invokes any function or subroutine, it provides the necessary input values according to the specified signature, and expects the function or subroutine to prepare the output and return it according to the specification of the return values in the signature.

- You can use most of the SAS programming statements and SAS functions that you can use in a DATA step for defining the FCMP functions and subroutines. However, there are a few differences in the capabilities of the DATA step and the FCMP procedure. To learn more, see the documentation of the FCMP procedure in the *Base SAS Procedures Guide*.

- You must specify either the PDF or the LOGPDF function. Similarly, you must specify either the CDF or the LOGCDF function. All other functions are optional, except when necessary for correct definition of the distribution. It is strongly recommended that you define the PARMINIT subroutine to provide a good set of initial values for the parameters. The information provided by PROC SEVERITY to the PARMINIT subroutine enables you to use popular initialization approaches based on the method of moments and the method of percentile matching, but you can implement any algorithm to initialize the parameters by using the values of the response variable and the estimate of its empirical distribution function.

- The LOWERBOUNDS subroutines should be defined if the lower bound on at least one distribution parameter is different from the default lower bound of 0. If you define a LOWERBOUNDS subroutine but do not set a lower bound for some parameter inside the subroutine, then that parameter is assumed to have no lower bound (or a lower bound of $-\infty$). Hence, it is recommended that you explicitly return the lower bound for each parameter when you define the LOWERBOUNDS subroutine.

- The UPPERBOUNDS subroutines should be defined if the upper bound on at least one distribution parameter is different from the default upper bound of ∞. If you define an UPPERBOUNDS subroutine but do not set an upper bound for some parameter inside the subroutine, then that parameter is assumed to have no upper bound (or a upper bound of ∞). Hence, it is recommended that you explicitly return the upper bound for each parameter when you define the UPPERBOUNDS subroutine.

- If you want to use the distribution in a model with regression effects, then make sure that the first parameter of the distribution is the scale parameter itself or a log-transformed scale parameter. If the first parameter is a log-transformed scale parameter, then you must define the SCALETRANSFORM function.

- In general, it is not necessary to define the gradient and Hessian functions, because the SEVERITY procedure uses an internal system to evaluate the required derivatives. The internal system typically computes the derivatives analytically. But it might not be able to do so if your function definitions use other functions that it cannot differentiate analytically. In such cases, derivatives are approximated using a finite difference method and a note is written to the SAS log to indicate the components that are differentiated using such approximations. PROC SEVERITY does reasonably well with these finite difference approximations. But, if you know of a way to compute the derivatives of such components analytically, then you should define the gradient and Hessian functions.

In order to use your distribution with PROC SEVERITY, you need to record the FCMP library that contains the functions and subroutines for your distribution and other FCMP libraries that contain FCMP functions or subroutines used within your distribution's functions and subroutines. Specify all those libraries in the CMPLIB= system option by using the OPTIONS global statement. For more information about the OPTIONS statement, see *SAS Statements: Reference*. For more information about the CMPLIB= system option, see *SAS System Options: Reference*.

Each predefined distribution mentioned in the section "Predefined Distributions" on page 2108 has a distribution model associated with it. The functions and subroutines of all those models are available in the Sashelp.Svrtdist library. The order of the parameters in the signatures of the functions and subroutines is the same as listed in Table 30.2. You do not need to use the CMPLIB= option in order to use the predefined distributions with PROC SEVERITY. However, if you need to use the functions or subroutines of the

predefined distributions in SAS statements other than the PROC SEVERITY step (such as in a DATA step), then specify the Sashelp.Svrtdist library in the CMPLIB= system option by using the OPTIONS global statement prior to using them.

Table 30.16 shows functions and subroutines that define a distribution model, and subsections after the table provide more detail. The functions are listed in alphabetical order of the keyword suffix.

Table 30.16 List of Functions and Subroutines That Define a Distribution Model

Name	Type	Required	Expected to Return
*dist*_CDF	Function	YES[1]	Cumulative distribution function value
*dist*_CDFGRADIENT	Subroutine	NO	Gradient of the CDF
*dist*_CDFHESSIAN	Subroutine	NO	Hessian of the CDF
*dist*_CONSTANTPARM	Subroutine	NO	Constant parameters
*dist*_DESCRIPTION	Function	NO	Description of the distribution
*dist*_LOGCDF	Function	YES[1]	Log of cumulative distribution function value
*dist*_LOGCDFGRADIENT	Subroutine	NO	Gradient of the LOGCDF
*dist*_LOGCDFHESSIAN	Subroutine	NO	Hessian of the LOGCDF
*dist*_LOGPDF	Function	YES[2]	Log of probability density function value
*dist*_LOGPDFGRADIENT	Subroutine	NO	Gradient of the LOGPDF
*dist*_LOGPDFHESSIAN	Subroutine	NO	Hessian of the LOGPDF
*dist*_LOGSDF	Function	NO	Log of survival function value
*dist*_LOGSDFGRADIENT	Subroutine	NO	Gradient of the LOGSDF
*dist*_LOGSDFHESSIAN	Subroutine	NO	Hessian of the LOGSDF
*dist*_LOWERBOUNDS	Subroutine	NO	Lower bounds on parameters
*dist*_PARMINIT	Subroutine	NO	Initial values for parameters
*dist*_PDF	Function	YES[2]	Probability density function value
*dist*_PDFGRADIENT	Subroutine	NO	Gradient of the PDF
*dist*_PDFHESSIAN	Subroutine	NO	Hessian of the PDF
*dist*_QUANTILE	Function	NO	Quantile for a given CDF value
*dist*_SCALETRANSFORM	Function	NO	Type of relationship between the first distribution parameter and the scale parameter
*dist*_SDF	Function	NO	Survival function value
*dist*_SDFGRADIENT	Subroutine	NO	Gradient of the SDF
*dist*_SDFHESSIAN	Subroutine	NO	Hessian of the SDF
*dist*_UPPERBOUNDS	Subroutine	NO	Upper bounds on parameters

Notes:
1. Either the *dist*_CDF or the *dist*_LOGCDF function must be defined.
2. Either the *dist*_PDF or the *dist*_LOGPDF function must be defined.

The signature syntax and semantics of each function or subroutine are as follows:

***dist*_CDF**

defines a function that returns the value of the cumulative distribution function (CDF) of the distribution at the specified values of the random variable and distribution parameters.

- *Type*: Function
- *Required*: YES
- *Number of arguments*: $m + 1$, where m is the number of distribution parameters
- *Sequence and type of arguments*:

  x Numeric value of the random variable at which the CDF value should be evaluated

  p1 Numeric value of the first parameter

  p2 Numeric value of the second parameter

  ...

  p*m* Numeric value of the *m*th parameter

- *Return value*: Numeric value that contains the CDF value $F(x; p_1, p_2, \ldots, p_m)$

If you want to consider this distribution as a candidate distribution when you estimate a response variable model with regression effects, then the first parameter of this distribution must be a scale parameter or log-transformed scale parameter. In other words, if the distribution has a scale parameter, then the following equation must be satisfied:

$$F(x; p_1, p_2, \ldots, p_m) = F(\frac{x}{p_1}; 1, p_2, \ldots, p_m)$$

If the distribution has a log-transformed scale parameter, then the following equation must be satisfied:

$$F(x; p_1, p_2, \ldots, p_m) = F(\frac{x}{\exp(p_1)}; 0, p_2, \ldots, p_m)$$

Here is a sample structure of the function for a distribution named 'FOO':

```
function FOO_CDF(x, P1, P2);
    /* Code to compute CDF by using x, P1, and P2 */

    F = <computed CDF>;
    return (F);
endsub;
```

***dist*_CONSTANTPARM**

defines a subroutine that specifies constant parameters. A parameter is *constant* if it is required for defining a distribution but is not subject to optimization in PROC SEVERITY. Constant parameters are required to be part of the model in order to compute the PDF or the CDF of the distribution. Typically, values of these parameters are known a priori or estimated using some means other than the maximum likelihood method used by PROC SEVERITY. You can estimate them inside the *dist*_PARMINIT subroutine. Once initialized, the parameters remain constant in the context of PROC SEVERITY; that is, they retain their initial value. PROC SEVERITY estimates only the nonconstant parameters.

- *Type*: Subroutine
- *Required*: NO
- *Number of arguments*: *k*, where *k* is the number of constant parameters
- *Sequence and type of arguments*:

  constant parameter 1 Name of the first constant parameter
  . . .
  constant parameter *k* Name of the *k*th constant parameter

- *Return value*: None

Here is a sample structure of the subroutine for a distribution named 'FOO' that has P3 and P5 as its constant parameters, assuming that distribution has at least three parameters:

```
subroutine FOO_CONSTANTPARM(p5, p3);
endsub;
```

Note the following points when you specify the constant parameters:

- At least one distribution parameter must be free to be optimized; that is, if a distribution has total *m* parameters, then *k* must be strictly less than *m*.
- If you want to use this distribution for modeling regression effects, then the first parameter must not be a constant parameter.
- The order of arguments in the signature of this subroutine does not matter as long as each argument's name matches the name of one of the parameters that are defined in the signature of the *dist*_PDF function.
- The constant parameters must be specified in signatures of all the functions and subroutines that accept distribution parameters as their arguments.
- You must provide a nonmissing initial value for each constant parameter by using one of the supported parameter initialization methods.

*dist*_DESCRIPTION

defines a function that returns a description of the distribution.

- *Type*: Function
- *Required*: NO
- *Number of arguments*: None
- *Sequence and type of arguments*: Not applicable
- *Return value*: Character value containing a description of the distribution

Here is a sample structure of the function for a distribution named 'FOO':

```
function FOO_DESCRIPTION() $48;
    length desc $48;
    desc = "A model for a continuous distribution named foo";
    return (desc);
endsub;
```

There is no restriction on the length of the description (the length of 48 used in the previous example is for illustration purposes only). However, if the length is greater than 256, then only the first 256 characters appear in the displayed output and in the _DESCRIPTION_ variable of the OUTMODELINFO= data set. Hence, the recommended length of the description is less than or equal to 256.

dist*_LOG*core

defines a function that returns the natural logarithm of the specified *core* function of the distribution at the specified values of the random variable and distribution parameters. The *core* keyword can be PDF, CDF, or SDF.

- *Type*: Function
- *Required*: YES only if *core* is PDF or CDF and you have not defined that *core* function; otherwise, NO
- *Number of arguments*: $m + 1$, where m is the number of distribution parameters
- *Sequence and type of arguments*:

  x Numeric value of the random variable at which the natural logarithm of the *core* function should be evaluated

  p1 Numeric value of the first parameter

  p2 Numeric value of the second parameter

  ...

  p*m* Numeric value of the *m*th parameter

- *Return value*: Numeric value that contains the natural logarithm of the *core* function

Here is a sample structure of the function for the core function PDF of a distribution named 'FOO':

```
function FOO_LOGPDF(x, P1, P2);
    /* Code to compute LOGPDF by using x, P1, and P2 */

    l = <computed LOGPDF>;
    return (l);
endsub;
```

***dist*_LOWERBOUNDS**

defines a subroutine that returns lower bounds for the parameters of the distribution. If this subroutine is not defined for a given distribution, then the SEVERITY procedure assumes a lower bound of 0 for each parameter. If a lower bound of l_i is returned for a parameter p_i, then the SEVERITY procedure assumes that $l_i < p_i$ (strict inequality). If a missing value is returned for some parameter, then the SEVERITY procedure assumes that there is no lower bound for that parameter (equivalent to a lower bound of $-\infty$).

- *Type*: Subroutine
- *Required*: NO
- *Number of arguments*: m, where m is the number of distribution parameters

- *Sequence and type of arguments*:

  p1 Output argument that returns the lower bound on the first parameter. You must specify this in the OUTARGS statement inside the subroutine's definition.

  p2 Output argument that returns the lower bound on the second parameter. You must specify this in the OUTARGS statement inside the subroutine's definition.

  ...

  p*m* Output argument that returns the lower bound on the *m*th parameter. You must specify this in the OUTARGS statement inside the subroutine's definition.

- *Return value*: The results, lower bounds on parameter values, should be returned in the parameter arguments of the subroutine.

Here is a sample structure of the subroutine for a distribution named 'FOO':

```
subroutine FOO_LOWERBOUNDS(p1, p2);
    outargs p1, p2;

    p1 = <lower bound for P1>;
    p2 = <lower bound for P2>;
endsub;
```

*dist*_PARMINIT

defines a subroutine that returns the initial values for the distribution's parameters given an empirical distribution function (EDF) estimate.

- *Type*: Subroutine
- *Required*: NO
- *Number of arguments*: $m + 4$, where m is the number of distribution parameters
- *Sequence and type of arguments*:

  dim Input numeric value that contains the dimension of the x, nx, and F array arguments.

  x{*} Input numeric array of dimension *dim* that contains values of the random variables at which the EDF estimate is available. It can be assumed that x contains values in an increasing order. In other words, if $i < j$, then x[i] < x[j].

  nx{*} Input numeric array of dimension *dim*. Each nx[i] contains the number of observations in the original data that have the value x[i].

  F{*} Input numeric array of dimension *dim*. Each F[i] contains the EDF estimate for x[i]. This estimate is computed by the SEVERITY procedure based on the options that you specify in the LOSS statement and the EMPIRICALCDF= option.

  Ftype Input numeric value that contains the type of the EDF estimate that is stored in x and F. See the section "Supplying EDF Estimates to Functions and Subroutines" on page 2143 for definition of types.

  p1 Output argument that returns the initial value of the first parameter. You must specify this in the OUTARGS statement inside the subroutine's definition.

p2 Output argument that returns the initial value of the second parameter. You must specify this in the OUTARGS statement inside the subroutine's definition.

...

p*m* Output argument that returns the initial value of the *m*th parameter. You must specify this in the OUTARGS statement inside the subroutine's definition.

- *Return value*: The results, initial values of the parameters, should be returned in the parameter arguments of the subroutine.

Here is a sample structure of the subroutine for a distribution named 'FOO':

```
subroutine FOO_PARMINIT(dim, x{*}, nx{*}, F{*}, Ftype, p1, p2);
    outargs p1, p2;

    /* Code to initialize values of P1 and P2 by using
       dim, x, nx, and F */

    p1 = <initial value for p1>;
    p2 = <initial value for p2>;
endsub;
```

*dist*_**PDF**

defines a function that returns the value of the probability density function (PDF) of the distribution at the specified values of the random variable and distribution parameters.

- *Type*: Function
- *Required*: YES
- *Number of arguments*: $m + 1$, where m is the number of distribution parameters
- *Sequence and type of arguments*:

  x Numeric value of the random variable at which the PDF value should be evaluated

  p1 Numeric value of the first parameter

  p2 Numeric value of the second parameter

  ...

  p*m* Numeric value of the *m*th parameter

- *Return value*: Numeric value that contains the PDF value $f(x; p_1, p_2, \ldots, p_m)$

If you want to consider this distribution as a candidate distribution when you estimate a response variable model with regression effects, then the first parameter of this distribution must be a scale parameter or log-transformed scale parameter. In other words, if the distribution has a scale parameter, then the following equation must be satisfied:

$$f(x; p_1, p_2, \ldots, p_m) = \frac{1}{p_1} f(\frac{x}{p_1}; 1, p_2, \ldots, p_m)$$

If the distribution has a log-transformed scale parameter, then the following equation must be satisfied:

$$f(x; p_1, p_2, \ldots, p_m) = \frac{1}{\exp(p_1)} f(\frac{x}{\exp(p_1)}; 0, p_2, \ldots, p_m)$$

Here is a sample structure of the function for a distribution named 'FOO':

```
function FOO_PDF(x, P1, P2);
    /* Code to compute PDF by using x, P1, and P2 */

    f = <computed PDF>;
    return (f);
endsub;
```

*dist*_**QUANTILE**

defines a function that returns the quantile of the distribution at the specified value of the CDF for the specified values of distribution parameters.

- *Type*: Function
- *Required*: NO
- *Number of arguments*: $m + 1$, where m is the number of distribution parameters
- *Sequence and type of arguments*:

  cdf Numeric value of the cumulative distribution function (CDF) for which the quantile should be evaluated

  p1 Numeric value of the first parameter

  p2 Numeric value of the second parameter

  ...

  p*m* Numeric value of the *m*th parameter

- *Return value*: Numeric value that contains the quantile $F^{-1}(\text{cdf}; p_1, p_2, \ldots, p_m)$

Here is a sample structure of the function for a distribution named 'FOO':

```
function FOO_QUANTILE(c, P1, P2);
    /* Code to compute quantile by using c, P1, and P2 */

    Q = <computed quantile>;
    return (Q);
endsub;
```

*dist*_**SCALETRANSFORM**

defines a function that returns a keyword to identify the transform that needs to be applied to the scale parameter to convert it to the first parameter of the distribution.

If you want to use this distribution for modeling regression effects, then the first parameter of this distribution must be a scale parameter. However, for some distributions, a typical or convenient parameterization might not have a scale parameter, but one of the parameters can be a simple transform of the scale parameter. As an example, consider a typical parameterization of the lognormal distribution with two parameters, location μ and shape σ, for which the PDF is defined as follows:

$$f(x;\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\log(x)-\mu}{\sigma}\right)^2}$$

You can reparameterize this distribution to contain a parameter θ instead of the parameter μ such that $\mu = \log(\theta)$. The parameter θ would then be a scale parameter. However, if you want to specify the distribution in terms of μ and σ (which is a more recognized form of the lognormal distribution) and still allow it as a candidate distribution for estimating regression effects, then instead of writing another distribution with parameters θ and σ, you can simply define the distribution with μ as the first parameter and specify that it is the logarithm of the scale parameter.

- *Type*: Function
- *Required*: NO
- *Number of arguments*: None
- *Sequence and type of arguments*: Not applicable
- *Return value*: Character value that contains one of the following keywords:

LOG	specifies that the first parameter is the logarithm of the scale parameter.
IDENTITY	specifies that the first parameter is a scale parameter without any transformation.

If you do not specify this function, then the IDENTITY transform is assumed.

Here is a sample structure of the function for a distribution named 'FOO':

```
function FOO_SCALETRANSFORM() $8;
    length xform $8;
    xform = "IDENTITY";
    return (xform);
endsub;
```

*dist*_**SDF**

defines a function that returns the value of the survival distribution function (SDF) of the distribution at the specified values of the random variable and distribution parameters.

- *Type*: Function
- *Required*: NO
- *Number of arguments*: $m + 1$, where m is the number of distribution parameters
- *Sequence and type of arguments*:

x	Numeric value of the random variable at which the SDF value should be evaluated
p1	Numeric value of the first parameter
p2	Numeric value of the second parameter
...	
p*m*	Numeric value of the *m*th parameter

- *Return value*: Numeric value that contains the SDF value $S(x; p_1, p_2, \ldots, p_m)$

If you want to consider this distribution as a candidate distribution when estimating a response variable model with regression effects, then the first parameter of this distribution must be a scale parameter

or log-transformed scale parameter. In other words, if the distribution has a scale parameter, then the following equation must be satisfied:

$$S(x; p_1, p_2, \ldots, p_m) = S(\frac{x}{p_1}; 1, p_2, \ldots, p_m)$$

If the distribution has a log-transformed scale parameter, then the following equation must be satisfied:

$$S(x; p_1, p_2, \ldots, p_m) = S(\frac{x}{\exp(p_1)}; 0, p_2, \ldots, p_m)$$

Here is a sample structure of the function for a distribution named 'FOO':

```
function FOO_SDF(x, P1, P2);
    /* Code to compute SDF by using x, P1, and P2 */

    S = <computed SDF>;
    return (S);
endsub;
```

*dist*_UPPERBOUNDS

defines a subroutine that returns upper bounds for the parameters of the distribution. If this subroutine is not defined for a given distribution, then the SEVERITY procedure assumes that there is no upper bound for any of the parameters. If an upper bound of u_i is returned for a parameter p_i, then the SEVERITY procedure assumes that $p_i < u_i$ (strict inequality). If a missing value is returned for some parameter, then the SEVERITY procedure assumes that there is no upper bound for that parameter (equivalent to an upper bound of ∞).

- *Type*: Subroutine
- *Required*: NO
- *Number of arguments*: *m*, where *m* is the number of distribution parameters
- *Sequence and type of arguments*:

  p1 — Output argument that returns the upper bound on the first parameter. You must specify this in the OUTARGS statement inside the subroutine's definition.

  p2 — Output argument that returns the upper bound on the second parameter. You must specify this in the OUTARGS statement inside the subroutine's definition.

  ...

  p*m* — Output argument that returns the upper bound on the *m*th parameter. You must specify this in the OUTARGS statement inside the subroutine's definition.

- *Return value*: The results, upper bounds on parameter values, should be returned in the parameter arguments of the subroutine.

Here is a sample structure of the subroutine for a distribution named 'FOO':

```
subroutine FOO_UPPERBOUNDS(p1, p2);
    outargs p1, p2;

    p1 = <upper bound for P1>;
    p2 = <upper bound for P2>;
endsub;
```

*dist_core*__GRADIENT__

defines a subroutine that returns the gradient vector of the specified *core* function of the distribution at the specified values of the random variable and distribution parameters. The *core* keyword can be PDF, CDF, SDF, LOGPDF, LOGCDF, or LOGSDF.

- *Type*: Subroutine
- *Required*: NO
- *Number of arguments*: $m + 2$, where m is the number of distribution parameters
- *Sequence and type of arguments*:

x	Numeric value of the random variable at which the gradient should be evaluated
p1	Numeric value of the first parameter
p2	Numeric value of the second parameter
	...
p*m*	Numeric value of the *m*th parameter
grad{*}	Output numeric array of size m that contains the gradient vector evaluated at the specified values. If h denotes the value of the *core* function, then the expected order of the values in the array is as follows: $\frac{\partial h}{\partial p_1}\ \frac{\partial h}{\partial p_2} \cdots \frac{\partial h}{\partial p_m}$

- *Return value*: Numeric array that contains the gradient evaluated at x for the parameter values $(p_1, p_2, \ldots, p_m)$

Here is a sample structure of the function for the core function CDF of a distribution named 'FOO':

```
subroutine FOO_CDFGRADIENT(x, P1, P2, grad{*});
    outargs grad;

    /* Code to compute gradient by using x, P1, and P2 */
    grad[1] = <partial derivative of CDF w.r.t. P1
               evaluated at x, P1, P2>;
    grad[2] = <partial derivative of CDF w.r.t. P2
               evaluated at x, P1, P2>;
endsub;
```

*dist_core*__HESSIAN__

defines a subroutine that returns the Hessian matrix of the specified *core* function of the distribution at the specified values of the random variable and distribution parameters. The *core* keyword can be PDF, CDF, SDF, LOGPDF, LOGCDF, or LOGSDF.

- *Type*: Subroutine
- *Required*: NO
- *Number of arguments*: $m + 2$, where m is the number of distribution parameters
- *Sequence and type of arguments*:

x	Numeric value of the random variable at which the Hessian matrix should be evaluated
p1	Numeric value of the first parameter

p2 Numeric value of the second parameter

...

p*m* Numeric value of the *m*th parameter

hess{*} Output numeric array of size $m(m+1)/2$ that contains the lower triangular portion of the Hessian matrix in a packed vector form, evaluated at the specified values. If h denotes the value of the *core* function, then the expected order of the values in the array is as follows: $\frac{\partial^2 h}{\partial p_1^2} \mid \frac{\partial^2 h}{\partial p_1 \partial p_2} \; \frac{\partial^2 h}{\partial p_2^2} \mid \cdots \mid \frac{\partial^2 h}{\partial p_1 \partial p_m} \; \frac{\partial^2 h}{\partial p_2 \partial p_m} \cdots \frac{\partial^2 h}{\partial p_m^2}$

- *Return value*: Numeric array that contains the lower triangular portion of the Hessian matrix evaluated at x for the parameter values $(p_1, p_2, \ldots, p_m)$

Here is a sample structure of the subroutine for the core function LOGSDF of a distribution named 'FOO':

```
subroutine FOO_LOGSDFHESSIAN(x, P1, P2, hess{*});
    outargs hess;

    /* Code to compute Hessian by using x, P1, and P2 */
    hess[1] = <second order partial derivative of LOGSDF
               w.r.t. P1 evaluated at x, P1, P2>;
    hess[2] = <second order partial derivative of LOGSDF
               w.r.t. P1 and P2 evaluated at x, P1, P2>;
    hess[3] = <second order partial derivative of LOGSDF
               w.r.t. P2 evaluated at x, P1, P2>;
endsub;
```

Predefined Utility Functions

The following predefined utility functions are provided with the SEVERITY procedure and are available in the Sashelp.Svrtdist library:

SVRTUTIL_EDF:

This function computes the empirical distribution function (EDF) estimate at the specified value of the random variable given the EDF estimate for a sample.

- *Type*: Function
- *Signature*: SVRTUTIL_EDF(y, n, x{*}, F{*}, Ftype)
- *Argument Description*:

  y Value of the random variable at which the EDF estimate is desired.

  n Dimension of the *x* and *F* input arrays.

  x{*} Input numeric array of dimension *n* that contains values of the random variable observed in the sample. These values are sorted in nondecreasing order.

  F{*} Input numeric array of dimension *n* in which each F[*i*] contains the EDF estimate for x[*i*]. These values must be sorted in nondecreasing order.

Ftype — Type of the empirical estimate that is stored in the *x* and *F* arrays. See the section "Supplying EDF Estimates to Functions and Subroutines" on page 2143 for definition of types.

- *Return value*: The EDF estimate at *y*.

The type of the sample EDF estimate determines how the EDF estimate at *y* is computed. For more information, see the section "Supplying EDF Estimates to Functions and Subroutines" on page 2143.

SVRTUTIL_EMPLIMMOMENT:

This function computes the empirical estimate of the limited moment of specified order for the specified upper limit, given the EDF estimate for a sample.

- *Type*: Function
- *Signature*: SVRTUTIL_EMPLIMMOMENT(k, u, n, x{*}, F{*}, Ftype)
- *Argument Description*:

  k — Order of the desired empirical limited moment.

  u — Upper limit on the value of the random variable to be used in the computation of the desired empirical limited moment.

  n — Dimension of the *x* and *F* input arrays.

  x{*} — Input numeric array of dimension *n* that contains values of the random variable observed in the sample. These values are sorted in nondecreasing order.

  F{*} — Input numeric array of dimension *n* in which each F[*i*] contains the EDF estimate for x[*i*]. These values must be sorted in nondecreasing order.

  Ftype — Type of the empirical estimate that is stored in the *x* and *F* arrays. See the section "Supplying EDF Estimates to Functions and Subroutines" on page 2143 for definition of types.

- *Return value*: The desired empirical limited moment.

The empirical limited moment is computed by using the empirical estimate of the CDF. If $F_n(x)$ denotes the EDF at x, then the empirical limited moment of order k with upper limit u is defined as

$$E_n[(X \wedge u)^k] = k \int_0^u (1 - F_n(x)) x^{k-1} dx$$

The SVRTUTIL_EMPLIMMOMENT function uses the piecewise linear nature of $F_n(x)$ as described in the section "Supplying EDF Estimates to Functions and Subroutines" on page 2143 to compute the integration.

SVRTUTIL_HILLCUTOFF:

This function computes an estimate of the value where the right tail of a distribution is expected to begin. The function implements the algorithm described in Danielsson et al. 2001. The description of the algorithm uses the following notation:

n — Number of observations in the original sample.

B — Number of bootstrap samples to draw.

m_1 — Size of the bootstrap sample in the first step of the algorithm ($m_1 < n$).

$x_{(i)}^{j,m}$ ith order statistic of jth bootstrap sample of size m ($1 \le i \le m, 1 \le j \le B$).

$x_{(i)}$ ith order statistic of the original sample ($1 \le i \le n$).

Given the input sample x and values of B and m_1, the steps of the algorithm are as follows:

1. Take B bootstrap samples of size m_1 from the original sample.
2. Find the integer k_1 that minimizes the bootstrap estimate of the mean squared error:

$$k_1 = \arg \min_{1 \le k < m_1} Q(m_1, k)$$

3. Take B bootstrap samples of size $m_2 = m_1^2/n$ from the original sample.
4. Find the integer k_2 that minimizes the bootstrap estimate of the mean squared error:

$$k_2 = \arg \min_{1 \le k < m_2} Q(m_2, k)$$

5. Compute the integer k_{opt}, which is used for computing the cutoff point:

$$k_{\text{opt}} = \frac{k_1^2}{k_2} \left(\frac{\log(k_1)}{2\log(m_1) - \log(k_1)} \right)^{2 - 2\log(k_1)/\log(m_1)}$$

6. Set the cutoff point equal to $x_{(k_{\text{opt}}+1)}$.

The bootstrap estimate of the mean squared error is computed as

$$Q(m,k) = \frac{1}{B} \sum_{j=1}^{B} \text{MSE}_j(m,k)$$

The mean squared error of jth bootstrap sample is computed as

$$\text{MSE}_j(m,k) = (M_j(m,k) - 2(\gamma_j(m,k))^2)^2$$

where $M_j(m,k)$ is a control variate proposed by Danielsson et al. 2001,

$$M_j(m,k) = \frac{1}{k} \sum_{i=1}^{k} \left(\log(x_{(m-i+1)}^{j,m}) - \log(x_{(m-k)}^{j,m}) \right)^2$$

and $\gamma_j(m,k)$ is the Hill's estimator of the tail index (Hill 1975),

$$\gamma_j(m,k) = \frac{1}{k} \sum_{i=1}^{k} \log(x_{(m-i+1)}^{j,m}) - \log(x_{(m-k)}^{j,m})$$

This algorithm has two tuning parameters, B and m_1. The number of bootstrap samples B is chosen based on the availability of computational resources. The optimal value of m_1 is chosen such that the following ratio, $R(m_1)$, is minimized:

$$R(m_1) = \frac{(Q(m_1, k_1))^2}{Q(m_2, k_2)}$$

The SVRTUTIL_HILLCUTOFF utility function implements the preceding algorithm. It uses the grid search method to compute the optimal value of m_1.

- *Type*: Function
- *Signature*: SVRTUTIL_HILLCUTOFF(n, x{*}, b, s, status)

- *Argument Description*:

  n — Dimension of the array *x*.

  x{*} — Input numeric array of dimension *n* that contains the sample.

  b — Number of bootstrap samples used to estimate the mean squared error. If *b* is less than 10, then a default value of 50 is used.

  s — Approximate number of steps used to search the optimal value of m_1 in the range $[n^{0.75}, n-1]$. If *s* is less than or equal to 1, then a default value of 10 is used.

  status — Output argument that contains the status of the algorithm. If the algorithm succeeds in computing a valid cutoff point, then *status* is set to 0. If the algorithm fails, then *status* is set to 1.

- *Return value*: The cutoff value where the right tail is estimated to start. If the size of the input sample is inadequate ($n \leq 5$), then a missing value is returned and *status* is set to a missing value. If the algorithm fails to estimate a valid cutoff value (*status* = 1), then the fifth largest value in the input sample is returned.

SVRTUTIL_PERCENTILE:

This function computes the specified empirical percentile given the EDF estimates.

- *Type*: Function
- *Signature*: SVRTUTIL_PERCENTILE(p, n, x{*}, F{*}, Ftype)
- *Argument Description*:

  p — Desired percentile. The value must be in the interval (0,1). The function returns the 100*p*th percentile.

  n — Dimension of the *x* and *F* input arrays.

  x{*} — Input numeric array of dimension *n* that contains values of the random variable observed in the sample. These values are sorted in nondecreasing order.

  F{*} — Input numeric array of dimension *n* in which each F[*i*] contains the EDF estimate for x[*i*]. These values must be sorted in nondecreasing order.

  Ftype — Type of the empirical estimate that is stored in the *x* and *F* arrays. See the section "Supplying EDF Estimates to Functions and Subroutines" on page 2143 for definition of types.

- *Return value*: The 100*p*th percentile of the input sample.

The method used to compute the percentile depends on the type of the EDF estimate (Ftype argument).

Ftype = 1 — Smoothed empirical estimates are computed using the method described in Klugman, Panjer, and Willmot (1998). Let $\lfloor x \rfloor$ denote the greatest integer less than or equal to x. Define $g = \lfloor p(n+1) \rfloor$ and $h = p(n+1) - g$. Then the empirical percentile $\hat{\pi}_p$ is defined as

$$\hat{\pi}_p = (1-h)x[g] + hx[g+1]$$

This method does not work if $p < 1/(n+1)$ or $p > n/(n+1)$. If $p < 1/(n+1)$, then the function returns $\hat{\pi}_p = x[1]/2$, which assumes that the EDF is 0 in the interval $[0, x[1])$. If $p > n/(n+1)$, then $\hat{\pi}_p = x[n]$.

Ftype = 2

If $p < F[1]$, then $\hat{\pi}_p = x[1]/2$, which assumes that the EDF is 0 in the interval $[0, x[1])$. If $|p - F[i]| < \epsilon$ for some value of i and $i < n$, then $\hat{\pi}_p$ is computed as

$$\hat{\pi}_p = \frac{x[i] + x[i+1]}{2}$$

where ϵ is a machine-precision constant as returned by the SAS function CONSTANT('MACEPS'). If $F[i-1] < p < F[i]$, then $\hat{\pi}_p$ is computed as

$$\hat{\pi}_p = x[i]$$

If $p \geq F[n]$, then $\hat{\pi}_p = x[n]$.

Ftype = 3

If $p < F[1]$, then $\hat{\pi}_p = x[1]/2$, which assumes that the EDF is 0 in the interval $[0, x[1])$. If $|p - F[i]| < \epsilon$ for some value of i and $i < n$, then $\hat{\pi}_p$ is computed as

$$\hat{\pi}_p = \frac{x[i] + x[i+1]}{2}$$

where ϵ is a machine-precision constant as returned by the SAS function CONSTANT('MACEPS'). If $F[i-1] < p < F[i]$, then $\hat{\pi}_p$ is computed as

$$\hat{\pi}_p = x[i-1] + (p - F[i-1])\frac{x[i] - x[i-1]}{F[i] - F[i-1]}$$

If $p \geq F[n]$, then $\hat{\pi}_p = x[n]$.

SVRTUTIL_RAWMOMENTS:

This subroutine computes the raw moments of a sample.

- *Type*: Subroutine
- *Signature*: SVRTUTIL_RAWMOMENTS(n, x{*}, nx{*}, nRaw, raw{*})
- *Argument Description*:

  n — Dimension of the *x* and *nx* input arrays.

  x{*} — Input numeric array of dimension *n* that contains distinct values of the random variable that are observed in the sample.

  nx{*} — Input numeric array of dimension *n* in which each *nx*[*i*] contains the number of observations in the sample that have the value x[*i*].

  nRaw — Desired number of raw moments. The output array *raw* contains the first *nRaw* raw moments.

  raw{*} — Output array of raw moments. The *k*th element in the array (raw{*k*}) contains the *k*th raw moment, where $1 \leq k \leq$ nRaw.

- *Return value*: Numeric array *raw* that contains the first *nRaw* raw moments. The array contains missing values if the sample has no observations (that is, if all the values in the *nx* array add up to zero).

SVRTUTIL_SORT:

This function sorts the given array of numeric values in an ascending or descending order.

- *Type*: Subroutine
- *Signature*: SVRTUTIL_SORT(n, x{*}, flag)

- *Argument Description*:

  n — Dimension of the input array *x*.

  x{*} — Numeric array that contains the values to be sorted at input. The subroutine uses the same array to return the sorted values.

  flag — A numeric value that controls the sort order. If *flag* is 0, then the values are sorted in an ascending order. If *flag* has any value other than 0, then the values are sorted in descending order.

- *Return value*: Numeric array *x*, which is sorted in place (that is, the sorted array is stored in the same storage area occupied by the input array *x*).

You can use the following predefined functions when you use the FCMP procedure to define functions and subroutines. They are summarized here for your information. For more information, see the FCMP procedure documentation in *Base SAS Procedures Guide*.

INVCDF:
This function computes the quantile from any continuous probability distribution by numerically inverting the CDF of that distribution. You need to specify the CDF function of the distribution, the values of its parameters, and the cumulative probability to compute the quantile.

LIMMOMENT:
This function computes the limited moment of order k with upper limit u for any continuous probability distribution. The limited moment is defined as

$$\begin{aligned} E[(X \wedge u)^k] &= \int_0^u x^k f(x)dx + \int_u^\infty u^k f(x)dx \\ &= \int_0^u x^k f(x)dx + u^k(1 - F(u)) \end{aligned}$$

where $f(x)$ and $F(x)$ denote the PDF and the CDF of the distribution, respectively. The LIMMOMENT function uses the following alternate definition, which can be derived using integration-by-parts:

$$E[(X \wedge u)^k] = k \int_0^u (1 - F(x))x^{k-1}dx$$

You need to specify the CDF function of the distribution, the values of its parameters, and the values of k and u to compute the limited moment.

Scoring Functions

Scoring refers to the act of evaluating a distribution function, such as LOGPDF, SDF, or QUANTILE, on an observation by using the fitted parameter estimates of that distribution. You can do scoring in a DATA step by using the OUTEST= data set that you create with PROC SEVERITY. However, that approach requires some cumbersome programming. In order to simplify the scoring process, you can specify that PROC SEVERITY create scoring functions for each fitted distribution.

As an example, assume that you have fitted the Pareto distribution by using PROC SEVERITY and that it converges. Further assume that you want to use the fitted distribution to evaluate the probability of observing a loss value greater than some set of regulatory limits {L} that are encoded in a data set. You can simplify this scoring process as follows. First, in the PROC SEVERITY step that fits your distributions, you create the scoring functions library by specifying the OUTSCORELIB statement as illustrated in the following steps:

```
proc severity data=input;
   loss lossclaim;
   dist pareto;
   outscorelib outlib=sasuser.fitdist;
run;
```

Upon successful completion, if the Pareto distribution model has converged, then the Sasuser.Fitdist library contains the *SEV_SDF* scoring function in addition to other scoring functions, such as *SEV_PDF*, *SEV_LOGPDF*, and so on. Further, PROC SEVERITY also sets the CMPLIB system option to include the Sasuser.Fitdist library. If the set of limits {L} is recorded in the variable Limit in the scoring data set Work.Limits, then you can submit the following DATA step to compute the probability of seeing a loss greater than each limit:

```
data prob;
   set work.limits;
   exceedance_probability = sev_sdf(limit);
run;
```

Without the use of scoring functions, you can still perform this scoring task, but the DATA step that you need to write to accomplish it becomes more complicated and less flexible. For example, you would need to read the parameter estimates from some output created by PROC SEVERITY. To do that, you would need to know the parameter names, which are different for different distributions; this in turn would require you to write a specific DATA step for each distribution or to write a SAS macro. With the use of scoring functions, you can accomplish that task much more easily.

If you fit multiple distributions, then you can specify the COMMONPACKAGE option in the OUTSCORELIB statement as follows:

```
proc severity data=input;
   loss lossclaim;
   dist exp pareto weibull;
   outscorelib outlib=sasuser.fitdist commonpackage;
run;
```

The preceding step creates scoring functions such as *SEV_SDF_Exp*, *SEV_SDF_Pareto*, and *SEV_SDF_Weibull*. You can use them to compare the probabilities of exceeding the limit for different distributions by using the following DATA step:

```
data prob;
   set work.limits;
   exceedance_exp = sev_sdf_exp(limit);
   exceedance_pareto = sev_sdf_pareto(limit);
   exceedance_weibull = sev_sdf_weibull(limit);
run;
```

Formal Description

PROC SEVERITY creates a scoring function for each distribution function. A distribution function is defined as any function named *dist_suffix*, where *dist* is the name of a distribution that you specify in the DIST statement and the function's signature is identical to the signature of the required distribution function such as *dist*_CDF or *dist*_LOGCDF. For example, for the function 'FOO_BAR' to be a distribution function, you must specify the distribution 'FOO' in the DIST statement and you must define 'FOO_BAR' in the following manner if the distribution 'FOO' has parameters named 'P1' and 'P2':

```
function FOO_BAR(y, P1, P2);
   /* Code to compute BAR by using y, P1, and P2 */
   R = <computed BAR>;
   return (R);
endsub;
```

For more information about the signature that defines a distribution function, see the description of the *dist*_CDF function in the section "Defining a Severity Distribution Model with the FCMP Procedure" on page 2148.

The name and package of the scoring function of a distribution function depend on whether you specify the COMMONPACKAGE option in the OUTSCORELIB statement.

When you do not specify the COMMONPACKAGE option, the scoring function that corresponds to the distribution function *dist_suffix* is named *SEV_suffix*, where *SEV_* is the standard prefix of all scoring functions. The scoring function is created in a package named *dist*. Each scoring function accepts only one argument, the value of the loss variable, and returns the same value as the value returned by the corresponding distribution function for the final estimates of the distribution's parameters. For example, for the preceding 'FOO_BAR' distribution function, the scoring function named 'SEV_BAR' is created in the package named 'FOO' and 'SEV_BAR' has the following signature:

```
function SEV_BAR(y);
   /* returns value of FOO_BAR for the supplied value
      of y and fitted values of P1, P2 */
endsub;
```

If you specify the COMMONPACKAGE option in the OUTSCORELIB statement, then the scoring function that corresponds to the distribution function *dist_suffix* is named *SEV_suffix_dist*, where *SEV_* is the standard prefix of all scoring functions. The scoring function is created in a package named *sevfit*. For example, for the preceding 'FOO_BAR' distribution function, if you specify the COMMONPACKAGE option, the scoring function named 'SEV_BAR_FOO' is created in the *sevfit* package and 'SEV_BAR_FOO' has the following signature:

```
function SEV_BAR_FOO(y);
   /* returns value of FOO_BAR for the supplied value
      of y and fitted values of P1, P2 */
endsub;
```

Scoring Functions for the Scale Regression Model

If you use the SCALEMODEL statement to specify a scale regression model, then PROC SEVERITY generates the scoring functions when you specify only singleton continuous effects. If you specify interaction or classification effects, then scoring functions are not generated.

For a scale regression model, the estimate of the scale parameter or the log-transformed scale parameter of the distribution depends on the values of the regressors. So PROC SEVERITY creates a scoring function that has the following signature, where *x{*}* represents the array of regressors:

```
function SEV_BAR(y, x{*});
   /* returns value of FOO_BAR for the supplied value of x and fitted values of P1, P2 */
endsub;
```

As an illustration of using this form, assume that you submit the following PROC SEVERITY step to create the scoring library Sasuser.Scalescore:

```
proc severity data=input;
   loss lossclaim;
   scalemodel x1-x3;
   dist pareto;
   outscorelib outlib=sasuser.scalescore;
run;
```

Your scoring data set must contain all the regressors that you specify in the SCALEMODEL statement. You can submit the following DATA step to score observations by using the scale regression model:

```
data prob;
   array regvals{*} x1-x3;
   set work.limits;
   exceedance_probability = sev_sdf(limit, regvals);
run;
```

PROC SEVERITY creates two utility functions, SEV_NUMREG and SEV_REGNAME, in the OUTLIB= library that return the number of regressors and name of a given regressor, respectively. They are described in detail in the next section. These utility functions are useful when you do not have easy access to the regressor names in the SCALEMODEL statement. You can use the utility functions as follows:

```
data prob;
   array regvals{10} _temporary_;
   set work.limits;
   do i = 1 to sev_numreg();
      regvals(i) = input(vvaluex(sev_regname(i)), best12.);
   end;
   exceedance_probability = sev_sdf(limit, regvals);
run;
```

The dimension of the regressor values array that you supply to the scoring function must be equal to $K + L$, where K is the number of regressors that you specify in the SCALEMODEL statement irrespective of whether PROC SEVERITY deems any of those regressors to be redundant. L is 1 if you specify an OFFSET= variable in the SCALEMODEL statement, and 0 otherwise.

Utility Functions and Subroutines in the OUTLIB= Library

In addition to creating the scoring functions for all distribution functions, PROC SEVERITY creates the following utility functions and subroutines in the OUTLIB= library.

SEV_NUMPARM | SEV_NUMPARM_*dist*
is a function that returns the number of distribution parameters and has the following signature:

- *Type*: Function
- *Number of arguments*: 0
- *Sequence and type of arguments*: Not applicable
- *Return value*: Numeric value that contains the number of distribution parameters

If you do not specify the COMMONPACKAGE option in the OUTSCORELIB statement, then a function named SEV_NUMPARM is created in the package of each distribution. Here is a sample structure of the code that PROC SEVERITY uses to define the function:

```
function SEV_NUMPARM();
    n = <number of distribution parameters>;
    return (n);
endsub;
```

If you specify the COMMONPACKAGE option in the OUTSCORELIB statement, then for each distribution *dist*, the function named SEV_NUMPARM_*dist* is created in the *sevfit* package. SEV_NUMPARM_*dist* has the same structure as the SEV_NUMPARM function that is described previously.

SEV_PARMEST | SEV_PARMEST_*dist*
is a subroutine that returns the estimate and standard error of a specified distribution parameter and has the following signature:

- *Type*: Subroutine
- *Number of arguments*: 3
- *Sequence and type of arguments*:

  index specifies the numeric value of the index of the distribution parameter for which you want the information. The value of *index* must be in the interval [1,*m*], where *m* is the number of parameters in the distribution to which this subroutine belongs.

  est specifies the output argument that returns the estimate of the requested parameter.

  stderr specifies the output argument that returns the standard error of the requested parameter.

- *Return value*: Estimate and standard error of the requested distribution parameter that are returned in the output arguments *est* and *stderr*, respectively

If you do not specify the COMMONPACKAGE option in the OUTSCORELIB statement, then a subroutine named SEV_PARMEST is created in the package of each distribution. Here is a sample structure of the code that PROC SEVERITY uses to define the subroutine:

```
subroutine SEV_PARMEST(index, est, stderr);
    outargs est, stderr;
    est = <value of the estimate for the distribution parameter
           at position 'index'>;
    stderr =  <value of the standard error for distribution parameter
               at position 'index'>;
endsub;
```

If you specify the COMMONPACKAGE option in the OUTSCORELIB statement, then for each distribution *dist*, the subroutine named SEV_PARMEST_*dist* is created in the *sevfit* package. SEV_PARMEST_*dist* has the same structure as the SEV_PARMEST subroutine that is described previously.

If you use the SCALEMODEL statement to specify a scale regression model, and if you specify only singleton continuous effects, then for *index*=1, the returned estimates are of θ_0, the base value of the scale parameter, or $\log(\theta_0)$ if the distribution has a log-scale parameter. For more information about θ_0, see the section "Estimating Regression Effects" on page 2123.

SEV_PARMNAME | SEV_PARMNAME_*dist*

is a function that returns the name of a specified distribution parameter and has the following signature:

- *Type*: Function
- *Number of arguments*: 1
- *Sequence and type of arguments*:

  index specifies the numeric value of the index of the distribution parameter for which you want the information. The value of *index* must be in the interval [1,*m*], where *m* is the number of parameters in the distribution to which this function belongs.

- *Return value*: Character value that contains the name of the distribution parameter that appears at the position *index* in the distribution's definition

If you do not specify the COMMONPACKAGE option in the OUTSCORELIB statement, then a function named SEV_PARMNAME is created in the package of each distribution. Here is a sample structure of the code that PROC SEVERITY uses to define the function:

```
function SEV_PARMNAME(index) $32;
    name = <name of the distribution parameter at position 'index'>;
    return (name);
endsub;
```

If you specify the COMMONPACKAGE option in the OUTSCORELIB statement, then for each distribution *dist*, a function named SEV_PARMNAME_*dist* is created in the *sevfit* package. SEV_PARMNAME_*dist* has the same structure as the SEV_PARMNAME function that is described previously.

If you use the SCALEMODEL statement to specify a scale regression model, and if you specify only singleton continuous effects, then the following helper functions and subroutines are also created in the OUTLIB= library.

SEV_NUMREG
is a function that returns the number of regressors and has the following signature:

- *Type*: Function
- *Number of arguments*: 0
- *Sequence and type of arguments*: Not applicable
- *Return value*: Numeric value that contains the number of regressors that you specify in the SCALEMODEL statement. If you specify an OFFSET= variable in the SCALEMODEL statement, then the returned value is equal to 1 plus the number of regressors that you specify in the SCALEMODEL statement.

Here is a sample structure of the code that PROC SEVERITY uses to define the function:

```
function SEV_NUMREG();
    m = <number of regressors>;
    if (<offset variable is specified>) then m = m + 1;
    return (m);
endsub;
```

This function does not depend on any distribution, so it is always created in the *sevfit* package.

SEV_REGEST | SEV_REGEST_*dist*
is a subroutine that returns the estimate and standard error of a specified regression parameter and has the following signature:

- *Type*: Subroutine
- *Number of arguments*: 3
- *Sequence and type of arguments*:

  index specifies the numeric value of the index of the regression parameter for which you want the information. The value of *index* must be in the interval [1,*K*], where *K* is the number of regressors as returned by the SEV_NUMREG function. If you specify an OFFSET= variable in the SCALEMODEL statement, then an *index* value of *K* corresponds to the offset variable.

  est specifies the output argument that returns the estimate of the requested regression parameter.

  stderr specifies the output argument that returns the standard error of the requested regression parameter.

- *Return value*: Estimate and standard error of the requested regression parameter that are returned in the output arguments *est* and *stderr*, respectively

If you do not specify the COMMONPACKAGE option in the OUTSCORELIB statement, then a subroutine named SEV_REGEST is created in the package of each distribution. Here is a sample structure of the code that PROC SEVERITY uses to define the subroutine:

```
subroutine SEV_REGEST(index, est, stderr);
    outargs est, stderr;
    est = <value of the estimate for the regression parameter
           at position 'index'>;
    stderr =  <value of the standard error for regression parameter
               at position 'index'>;
endsub;
```

If you specify the COMMONPACKAGE option in the OUTSCORELIB statement, then for each distribution *dist*, the subroutine named SEV_REGEST_*dist* is created in the *sevfit* package. SEV_REGEST_*dist* has the same structure as the SEV_REGEST subroutine that is described previously.

If the regressor that corresponds to the specified *index* value is a redundant regressor, the returned values of both *est* and *stderr* are equal to the special missing value of .R. If you specify an OFFSET= variable in the SCALEMODEL statement and if the *index* value corresponds to the offset variable—that is, it is equal to the value that the SEV_NUMREG function returns — then the returned value of *est* is equal to 1 and the returned value of *stderr* is equal to the special missing value of .F.

SEV_REGNAME

is a function that returns the name of a specified regressor and has the following signature:

- *Type*: Function
- *Number of arguments*: 1
- *Sequence and type of arguments*:

  index specifies the numeric value of the index of the regressor for which you want the name. The value of *index* must be in the interval [1,*K*], where *K* is the number of regressors as returned by the SEV_NUMREG function. If you specify an OFFSET= variable in the SCALEMODEL statement, then an *index* value of *K* corresponds to the offset variable.

- *Return value*: Character value that contains the name of the regressor that appears at the position *index* in the SCALEMODEL statement. If you specify an OFFSET= variable in the SCALEMODEL statement, then for an *index* value of *K*, the returned value contains the name of the offset variable.

Here is a sample structure of the code that PROC SEVERITY uses to define the function:

```
function SEV_REGNAME(index) $32;
    name = <name of regressor at position 'index'>;
    return (name);
endsub;
```

This function does not depend on any distribution, so it is always created in the *sevfit* package.

Custom Objective Functions

You can use a series of programming statements that use variables in the DATA= data set to assign a value to an objective function symbol. You must specify the objective function symbol by using the OBJECTIVE= option in the PROC SEVERITY statement.

The objective function can be programmed such that it is applicable to any distribution that is used in the model. For that purpose, PROC SEVERITY recognizes the following *keyword* functions in the programming statements:

PDF(x) returns the probability density function (PDF) of a distribution evaluated at the current value of a data set variable x.

CDF(x) returns the cumulative distribution function (CDF) of a distribution evaluated at the current value of a data set variable x.

SDF(x) returns the survival distribution function (SDF) of a distribution evaluated at the current value of a data set variable x.

LOGPDF(x) returns the natural logarithm of the PDF of a distribution evaluated at the current value of a data set variable x.

LOGCDF(x) returns the natural logarithm of the CDF of a distribution evaluated at the current value of a data set variable x.

LOGSDF(x) returns the natural logarithm of the SDF of a distribution evaluated at the current value of a data set variable x.

EDF(x) returns the empirical distribution function (EDF) estimate evaluated at the current value of a data set variable x. Internally, PROC SEVERITY computes the estimate using the SVRTUTIL_EDF function as described in the section "Predefined Utility Functions" on page 2160. The EDF estimate that is required by the SVRTUTIL_EDF function is computed by using the response variable values in the current BY group or in the entire input data set if you do not specify the BY statement.

EMPLIMMOMENT(k, u)
returns the empirical limited moment of order *k* evaluated at the current value of a data set variable u that represents the upper limit of the limited moment. The order *k* can also be a data set variable. Internally, PROC SEVERITY computes the moment using the SVRTUTIL_EMPLIMMOMENT function as described in the section "Predefined Utility Functions" on page 2160. The EDF estimate that is required by the SVRTUTIL_EMPLIMMOMENT function is computed by using the response variable values in the current BY group or in the entire input data set if you do not specify the BY statement.

LIMMOMENT(k, u)
returns the limited moment of order *k* evaluated at the current value of a data set variable u that represents the upper limit of the limited moment. The order *k* can be a data set variable or a constant. Internally, for each candidate distribution, PROC SEVERITY computes the moment using the LIMMOMENT function as described in the section "Predefined Utility Functions" on page 2160.

All the preceding functions are right-hand side functions. They act as placeholders for distribution-specific functions, with the exception of _EDF_ and _EMPLIMMOMENT_ functions.

As an example, let the data set Work.Test contain a response variable Y and a left-truncation threshold variable T. The following statements use the values in this data set to fit a model with distribution D such that the parameters of the model minimize the value of the objective function symbol MYOBJ:

```
options cmplib=(work.mydist);
proc severity data=work.test objective=myobj;
    loss y / lt=t;

    myobj = -_LOGPDF_(y);
    if (not(missing(t))) then
        myobj = myobj + log(1-_CDF_(t));

    dist d;
run;
```

The symbol MYOBJ is designated as an objective function symbol by using the OBJECTIVE= option in the PROC SEVERITY statement. The response variable Y and left-truncation variable T are specified in the LOSS statement. The distribution D is specified in the DIST statement. The remaining statements constitute a program that computes the value of the MYOBJ symbol.

Let the distribution D have parameters P1 and P2. In order to estimate the model for this distribution, PROC SEVERITY internally converts the generic program to the following program specific to distribution D:

```
myobj = -D_LOGPDF(y, p1, p2);
if (not(missing(t))) then
    myobj = myobj + log(1-D_CDF(t, p1, p2));
```

Note that the generic keyword functions _LOGPDF_ and _CDF_ have been replaced with distribution-specific functions D_LOGPDF and D_CDF, respectively, with appropriate distribution parameters. The D_LOGPDF and D_CDF functions must have been defined previously and are assumed to be available in the Work.Mydist library that you specify in the CMPLIB= option.

The program is executed for each observation in Work.Test to compute the value of MYOBJ by using the values of variables Y and T in that observation and internally computed values of the model parameters P1 and P2. The values of MYOBJ are then added over all the observations of the data set or over all the observations of the current BY group if you specify the BY statement. The resulting aggregate value is the value of the objective function, and it is supplied to the optimizer. If the optimizer requires derivatives of the objective function, then PROC SEVERITY automatically differentiates MYOBJ with respect to the parameters P1 and P2. The optimizer iterates over various combinations of the values of parameters P1 and P2, each time computing a new value of the objective function and the needed derivatives of it, until it finds a combination that minimizes the objective function.

Note the following points when you define your own program to compute the custom objective function:

- The value of the objective function is always minimized by PROC SEVERITY. If you want to maximize the value of a certain objective, then add a statement that assigns the negated value of the maximization objective to the objective function symbol that you specify in the OBJECTIVE= option. Minimization of the negated objective is equivalent to the maximization of the original objective.
- The contributions of individual observations are always added to compute the overall objective function in a given iteration of the optimizer. If you specify the WEIGHT statement, then the contribution of each observation is weighted by multiplying it with the normalized value of the weight variable for that observation.

- If you are fitting multiple distributions in one PROC SEVERITY step and use any of the keyword functions in your program, then it is recommended that you do not explicitly use the parameters of any of the specified distributions in your programming statements.
- If you use a specific keyword function in your programming statements, then the corresponding distribution functions must be defined in a library that you specify in the CMPLIB= system option or in Sashelp.Svrtdist, the predefined functions library. In the preceding example, it is assumed that the functions D_LOGPDF and D_CDF are defined in the Work.Mydist library that is specified in the CMPLIB= option.
- You can use most DATA step statements and functions in your program. The DATA step file and the data set I/O statements (for example, INPUT, FILE, SET, and MERGE) are not available. However, some functionality of the PUT statement is supported. See the section "PROC FCMP and DATA Step Differences" in *Base SAS Procedures Guide* for more information. In addition to the differences listed in that section, the following differences exist:
  - Only numeric-valued variables can be used in PROC SEVERITY programming statements. This restriction also implies that you cannot use SAS functions or call routines that require character-valued arguments, unless you pass those arguments as constant (literal) strings or characters.
  - You cannot use functions that create lagged versions of a variable in PROC SEVERITY programming statements. If you need lagged versions, then you can use a DATA step prior to the PROC SEVERITY step to add those versions to the input data set.
- When coding your programming statements, avoid defining variables that begin with an underscore (_), because they might conflict with internal variables created by PROC SEVERITY.

Custom Objective Functions and Regression Effects

If you specify regression effects by using the SCALEMODEL statement, then PROC SEVERITY automatically adds a statement prior to your programming statements to compute the value of the scale parameter or the log-transformed scale parameter of the distribution using the values of the regression variables and internally created regression parameters. For example, if your specification of the SCALEMODEL statement results in three regression effects x1, x2, and x3, then for a model that contains the distribution D with scale parameter S, PROC SEVERITY adds a statement that is equivalent to the following statement to the beginning of your program:

```
S = _SEVTHETA0 * exp(_SEVBETA1 * x1 + _SEVBETA2 * x2 + _SEVBETA3 * x3);
```

If a model contains a distribution D1 with a log-transformed scale parameter M, PROC SEVERITY adds a statement that is equivalent to the following statement to the beginning of your program:

```
M = _SEVTHETA0 + _SEVBETA1 * x1 + _SEVBETA2 * x2 + _SEVBETA3 * x3;
```

The _SEVTHETA0, _SEVBETA1, _SEVBETA2, and _SEVBETA3 are the internal regression parameters associated with the intercept and the regression effects x1, x2, and x3, respectively.

Since the names of the internal regression parameters start with a prefix _SEV, if you use a variable in your program with a name that begins with _SEV, then PROC SEVERITY writes an error message to the SAS log and stops processing.

Multithreaded Computation

PROC SEVERITY attempts to use all the computational resources of the machine where SAS is running in order to complete the estimation tasks as fast as possible. This section describes the options that control the use of multithreading by PROC SEVERITY.

Threading refers to the organization of computational work into multiple tasks (processing units that can be scheduled by the operating system). A task is associated with a thread. Multithreading refers to the concurrent execution of threads. When multithreading is possible, substantial performance gains can be realized compared to sequential (single-threaded) execution.

The number of threads spawned by the SEVERITY procedure is determined by the number of CPUs on a machine. You can control the number of threads by specifying either the CPUCOUNT= or the NOTHREADS SAS system option.

- You can specify the CPU count with the CPUCOUNT= SAS system option. For example, if you specify the following statement, then PROC SEVERITY schedules threads as if it executed on a system with four CPUs, regardless of the actual CPU count:

  ```
  options cpucount=4;
  ```

  On most systems, the default value of the CPUCOUNT= system option is set to the number of actual CPU cores available for processing.

- If you do not want PROC SEVERITY to use multithreading, then you can turn off the THREADS SAS system option by specifying the following statement:

  ```
  options nothreads;
  ```

  On most systems, the THREADS option is turned on by default.

You can examine the current settings of these system options in the SAS log by submitting the following PROC OPTIONS step:

```
proc options option=(threads cpucount);
run;
```

Input Data Sets

PROC SEVERITY accepts DATA= and INEST= data sets as input data sets. This section details the information they are expected to contain.

DATA= Data Set

The DATA= data set is expected to contain the values of the analysis variables that you specify in the LOSS statement and the SCALEMODEL statement.

If you specify the BY statement, then the DATA= data set must contain all the BY variables that you specify in the BY statement and the data set must be sorted by the BY variables unless you specify the NOTSORTED option in the BY statement.

INEST= Data Set

The INEST= data set is expected to contain the initial values of the parameters for the parameter estimation process.

If you specify the SCALEMODEL statement, then you can use the INEST= data set only if the SCALEMODEL statement contains singleton continuous effects.

If you specify the BY statement, then the INEST= data set must contain all the BY variables that you specify in the BY statement. If you do not specify the NOTSORTED option in the BY statement, then the INEST= data set must be sorted by the BY variables. However, it is not required to contain all the BY groups present in the DATA= data set. For the BY groups that are not present in the INEST= data set, the default parameter initialization method is used. If you specify the NOTSORTED option in the BY statement, then the INEST= data set must contain all the BY groups that are present in the DATA= data set and they must appear in the same order as they appear in the DATA= data set.

In addition to any variables that you specify in the BY statement, the data set must contain the following variables:

MODEL identifying name of the distribution for which the estimates are provided.

TYPE type of the estimate. The value of this variable must be EST for an observation to be valid.

<Parameter 1> . . . <Parameter M>
M variables, named after the parameters of all candidate distributions, that contain initial values of the respective parameters. *M* is the cardinality of the union of parameter name sets from all candidate distributions. In an observation, estimates are read only from variables for parameters that correspond to the distribution that is indicated by the _MODEL_ variable.

If you specify a missing value for some parameters, then default initial values are used unless the parameter is initialized by using the INIT= option in the DIST statement. If you want to use the *dist*_PARMINIT subroutine for initializing the parameters of a model, then you should either not specify the model in the INEST= data set or specify missing values for all the distribution parameters in the INEST= data set and not use the INIT= option in the DIST statement.

If you specify regressors, then the initial value that you provide for the first parameter of each distribution must be the base value of the scale or log-transformed scale parameter. For more information, see the section "Estimating Regression Effects" on page 2123.

<Regressor 1> . . . <Regressor K>
If you specify *K* regressors in the SCALEMODEL statement, then the INEST= data set must contain *K* variables that are named for each regressor. The variables contain initial values of the respective regression coefficients. If a regressor is linearly dependent on

other regressors for a given BY group, then you can indicate this by providing a special missing value of .R for the respective variable. In a given BY group, if you mark a variable as linearly dependent for one model, then you must mark that variable as linearly dependent for all the models. Similarly, in a given BY group, if you do not mark a variable as linearly dependent for one model, then you must not mark that variable as linearly dependent for all the models.

Output Data Sets

PROC SEVERITY writes the OUTCDF=, OUTEST=, OUTMODELINFO=, and OUTSTAT= data sets when requested by their respective options. The data sets and their contents are described in the following sections.

OUTCDF= Data Set

The OUTCDF= data set records the estimates of the cumulative distribution function (CDF) of each of the specified model distributions and an estimate of the empirical distribution function (EDF).

If you specify BY variables, then the data are organized in BY groups and the data set contains variables that you specify in the BY statement. In addition, the data set contains the following variables:

<response variable>
: value of the response variable. The values are sorted. If there are multiple BY groups, the values are sorted within each BY group.

OBSNUM
: observation number in the DATA= data set. This is a sequence number that indicates the order in which the procedure accesses the observation; it does not necessarily reflect the actual observation number in the data set.

EDF
: estimate of the empirical distribution function (EDF). This estimate is computed by using the EMPIRICALCDF= option that you specify in the PROC SEVERITY statement.

_EDF_STD
: estimate of the standard error of EDF. This estimate is computed by using a method that is appropriate for the EMPIRICALCDF= option that you specify in the PROC SEVERITY statement.

_EDF_LOWER
: estimate of the lower confidence limit of EDF for a pointwise $100(1-\alpha)\%$ confidence interval, where α is the value of the EDFALPHA= option that you specify in the PROC SEVERITY statement (default is $\alpha = 0.05$). For an EDF estimate F_n that has standard error σ_n, it is computed as MAX(0, $F_n - z_{(1-\alpha/2)}\sigma_n$), where z_p is the pth quantile from the standard normal distribution.

_EDF_UPPER
: estimate of the upper confidence limit of EDF for a pointwise $100(1-\alpha)\%$ confidence interval, where α is the value of the EDFALPHA= option that you specify in the PROC SEVERITY statement (default is $\alpha = 0.05$). For an EDF estimate F_n that has standard error σ_n, it is computed as MIN(1, $F_n + z_{(1-\alpha/2)}\sigma_n$), where z_p is the pth quantile from the standard normal distribution.

<distribution1>_CDF . . . <distribution$D$>_CDF
: estimate of the cumulative distribution function (CDF) for each of the D candidate distributions, computed by using the final parameter estimates for that distribution. This value is missing if the parameter estimation process does not converge for the given distribution.

If you specify regression effects, then the reported estimates are from a mixture distribution. For more information, see the section "CDF and PDF Estimates with Regression Effects" on page 2126.

If you specify truncation, then the data set contains the following additional variables:

<distribution1>_COND_CDF . . . <distribution*D*>_COND_CDF
: estimate of the conditional CDF for each of the *D* candidate distributions, computed by using the final parameter estimates for that distribution. This value is missing if the parameter estimation process does not converge for the distribution. The conditional estimates are computed by using the method that is described in the section "Truncation and Conditional CDF Estimates" on page 2119.

OUTEST= Data Set

The OUTEST= data set records the estimates of the model parameters. It also contains estimates of their standard errors and optionally their covariance structure. If you specify BY variables, then the data are organized in BY groups and the data set contains variables that you specify in the BY statement.

If you do not specify the COVOUT option, then the data set contains the following variables:

MODEL
: identifying name of the distribution model. The observation contains information about this distribution.

TYPE
: type of the estimates reported in this observation. It can take one of the following two values:

  EST — point estimates of model parameters

  STDERR — standard error estimates of model parameters

STATUS
: status of the reported estimates. The possible values are listed in the section "_STATUS_ Variable Values" on page 2182.

<Parameter 1> . . . <Parameter M>
: *M* variables, named after the parameters of all candidate distributions, that contain estimates of the respective parameters. *M* is the cardinality of the union of parameter name sets from all candidate distributions. In an observation, estimates are populated only for parameters that correspond to the distribution that is indicated by the _MODEL_ variable. If _TYPE_ is EST, then the estimates are missing if the model does not converge. If _TYPE_ is STDERR, then the estimates are missing if covariance estimates cannot be obtained.

  If you specify regression effects, then the estimate that is reported for the first parameter of each distribution is the estimate of the base value of the scale or log-transformed scale parameter. For more information, see the section "Estimating Regression Effects" on page 2123.

<Regression Effect 1> . . . <Regression Effect K>
: If your effect specification in the SCALEMODEL statement results in *K* regression effects, then the OUTEST= data set contains *K* regression variables. The name of each variable is formed by using the name of the effect and the names of the levels of the CLASS variables that the effect might contain. If the effect name or level names are too long, then the variable name is constructed by using partial effect name and integer

identifiers for BY groups and CLASS variable levels. The label of the variable is more descriptive than the name of the variable. The variables contain estimates for their respective regression coefficients. If an effect is deemed to be linearly dependent on other effects for a given BY group, then a warning message is written to the SAS log and a special missing value of .R is written in the respective variable. If _TYPE_ is EST, then the estimates are missing if the model does not converge. If _TYPE_ is STDERR, then the estimates are missing if covariance estimates cannot be obtained.

<Offset Variable>

If you specify an OFFSET= variable in the SCALEMODEL statement, then the OUTEST= data set contains a variable that is named after the offset variable. If _TYPE_ is EST, then the value of this variable is 1. If _TYPE_ is STDERR, then the value of this variable is a special missing value of .F.

If you specify the COVOUT option in the PROC SEVERITY statement, then the OUTEST= data set contains additional observations that contain the estimates of the covariance structure. Given the symmetric nature of the covariance structure, only the lower triangular portion is reported. In addition to the variables listed and described previously, the data set contains the following variables that are either new or have a modified description:

TYPE type of the estimates reported in this observation. For observations that contain rows of the covariance structure, the value is COV.

STATUS status of the reported estimates. For observations that contain rows of the covariance structure, the status is 0 if covariance estimation was successful. If estimation fails, the status is 1 and a single observation is reported with _TYPE_=COV and missing values for all the parameter variables.

NAME name of the parameter for the row of covariance matrix that is reported in the current observation.

OUTMODELINFO= Data Set

The OUTMODELINFO= data set records the information about each candidate distribution that you specify in the DIST statement. It contains the following variables:

MODEL identifying name of the distribution model. The observation contains information about this distribution.

DEPVAR name of the loss variable.

DESCRIPTION descriptive name of the model. This has a nonmissing value only if the DESCRIPTION function has been defined for this model.

VALID validity of the distribution definition. This has a value of 1 for valid definitions and a value of 0 for invalid definitions. If the definition is invalid, then PROC SEVERITY writes the reason for invalidity to the SAS log.

_PARMNAME1 ... _PARMNAME*M*

M variables that contain names of parameters of the distribution model, where *M* is the maximum number of parameters across all the specified distribution models. For a given distribution with *m* parameters, values of variables _PARMNAME*j* ($j > m$) are missing.

OUTSTAT= Data Set

The OUTSTAT= data set records statistics of fit and model selection information. If you specify BY variables, then the data are organized in BY groups and the data set contains variables that you specify in the BY statement. The data set contains the following variables:

MODEL	identifying name of the distribution model. The observation contains information about this distribution.
NMODELPARM	number of parameters in the distribution.
NESTPARM	number of estimated parameters. This includes the regression parameters, if you specify any regression effects.
NOBS	number of nonmissing observations used for parameter estimation.
STATUS	status of the parameter estimation process for this model. The possible values are listed in the section "_STATUS_ Variable Values" on page 2182.
SELECTED	indicator of the best distribution model. If the value is 1, then this model is the best model for the current BY group according to the specified model selection criterion. This value is missing if the parameter estimation process does not converge for this model.
Neg2LogLike	value of the log likelihood, multiplied by –2, that is attained at the end of the parameter estimation process. This value is missing if the parameter estimation process does not converge for this model.
AIC	value of the Akaike's information criterion (AIC) that is attained at the end of the parameter estimation process. This value is missing if the parameter estimation process does not converge for this model.
AICC	value of the corrected Akaike's information criterion (AICC) that is attained at the end of the parameter estimation process. This value is missing if the parameter estimation process does not converge for this model.
BIC	value of the Schwarz Bayesian information criterion (BIC) that is attained at the end of the parameter estimation process. This value is missing if the parameter estimation process does not converge for this model.
KS	value of the Kolmogorov-Smirnov (KS) statistic that is attained at the end of the parameter estimation process. This value is missing if the parameter estimation process does not converge for this model.
AD	value of the Anderson-Darling (AD) statistic that is attained at the end of the parameter estimation process. This value is missing if the parameter estimation process does not converge for this model.
CVM	value of the Cramér-von Mises (CvM) statistic that is attained at the end of the parameter estimation process. This value is missing if the parameter estimation process does not converge for this model.

STATUS Variable Values

The _STATUS_ variable in the OUTEST= and OUTSTAT= data sets contains a value that indicates the status of the parameter estimation process for the respective distribution model. The variable can take the following values in the OUTEST= data set for _TYPE_=EST observations and in the OUTSTAT= data set:

0 The parameter estimation process converged for this model.

301 The parameter estimation process might not have converged for this model because there is no improvement in the objective function value. This might indicate that the initial values of the parameters are optimal, or you can try different convergence criteria in the NLOPTIONS statement.

302 The parameter estimation process might not have converged for this model because the number of iterations exceeded the maximum allowed value. You can try setting a larger value for the MAXITER= options in the NLOPTIONS statement.

303 The parameter estimation process might not have converged for this model because the number of objective function evaluations exceeded the maximum allowed value. You can try setting a larger value for the MAXFUNC= options in the NLOPTIONS statement.

304 The parameter estimation process might not have converged for this model because the time taken by the process exceeded the maximum allowed value. You can try setting a larger value for the MAXTIME= option in the NLOPTIONS statement.

400 The parameter estimation process did not converge for this model.

The _STATUS_ variable can take the following values in the OUTEST= data set for _TYPE_=STDERR and _TYPE_=COV observations:

0 The covariance and standard error estimates are available and valid.

1 The covariance and standard error estimates are not available, because the process of computing covariance estimates failed.

Displayed Output

The SEVERITY procedure optionally produces displayed output by using the Output Delivery System (ODS). All output is controlled by the PRINT= option in the PROC SEVERITY statement. Table 30.17 relates the ODS tables to PRINT= options.

Table 30.17 ODS Tables Produced in PROC SEVERITY

ODS Table Name	Description	Option
AllFitStatistics	Statistics of fit for all the distribution models	PRINT=ALLFITSTATS
ConvergenceStatus	Convergence status of parameter estimation process	PRINT=CONVSTATUS
DescStats	Descriptive statistics for the response variable	PRINT=DESCSTATS
DistributionInfo	Distribution information	PRINT=DISTINFO

Table 30.17 *continued*

ODS Table Name	Description	Option
InitialValues	Initial parameter values and bounds	PRINT=INITIALVALUES
IterationHistory	Optimization iteration history	PRINT=NLOHISTORY
ModelSelection	Model selection summary	PRINT=SELECTION
OptimizationSummary	Optimization summary	PRINT=NLOSUMMARY
ParameterEstimates	Final parameter estimates	PRINT=ESTIMATES
RegDescStats	Descriptive statistics for the regression effects that do not contain a CLASS variable	PRINT=DESCSTATS
StatisticsOfFit	Statistics of fit	PRINT=STATISTICS
Timing	Timing information for various computational stages of the procedure	PRINT=ALL
TurnbullSummary	Turnbull EDF estimation summary	PRINT=ALL

If you do not specify the PRINT= option, then by default PROC SEVERITY produces ModelSelection, ConvergenceStatus, OptimizationSummary, StatisticsOfFit, and ParameterEstimates ODS tables.

The following describes the content that is displayed in each table:

AllFitStatistics (PRINT=ALLFITSTATS)
displays the comparison of all the statistics of fit for all the models in one table. The table does not include the models whose parameter estimation process does not converge. If all the models fail to converge, then this table is not produced. If the table contains more than one model, then the best model according to each statistic is indicated with an asterisk (*) in that statistic's column.

ConvergenceStatus (PRINT=CONVSTATUS)
displays the convergence status of the parameter estimation process.

DescStats (PRINT=DESCSTATS)
displays the descriptive statistics for the response variable.

DistributionInfo (PRINT=DISTINFO)
displays the information about all the candidate distribution. It includes the name, the description, the number of distribution parameters, and whether the distribution is valid for the specified modeling task.

InitialValues (PRINT=INITIALVALUES)
displays the initial values and bounds used for estimating each model.

IterationHistory (PRINT=NLOHISTORY)
displays the iteration history of the nonlinear optimization process used for estimating the parameters.

ModelSelection (PRINT=SELECTION)
displays the model selection table. The table shows the convergence status of each candidate model, and the value of the selection criterion along with an indication of the selected model.

OptimizationSummary (PRINT=NLOSUMMARY)
displays the summary of the nonlinear optimization process used for estimating the parameters.

ParameterEstimates (PRINT=ESTIMATES)
displays the final estimates of parameters. The estimates are not displayed for models whose parameter estimation process does not converge.

RegDescStats (PRINT=DESCSTATS)
displays the descriptive statistics for the regression effects in the SCALEMODEL statement that do not contain a CLASS variable.

StatisticsOfFit (PRINT=STATISTICS)
displays the statistics of fit for each model. The statistics of fit are not displayed for models whose parameter estimation process does not converge.

Timing (PRINT=ALL)
displays elapsed times (absolute and relative) for the main tasks of the procedure.

TurnbullSummary (PRINT=ALL)
displays the summary of Turnbull's estimation process if Turnbull's method is used for computing EDF estimates. The summary includes whether the nonlinear optimization converged, the number of iterations, the maximum absolute relative error, the maximum absolute reduced gradient, and whether the final estimates are maximum likelihood estimates. This table is produced only if you specify PRINT=ALL and Turnbull's method is used for computing EDF estimates.

ODS Graphics

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*).

Before you create graphs, ODS Graphics must be enabled (for example, with the ODS GRAPHICS ON statement). For more information about enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" in that chapter.

The overall appearance of graphs is controlled by ODS styles. Styles and other aspects of using ODS Graphics are discussed in the section "A Primer on ODS Statistical Graphics" in that chapter.

This section describes the use of ODS for creating graphics with the SEVERITY procedure.

ODS Graph Names

PROC SEVERITY assigns a name to each graph that it creates by using ODS. You can use these names to selectively reference the graphs. The names are listed in Table 30.18.

Table 30.18 ODS Graphics Produced by PROC SEVERITY

ODS Graph Name	Plot Description	PLOTS= Option
CDFPlot	Comparative CDF plot	CDF
CDFDistPlot	CDF plot per distribution	CDFPERDIST
PDFPlot	Comparative PDF plot	PDF
PDFDistPlot	PDF plot per distribution	PDFPERDIST
PPPlot	P-P plot of CDF and EDF	PP
QQPlot	Q-Q plot	QQ

Comparative CDF Plot

The comparative CDF plot helps you visually compare the cumulative distribution function (CDF) estimates of all the candidate distribution models and the empirical distribution function (EDF) estimate. The plot does not contain CDF estimates for models whose parameter estimation process does not converge. The horizontal axis represents the values of the response variable. The vertical axis represents the values of the CDF or EDF estimates.

If you specify truncation, then conditional CDF estimates are plotted. Otherwise, unconditional CDF estimates are plotted. The conditional estimates are computed by using the method that is described in the section "Truncation and Conditional CDF Estimates" on page 2119.

If you specify regression effects, then the plotted CDF estimates are from a mixture distribution. For more information, see the section "CDF and PDF Estimates with Regression Effects" on page 2126.

CDF Plot per Distribution

The CDF plot per distribution shows the CDF estimates of each candidate distribution model unless that model's parameter estimation process does not converge. The plot also contains estimates of the EDF. The horizontal axis represents the values of the response variable. The vertical axis represents the values of the CDF or EDF estimates.

This plot shows the lower and upper pointwise confidence limits for the EDF estimates. For an EDF estimate F_n with standard error σ_n, they are computed as $\text{MAX}(0, F_n - z_{(1-\alpha/2)}\sigma_n)$ and $\text{MIN}(1, F_n + z_{(1-\alpha/2)}\sigma_n)$, respectively, where z_p is the pth quantile from the standard normal distribution and α denotes the confidence level that you specify in the EDFALPHA= option (the default is $\alpha = 0.05$).

If you specify truncation, then conditional CDF estimates are plotted. Otherwise, unconditional CDF estimates are plotted. The conditional estimates are computed by using the method that is described in the section "Truncation and Conditional CDF Estimates" on page 2119.

If you specify regression effects, then the plotted CDF estimates are from a mixture distribution. For more information, see the section "CDF and PDF Estimates with Regression Effects" on page 2126.

Comparative PDF Plot

The comparative PDF plot helps you visually compare the probability density function (PDF) estimates of all the candidate distribution models. The plot does not contain PDF estimates for models whose parameter estimation process does not converge. The horizontal axis represents the values of the response variable. The vertical axis represents the values of the PDF estimates.

If you specify the HISTOGRAM option, then the plot also contains the histogram of response variable values. If you specify the KERNEL option, then the plot also contains the kernel density estimate of the response variable values.

If you specify regression effects, then the plotted PDF estimates are from a mixture distribution. For more information, see the section "CDF and PDF Estimates with Regression Effects" on page 2126.

PDF Plot per Distribution

The PDF plot per distribution shows the PDF estimates of each candidate distribution model unless that model's parameter estimation process does not converge. The horizontal axis represents the values of the response variable. The vertical axis represents the values of the PDF estimates.

If you specify the HISTOGRAM option, then the plot also contains the histogram of response variable values. If you specify the KERNEL option, then the plot also contains the kernel density estimate of the response variable values.

If you specify regression effects, then the plotted PDF estimates are from a mixture distribution. For more information, see the section "CDF and PDF Estimates with Regression Effects" on page 2126.

P-P Plot of CDF and EDF

The P-P plot of CDF and EDF is the probability-probability plot that compares the CDF estimates of a distribution to the EDF estimates. A plot is not prepared for models whose parameter estimation process does not converge. The horizontal axis represents the CDF estimates of a candidate distribution, and the vertical axis represents the EDF estimates.

This plot can be interpreted as displaying the data that are used for computing the EDF-based statistics of fit for the given candidate distribution. As described in the section "EDF-Based Statistics" on page 2145, these statistics are computed by comparing the EDF, denoted by $F_n(y)$, to the CDF, denoted by $F(y)$, at each of the response variable values y. Using the probability inverse transform $z = F(y)$, this is equivalent to comparing the EDF of the z, denoted by $F_n(z)$, to the CDF of z, denoted by $F(z)$ (D'Agostino and Stephens 1986, Ch. 4). Because the CDF of z is a uniform distribution ($F(z) = z$), the EDF-based statistics can be computed by comparing the EDF estimate of z to the estimate of z. The horizontal axis of the plot represents the estimated CDF $\hat{z} = \hat{F}(y)$. The vertical axis represents the estimated EDF of z, $\hat{F}_n(z)$. The

plot contains a scatter plot of $(\hat{z},\ \hat{F}_n(z))$ points and a reference line $F_n(z) = z$ that represents the expected uniform distribution of z. Points that are scattered closer to the reference line indicate a better fit than the points that are scattered farther away from the reference line.

If you specify truncation, then the EDF estimates are conditional, as described in the section "EDF Estimates and Truncation" on page 2142. So conditional estimates of CDF are displayed, which are computed by using the method that is described in the section "Truncation and Conditional CDF Estimates" on page 2119.

If you specify regression effects, then the displayed CDF estimates, both unconditional and conditional, are from a mixture distribution. For more information, see the section "CDF and PDF Estimates with Regression Effects" on page 2126.

Q-Q Plot

The Q-Q plot is a quantile-quantile scatter plot that compares the empirical quantiles to the quantiles from a candidate distribution. A plot is not prepared for models whose parameter estimation process does not converge. The horizontal axis represents the quantiles from a candidate distribution, and the vertical axis represents the empirical quantiles.

Each point in the plot corresponds to a specific value of the EDF estimate, F_n. The Y coordinate is the value of the response variable for which F_n is computed. The X coordinate is computed by using one of the two following methods for a candidate distribution named *dist*:

- If you have defined the *dist*_QUANTILE function that satisfies the requirements listed in the section "*dist*_QUANTILE" on page 2156, then that function is invoked by using F_n and estimated distribution parameters as arguments. The QUANTILE function is defined in the Sashelp.Svrtdist library for all the predefined distributions.
- If the *dist*_QUANTILE function is not defined, then PROC SEVERITY numerically inverts the *dist*_CDF function at the CDF value of F_n for the estimated distribution parameters. If the *dist*_CDF function is not defined, then the exp(*dist*_LOGCDF) function is inverted. If the inversion fails, the corresponding point is not plotted in the Q-Q plot.

If you specify truncation, then the EDF estimates are conditional, as described in the section "EDF Estimates and Truncation" on page 2142. The CDF inversion process, whether done numerically or by evaluating the *dist*_QUANTILE function, needs to accept an unconditional CDF value. So the F_n value is first transformed to an unconditional estimate F_n^u as

$$F_n^u = F_n \cdot (\hat{F}(t_{\max}^r) - \hat{F}(t_{\min}^l)) + \hat{F}(t_{\min}^l)$$

where $\hat{F}(t_{\max}^r)$ and $\hat{F}(t_{\min}^l)$ are as defined in the section "Truncation and Conditional CDF Estimates" on page 2119.

If you specify regression effects, then the value of the first distribution parameter is determined by using the DFMIXTURE=MEAN method that is described in the section "CDF and PDF Estimates with Regression Effects" on page 2126.

Examples: SEVERITY Procedure

Example 30.1: Defining a Model for Gaussian Distribution

Suppose you want to fit a distribution model other than one of the predefined ones available to you. Suppose you want to define a model for the Gaussian distribution with the following typical parameterization of the PDF (f) and CDF (F):

$$f(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$
$$F(x;\mu,\sigma) = \frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right)$$

For PROC SEVERITY, a *distribution model* consists of a set of functions and subroutines that are defined with the FCMP procedure. Each function and subroutine should be written following certain rules. For more information, see the section "Defining a Severity Distribution Model with the FCMP Procedure" on page 2148.

NOTE: The Gaussian distribution is not a commonly used severity distribution. It is used in this example primarily to illustrate the process of defining your own distribution models. Although the distribution has a support over the entire real line, you can fit the distribution with PROC SEVERITY only if the input sample contains nonnegative values.

The following SAS statements define a distribution model named NORMAL for the Gaussian distribution. The OUTLIB= option in the PROC FCMP statement stores the compiled versions of the functions and subroutines in the 'models' package of the Work.Sevexmpl library. The LIBRARY= option in the PROC FCMP statement enables this PROC FCMP step to use the SVRTUTIL_RAWMOMENTS utility subroutine that is available in the Sashelp.Svrtdist library. The subroutine is described in the section "Predefined Utility Functions" on page 2160.

```
/*-------- Define Normal Distribution with PROC FCMP  ----------*/
proc fcmp library=sashelp.svrtdist outlib=work.sevexmpl.models;
   function normal_pdf(x,Mu,Sigma);
      /* Mu    : Location */
      /* Sigma : Standard Deviation */
      return ( exp(-(x-Mu)**2/(2 * Sigma**2)) /
             (Sigma * sqrt(2*constant('PI'))) );
   endsub;

   function normal_cdf(x,Mu,Sigma);
      /* Mu    : Location */
      /* Sigma : Standard Deviation */
      z = (x-Mu)/Sigma;
      return (0.5 + 0.5*erf(z/sqrt(2)));
   endsub;

   subroutine normal_parminit(dim, x[*], nx[*], F[*], Ftype, Mu, Sigma);
      outargs Mu, Sigma;
      array m[2] / nosymbols;
```

```
      /* Compute estimates by using method of moments */
      call svrtutil_rawmoments(dim, x, nx, 2, m);
      Mu    = m[1];
      Sigma = sqrt(m[2] - m[1]**2);
   endsub;

   subroutine normal_lowerbounds(Mu, Sigma);
      outargs Mu, Sigma;
      Mu = .;   /* Mu has no lower bound */
      Sigma = 0; /* Sigma > 0 */
   endsub;
quit;
```

The statements define the two functions required of any distribution model (NORMAL_PDF and NORMAL_CDF) and two optional subroutines (NORMAL_PARMINIT and NORMAL_LOWERBOUNDS). The name of each function or subroutine must follow a specific structure. It should start with the model's short or identifying name, which is 'NORMAL' in this case, followed by an underscore '_', followed by a keyword suffix such as 'PDF'. Each function or subroutine has a specific purpose. For more information about all the functions and subroutines that you can define for a distribution model, see the section "Defining a Severity Distribution Model with the FCMP Procedure" on page 2148. Following is the description of each function and subroutine defined in this example:

- The PDF and CDF suffixes define functions that return the probability density function and cumulative distribution function values, respectively, given the values of the random variable and the distribution parameters.
- The PARMINIT suffix defines a subroutine that returns the initial values for the parameters by using the sample data or the empirical distribution function (EDF) estimate computed from it. In this example, the parameters are initialized by using the method of moments. Hence, you do not need to use the EDF estimates, which are available in the F array. The first two raw moments of the Gaussian distribution are as follows:

  $$E[x] = \mu,\ E[x^2] = \mu^2 + \sigma^2$$

  Given the sample estimates, m_1 and m_2, of these two raw moments, you can solve the equations $E[x] = m_1$ and $E[x^2] = m_2$ to get the following estimates for the parameters: $\hat{\mu} = m_1$ and $\hat{\sigma} = \sqrt{m_2 - m_1^2}$. The NORMAL_PARMINIT subroutine implements this solution. It uses the SVRTUTIL_RAWMOMENTS utility subroutine to compute the first two raw moments.
- The LOWERBOUNDS suffix defines a subroutine that returns the lower bounds on the parameters. PROC SEVERITY assumes a default lower bound of 0 for all the parameters when a LOWERBOUNDS subroutine is not defined. For the parameter μ (*Mu*), there is no lower bound, so you need to define the NORMAL_LOWERBOUNDS subroutine. It is recommended that you assign bounds for all the parameters when you define the LOWERBOUNDS subroutine or its counterpart, the UPPERBOUNDS subroutine. Any unassigned value is returned as a missing value, which PROC SEVERITY interprets to mean that the parameter is unbounded, and that might not be what you want.

You can now use this distribution model with PROC SEVERITY. Let the following DATA step statements simulate a normal sample with $\mu = 10$ and $\sigma = 2.5$:

```
/*-------- Simulate a Normal sample ----------*/
data testnorm(keep=y);
   call streaminit(12345);
   do i=1 to 100;
      y = rand('NORMAL', 10, 2.5);
      output;
   end;
run;
```

Prior to using your distribution with PROC SEVERITY, you must communicate the location of the library that contains the definition of the distribution and the locations of libraries that contain any functions and subroutines used by your distribution model. The following OPTIONS statement sets the CMPLIB= system option to include the FCMP library Work.Sevexmpl in the search path used by PROC SEVERITY to find FCMP functions and subroutines.

```
/*--- Set the search path for functions defined with PROC FCMP ---*/
options cmplib=(work.sevexmpl);
```

Now, you are ready to fit the NORMAL distribution model with PROC SEVERITY. The following statements fit the model to the values of Y in the Work.Testnorm data set:

```
/*--- Fit models with PROC SEVERITY ---*/
proc severity data=testnorm print=all;
   loss y;
   dist Normal;
run;
```

The DIST statement specifies the identifying name of the distribution model, which is 'NORMAL'. Neither the INEST= option nor the INSTORE= option is specified in the PROC SEVERITY statement, and the INIT= option is not specified in the DIST statement. So PROC SEVERITY initializes the parameters by invoking the NORMAL_PARMINIT subroutine.

Some of the results prepared by the preceding PROC SEVERITY step are shown in Output 30.1.1 and Output 30.1.2. The descriptive statistics of variable Y and the "Model Selection" table, which includes just the normal distribution, are shown in Output 30.1.1.

Output 30.1.1 Summary of Results for Fitting the Normal Distribution

The SEVERITY Procedure

Input Data Set	
Name	WORK.TESTNORM

Output 30.1.1 *continued*

Descriptive Statistics for y	
Observations	100
Observations Used for Estimation	100
Minimum	3.88249
Maximum	16.00864
Mean	10.02059
Standard Deviation	2.37730

Model Selection

Distribution	Converged	-2 Log Likelihood	Selected
Normal	Yes	455.97541	Yes

The initial values for the parameters, the optimization summary, and the final parameter estimates are shown in Output 30.1.2. No iterations are required to arrive at the final parameter estimates, which are identical to the initial values. This confirms the fact that the maximum likelihood estimates for the Gaussian distribution are identical to the estimates obtained by the method of moments that was used to initialize the parameters in the NORMAL_PARMINIT subroutine.

Output 30.1.2 Details of the Fitted Normal Distribution Model

The SEVERITY Procedure
Normal Distribution

Distribution Information	
Name	Normal
Distribution Parameters	2

Initial Parameter Values and Bounds

Parameter	Initial Value	Lower Bound	Upper Bound
Mu	10.02059	-Infty	Infty
Sigma	2.36538	1.05367E-8	Infty

Optimization Summary	
Optimization Technique	Trust Region
Iterations	0
Function Calls	4
Log Likelihood	-227.98770

Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|
Mu	10.02059	0.23894	41.94	<.0001
Sigma	2.36538	0.16896	14.00	<.0001

The NORMAL distribution defined and illustrated here has no scale parameter, because all the following inequalities are true:

$$f(x;\mu,\sigma) \neq \frac{1}{\mu} f(\frac{x}{\mu};1,\sigma)$$
$$f(x;\mu,\sigma) \neq \frac{1}{\sigma} f(\frac{x}{\sigma};\mu,1)$$
$$F(x;\mu,\sigma) \neq F(\frac{x}{\mu};1,\sigma)$$
$$F(x;\mu,\sigma) \neq F(\frac{x}{\sigma};\mu,1)$$

This implies that you cannot estimate the influence of regression effects on a model for the response variable based on this distribution.

Example 30.2: Defining a Model for the Gaussian Distribution with a Scale Parameter

If you want to estimate the influence of regression effects, then the model needs to be parameterized to have a scale parameter. Although this might not be always possible, it is possible for the Gaussian distribution by replacing the location parameter μ with another parameter, $\alpha = \mu/\sigma$, and defining the PDF (f) and the CDF (F) as follows:

$$f(x;\sigma,\alpha) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x}{\sigma}-\alpha\right)^2\right)$$
$$F(x;\sigma,\alpha) = \frac{1}{2}\left(1+\operatorname{erf}\left(\frac{1}{\sqrt{2}}\left(\frac{x}{\sigma}-\alpha\right)\right)\right)$$

You can verify that σ is the scale parameter, because both of the following equalities are true:

$$f(x;\sigma,\alpha) = \frac{1}{\sigma} f(\frac{x}{\sigma};1,\alpha)$$
$$F(x;\sigma,\alpha) = F(\frac{x}{\sigma};1,\alpha)$$

NOTE: The Gaussian distribution is not a commonly used severity distribution. It is used in this example primarily to illustrate the concept of parameterizing a distribution such that it has a scale parameter. Although the distribution has a support over the entire real line, you can fit the distribution with PROC SEVERITY only if the input sample contains nonnegative values.

The following statements use the alternate parameterization to define a new model named NORMAL_S. The definition is stored in the Work.Sevexmpl library.

```
/*-------- Define Normal Distribution With Scale Parameter  ----------*/
proc fcmp library=sashelp.svrtdist outlib=work.sevexmpl.models;
   function normal_s_pdf(x, Sigma, Alpha);
      /* Sigma : Scale & Standard Deviation */
      /* Alpha : Scaled mean */
```

```
      return ( exp(-(x/Sigma - Alpha)**2/2) /
             (Sigma * sqrt(2*constant('PI'))) );
   endsub;

   function normal_s_cdf(x, Sigma, Alpha);
      /* Sigma : Scale & Standard Deviation */
      /* Alpha : Scaled mean */
      z = x/Sigma - Alpha;
      return (0.5 + 0.5*erf(z/sqrt(2)));
   endsub;

   subroutine normal_s_parminit(dim, x[*], nx[*], F[*], Ftype, Sigma, Alpha);
      outargs Sigma, Alpha;
      array m[2] / nosymbols;

      /* Compute estimates by using method of moments */
      call svrtutil_rawmoments(dim, x, nx, 2, m);
      Sigma = sqrt(m[2] - m[1]**2);
      Alpha = m[1]/Sigma;
   endsub;

   subroutine normal_s_lowerbounds(Sigma, Alpha);
      outargs Sigma, Alpha;
      Alpha = .; /* Alpha has no lower bound */
      Sigma = 0; /* Sigma > 0 */
   endsub;
quit;
```

An important point to note is that the scale parameter *Sigma* is the first distribution parameter (after the 'x' argument) listed in the signatures of NORMAL_S_PDF and NORMAL_S_CDF functions. *Sigma* is also the first distribution parameter listed in the signatures of other subroutines. This is required by PROC SEVERITY, so that it can identify which is the scale parameter. When you specify regression effects, PROC SEVERITY checks whether the first parameter of each candidate distribution is a scale parameter (or a log-transformed scale parameter if *dist*_SCALETRANSFORM subroutine is defined for the distribution with LOG as the transform). If it is not, then an appropriate message is written the SAS log and that distribution is not fitted.

Let the following DATA step statements simulate a sample from the normal distribution where the parameter σ is affected by the regressors as follows:

$$\sigma = \exp(1 + 0.5\ \mathrm{X1} + 0.75\ \mathrm{X3} - 2\ \mathrm{X4} + \mathrm{X5})$$

The sample is simulated such that the regressor X2 is linearly dependent on regressors X1 and X3.

```
/*--- Simulate a Normal sample affected by Regressors ---*/
data testnorm_reg(keep=y x1-x5 Sigma);
   array x{*} x1-x5;
   array b{6} _TEMPORARY_ (1 0.5 . 0.75 -2 1);
   call streaminit(34567);
   label y='Normal Response Influenced by Regressors';

   do n = 1 to 100;
      /* simulate regressors */
      do i = 1 to dim(x);
         x(i) = rand('UNIFORM');
```

```
      end;
      /* make x2 linearly dependent on x1 and x3 */
      x(2) = x(1) + 5 * x(3);

      /* compute log of the scale parameter */
      logSigma = b(1);
      do i = 1 to dim(x);
         if (i ne 2) then
            logSigma = logSigma + b(i+1) * x(i);
      end;

      Sigma = exp(logSigma);
      y = rand('NORMAL', 25, Sigma);
      output;
   end;
run;
```

The following statements use PROC SEVERITY to fit the NORMAL_S distribution model along with some of the predefined distributions to the simulated sample:

```
/*--- Set the search path for functions defined with PROC FCMP ---*/
options cmplib=(work.sevexmpl);

/*-------- Fit models with PROC SEVERITY --------*/
proc severity data=testnorm_reg print=all plots=none;
   loss y;
   scalemodel x1-x5;
   dist Normal_s burr logn pareto weibull;
run;
```

The "Model Selection" table in Output 30.2.1 indicates that all the models, except the Burr distribution model, have converged. Also, only three models, Normal_s, Burr, and Weibull, seem to have a good fit for the data. The table that compares all the fit statistics indicates that Normal_s model is the best according to the likelihood-based statistics; however, the Burr model is the best according to the EDF-based statistics.

Output 30.2.1 Summary of Results for Fitting the Normal Distribution with Regressors

The SEVERITY Procedure

Input Data Set	
Name	WORK.TESTNORM_REG

Model Selection

Distribution	Converged	-2 Log Likelihood	Selected
Normal_s	Yes	603.95786	Yes
Burr	Maybe	612.81685	No
Logn	Yes	749.20125	No
Pareto	Yes	841.07022	No
Weibull	Yes	612.77496	No

Output 30.2.1 *continued*

All Fit Statistics

Distribution	-2 Log Likelihood		AIC		AICC		BIC		KS		AD
Normal_s	603.95786	*	615.95786	*	616.86108	*	631.58888	*	1.52388	4.00152	
Burr	612.81685		626.81685		628.03424		645.05304		1.50448	* 3.90072	*
Logn	749.20125		761.20125		762.10448		776.83227		2.88110	16.20558	
Pareto	841.07022		853.07022		853.97345		868.70124		4.83810	31.60568	
Weibull	612.77496		624.77496		625.67819		640.40598		1.50490	3.90559	

Note: The asterisk (*) marks the best model according to each column's criterion.

All Fit Statistics

Distribution	CvM	
Normal_s	0.70769	
Burr	0.63399	*
Logn	3.04825	
Pareto	6.84046	
Weibull	0.63458	

Note: The asterisk (*) marks the best model according to each column's criterion.

This prompts you to further evaluate why the model with Burr distribution has not converged. The initial values, convergence status, and the optimization summary for the Burr distribution are shown in Output 30.2.2. The initial values table indicates that the regressor X2 is redundant, which is expected. More importantly, the convergence status indicates that it requires more than 50 iterations. PROC SEVERITY enables you to change several settings of the optimizer by using the NLOPTIONS statement. In this case, you can increase the limit of 50 on the iterations, change the convergence criterion, or change the technique to something other than the default trust-region technique.

Output 30.2.2 Details of the Fitted Burr Distribution Model

The SEVERITY Procedure
Burr Distribution

Distribution Information	
Name	Burr
Description	Burr Distribution
Distribution Parameters	3
Regression Parameters	4

Output 30.2.2 *continued*

Initial Parameter Values and Bounds			
Parameter	Initial Value	Lower Bound	Upper Bound
Theta	25.75198	1.05367E-8	Infty
Alpha	2.00000	1.05367E-8	Infty
Gamma	2.00000	1.05367E-8	Infty
x1	0.07345	-709.78271	709.78271
x2	Redundant		
x3	-0.14056	-709.78271	709.78271
x4	0.27064	-709.78271	709.78271
x5	-0.23230	-709.78271	709.78271

Convergence Status
Needs more than 50 iterations.

Optimization Summary	
Optimization Technique	Trust Region
Iterations	50
Function Calls	137
Log Likelihood	-306.40842

The following PROC SEVERITY step uses the NLOPTIONS statement to change the convergence criterion and the limits on the iterations and function evaluations, exclude the lognormal and Pareto distributions that have been confirmed previously to fit the data poorly, and exclude the redundant regressor X2 from the model:

```
/*--- Refit and compare models with higher limit on iterations ---*/
proc severity data=testnorm_reg print=all plots=pp;
   loss y;
   scalemodel x1 x3-x5;
   dist Normal_s burr weibull;
   nloptions absfconv=2.0e-5 maxiter=100 maxfunc=500;
run;
```

The results shown in Output 30.2.3 indicate that the Burr distribution has now converged and that the Burr and Weibull distributions have an almost identical fit for the data. The NORMAL_S distribution is still the best distribution according to the likelihood-based criteria.

Output 30.2.3 Summary of Results after Changing Maximum Number of Iterations

The SEVERITY Procedure

Input Data Set	
Name	WORK.TESTNORM_REG

Model Selection

Distribution	Converged	-2 Log Likelihood	Selected
Normal_s	Yes	603.95786	Yes
Burr	Yes	612.79276	No
Weibull	Yes	612.77496	No

All Fit Statistics

Distribution	-2 Log Likelihood		AIC		AICC		BIC		KS		AD	
Normal_s	603.95786	*	615.95786	*	616.86108	*	631.58888	*	1.52388		4.00152	
Burr	612.79276		626.79276		628.01015		645.02895		1.50472	*	3.90351	*
Weibull	612.77496		624.77496		625.67819		640.40598		1.50490		3.90559	

Note: The asterisk (*) marks the best model according to each column's criterion.

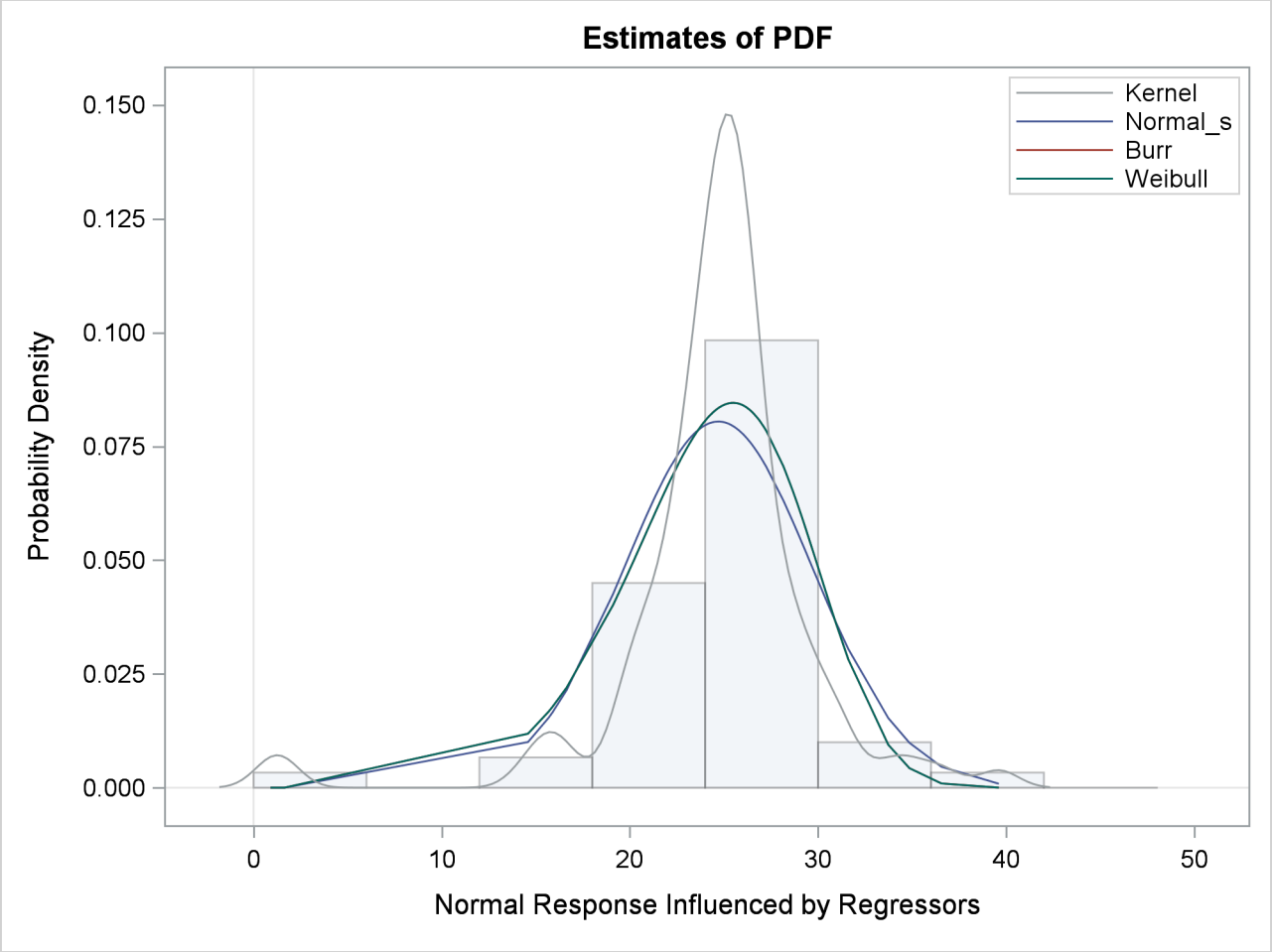
All Fit Statistics

Distribution		CvM
Normal_s	0.70769	
Burr	0.63433	*
Weibull	0.63458	

Note: The asterisk (*) marks the best model according to each column's criterion.

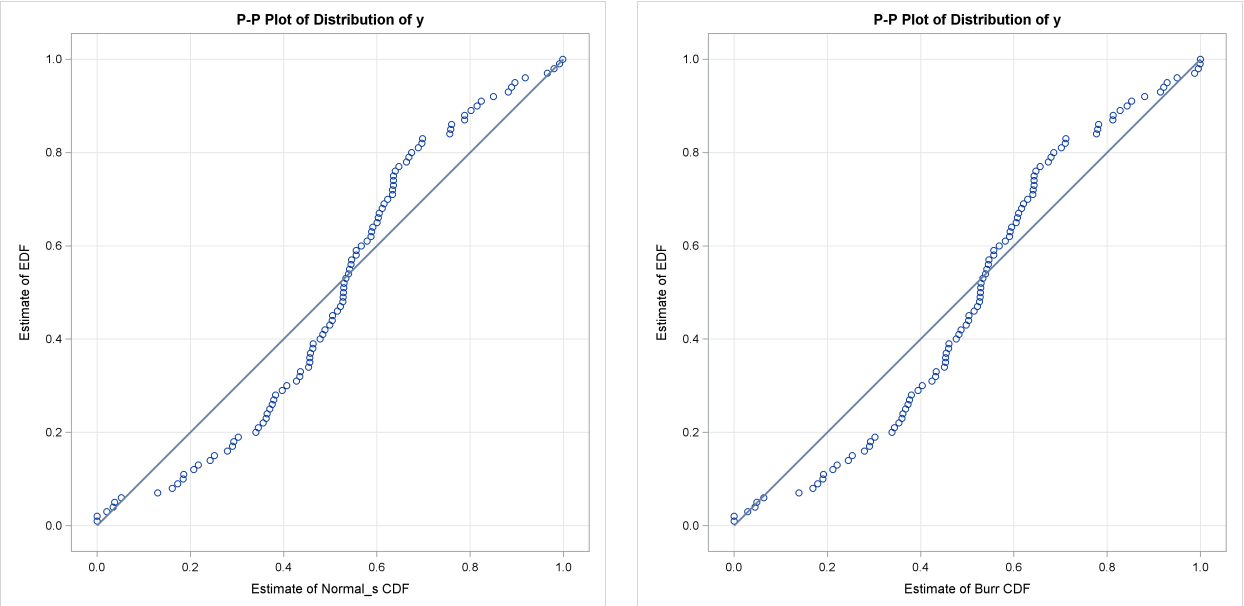
The comparison of the PDF estimates of all the candidates is shown in Output 30.2.4. Each plotted PDF estimate is an average computed over the N PDF estimates that are obtained with the scale parameter determined by each of the N observations in the input data set. The PDF plot shows that the Burr and Weibull models result in almost identical estimates. All the estimates have a slight left skew with the mode closer to Y=25, which is the mean of the simulated sample.

Output 30.2.4 Comparison of EDF and CDF Estimates of the Fitted Models



The P-P plots for the Normal_s and Burr distributions are shown in Output 30.2.5. These plots show how the EDF estimates compare against the CDF estimates. Each plotted CDF estimate is an average computed over the N CDF estimates that are obtained with the scale parameter determined by each of the N observations in the input data set. Comparing the P-P plots of Normal_s and Burr distributions indicates that both fit the data almost similarly, but the Burr distribution fits the right tail slightly better, which explains why the EDF-based statistics prefer it over the Normal_s distribution.

Output 30.2.5 Comparison of EDF and CDF Estimates of NORMAL_S and BURR Models



Example 30.3: Defining a Model for Mixed-Tail Distributions

In some applications, a few severity values tend to be extreme as compared to the typical values. The extreme values represent the worst case scenarios and cannot be discarded as outliers. Instead, their distribution must be modeled to prepare for their occurrences. In such cases, it is often useful to fit one distribution to the non-extreme values and another distribution to the extreme values. The *mixed-tail* distribution mixes two distributions: one for the *body* region, which contains the non-extreme values, and another for the *tail* region, which contains the extreme values. The tail distribution is usually a generalized Pareto distribution (GPD), because it is usually good for modeling the conditional excess severity above a threshold. The body distribution can be any distribution. The following definitions are used in describing a generic formulation of a mixed-tail distribution:

$g(x)$	PDF of the body distribution
$G(x)$	CDF of the body distribution
$h(x)$	PDF of the tail distribution
$H(x)$	CDF of the tail distribution
θ	scale parameter for the body distribution
Ω	set of nonscale parameters for the body distribution
ξ	shape parameter for the GPD tail distribution
x_r	normalized value of the response variable at which the tail starts
p_n	mixing probability

Given these notations, the PDF $f(x)$ and the CDF $F(x)$ of the mixed-tail distribution are defined as

$$f(x) = \begin{cases} \frac{p_n}{G(x_b)} g(x) & \text{if } x \le x_b \\ (1 - p_n) h(x - x_b) & \text{if } x > x_b \end{cases}$$

$$F(x) = \begin{cases} \frac{p_n}{G(x_b)} G(x) & \text{if } x \le x_b \\ p_n + (1 - p_n) H(x - x_b) & \text{if } x > x_b \end{cases}$$

where $x_b = \theta x_r$ is the value of the response variable at which the tail starts.

These definitions indicate the following:

- The body distribution is conditional on $X \le x_b$, where X denotes the random response variable.
- The tail distribution is the generalized Pareto distribution of the $(X - x_b)$ values.
- The probability that a response variable value belongs to the body is p_n. Consequently the probability that the value belongs to the tail is $(1 - p_n)$.

The parameters of this distribution are θ, Ω, ξ, x_r, and p_n. The scale of the GPD tail distribution θ_t is computed as

$$\theta_t = \frac{G(x_b; \theta, \Omega)}{g(x_b; \theta, \Omega)} \frac{(1 - p_n)}{p_n} = \theta \frac{G(x_r; \theta = 1, \Omega)}{g(x_r; \theta = 1, \Omega)} \frac{(1 - p_n)}{p_n}$$

The parameter x_r is usually estimated using a tail index estimation algorithm. One such algorithm is the Hill's algorithm (Danielsson et al. 2001), which is implemented by the predefined utility function SVRTUTIL_HILLCUTOFF available to you in the Sashelp.Svrtdist library. The algorithm and the utility function are described in detail in the section "Predefined Utility Functions" on page 2160. The function computes an estimate of x_b, which can be used to compute an estimate of x_r because $x_r = x_b/\hat{\theta}$, where $\hat{\theta}$ is the estimate of the scale parameter of the body distribution.

The parameter p_n is usually determined by the domain expert based on the fraction of losses that are expected to belong to the tail.

The following SAS statements define the LOGNGPD distribution model for a mixed-tail distribution with the lognormal distribution as the body distribution and GPD as the tail distribution:

```
/*------- Define Lognormal Body-GPD Tail Mixed Distribution -------*/
proc fcmp library=sashelp.svrtdist outlib=work.sevexmpl.models;
   function LOGNGPD_DESCRIPTION() $256;
      length desc $256;
      desc1 = "Lognormal Body-GPD Tail Distribution.";
      desc2 = " Mu, Sigma, and Xi are free parameters.";
      desc3 = " Xr and Pn are constant parameters.";
      desc = desc1 || desc2 || desc3;
      return(desc);
   endsub;

   function LOGNGPD_SCALETRANSFORM() $3;
      length xform $3;
      xform = "LOG";
      return (xform);
   endsub;
```

```
subroutine LOGNGPD_CONSTANTPARM(Xr,Pn);
endsub;

function LOGNGPD_PDF(x, Mu,Sigma,Xi,Xr,Pn);
   cutoff = exp(Mu) * Xr;
   p = CDF('LOGN',cutoff, Mu, Sigma);
   if (x < cutoff + constant('MACEPS')) then do;
      return ((Pn/p)*PDF('LOGN', x, Mu, Sigma));
   end;
   else do;
      gpd_scale = p*((1-Pn)/Pn)/PDF('LOGN', cutoff, Mu, Sigma);
      h = (1+Xi*(x-cutoff)/gpd_scale)**(-1-(1/Xi))/gpd_scale;
      return ((1-Pn)*h);
   end;
endsub;

function LOGNGPD_CDF(x, Mu,Sigma,Xi,Xr,Pn);
   cutoff = exp(Mu) * Xr;
   p = CDF('LOGN',cutoff, Mu, Sigma);
   if (x < cutoff + constant('MACEPS')) then do;
      return ((Pn/p)*CDF('LOGN', x, Mu, Sigma));
   end;
   else do;
      gpd_scale = p*((1-Pn)/Pn)/PDF('LOGN', cutoff, Mu, Sigma);
      H = 1 - (1 + Xi*((x-cutoff)/gpd_scale))**(-1/Xi);
      return (Pn + (1-Pn)*H);
   end;
endsub;

subroutine LOGNGPD_PARMINIT(dim,x[*],nx[*],F[*],Ftype,
                     Mu,Sigma,Xi,Xr,Pn);
   outargs Mu,Sigma,Xi,Xr,Pn;
   array xe[1] / nosymbols;
   array nxe[1] / nosymbols;

   eps = constant('MACEPS');

   Pn = 0.8; /* Set mixing probability */
   _status_ = .;
   call streaminit(56789);
   Xb = svrtutil_hillcutoff(dim, x, 100, 25, _status_);
   if (missing(_status_) or _status_ = 1) then
      Xb = svrtutil_percentile(Pn, dim, x, F, Ftype);

   /* Initialize lognormal parameters */
   call logn_parminit(dim, x, nx, F, Ftype, Mu, Sigma);
   if (not(missing(Mu))) then
      Xr = Xb/exp(Mu);
   else
      Xr = .;

   /* prepare arrays for excess values */
   i = 1;
   do while (i <= dim and x[i] < Xb+eps);
```

```
         i = i + 1;
      end;
      dime = dim-i+1;
      if (dime > 0) then do;
         call dynamic_array(xe, dime);
         call dynamic_array(nxe, dime);
         j = 1;
         do while(i <= dim);
            xe[j] = x[i] - Xb;
            nxe[j] = nx[i];
            i = i + 1;
            j = j + 1;
         end;

         /* Initialize GPD's shape parameter using excess values */
         call gpd_parminit(dime, xe, nxe, F, Ftype, theta_gpd, Xi);
      end;
      else do;
         Xi = .;
      end;
   endsub;

   subroutine LOGNGPD_LOWERBOUNDS(Mu,Sigma,Xi,Xr,Pn);
      outargs Mu,Sigma,Xi,Xr,Pn;

      Mu   = .; /* Mu has no lower bound */
      Sigma = 0; /* Sigma > 0 */
      Xi   = 0; /* Xi > 0 */
   endsub;
quit;
```

Note the following points about the LOGNGPD definition:

- The parameters x_r and p_n are not estimated with the maximum likelihood method used by PROC SEVERITY, so you need to specify them as *constant* parameters by defining the *dist*_CONSTANTPARM subroutine. The signature of LOGNGPD_CONSTANTPARM subroutine lists only the constant parameters *Xr* and *Pn*.

- The parameter x_r is estimated by first using the SVRTUTIL_HILLCUTOFF utility function to compute an estimate of the cutoff point $\hat{x}_b$ and then computing $x_r = \hat{x}_b / e^{\hat{\mu}}$. If SVRTUTIL_HILLCUTOFF fails to compute a valid estimate, then the SVRTUTIL_PERCENTILE utility function is used to set $\hat{x}_b$ to the p_nth percentile of the data. The parameter p_n is fixed to 0.8.

- The Sashelp.Svrtdist library is specified with the LIBRARY= option in the PROC FCMP statement to enable the LOGNGPD_PARMINIT subroutine to use the predefined utility functions (SVRTUTIL_HILLCUTOFF and SVRTUTIL_PERCENTILE) and parameter initialization subroutines (LOGN_PARMINIT and GPD_PARMINIT).

- The LOGNGPD_LOWERBOUNDS subroutine defines the lower bounds for all parameters. This subroutine is required because the parameter *Mu* has a non-default lower bound. The bounds for *Sigma* and *Xi* must be specified. If they are not specified, they are returned as missing values, which PROC

SEVERITY interprets as having no lower bound. You need not specify any bounds for the constant parameters *Xr* and *Pn*, because they are not subject to optimization.

The following DATA step statements simulate a sample from a mixed-tail distribution with a lognormal body and GPD tail. The parameter p_n is fixed to 0.8, the same value used in the LOGNGPD_PARMINIT subroutine defined previously.

```
/*----- Simulate a sample for the mixed-tail distribution -----*/
data testmixdist(keep=y label='Lognormal Body-GPD Tail Sample');
   call streaminit(45678);
   label y='Response Variable';
   N = 100;
   Mu = 1.5;
   Sigma = 0.25;
   Xi = 1.5;
   Pn = 0.8;

   /* Generate data comprising the lognormal body */
   Nbody = N*Pn;
   do i=1 to Nbody;
      y = exp(Mu) * rand('LOGNORMAL')**Sigma;
      output;
   end;

   /* Generate data comprising the GPD tail */
   cutoff = quantile('LOGNORMAL', Pn, Mu, Sigma);
   gpd_scale = (1-Pn) / pdf('LOGNORMAL', cutoff, Mu, Sigma);
   do i=Nbody+1 to N;
      y = cutoff + ((1-rand('UNIFORM'))**(-Xi) - 1)*gpd_scale/Xi;
      output;
   end;
run;
```

The following statements use PROC SEVERITY to fit the LOGNGPD distribution model to the simulated sample. They also fit three other predefined distributions (BURR, LOGN, and GPD). The final parameter estimates are written to the Work.Parmest data set.

```
/*--- Set the search path for functions defined with PROC FCMP ---*/
options cmplib=(work.sevexmpl);

/*-------- Fit LOGNGPD model with PROC SEVERITY --------*/
proc severity data=testmixdist print=all plots(histogram kernel)=all
           outest=parmest;
   loss y;
   dist logngpd burr logn gpd;
run;
```

Some of the results prepared by PROC SEVERITY are shown in Output 30.3.1 through Output 30.3.4. The "Model Selection" table in Output 30.3.1 indicates that all models converged. The last table in Output 30.3.1 shows that the model with LOGNGPD distribution has the best fit according to almost all the statistics of fit. The Burr distribution model is the closest contender to the LOGNGPD model, but the GPD distribution model fits the data very poorly.

Output 30.3.1 Summary of Fitting Mixed-Tail Distribution

The SEVERITY Procedure

Input Data Set	
Name	WORK.TESTMIXDIST
Label	Lognormal Body-GPD Tail Sample

Model Selection

Distribution	Converged	-2 Log Likelihood	Selected
logngpd	Yes	418.78232	Yes
Burr	Yes	424.93728	No
Logn	Yes	459.43471	No
Gpd	Yes	558.13444	No

All Fit Statistics

Distribution	-2 Log Likelihood		AIC		AICC		BIC		KS		AD	
logngpd	418.78232	*	428.78232	*	429.42062	*	441.80817		0.62140	*	0.31670	*
Burr	424.93728		430.93728		431.18728		438.75280	*	0.71373		0.57649	
Logn	459.43471		463.43471		463.55842		468.64505		1.55267		3.27122	
Gpd	558.13444		562.13444		562.25815		567.34478		3.43470		16.74156	

Note: The asterisk (*) marks the best model according to each column's criterion.

All Fit Statistics

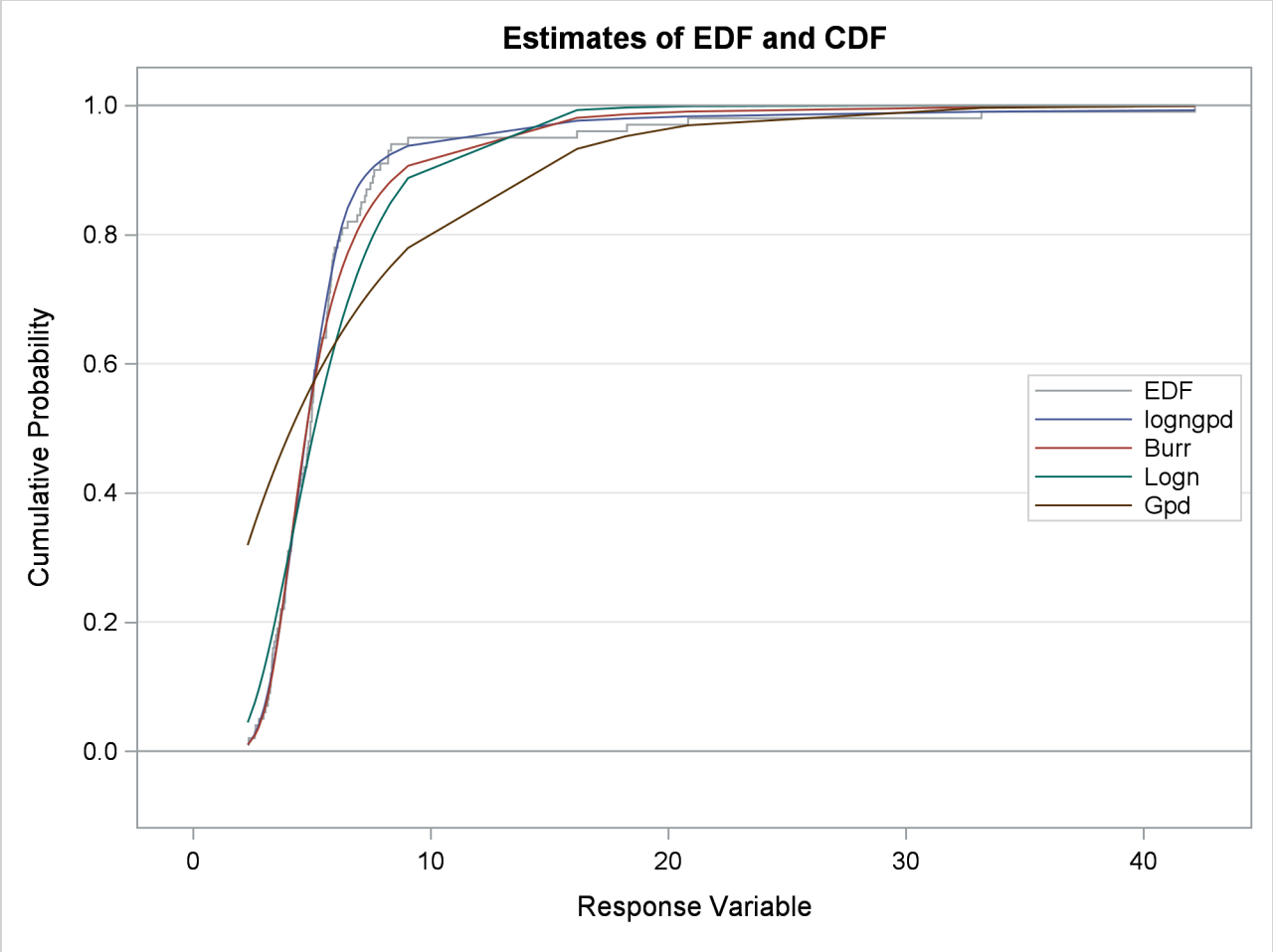
Distribution		CvM
logngpd	0.04972	*
Burr	0.07860	
Logn	0.48448	
Gpd	3.31860	

Note: The asterisk (*) marks the best model according to each column's criterion.

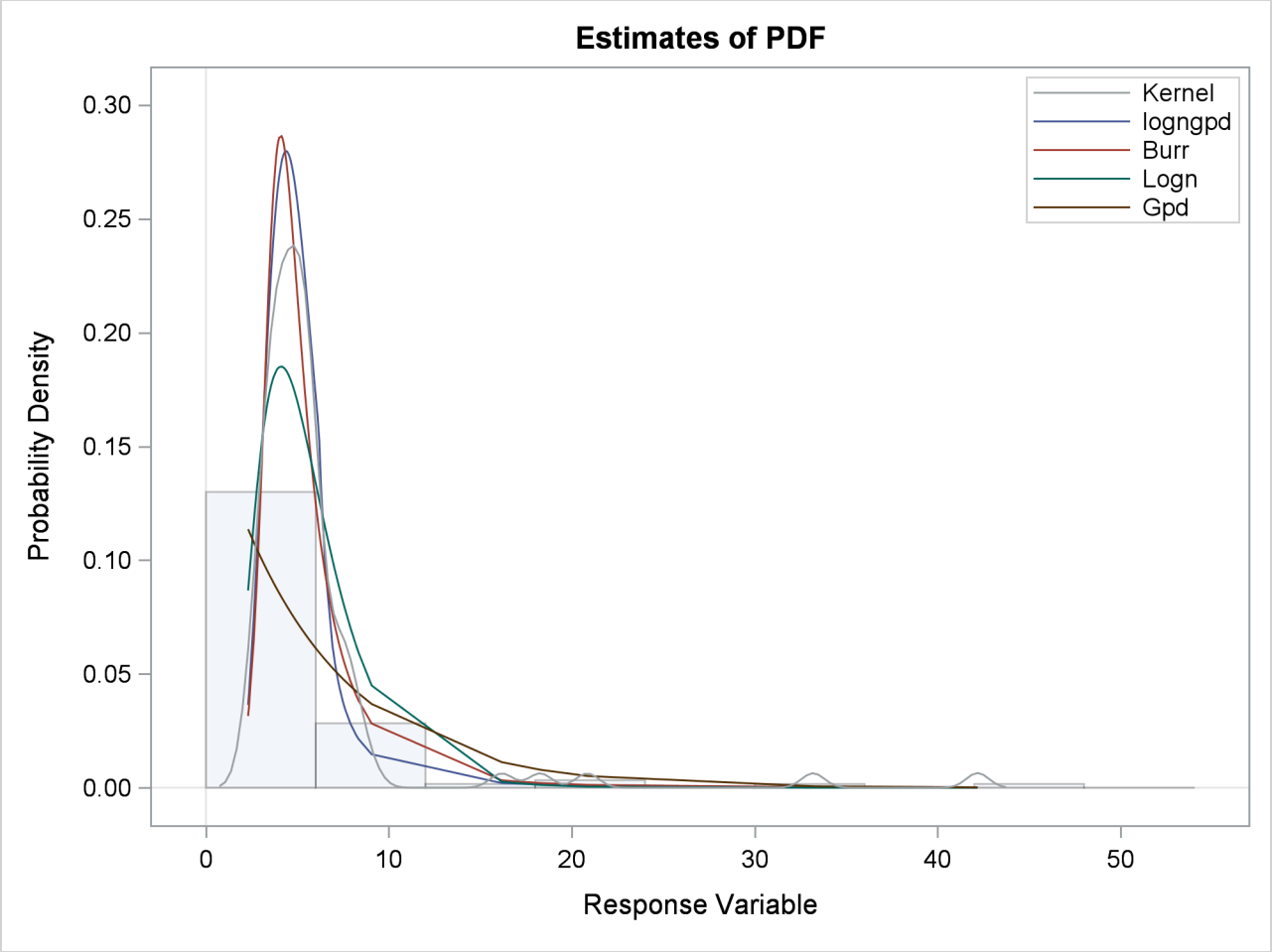
The plots in Output 30.3.2 show that both the lognormal and GPD distributions fit the data poorly, GPD being the worst. The Burr distribution fits the data as well as the LOGNGPD distribution in the body region, but has a poorer fit in the tail region than the LOGNGPD distribution.

The P-P plots of Output 30.3.3 provide a better visual confirmation that the LOGNGPD distribution fits the tail region better than the Burr distribution.

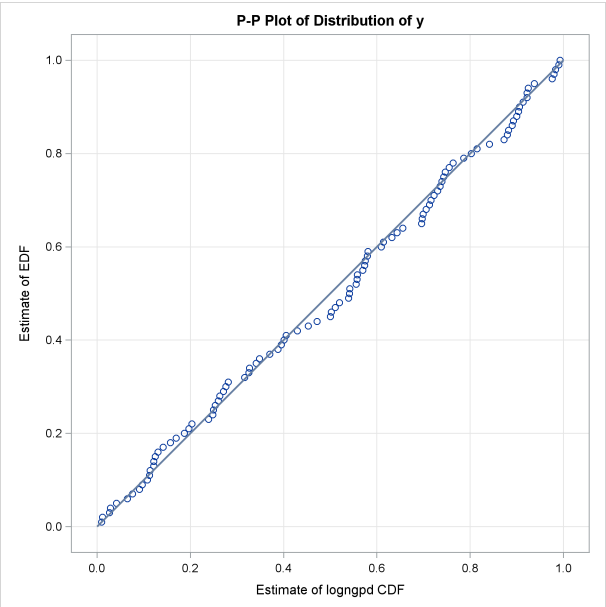
Output 30.3.2 Comparison of the CDF and PDF Estimates of the Fitted Models

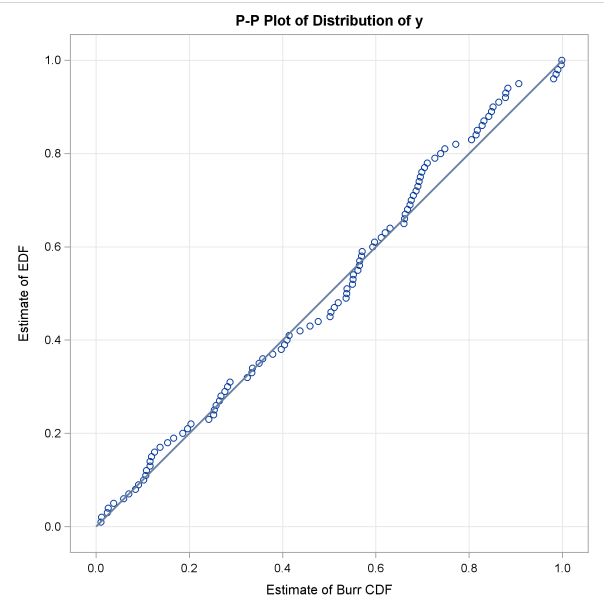


Output 30.3.2 *continued*



Output 30.3.3 P-P Plots for the LOGNGPD and BURR Distribution Models





The detailed results for the LOGNGPD distribution are shown in Output 30.3.4. The initial values table indicates the values computed by LOGNGPD_PARMINIT subroutine for the *Xr* and *Pn* parameters. It also uses the bounds columns to indicate the constant parameters. The last table in the figure shows the final parameter estimates. The estimates of all free parameters are significantly different from 0. As expected, the final estimates of the constant parameters *Xr* and *Pn* have not changed from their initial values.

Output 30.3.4 Detailed Results for the LOGNGPD Distribution

The SEVERITY Procedure
lognpgd Distribution

Distribution Information	
Name	lognpgd
Description	Lognormal Body-GPD Tail Distribution. Mu, Sigma, and Xi are free parameters. Xr and Pn are constant parameters.
Distribution Parameters	5

Initial Parameter Values and Bounds

Parameter	Initial Value	Lower Bound	Upper Bound
Mu	1.49954	-Infty	Infty
Sigma	0.76306	1.05367E-8	Infty
Xi	0.36661	1.05367E-8	Infty
Xr	1.27395	Constant	Constant
Pn	0.80000	Constant	Constant

Convergence Status
Convergence criterion (GCONV=1E-8) satisfied.

Optimization Summary	
Optimization Technique	Trust Region
Iterations	11
Function Calls	33
Failed Function Calls	1
Log Likelihood	-209.39116

Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|
Mu	1.57921	0.06426	24.57	<.0001
Sigma	0.31868	0.04459	7.15	<.0001
Xi	1.03771	0.38205	2.72	0.0078
Xr	1.27395	Constant	.	.
Pn	0.80000	Constant	.	.

The following SAS statements use the parameter estimates to compute the value where the tail region is estimated to start ($x_b = e^{\hat{\mu}}\hat{x_r}$) and the scale of the GPD tail distribution ($\theta_t = \frac{G(x_b)}{g(x_b)}\frac{(1-p_n)}{p_n}$):

```
/*-------- Compute tail cutoff and tail distribution's scale --------*/
data xb_thetat(keep=x_b theta_t);
   set parmest(where=(_MODEL_='logngpd' and _TYPE_='EST'));
   x_b = exp(Mu) * Xr;
   theta_t = (CDF('LOGN',x_b,Mu,Sigma)/PDF('LOGN',x_b,Mu,Sigma)) *
             ((1-Pn)/Pn);
run;

proc print data=xb_thetat noobs;
run;
```

Output 30.3.5 Start of the Tail and Scale of the GPD Tail Distribution

x_b	theta_t
6.18005	1.27865

The computed values of x_b and θ_t are shown as *x_b* and *theta_t* in Output 30.3.5. Equipped with this additional derived information, you can now interpret the results of fitting the mixed-tail distribution as follows:

- The tail starts at $y \approx 6.18$. The primary benefit of using the scale-normalized cutoff (x_r) as the constant parameter instead of using the actual cutoff (x_b) is that the absolute cutoff gets optimized by virtue of optimizing the scale of the body region ($\theta = e^{\mu}$).
- The values $y \leq 6.18$ follow the lognormal distribution with parameters $\mu \approx 1.58$ and $\sigma \approx 0.32$. These parameter estimates are reasonably close to the parameters used for simulating the sample.
- The values $y_t = y - 6.18$ ($y_t > 0$) follow the GPD distribution with scale $\theta_t \approx 1.28$ and shape $\xi \approx 1.04$.

Example 30.4: Estimating Parameters Using Cramér-von Mises Estimator

PROC SEVERITY enables you to estimate model parameters by minimizing your own objective function. This example illustrates how you can use PROC SEVERITY to implement the Cramér-von Mises estimator. Let $F(y_i;\Theta)$ denote the estimate of CDF at y_i for a distribution with parameters Θ, and let $F_n(y_i)$ denote the empirical estimate of CDF (EDF) at y_i that is computed from a sample y_i, $1 \leq i \leq N$. Then, the Cramér-von Mises estimator of the parameters is defined as

$$\hat{\Theta} = \arg\min_{\Theta} \sum_{i=1}^{N} (F(y_i;\Theta) - F_n(y_i))^2$$

This estimator belongs to the class of minimum distance estimators. It attempts to estimate the parameters such that the squared distance between the CDF and EDF estimates is minimized.

The following PROC SEVERITY step uses the Cramér-von Mises estimator to fit four candidate distribution models, including the LOGNGPD mixed-tail distribution model that was defined in "Example 30.3: Defining a Model for Mixed-Tail Distributions" on page 2199. The input sample is the same as is used in that example.

```
/*--- Set the search path for functions defined with PROC FCMP ---*/
options cmplib=(work.sevexmpl);

/*-------- Fit LOGNGPD model with PROC SEVERITY by using -------
  -------- the Cramer-von Mises minimum distance estimator -------*/
proc severity data=testmixdist obj=cvmobj print=all plots=pp;
   loss y;
   dist logngpd burr logn gpd;

   * Cramer-von Mises estimator (minimizes the distance *
   * between parametric and nonparametric estimates)    *;
   cvmobj = _cdf_(y);
   cvmobj = (cvmobj -_edf_(y))**2;
run;
```

The OBJ= option in the PROC SEVERITY statement specifies that the objective function cvmobj should be minimized. The programming statements compute the contribution of each observation in the input data set to the objective function cvmobj. The use of keyword functions _CDF_ and _EDF_ makes the program applicable to all the distributions.

Some of the key results prepared by PROC SEVERITY are shown in Output 30.4.1. The "Model Selection" table indicates that all models converged. When you specify a custom objective function, the default selection criterion is the value of the custom objective function. The "All Fit Statistics" table indicates that LOGNGPD is the best distribution according to all the statistics of fit. Comparing the fit statistics of Output 30.4.1 with those of Output 30.3.1 indicates that the use of the Cramér-von Mises estimator has resulted in smaller values for all the EDF-based statistics of fit for all the models, which is expected from a minimum distance estimator.

Output 30.4.1 Summary of Cramér-von Mises Estimation

The SEVERITY Procedure

Input Data Set	
Name	WORK.TESTMIXDIST
Label	Lognormal Body-GPD Tail Sample

Model Selection

Distribution	Converged	cvmobj	Selected
logngpd	Yes	0.02694	Yes
Burr	Yes	0.03325	No
Logn	Yes	0.03633	No
Gpd	Yes	2.96090	No

Output 30.4.1 *continued*

All Fit Statistics											
Distribution	cvmobj		-2 Log Likelihood		AIC		AICC		BIC		KS
logngpd	0.02694	*	419.49635	*	429.49635	*	430.13464	*	442.52220	*	0.51332	*
Burr	0.03325		436.58823		442.58823		442.83823		450.40374		0.53084	
Logn	0.03633		491.88659		495.88659		496.01030		501.09693		0.52469	
Gpd	2.96090		560.35409		564.35409		564.47780		569.56443		2.99095	

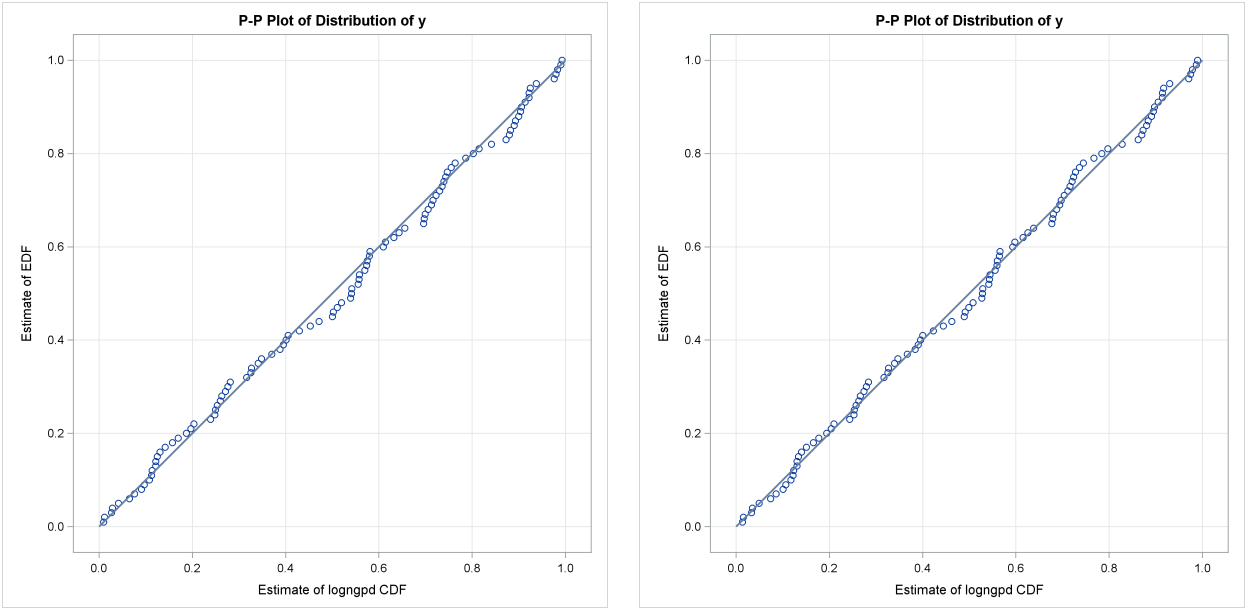
Note: The asterisk (*) marks the best model according to each column's criterion.

All Fit Statistics				
Distribution	AD		CvM	
logngpd	0.21563	*	0.03030	*
Burr	0.82875		0.03807	
Logn	2.08312		0.04173	
Gpd	15.51378		2.97806	

Note: The asterisk (*) marks the best model according to each column's criterion.

The P-P plots in Output 30.4.2 provide a visual confirmation that the CDF estimates match the EDF estimates more closely when compared to the estimates that are obtained with the maximum likelihood estimator.

Output 30.4.2 P-P Plots for LOGNGPD Model with Maximum Likelihood (Left) and Cramér-von Mises (Right) Estimators



Example 30.5: Fitting a Scaled Tweedie Model with Regressors

The Tweedie distribution is often used in the insurance industry to explain the influence of regression effects on the distribution of losses. PROC SEVERITY provides a predefined scaled Tweedie distribution (STWEEDIE) that enables you to model the influence of regression effects on the scale parameter. The

scale regression model has its own advantages such as the ability to easily account for inflation effects. This example illustrates how that model can be used to evaluate the influence of regression effects on the *mean* of the Tweedie distribution, which is useful in problems such rate-making and pure premium modeling.

Assume a Tweedie process, whose mean μ is affected by k regression effects x_j, $j = 1, \ldots, k$ as follows:

$$\mu = \mu_0 \exp\left(\sum_{j=1}^{k} \beta_j x_j\right)$$

where μ_0 represents the base value of the mean (you can think of μ_0 as $\exp(\beta_0)$, where β_0 is the intercept). This model for the mean is identical to the popular generalized linear model for the mean with a logarithmic link function.

More interestingly, it parallels the model used by PROC SEVERITY for the scale parameter θ,

$$\theta = \theta_0 \exp\left(\sum_{j=1}^{k} \beta_j x_j\right)$$

where θ_0 represents the base value of the scale parameter. As described in the section "Tweedie Distributions" on page 2110, for the parameter range $p \in (1, 2)$, the mean of the Tweedie distribution is given by

$$\mu = \theta\lambda\frac{2-p}{p-1}$$

where λ is the Poisson mean parameter of the scaled Tweedie distribution. This relationship enables you to use the scale regression model to infer the influence of regression effects on the mean of the distribution.

Let the data set Work.Test_Sevtw contain a sample generated from a Tweedie distribution with dispersion parameter $\phi = 0.5$, index parameter $p = 1.75$, and the mean parameter that is affected by three regression variables x1, x2, and x3 as follows:

$$\mu = 5\ \exp(0.25\,\mathrm{x1} - \mathrm{x2} + 3\,\mathrm{x3})$$

Thus, the population values of regression parameters are $\mu_0 = 5$, $\beta_1 = 0.25$, $\beta_2 = -1$, and $\beta_3 = 3$. You can find the code used to generate the sample in the PROC SEVERITY sample program *sevex05.sas*.

The following PROC SEVERITY step uses the sample in Work.Test_Sevtw data set to estimate the parameters of the scale regression model for the predefined scaled Tweedie distribution (STWEEDIE) with the dual quasi-Newton (QUANEW) optimization technique:

```
/*--- Fit the scale parameter version of the Tweedie distribution ---*/
proc severity data=test_sevtw outest=estw covout print=all plots=none;
   loss y;
   scalemodel x1-x3;

   dist stweedie;
   nloptions tech=quanew;
run;
```

The dual quasi-Newton technique is used because it requires only the first-order derivatives of the objective function, and it is harder to compute reasonably accurate estimates of the second-order derivatives of Tweedie distribution's PDF with respect to the parameters.

Some of the key results prepared by PROC SEVERITY are shown in Output 30.5.1 and Output 30.5.2. The distribution information and the convergence results are shown in Output 30.5.1.

Output 30.5.1 Convergence Results for the STWEEDIE Model with Regressors

The SEVERITY Procedure
stweedie Distribution

Distribution Information	
Name	stweedie
Description	Tweedie Distribution with Scale Parameter
Distribution Parameters	3
Regression Parameters	3

Convergence Status
Convergence criterion (FCONV=2.220446E-16) satisfied.

Optimization Summary	
Optimization Technique	Dual Quasi-Newton
Iterations	42
Function Calls	243
Log Likelihood	-1044.3

The final parameter estimates of the STWEEDIE regression model are shown in Output 30.5.2. The estimate that is reported for the parameter Theta is the estimate of the base value θ_0. The estimates of regression coefficients β_1, β_2, and β_3 are indicated by the rows of x1, x2, and x3, respectively.

Output 30.5.2 Parameter Estimates for the STWEEDIE Model with Regressors

Parameter Estimates				
Parameter	**Estimate**	**Standard Error**	**t Value**	**Approx Pr > |t|**
Theta	0.82345	0.29281	2.81	0.0053
Lambda	16.24203	14.02459	1.16	0.2478
P	1.74930	0.22497	7.78	<.0001
x1	0.27930	0.09879	2.83	0.0050
x2	-0.76730	0.10318	-7.44	<.0001
x3	3.03183	0.10143	29.89	<.0001

If your goal is to explain the influence of regression effects on the scale parameter, then the output displayed in Output 30.5.2 is sufficient. But, if you want to compute the influence of regression effects on the mean of the distribution, then you need to do some postprocessing. Using the relationship between μ and θ, μ can be written in terms of the parameters of the STWEEDIE model as

$$\mu = \theta_0 \exp\left(\sum_{j=1}^{k} \beta_j x_j\right) \lambda \frac{2-p}{p-1}$$

This shows that the parameters β_j are identical for the mean and the scale model, and the base value μ_0 of the mean model is

$$\mu_0 = \theta_0 \lambda \frac{2-p}{p-1}$$

The estimate of μ_0 and the standard error associated with it can be computed by using the property of the functions of maximum likelihood estimators (MLE). If $g(\Omega)$ represents a totally differentiable function of parameters Ω, then the MLE of g has an asymptotic normal distribution with mean $g(\hat{\Omega})$ and covariance $C = (\partial \mathbf{g})' \Sigma (\partial \mathbf{g})$, where $\hat{\Omega}$ is the MLE of Ω, Σ is the estimate of covariance matrix of Ω, and $\partial \mathbf{g}$ is the gradient vector of g with respect to Ω evaluated at $\hat{\Omega}$. For μ_0, the function is $g(\Omega) = \theta_0 \lambda (2-p)/(p-1)$. The gradient vector is

$$\begin{aligned} \partial \mathbf{g} &= \left(\frac{\partial g}{\partial \theta_0} \quad \frac{\partial g}{\partial \lambda} \quad \frac{\partial g}{\partial p} \quad \frac{\partial g}{\partial \beta_1} \cdots \frac{\partial g}{\partial \beta_k} \right) \\ &= \left(\frac{\mu_0}{\theta_0} \quad \frac{\mu_0}{\lambda} \quad \frac{-\mu_0}{(p-1)(2-p)} \quad 0 \ldots 0 \right) \end{aligned}$$

You can write a DATA step that implements these computations by using the parameter and covariance estimates prepared by PROC SEVERITY step. The DATA step program is available in the sample program *sevex05.sas*. The estimates of μ_0 prepared by that program are shown in Output 30.5.3. These estimates and the estimates of β_j as shown in Output 30.5.2 are reasonably close (that is, within one or two standard errors) to the parameters of the population from which the sample in Work.Test_Sevtw data set was drawn.

Output 30.5.3 Estimate of the Base Value Mu0 of the Mean Parameter

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|
Mu0	4.47472	0.42271	10.5858	0

Another outcome of using the scaled Tweedie distribution to model the influence of regression effects is that the regression effects also influence the variance V of the Tweedie distribution. The variance is related to the mean as $V = \phi \mu^p$, where ϕ is the dispersion parameter. Using the relationship between the parameters TWEEDIE and STWEEDIE distributions as described in the section "Tweedie Distributions" on page 2110, the regression model for the dispersion parameter is

$$\begin{aligned} \log(\phi) &= (2-p)\log(\mu) - \log(\lambda(2-p)) \\ &= ((2-p)\log(\mu_0) - \log(\lambda(2-p))) + (2-p)\sum_{j=1}^{k} \beta_j x_j \end{aligned}$$

Subsequently, the regression model for the variance is

$$\begin{aligned} \log(V) &= 2\log(\mu) - \log(\lambda(2-p)) \\ &= (2\log(\mu_0) - \log(\lambda(2-p))) + 2\sum_{j=1}^{k} \beta_j x_j \end{aligned}$$

In summary, PROC SEVERITY enables you to estimate regression effects on various parameters and statistics of the Tweedie model.

Example 30.6: Fitting Distributions to Interval-Censored Data

In some applications, the data available for modeling might not be exact. A commonly encountered scenario is the use of grouped data from an external agency, which for several reasons, including privacy, does not provide information about individual loss events. The losses are grouped into disjoint bins, and you know only the range and number of values in each bin. Each group is essentially interval-censored, because you know that a loss magnitude is in certain interval, but you do not know the exact magnitude. This example illustrates how you can use PROC SEVERITY to model such data.

The following DATA step generates sample grouped data for dental insurance claims, which is taken from Klugman, Panjer, and Willmot (1998):

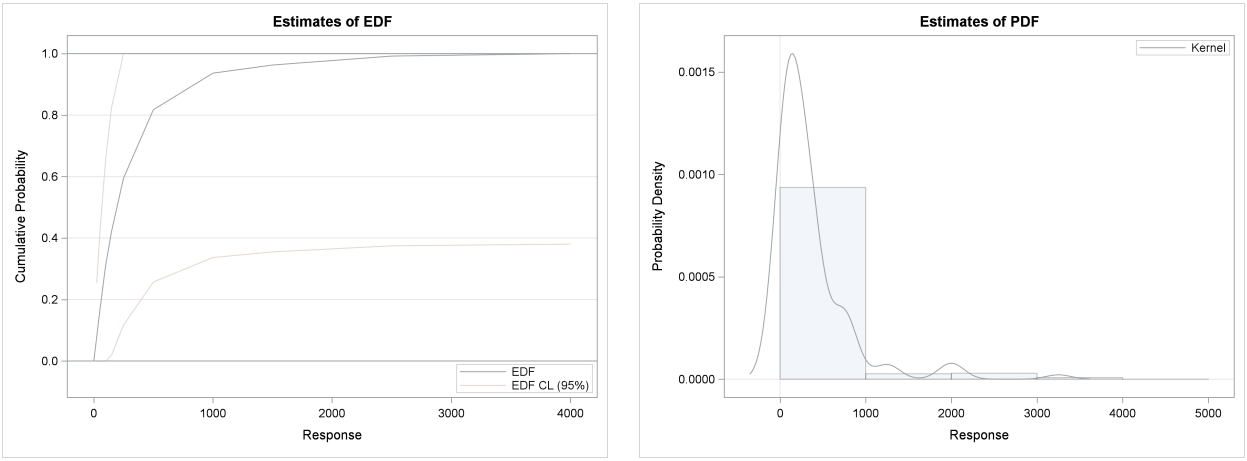
```
/* Grouped dental insurance claims data
    (Klugman, Panjer, and Willmot, 1998) */
data gdental;
   input lowerbd upperbd count @@;
   datalines;
0 25 30  25 50 31  50 100 57  100 150 42  150 250 65  250 500 84
500 1000 45  1000 1500 10  1500 2500 11  2500 4000 3
;
run;
```

Often, when you do not know the nature of the data, it is recommended that you first explore the nature of the sample distribution by examining the nonparametric estimates of PDF and CDF. The following PROC SEVERITY step prepares the nonparametric estimates, but it does not fit any distribution because there is no DIST statement specified:

```
/* Prepare nonparametric estimates */
proc severity data=gdental print=all plots(histogram kernel)=all;
   loss / rc=lowerbd lc=upperbd;
   weight count;
run;
```

The LOSS statement specifies the left and right boundary of each group as the right-censoring and left-censoring limits, respectively. The variable count records the number of losses in each group and is specified in the WEIGHT statement. Note that there is no response or loss variable specified in the LOSS statement, which is allowed as long as each observation in the input data set is censored. The nonparametric estimates prepared by this step are shown in Output 30.6.1. The histogram, kernel density, and EDF plots all indicate that the data is heavy-tailed. For interval-censored data, PROC SEVERITY uses Turnbull's algorithm to compute the EDF estimates. The plot of Turnbull's EDF estimates is shown to be linear between the endpoints of a censored group. The linear relationship is chosen for convenient visualization and ease of computation of EDF-based statistics, but you should note that theoretically the behavior of Turnbull's EDF estimates is undefined within a group.

Output 30.6.1 Nonparametric Distribution Estimates for Interval-Censored Data



With the PRINT=ALL option, PROC SEVERITY prints the summary of the Turnbull EDF estimation process as shown in Output 30.6.2. It indicates that the final EDF estimates have converged and are in fact maximum likelihood (ML) estimates. If they were not ML estimates, then you could have used the ENSUREMLE option to force the algorithm to search for ML estimates.

Output 30.6.2 Turnbull EDF Estimation Summary for Interval-Censored Data

Turnbull EDF Estimation Summary	
Technique	EM with Maximum Likelihood Check
Convergence Status	Converged
Iterations	2
Maximum Absolute Relative Error	1.8406E-16
Maximum Absolute Reduced Gradient	1.7764E-15
Estimates	Maximum Likelihood

After exploring the nature of the data, you can now fit a set of heavy-tailed distributions to this data. The following PROC SEVERITY step fits all the predefined distributions to the data in Work.Gdental data set:

```
/* Fit all predefined distributions */
proc severity data=gdental print=all plots(histogram kernel)=all
              criterion=ad;
   loss / rc=lowerbd lc=upperbd;
   weight count;
   dist _predef_;
run;
```

Some of the key results prepared by PROC SEVERITY are shown in Output 30.6.3 through Output 30.6.4. According to the "Model Selection" table in Output 30.6.3, all distribution models have converged. The "All Fit Statistics" table in Output 30.6.3 indicates that the generalized Pareto distribution (GPD) has the best fit for data according to a majority of the likelihood-based statistics and that the Burr distribution (BURR) has the best fit according to all the EDF-based statistics.

Output 30.6.3 Statistics of Fit for Interval-Censored Data

The SEVERITY Procedure

Input Data Set	
Name	WORK.GDENTAL

Model Selection			
Distribution	**Converged**	**AD**	**Selected**
Burr	Yes	0.00103	Yes
Exp	Yes	0.09936	No
Gamma	Yes	0.04608	No
Igauss	Yes	0.12301	No
Logn	Yes	0.01884	No
Pareto	Yes	0.00739	No
Gpd	Yes	0.00739	No
Weibull	Yes	0.03293	No

All Fit Statistics

Distribution	-2 Log Likelihood		AIC		AICC		BIC		KS		AD
Burr	41.41112	*	47.41112		51.41112		48.31888		0.08974	*	0.00103	*
Exp	42.14768		44.14768	*	44.64768	*	44.45026	*	0.26412		0.09936	
Gamma	41.92541		45.92541		47.63969		46.53058		0.19569		0.04608	
Igauss	42.34445		46.34445		48.05874		46.94962		0.34514		0.12301	
Logn	41.62598		45.62598		47.34027		46.23115		0.16853		0.01884	
Pareto	41.45480		45.45480		47.16908		46.05997		0.11423		0.00739	
Gpd	41.45480		45.45480		47.16908		46.05997		0.11423		0.00739	
Weibull	41.76272		45.76272		47.47700		46.36789		0.17238		0.03293	

Note: The asterisk (*) marks the best model according to each column's criterion.

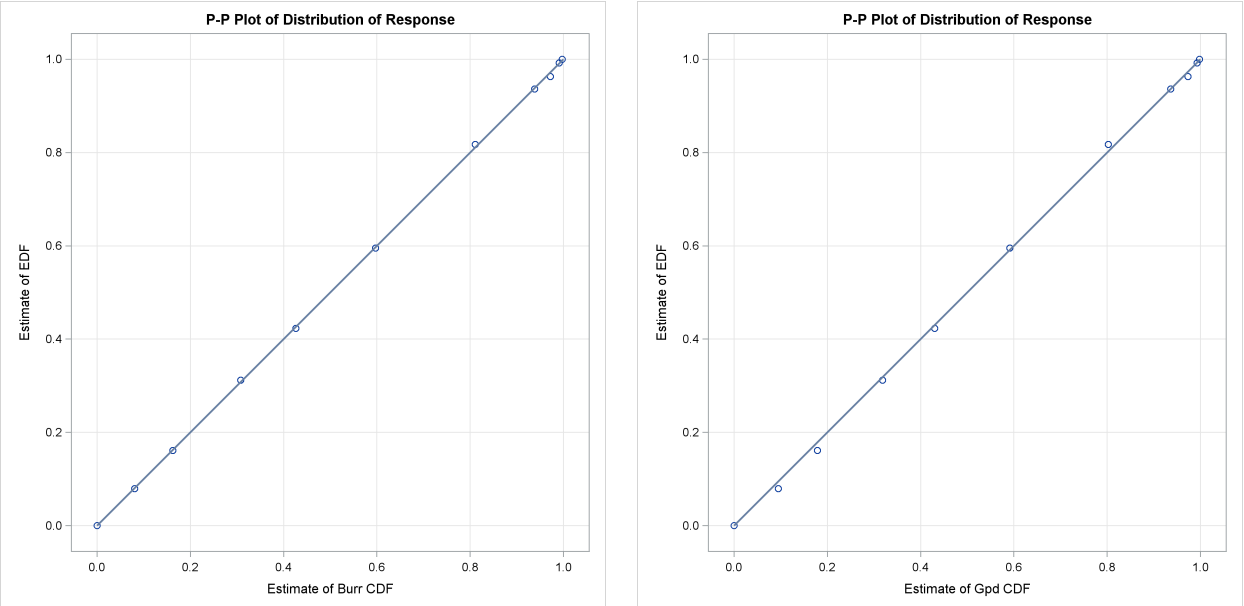
All Fit Statistics

Distribution		CvM
Burr	0.0000816	*
Exp	0.01866	
Gamma	0.00759	
Igauss	0.02562	
Logn	0.00333	
Pareto	0.0009084	
Gpd	0.0009084	
Weibull	0.00472	

Note: The asterisk (*) marks the best model according to each column's criterion.

The P-P plots of Output 30.6.4 show that both GPD and BURR have a close fit between EDF and CDF estimates, although BURR has slightly better fit, which is also indicated by the EDF-based statistics. Given that BURR is a generalization of the GPD and that the plots do not offer strong evidence in support of the more complex distribution, GPD seems like a good choice for this data.

Output 30.6.4 P-P Plots of Burr and GPD for Interval-Censored Data



Example 30.7: Defining a Finite Mixture Model That Has a Scale Parameter

A finite mixture model is a stochastic model that postulates that the probability distribution of the data generation process is a mixture of a finite number of probability distributions. For example, when an insurance company analyzes loss data from multiple policies that are underwritten in different geographic regions, some regions might behave similarly, but the distribution that governs some regions might be different from the distribution that governs other regions. Further, it might not be known which regions behave similarly. Also, the larger amounts of losses might follow a different stochastic process from the stochastic process that governs the smaller amounts of losses. It helps to model all policies together in order to pool the data together and exploit any commonalities among the regions, and the use of a finite mixture model can help capture the differences in distributions across regions and ranges of loss amounts.

Formally, if f_i and F_i denote the PDF and CDF, respectively, of component distribution i and p_i represents the mixing probability that is associated with component i, then the PDF and CDF of the finite mixture of K distribution components are

$$f(x;\Theta,\mathbf{p}) = \sum_{i=1}^{K} p_i\, f_i(x;\Theta_i)$$

$$F(x;\Theta,\mathbf{p}) = \sum_{i=1}^{K} p_i\, F_i(x;\Theta_i)$$

where Θ_i denotes the parameters of component distribution i and Θ denotes the parameters of the mixture distribution, which is a union of all the Θ_i parameters. $\mathbf{p}$ denotes the set of mixing probabilities. All mixing probabilities must add up to 1 ($\sum_{i=1}^{K} p_i = 1$).

You can define the finite mixture of a specific number of components and specific distributions for each of the components by defining the FCMP functions for the PDF and CDF. However, in general, it is not possible to fit a scale regression model by using any finite mixture distribution unless you take special care to ensure

that the mixture distribution has a scale parameter. This example provides a formulation of a two-component finite mixture model that has a scale parameter.

To start with, each component distribution must have either a scale parameter or a log-transformed scale parameter. Let θ_1 and θ_2 denote the scale parameters of the first and second components, respectively. Let $p_1 = p$ be the mixing probability, which makes $p_2 = 1 - p$ by using the constraint on $\mathbf{p}$. The PDF of the mixture of these two distributions can be written as

$$f(x; \theta_1, \theta_2, \Phi, p) = \frac{p}{\theta_1} f_1(\frac{x}{\theta_1}; \Phi_1) + \frac{1-p}{\theta_2} f_2(\frac{x}{\theta_2}; \Phi_2)$$

where Φ_1 and Φ_2 denote the sets of nonscale parameters of the first and second components, respectively, and Φ denotes a union of Φ_1 and Φ_2. For the mixture to have the scale parameter θ, the PDF must be of the form

$$f(x; \theta, \Phi', p) = \frac{1}{\theta}\left(p f_1(\frac{x}{\theta}; \Phi'_1) + (1-p) f_2(\frac{x}{\theta}; \Phi'_2)\right)$$

where Φ', Φ'_1, and Φ'_2 denote the modified sets of nonscale parameters. One simple way to achieve this is to make $\theta_1 = \theta_2 = \theta$ and $\Phi' = \Phi$; that is, you simply equate the scale parameters of both components and keep the set of nonscale parameters unchanged. However, forcing the scale parameters to be equal in both components is restrictive, because the mixture cannot model potential differences in the scales of the two components. A better approach is to tie the scale parameters of the two components by a ratio such that $\theta_1 = \theta$ and $\theta_2 = \rho\theta$. If the ratio parameter ρ is estimated along with the other parameters, then the mixture distribution becomes flexible enough to model the variations across the scale parameters of individual components.

To summarize, the PDF and CDF are of the following form for the two-component mixture that has a scale parameter:

$$f(x; \theta, \rho, \Phi, p) = \frac{1}{\theta}\left(p f_1(\frac{x}{\theta}; \Phi_1) + (1-p) f_2(\frac{x}{\theta}; \rho, \Phi_2)\right)$$
$$F(x; \theta, \rho, \Phi, p) = p F_1(\frac{x}{\theta}; \Phi_1) + (1-p) F_2(\frac{x}{\theta}; \rho, \Phi_2)$$

This can be generalized to a mixture of K components by introducing the $K-1$ ratio parameters ρ_i that relate the scale parameters of each of the K components to the scale parameter θ of the mixture distribution as follows:

$$\theta_1 = \theta$$
$$\theta_i = \rho_i \theta;\ i \in [2, K]$$

In order to illustrate this approach, define a mixture of two lognormal distributions by using the following PDF function:

$$f(x; \mu, \sigma_1, p_2, \rho_2, \sigma_2) = \frac{(1-p_2)}{\sigma_1 x\sqrt{2\pi}} \exp\left(\frac{-(\log(x)-\mu)^2}{2\sigma_1^2}\right) + \frac{p_2}{\sigma_2 x\sqrt{2\pi}} \exp\left(\frac{-(\log(x)-\mu-\log(\rho_2))^2}{2\sigma_2^2}\right)$$

You can verify that μ serves as the log of the scale parameter θ ($\mu = \log(\theta)$). The following PROC FCMP steps encode this formulation in a distribution named SLOGNMIX2 for use with PROC SEVERITY:

```
/*- Define Mixture of 2 Lognormal Distributions with a Log-Scale Parameter -*/
proc fcmp library=sashelp.svrtdist outlib=work.sevexmpl.models;
   function slognmix2_description() $128;
      return ("Mixture of two lognormals with a log-scale parameter Mu");
   endsub;

   function slognmix2_scaletransform() $8;
      return ("LOG");
   endsub;

   function slognmix2_pdf(x, Mu, Sigma1, p2, Rho2, Sigma2);
      Mu1 = Mu;
      Mu2 = Mu + log(Rho2);
      pdf1 = logn_pdf(x, Mu1, Sigma1);
      pdf2 = logn_pdf(x, Mu2, Sigma2);
      return ((1-p2)*pdf1 + p2*pdf2);
   endsub;

   function slognmix2_cdf(x, Mu, Sigma1, p2, Rho2, Sigma2);
      Mu1 = Mu;
      Mu2 = Mu + log(Rho2);
      cdf1 = logn_cdf(x, Mu1, Sigma1);
      cdf2 = logn_cdf(x, Mu2, Sigma2);
      return ((1-p2)*cdf1 + p2*cdf2);
   endsub;

   subroutine slognmix2_parminit(dim, x[*], nx[*], F[*], Ftype,
                                 Mu, Sigma1, p2, Rho2, Sigma2);
      outargs Mu, Sigma1, p2, Rho2, Sigma2;
      array m[1] / nosymbols;
      p2 = 0.5;
      Rho2 = 0.5;
      median = svrtutil_percentile(0.5, dim, x, F, Ftype);
      Mu = log(2*median/1.5);
      call svrtutil_rawmoments(dim, x, nx, 1, m);
      lm1 = log(m[1]);

      /* Search Rho2 that makes log(sample mean) > Mu */
      do while (lm1 <= Mu and Rho2 < 1);
         Rho2 = Rho2 + 0.01;
         Mu = log(2*median/(1+Rho2));
      end;
      if (Rho2 >= 1) then
         /* If Mu cannot be decreased enough to make it less
            than log(sample mean), then revert to Rho2=0.5.
            That will set Sigma1 and possibly Sigma2 to missing.
            PROC SEVERITY replaces missing initial values with 0.001. */
         Mu = log(2*median/1.5);

      Sigma1 = sqrt(2.0*(log(m[1])-Mu));
      Sigma2 = sqrt(2.0*(log(m[1])-Mu-log(Rho2)));
   endsub;
```

```
   subroutine slognmix2_lowerbounds(Mu, Sigma1, p2, Rho2, Sigma2);
      outargs Mu, Sigma1, p2, Rho2, Sigma2;
      Mu = .; /* Mu has no lower bound */
      Sigma1 = 0; /* Sigma1 > 0 */
      p2 = 0; /* p2 > 0 */
      Rho2 = 0; /* Rho2 > 0 */
      Sigma2 = 0; /* Sigma2 > 0 */
   endsub;

   subroutine slognmix2_upperbounds(Mu, Sigma1, p2, Rho2, Sigma2);
      outargs Mu, Sigma1, p2, Rho2, Sigma2;
      Mu = .; /* Mu has no upper bound */
      Sigma1 = .; /* Sigma1 has no upper bound */
      p2 = 1; /* p2 < 1 */
      Rho2 = 1; /* Rho2 < 1 */
      Sigma2 = .; /* Sigma2 has no upper bound */
   endsub;
quit;
```

As shown in previous examples, an important aspect of defining a distribution for use with PROC SEVERITY is the definition of the PARMINIT subroutine that initializes the parameters. For mixture distributions, in general, the parameter initialization is a nontrivial task. For a two-component mixture, some simplifying assumptions make the problem easier to handle. For the initialization of SLOGNMIX2, the initial values of p_2 and ρ_2 are fixed at 0.5, and the following two simplifying assumptions are made:

- The median of the mixture is the average of the medians of the two components:

  $$F^{-1}(0.5) = (\exp(\mu_1) + \exp(\mu_2))/2 = \exp(\mu)(1 + \rho_2)/2$$

  Solution of this equation yields the value of μ in terms of ρ_2 and the sample median.

- Each component has the same mean, which implies the following:

  $$\exp(\mu + \sigma_1^2/2) = \exp(\mu + \log(\rho_2) + \sigma_2^2/2)$$

  If X_i represents the random variable of component distribution i and X represents the random variable of the mixture distribution, then the following equation holds for the raw moment of any order k:

  $$E[X^k] = \sum_{i=1}^{K} p_i E[X_i^k]$$

  This, in conjunction with the assumption on component means, leads to the equations

  $$\log(m_1) = \mu + \frac{\sigma_1^2}{2} \qquad \log(m_1) = \mu + \log(\rho_2) + \frac{\sigma_2^2}{2}$$

  where m_1 denotes the first raw moment of the sample. Solving these equations leads to the following values of σ_1 and σ_2:

  $$\sigma_1^2 = 2(\log(m_1) - \mu) \qquad \sigma_2^2 = 2(\log(m_1) - \mu - \log(\rho_2))$$

  Note that σ_1 has a valid value only if $\log(m_1) > \mu$. Among the many possible methods of ensuring this condition, the SLOGNMIX2_PARMINIT subroutine uses the method of doing a linear search over ρ_2.

Even when the preceding assumptions are not true for a given problem, they produce reasonable initial values to help guide the nonlinear optimizer to an acceptable optimum if the mixture of two lognormal distributions is indeed a good fit for your input data. This is illustrated by the results of the following steps that fit the SLOGNMIX2 distribution to simulated data, which have different means for the two components (12.18 and 22.76, respectively), and the median of the sample (15.94) is not equal to the average of the medians of the two components (7.39 and 20.09, respectively):

```
/*-------- Simulate a lognormal mixture sample ----------*/
data testlognmix(keep=y);
   call streaminit(12345);
   Mu1 = 2;
   Sigma1 = 1;
   i = 0;
   do j=1 to 2000;
      y = exp(Mu1) * rand('LOGNORMAL')**Sigma1;
      output;
   end;
   Mu2 = 3;
   Sigma2 = 0.5;
   do j=1 to 3000;
      y = exp(Mu2) * rand('LOGNORMAL')**Sigma2;
      output;
   end;
run;

/*-- Fit and compare scale regression models with 2-component  --*/
/*-- lognormal mixture and the standard lognormal distribution --*/
options cmplib=(work.sevexmpl);

proc severity data=testlognmix print=all plots(histogram kernel)=all;
   loss y;
   dist slognmix2 logn;
run;
```

The comparison of the fit statistics of SLOGNMIX2 and LOGN, as shown in Output 30.7.1, confirms that the two-component mixture is certainly a better fit to these data than the single lognormal distribution.

Output 30.7.1 Comparison of Fitting One versus Two Lognormal Components to Mixture Data

All Fit Statistics

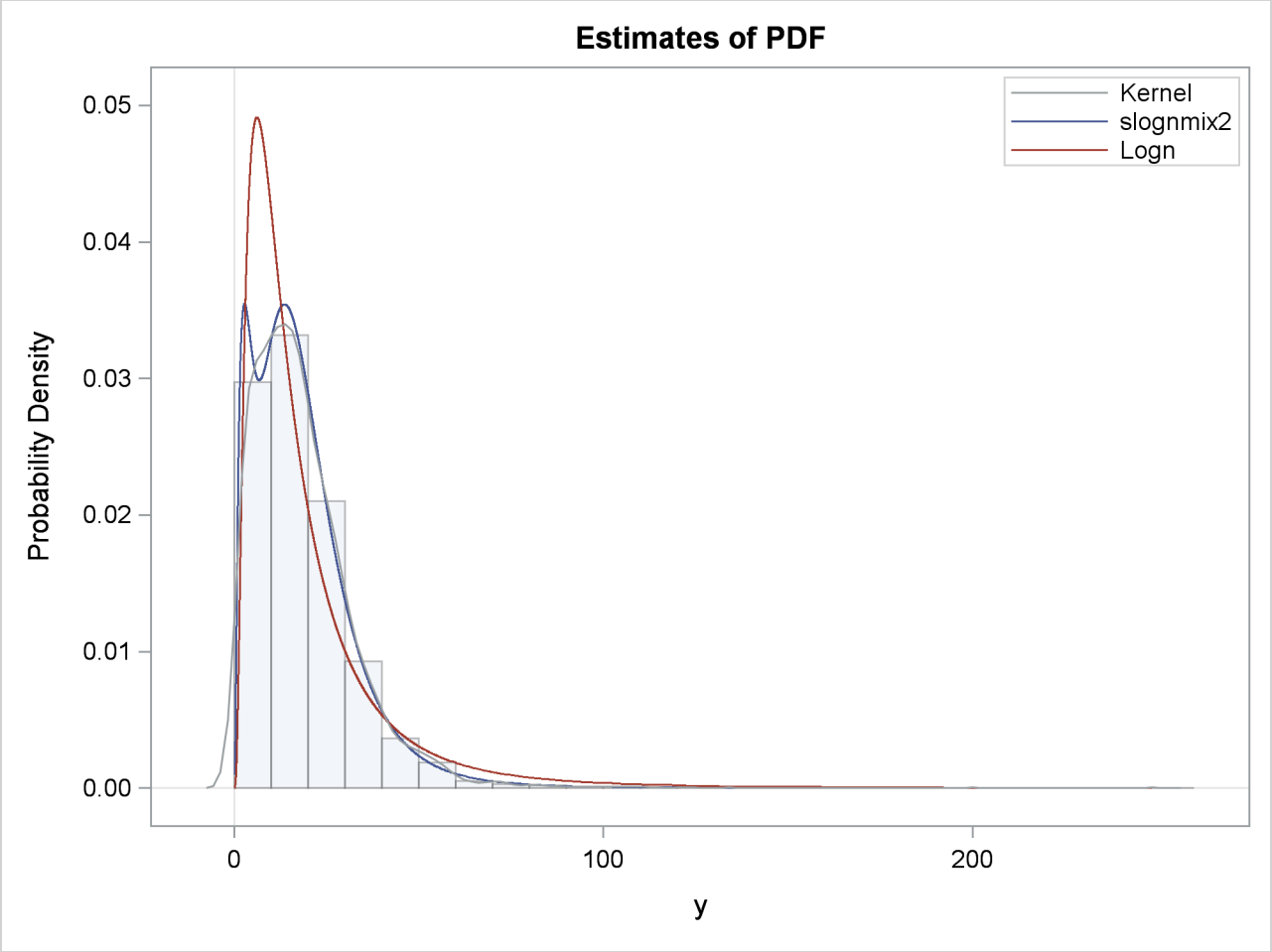
Distribution	-2 Log Likelihood		AIC		AICC		BIC		KS		AD		CvM	
slognmix2	38343	*	38353	*	38353	*	38386	*	0.52221	*	0.19843	*	0.02728	*
Logn	39073		39077		39077		39090		5.86522		66.93414		11.72703	

Note: The asterisk (*) marks the best model according to each column's criterion.

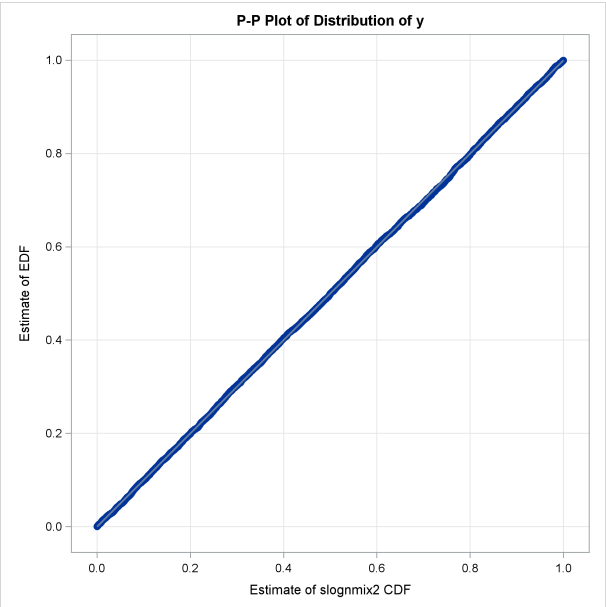
The comparative plot of probability densities in Output 30.7.2 shows that the density function of the mixture distribution is bimodal. In fact, one of the key motivations for using mixture distributions is to find better-fitting models for multimodal data.

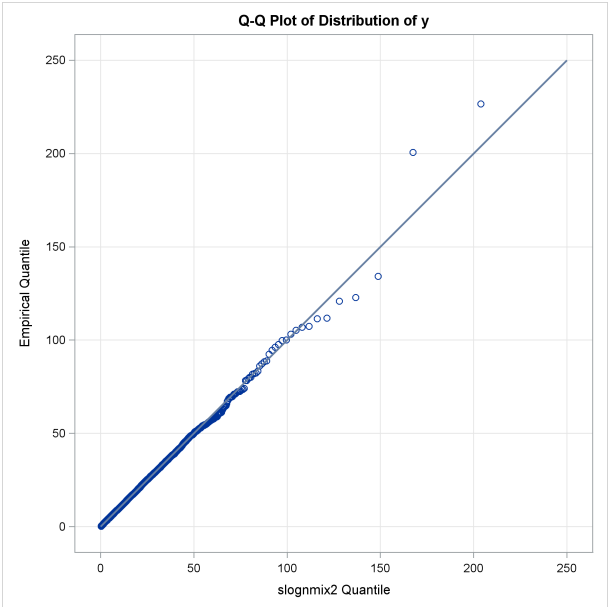
The P-P and Q-Q plots in Output 30.7.3 visually confirm that SLOGNMIX2 fits the data very well.

Output 30.7.2 Comparison of PDF Estimates of the Fitted Models



Output 30.7.3 P-P and Q-Q Plots to Evaluate SLOGNMIX2 Fit





The detailed results for the SLOGNMIX2 distribution are shown in Output 30.7.4. According to the "Initial Parameter Values and Bounds" table, the initial value of ρ_2 is not 0.5, indicating that a linear search was conducted to ensure $\log(m_1) > \mu$.

Output 30.7.4 Detailed Estimation Results for the SLOGNMIX2 Distribution

The SEVERITY Procedure
slognmix2 Distribution

Distribution Information	
Name	slognmix2
Description	Mixture of two lognormals with a log-scale parameter Mu
Distribution Parameters	5

Initial Parameter Values and Bounds

Parameter	Initial Value	Lower Bound	Upper Bound
Mu	2.92006	-Infty	Infty
Sigma1	0.10455	1.05367E-8	Infty
P2	0.50000	1.05367E-8	1.00000
Rho2	0.72000	1.05367E-8	1.00000
Sigma2	0.81728	1.05367E-8	Infty

Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

Optimization Summary

Optimization Technique	Trust Region
Iterations	7
Function Calls	18
Log Likelihood	-19171.5

Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|
Mu	3.00922	0.01554	193.68	<.0001
Sigma1	0.49516	0.01451	34.13	<.0001
P2	0.40619	0.02600	15.62	<.0001
Rho2	0.37212	0.02038	18.26	<.0001
Sigma2	1.00019	0.02124	47.09	<.0001

By using the relationship that $\mu_2 = \mu + \log(\rho_2)$, you can see that the final parameter estimates are indeed close to the true parameter values that were used to simulate the input sample.

Example 30.8: Predicting Mean and Value-at-Risk by Using Scoring Functions

If you work in the risk management department of an insurance company or a bank, then one of your primary applications of severity loss distribution models is to predict the value-at-risk (VaR) so that there is a very low probability of experiencing a loss value that is greater than the VaR. The probability level at which VaR is measured is prescribed by industry regulations such as Basel III and Solvency II. The VaR level is usually specified in terms of $(1 - \alpha)$, where $\alpha \in (0, 1)$ is the probability that a loss value exceeds the VaR. Typical VaR levels are 0.95, 0.975, and 0.995.

In addition to predicting the VaR, which is regarded as an estimate of the worst-case loss, businesses are often interested in predicting the average loss by estimating either the mean or median of the distribution.

The estimation of the mean and VaR combined with the scale regression model is very potent tool for analyzing worst-case and average losses for various scenarios. For example, if the regressors that are used in a scale regression model represent some key macroeconomic and operational indicators, which are widely referred to as key risk indicators (KRIs), then you can analyze the VaR and mean loss estimates over various values for the KRIs to get a more comprehensive picture of the risk profile of your organization across various market and internal conditions.

This example illustrates the use of scoring functions to simplify the process of predicting the mean and VaR of scale regression models.

To compute the mean, you need to ensure that the function to compute the mean of a distribution is available in the function library. If you define and fit your own distribution and you want to compute its mean, then you need to use the FCMP procedure to define that function and you need to use the CMPLIB= system option to specify the location of that function. For your convenience, the *dist*_MEAN function (which computes the mean of the *dist* distribution) is already defined in the Sashelp.Svrtdist library for each of the 10 predefined distributions. The following statements display the definitions of MEAN functions of all distributions. Note that the MEAN functions for the Burr, Pareto, and generalized Pareto distributions check the existence of the first moment for specified parameter values.

```
/*--------- Define distribution functions that compute the mean ----------*/
proc fcmp library=sashelp.svrtdist outlib=work.means.scalemod;
   function BURR_MEAN(x, Theta, Alpha, Gamma);
      if not(Alpha * Gamma > 1) then
         return (.); /* first moment does not exist */
      return (Theta*gamma(1 + 1/Gamma)*gamma(Alpha - 1/Gamma)/gamma(Alpha));
   endsub;
   function EXP_MEAN(x, Theta);
      return (Theta);
   endsub;
   function GAMMA_MEAN(x, Theta, Alpha);
      return (Theta*Alpha);
   endsub;
   function GPD_MEAN(x, Theta, Xi);
      if not(Xi < 1) then
         return (.); /* first moment does not exist */
      return (Theta/(1 - Xi));
   endsub;
   function IGAUSS_MEAN(x, Theta, Alpha);
      return (Theta);
   endsub;
```

```
   function LOGN_MEAN(x, Mu, Sigma);
      return (exp(Mu + Sigma*Sigma/2.0));
   endsub;


   function PARETO_MEAN(x, Theta, Alpha);
      if not(Alpha > 1) then
         return (.); /* first moment does not exist */
      return (Theta/(Alpha - 1));
   endsub;
   function STWEEDIE_MEAN(x, Theta, Lambda, P);
      return (Theta* Lambda * (2 - P) / (P - 1));
   endsub;
   function TWEEDIE_MEAN(x, P, Mu, Phi);
      return (Mu);
   endsub;
   function WEIBULL_MEAN(x, Theta, Tau);
      return (Theta*gamma(1 + 1/Tau));
   endsub;
quit;
```

For your further convenience, the *dist*_QUANTILE function (which computes the quantile of the *dist* distribution) is also defined in the Sashelp.Svrtdist library for each of the 10 predefined distributions. Because the MEAN and QUANTILE functions satisfy the definition of a distribution function as described in the section "Formal Description" on page 2167, you can submit the following PROC SEVERITY step to fit all regression-friendly predefined distributions and generate the scoring functions for the MEAN, QUANTILE, and other distribution functions:

```
/*----- Fit all distributions and generate scoring functions ------*/
proc severity data=test_sev8 outest=est print=all plots=none;
   loss y;
   scalemodel x1-x5;
   dist _predefined_ stweedie;
   outscorelib outlib=scorefuncs commonpackage;
run;
```

The SAS statements that simulate the sample in the Work.Test_sev8 data set are available in the PROC SEVERITY sample program *sevex08.sas*. The OUTLIB= option in the OUTSCORELIB statement requests that the scoring functions be written to the Work.Scorefuncs library, and the COMMONPACKAGE option in the OUTSCORELIB statement requests that all the functions be written to the same package. Upon completion, PROC SEVERITY sets the CMPLIB system option to the following value:

```
(sashelp.svrtdist work.scorefuncs)
```

The "All Fit Statistics" table in Output 30.8.1 shows that the lognormal distribution's scale model is the best and the inverse Gaussian's scale model is a close second according to the likelihood-based statistics.

Output 30.8.1 Comparison of Fitted Scale Models for Mean and VaR Illustration

The SEVERITY Procedure

All Fit Statistics

Distribution	-2 Log Likelihood		AIC		AICC		BIC		KS		AD
stweedie	460.65755		476.65755		476.95083		510.37441		10.44548		64571	
Burr	451.42238		467.42238		467.71565		501.13924		10.32782		42254	
Exp	1515		1527		1527		1552		8.85827		29917	
Gamma	448.28222		462.28222		462.50986		491.78448		10.42272		63712	
Igauss	444.44512		458.44512		458.67276		487.94738		10.33028		83195	
Logn	444.43670	*	458.43670	*	458.66434	*	487.93895	*	10.37035		68631	
Pareto	1515		1529		1529		1559		8.85775	*	29916	*
Gpd	1515		1529		1529		1559		8.85827		29917	
Weibull	527.28676		541.28676		541.51440		570.78902		10.48084		72814	

Note: The asterisk (*) marks the best model according to each column's criterion.

All Fit Statistics

Distribution	CvM	
stweedie	37.07705	
Burr	37.19808	
Exp	23.98267	
Gamma	37.19450	
Igauss	37.30880	
Logn	37.18553	
Pareto	23.98149	*
Gpd	23.98267	
Weibull	36.36039	

Note: The asterisk (*) marks the best model according to each column's criterion.

You can examine the scoring functions that are written to the Work.Scorefuncs library by using the FCMP Function Editor, which is available in the Display Manager session of Base SAS when you select **Solutions→Analysis** from the main menu. For example, PROC SEVERITY automatically generates and submits the following PROC FCMP statements to define the scoring functions SEV_MEAN_LOGN and SEV_QUANTILE_IGAUSS:

```
proc fcmp library=(sashelp.svrtdist) outlib=work.scorefuncs.sevfit;
   function SEV_MEAN_LOGN(y, x{*});
      _logscale_=0;
      _logscale_ = _logscale_ + ( 7.64722278930350E-01 * x{1});
      _logscale_ = _logscale_ + ( 2.99209540369860E+00 * x{2});
      _logscale_ = _logscale_ + (-1.00788916253430E+00 * x{3});
      _logscale_ = _logscale_ + ( 2.58883602184890E-01 * x{4});
      _logscale_ = _logscale_ + ( 5.00927479793970E+00 * x{5});
      _logscale_ = _logscale_ + ( 9.95078833050690E-01);
      return (LOGN_MEAN(y, _logscale_,  2.31592981635590E-01));
   endsub;
```

```
   function SEV_QUANTILE_IGAUSS(y, x{*});
      _logscale_=0;
      _logscale_ = _logscale_ + ( 7.64581738373520E-01 * x{1});
      _logscale_ = _logscale_ + ( 2.99159055015310E+00 * x{2});
      _logscale_ = _logscale_ + (-1.00793496641510E+00 * x{3});
      _logscale_ = _logscale_ + ( 2.58870460543840E-01 * x{4});
      _logscale_ = _logscale_ + ( 5.00996884646730E+00 * x{5});
      _scale_ =  2.77854870591020E+00 * exp(_logscale_);
      return (IGAUSS_QUANTILE(y, _scale_,  1.81511227238720E+01));
   endsub;
quit;
```

PROC SEVERITY detects all the distribution functions that are available in the current CMPLIB= search path (which always includes the Sashelp.Svrtdist library) for the distributions that you specify in the DIST statement, and it creates the corresponding scoring functions. You can define any distribution function that has the desired signature to compute an estimate of your choice, include its library in the CMPLIB= system option, and then specify the OUTSCORELIB statement to generate the corresponding scoring functions. Specifying the COMMONPACKAGE option in the OUTSCORELIB statement causes the name of the scoring function to take the form SEV_*function-suffix_dist*. If you do not specify the COMMONPACKAGE option, PROC SEVERITY creates a scoring function named SEV_*function-suffix* in a package named *dist*. You can invoke functions from a specific package only inside the FCMP procedure. If you want to invoke the scoring functions from a DATA step, then it is recommended that you specify the COMMONPACKAGE option when you specify multiple distributions in the DIST statement.

To illustrate the use of scoring functions, let Work.Reginput contain the scoring data, where the values of regressors in each observation define one scenario. Scoring functions make it very easy to compute the mean and VaR of each distribution's scale model for each of the scenarios, as the following steps illustrate for the lognormal and inverse Gaussian distributions:

```
/*--- Set VaR level ---*/
%let varLevel=0.975;

/*--- Compute scores (mean and var) for the       ---
  --- scoring data by using the scoring functions ---*/
data scores;
   array x{*} x1-x5;
   set reginput;

   igauss_mean = sev_mean_igauss(., x);
   igauss_var  = sev_quantile_igauss(&varLevel, x);
   logn_mean   = sev_mean_logn(., x);
   logn_var    = sev_quantile_logn(&varLevel, x);
run;
```

The preceding steps use a VaR level of 97.5%.

The following DATA step accomplishes the same task by reading the parameter estimates that were written to the Work.Est data set by the previous PROC SEVERITY step:

```
/*--- Compute scores (mean and var) for the       ---
  --- scoring data by using the OUTEST= data set  ---*/
data scoresWithOutest(keep=x1-x5 igauss_mean igauss_var logn_mean logn_var);
   array _x_{*} x1-x5;
   array _xparmIgauss_{5} _temporary_;
```

```
   array _xparmLogn_{5} _temporary_;
   retain _Theta0_ Alpha0;
   retain _Mu0_ Sigma0;
   *--- read parameter estimates for igauss and logn models ---*;
   if (_n_ = 1) then do;
      set est(where=(upcase(_MODEL_)='IGAUSS' and _TYPE_='EST'));
      _Theta0_ = Theta; Alpha0 = Alpha;
      do _i_=1 to dim(_x_);
         if (_x_(_i_) = .R) then _xparmIgauss_(_i_) = 0;
         else _xparmIgauss_(_i_) = _x_(_i_);
      end;
      set est(where=(upcase(_MODEL_)='LOGN' and _TYPE_='EST'));
      _Mu0_ = Mu; Sigma0 = Sigma;
      do _i_=1 to dim(_x_);
         if (_x_(_i_) = .R) then _xparmLogn_(_i_) = 0;
         else _xparmLogn_(_i_) = _x_(_i_);
      end;
   end;
   set reginput;

   *---  predict mean and VaR for inverse Gaussian  ---*;
   * first compute X'*beta for inverse Gaussian *;
   _xbeta_ = 0.0;
   do _i_ = 1 to dim(_x_);
      _xbeta_ = _xbeta_ + _xparmIgauss_(_i_) * _x_(_i_);
   end;
   * now compute scale for inverse Gaussian *;
   _SCALE_ = _Theta0_ * exp(_xbeta_);
   igauss_mean = igauss_mean(., _SCALE_, Alpha0);
   igauss_var = igauss_quantile(&varLevel, _SCALE_, Alpha0);
   *---  predict mean and VaR for lognormal         ---*;
   * first compute X'*beta for lognormal*;
   _xbeta_ = 0.0;
   do _i_ = 1 to dim(_x_);
      _xbeta_ = _xbeta_ + _xparmLogn_(_i_) * _x_(_i_);
   end;
   * now compute Mu=log(scale) for lognormal *;
   _MU_ = _Mu0_ + _xbeta_;
   logn_mean = logn_mean(., _MU_, Sigma0);
   logn_var = logn_quantile(&varLevel, _MU_, Sigma0);
run;
```

The "Values Comparison Summary" table in Output 30.8.2 shows that the difference between the estimates that are produced by both methods is within the acceptable machine precision. However, the comparison of the DATA step complexity of each method clearly shows that the method that uses the scoring functions is much easier because it saves a lot of programming effort. Further, new distribution functions, such as the *dist*_MEAN functions that are illustrated here, are automatically discovered and converted to scoring functions by PROC SEVERITY. That enables you to focus your efforts on writing the distribution function that computes your desired score, which needs to be done only once. Then, you can create and use the corresponding scoring functions multiple times with much less effort.

Output 30.8.2 Comparison of Mean and VaR Estimates of Two Scoring Methods

```
                    Observation Summary

                Observation      Base  Compare

                First Obs           1        1
                Last  Obs          10       10

Number of Observations in Common: 10.
Total Number of Observations Read from WORK.SCORESWITHOUTEST: 10.
Total Number of Observations Read from WORK.SCORES: 10.

Number of Observations with Some Compared Variables Unequal: 0.
Number of Observations with All Compared Variables Equal: 10.


                  Values Comparison Summary

 Number of Variables Compared with All Observations Equal: 9.
 Number of Variables Compared with Some Observations Unequal: 0.
 Total Number of Values which Compare Unequal: 0.
 Total Number of Values not EXACTLY Equal: 40.
 Maximum Difference Criterion Value: 2.0963E-13.
```

Example 30.9: Scale Regression with Rich Regression Effects

This example illustrates the use of regression effects that include CLASS variables and interaction effects.

Consider that you, as an actuary at an automobile insurance company, want to evaluate the effect of certain external factors on the distribution of the severity of the losses that your policyholders incur. Such analysis can help you determine the relative differences in premiums that you should charge to policyholders who have different characteristics. Assume that when you collect and record the information about each claim, you also collect and record some key characteristics of the policyholder and the vehicle that is involved in the claim. This example focuses on the following five factors: type of car, safety rating of the car, gender of the policyholder, education level of the policyholder, and annual household income of the policyholder (which can be thought of as a proxy for the luxury level of the car). Let these regressors be recorded in the variables CarType (1: sedan, 2: sport utility vehicle), CarSafety (scaled to be between 0 and 1, the safest being 1), Gender (1: female, 2: male), Education (1: high school graduate, 2: college graduate, 3: advanced degree holder), and Income (scaled by a factor of 1/100,000), respectively. Let the historical data about the severity of each loss be recorded in the LossAmount variable of the Work.Losses data set. Let the data set also contain two additional variables, Deductible and Limit, that record the deductible and ground-up loss limit provisions, respectively, of the insurance policy that the policyholder has. The limit on ground-up loss is usually derived from the payment limit that a typical insurance policy states. Deductible serves as the left-truncation variable, and Limit serves as the right-censoring variable. The SAS statements that simulate an example of the Work.Losses data set are available in the PROC SEVERITY sample program *sevex09.sas*.

The variables CarType, Education, and Gender each contain a known, finite set of discrete values. By specifying such variables as classification variables, you can separately identify the effect of each level of the variable on the severity distribution. For example, you might be interested in finding out how the magnitude of loss for a sport utility vehicle (SUV) differs from that for a sedan. This is an example of a main effect. You might also want to evaluate how the distribution of losses that are incurred by a policyholder with a college degree who drives a SUV differs from that of a policyholder with an advanced degree who drives a sedan. This is an example of an interaction effect. You can include various such types of effects in the scale regression model. For more information about the effect types, see the section "Specification and Parameterization of Model Effects" on page 2131. Analyzing such a rich set of regression effects can help you make more accurate predictions about the losses that a new applicant with certain characteristics might incur when he or she requests insurance for a specific vehicle, which can further help you with ratemaking decisions.

The following PROC SEVERITY step fits the scale regression model with a lognormal distribution to data in the Work.Losses data set, and stores the model and parameter estimate information in the Work.EstStore item store:

```
/* Fit scale regression model with different types of regression effects */
proc severity data=losses outstore=eststore
     print=all plots=none;
   loss lossAmount / lt=deductible rc=limit;
   class carType gender education;
   scalemodel carType gender carSafety income education*carType
              income*gender carSafety*income;
   dist logn;
run;
```

The SCALEMODEL statement in the preceding PROC SEVERITY step includes two main effects (carType and gender), two singleton continuous effects (carSafety and income), one interaction effect (education*carType), one continuous-by-class effect (income*gender), and one polynomial continuous effect (carSafety*income). For more information about effect types, see Table 30.9, "GLM Parameterization of Classification Variables and Effects," on page 2134.

When you specify a CLASS statement, it is recommended that you observe the "Class Level Information" table. For this example, the table is shown in Output 30.9.1. Note that if you specify BY-group processing, then the class level information might change from one BY group to the next, potentially resulting in a different parameterization for each BY group.

Output 30.9.1 Class Level Information Table

The SEVERITY Procedure

Class Level Information

Class	Levels	Values
carType	2	SUV Sedan
gender	2	Female Male
education	3	AdvancedDegree College High School

The regression modeling results for the lognormal distribution are shown in Output 30.9.2. The "Initial Parameter Values and Bounds" table is important especially because the preceding PROC SEVERITY step uses the default GLM parameterization, which is a singular parameterization—that is, it results in some

redundant parameters. As shown in the table, the redundant parameters correspond to the last level of each classification variable; this correspondence is a defining characteristic of a GLM parameterization. An alternative would be to use the reference parameterization by specifying the PARAM=REFERENCE option in the CLASS statement, which does not generate redundant parameters for effects that contain CLASS variables and enables you to specify a reference level for each CLASS variable.

Output 30.9.2 Initial Values for the Scale Regression Model with Class and Interaction Effects

Initial Parameter Values and Bounds

Parameter	Initial Value	Lower Bound	Upper Bound
Mu	4.88526	-709.78271	709.78271
Sigma	0.51283	1.05367E-8	Infty
carType SUV	0.56953	-709.78271	709.78271
carType Sedan	Redundant		
gender Female	0.41154	-709.78271	709.78271
gender Male	Redundant		
carSafety	-0.72742	-709.78271	709.78271
income	-0.33216	-709.78271	709.78271
carType*education SUV AdvancedDegree	0.31686	-709.78271	709.78271
carType*education SUV College	0.66361	-709.78271	709.78271
carType*education SUV High School	Redundant		
carType*education Sedan AdvancedDegree	-0.47841	-709.78271	709.78271
carType*education Sedan College	-0.25968	-709.78271	709.78271
carType*education Sedan High School	Redundant		
income*gender Female	-0.02112	-709.78271	709.78271
income*gender Male	Redundant		
carSafety*income	0.13084	-709.78271	709.78271

The convergence and optimization summary information in Output 30.9.3 indicates that the scale regression model for the lognormal distribution has converged with the default optimization technique in five iterations.

Output 30.9.3 Optimization Summary for the Scale Regression Model with Class and Interaction Effects

Convergence Status
Convergence criterion (GCONV=1E-8) satisfied.

Optimization Summary	
Optimization Technique	Trust Region
Iterations	5
Function Calls	14
Log Likelihood	-8286.8

The "Parameter Estimates" table in Output 30.9.4 shows the distribution parameter estimates and estimates for various regression effects. You can use the estimates for effects that contain CLASS variables to infer the relative influence of various CLASS variable levels. For example, on average, the magnitude of losses that are incurred by the female drivers is $\exp(0.44145) \approx 1.56$ times greater than that of male drivers, and an SUV driver with an advanced degree incurs a loss that is on average $\exp(0.39393)/\exp(-0.35210) \approx 2.11$ times greater than the loss that a college-educated sedan driver incurs. Neither the continuous-by-class effect income*gender nor the polynomial continuous effect carSafety*income is significant in this example.

Output 30.9.4 Parameter Estimates for the Scale Regression with Class and Interaction Effects

Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|
Mu	5.08874	0.05768	88.23	<.0001
Sigma	0.55774	0.01119	49.86	<.0001
carType SUV	0.62459	0.04452	14.03	<.0001
gender Female	0.44145	0.04885	9.04	<.0001
carSafety	-0.82942	0.08371	-9.91	<.0001
income	-0.35212	0.07657	-4.60	<.0001
carType*education SUV AdvancedDegree	0.39393	0.07351	5.36	<.0001
carType*education SUV College	0.76532	0.05723	13.37	<.0001
carType*education Sedan AdvancedDegree	-0.61064	0.05387	-11.34	<.0001
carType*education Sedan College	-0.35210	0.03942	-8.93	<.0001
income*gender Female	-0.01486	0.06629	-0.22	0.8226
carSafety*income	0.07045	0.11447	0.62	0.5383

If you want to update the model when new claims data arrive, then you can potentially speed up the estimation process by specifying the OUTSTORE= item store that is created by the preceding PROC SEVERITY step as an INSTORE= item store in a new PROC SEVERITY step as follows:

```
/* Refit scale regression model on new data different types of regression effects */
proc severity data=withNewLosses instore=eststore
     print=all plots=all;
   loss lossAmount / lt=deductible rc=limit;
   class carType gender education;
   scalemodel carType gender carSafety income education*carType
              income*gender carSafety*income;
   dist logn;
run;
```

PROC SEVERITY uses the parameter estimates in the INSTORE= item store to initialize the distribution and regression parameters.

References

D'Agostino, R. B., and Stephens, M., eds. (1986). *Goodness-of-Fit Techniques*. New York: Marcel Dekker.

Danielsson, J., de Haan, L., Peng, L., and de Vries, C. G. (2001). "Using a Bootstrap Method to Choose the Sample Fraction in Tail Index Estimation." *Journal of Multivariate Analysis* 76:226–248.

Dunn, P. K., and Smyth, G. K. (2005). "Series Evaluation of Tweedie Exponential Dispersion Model Densities." *Statistics and Computing* 15:267–280.

Frydman, H. (1994). "A Note on Nonparametric Estimation of the Distribution Function from Interval-Censored and Truncated Observations." *Journal of the Royal Statistical Society, Series B* 56:71–74.

Gentleman, R., and Geyer, C. J. (1994). "Maximum Likelihood for Interval Censored Data: Consistency and Computation." *Biometrika* 81:618–623.

Greenwood, M. (1926). "The Natural Duration of Cancer." In *Reports of Public Health and Related Subjects*, vol. 33, 1–26. London: Her Majesty's Stationery Office.

Hill, B. M. (1975). "A Simple General Approach to Inference about the Tail of a Distribution." *Annals of Statistics* 3:1163–1173.

Jørgensen, B. (1987). "Exponential Dispersion Models (with Discussion)." *Journal of the Royal Statistical Society, Series B* 49:127–162.

Kaplan, E. L., and Meier, P. (1958). "Nonparametric Estimation from Incomplete Observations." *Journal of the American Statistical Association* 53:457–481.

Klein, J. P., and Moeschberger, M. L. (1997). *Survival Analysis: Techniques for Censored and Truncated Data*. New York: Springer-Verlag.

Klugman, S. A., Panjer, H. H., and Willmot, G. E. (1998). *Loss Models: From Data to Decisions*. New York: John Wiley & Sons.

Koziol, J. A., and Green, S. B. (1976). "A Cramér–von Mises Statistic for Randomly Censored Data." *Biometrika* 63:466–474.

Lai, T. L., and Ying, Z. (1991). "Estimating a Distribution Function with Truncated and Censored Data." *Annals of Statistics* 19:417–442.

Lynden-Bell, D. (1971). "A Method of Allowing for Known Observational Selection in Small Samples Applied to 3CR Quasars." *Monthly Notices of the Royal Astronomical Society* 155:95–118.

Searle, S. R. (1971). *Linear Models*. New York: John Wiley & Sons.

Turnbull, B. W. (1976). "The Empirical Distribution Function with Arbitrarily Grouped, Censored, and Truncated Data." *Journal of the Royal Statistical Society, Series B* 38:290–295.

Tweedie, M. C. K. (1984). "An Index Which Distinguishes between Some Important Exponential Families." In *Statistics: Applications and New Directions—Proceedings of the Indian Statistical Institute Golden Jubilee International Conference*, edited by J. K. Ghosh, and J. Roy, 579–604. Calcutta: Indian Statistical Institute.

Chapter 31
The SIMILARITY Procedure

Contents

Overview: SIMILARITY Procedure **2236**
Getting Started: SIMILARITY Procedure **2238**
Syntax: SIMILARITY Procedure **2240**
Functional Summary 2240
PROC SIMILARITY Statement 2241
BY Statement 2244
FCMPOPT Statement 2245
ID Statement 2245
INPUT Statement 2248
TARGET Statement 2250
Details: SIMILARITY Procedure **2255**
Accumulation 2256
Missing Value Interpretation 2258
Zero Value Interpretation 2258
Time Series Transformation 2258
Time Series Differencing 2259
Time Series Missing Value Trimming 2259
Time Series Descriptive Statistics 2260
Input and Target Sequences 2260
Sliding Sequences 2260
Time Warping 2260
Sequence Normalization 2260
Sequence Scaling 2261
Similarity Measures 2261
User-Defined Functions and Subroutines 2261
Output Data Sets 2269
OUT= Data Set 2269
OUTMEASURE= Data Set 2270
OUTPATH= Data Set 2270
OUTSEQUENCE= Data Set 2271
OUTSUM= Data Set 2272
STATUS Variable Values 2273
Printed Output 2273
ODS Table Names 2274
ODS Graphics 2275
Examples: SIMILARITY Procedure **2277**

Example 31.1: Accumulating Transactional Data into Time Series Data 2277
Example 31.2: Similarity Analysis 2279
Example 31.3: Sliding Similarity Analysis 2295
Example 31.4: Searching for Historical Analogies 2298
Example 31.5: Clustering Time Series 2300
References **2300**

Overview: SIMILARITY Procedure

The SIMILARITY procedure computes similarity measures associated with time-stamped data, time series, and other sequentially ordered numeric data. PROC SIMILARITY computes similarity measures for time-stamped transactional data (transactions) with respect to time by accumulating the data into a time series format, and it computes similarity measures for sequentially ordered numeric data (sequences) by respecting the ordering of the data.

Given two ordered numeric sequences (input and target), a similarity measure is a metric that measures the distance between the input and target sequences while taking into account the ordering of the data. The SIMILARITY procedure computes similarity measures between an input sequence and a target sequence, in addition to similarity measures that "slide" the target sequence with respect to the input sequence. The "slides" can be by observation index (sliding-sequence similarity measures) or by seasonal index (seasonal-sliding-sequence similarity measures).

In order to compare the raw input and the raw target time-stamped data, the raw data must be accumulated to a time series format. After the input and target time series are formed, the two accumulated time series can be compared as two ordered numeric sequences.

For raw time-stamped data, after the transactional data are accumulated to form time series and any missing values are interpreted, each accumulated time series can be functionally transformed, if desired. Transformations are useful when you want to stabilize the time series before computing the similarity measures. Transformations performed by the SIMILARITY procedure include the following:

- log (LOG)
- square-root (SQRT)
- logistic (LOGISTIC)
- Box-Cox (BOXCOX)
- user-defined transformations

Each time series can be transformed further by using simple differencing or seasonal differencing or both. Additional time series transformations can be performed by using various time series transformation and analysis techniques provided by this procedure or other SAS/ETS procedures.

After optionally transforming each time series, the accumulated and transformed time series can be stored in an output data set (OUT= data set).

After optional accumulation and transformation, each of these time series are the "working series," which can now be analyzed as sequences of numeric data. Each of these sequences can be a target sequence, an input sequence, or both a target and an input sequence. Throughout the remainder of this chapter, the term "original sequence" applies to both the original input and target sequence. The term "working sequence" applies to a version of both the original input and target sequence under investigation.

Each original sequence can be normalized prior to similarity analysis. Normalizations are useful when you want to compare the "shape" or "profile" of the time series. Normalizations performed by the SIMILARITY procedure include the following:

- standard (STANDARD)
- absolute (ABSOLUTE)
- user-defined normalizations

After each original sequence is optionally normalized, each working input sequence can be scaled to the target sequence prior to similarity analysis. Scaling is useful when you want to compare the input sequence to the target sequence while discounting the variation of the target sequence. Input sequence scaling performed by the SIMILARITY procedure include the following:

- standard (STANDARD)
- absolute (ABSOLUTE)
- user-defined scaling

After the working input sequence is optionally scaled to the target sequence, similarity measures can be computed. Similarity measures computed by the SIMILARITY procedure include:

- squared deviation (SQRDEV)
- absolute deviation (ABSDEV)
- mean square deviation (MSQRDEV)
- mean absolute deviation (MABSDEV)
- user-defined similarity measures

In computing the similarity measure between two time series, tasks are needed for transforming time series, normalizing sequences, scaling sequences, and computing metrics or measures. The SIMILARITY procedure provides built-in routines to perform these tasks. The SIMILARITY procedure also enables you to extend the procedure with user-defined routines.

All results of the similarity analysis can be stored in output data sets, printed, or graphed using the Output Delivery System (ODS).

The SIMILARITY procedure can process large amounts of time-stamped transactional data, time series, or sequential data. Therefore, the analysis results are useful for large-scale time series analysis, analogous time series forecasting, new product forecasting, or time series (temporal) data mining.

The SAS/ETS EXPAND procedure can be used for frequency conversion and transformations of time series. The TIMESERIES procedure can be used for large-scale time series analysis. The SAS/STAT DISTANCE procedure can be used to compute various measures of distance, dissimilarity, or similarity between observations (rows) of a SAS data set.

Getting Started: SIMILARITY Procedure

This section outlines the use of the SIMILARITY procedure and gives a cursory description of some of the analysis techniques that can be performed on time-stamped transactional data, time series, or sequentially ordered numeric data.

Given an input data set that contains numerous transaction variables recorded over time at no specific frequency, the SIMILARITY procedure can form equally spaced input and target time series as follows:

```
PROC SIMILARITY DATA=<input-data-set>
                OUT=<output-data-set>
                OUTSUM=<summary-data-set>;
   ID <time-ID-variable> INTERVAL=<frequency>
                         ACCUMULATE=<statistic>;
   INPUT <input-time-stamp-variables>;
   TARGET <target-time-stamp-variables>;
RUN;
```

The SIMILARITY procedure forms time series from the input time-stamped transactional data. It can provide results in output data sets or in other output formats using the Output Delivery System (ODS). The examples in this section are more fully illustrated in the section "Examples: SIMILARITY Procedure" on page 2277.

Time-stamped transactional data are often recorded at no fixed interval. Analysts often want to use time series analysis techniques that require fixed-time intervals. Therefore, the transactional data must be accumulated to form a fixed-interval time series.

Suppose that a bank wants to analyze the transactions that are associated with each of its customers over time. Further, suppose that the data set WORK.TRANSACTIONS contains three variables that are related to the customer transactions (CUSTOMER, DATE, and WITHDRAWAL) and one variable that contains an example fraudulent behavior (FRAUD).

The following statements illustrate how to use the SIMILARITY procedure to accumulate time-stamped transactional data to form a daily time series based on the accumulated daily totals of each type of transaction (WITHDRAWALS and FRAUD):

```
proc similarity data=transactions out=timedata;
   by customer;
   id date interval=day accumulate=total;
   input withdrawals;
   target fraud;
run;
```

The OUT=TIMEDATA option specifies that the resulting time series data for each customer are to be stored in the data set WORK.TIMEDATA. The INTERVAL=DAY option specifies that the transactions are to be accumulated on a daily basis. The ACCUMULATE=TOTAL option specifies that the sum of the transactions are to be accumulated. After the transactional data are accumulated into a time series format, the time series data can be normalized so that the "shape" or "profile" is analyzed.

For example, the following statements build on the previous statements and demonstrate normalization of the accumulated time series:

```
proc similarity data=transactions out=timedata;
   by customer;
   id date interval=day accumulate=total;
   input withdrawals / NORMALIZE=STANDARD;
   target fraud      / NORMALIZE=STANDARD;
run;
```

The NORMALIZE=STANDARD option specifies that each accumulated time series observation is normalized by subtracting the mean and then dividing by the standard deviation of the accumulated time series. The WORK.TIMEDATA data set now contains the accumulated and normalized time series data for each customer.

After the transactional data are accumulated into a time series format and normalized to a mean of zero and standard deviation of one, similarity analysis can be performed on the accumulated and normalized time series.

For example, the following statements build on the previous statements and demonstrate similarity analysis of the accumulated and normalized time series:

```
proc similarity data=transactions
                out=timedata OUTSUM=SUMMARY;
   by customer;
   id date interval=day accumulate=total;
   input withdrawals / normalize=standard;
   target fraud      / normalize=standard MEASURE=MABSDEV;
run;
```

The MEASURE=MABSDEV option specifies the accumulated and normalized time series data that are associated with the variables WITHDRAWALS and FRAUD are to be compared by using mean absolute deviation. The OUTSUM=SUMMARY option specifies that the similarity analysis summary for each customer is to be stored in the data set WORK.SUMMARY.

Syntax: SIMILARITY Procedure

The following statements are used with the SIMILARITY procedure.

PROC SIMILARITY *options* **;**
BY *variables* **;**
ID *variable* **INTERVAL=** *interval options* **;**
FCMPOPT *options* **;**
INPUT *variable-list / options* **;**
TARGET *variable-list / options* **;**

Functional Summary

The statements and options that control the SIMILARITY procedure are summarized in the following table.

Table 31.1 SIMILARITY Functional Summary

Description	Statement	Option
Statements		
Specifies BY-group processing	BY	
Specifies the time ID variable	ID	
Specifies the FCMP options	FCMPOPT	
Specifies input variables to analyze	INPUT	
Specifies target variables to analyze	TARGET	
Data Set Options		
Specifies the input data set	PROC SIMILARITY	DATA=
Specifies the time series output data set	PROC SIMILARITY	OUT=
Specifies the measure summary output data set	PROC SIMILARITY	OUTMEASURE=
Specifies the path output data set	PROC SIMILARITY	OUTPATH=
Specifies the sequence output data set	PROC SIMILARITY	OUTSEQUENCE=
Specifies the summary output data set	PROC SIMILARITY	OUTSUM=
User-Defined Functions and Subroutine Options		
Specifies FCMP quiet mode	FCMPOPT	QUIET=
Specifies FCMP trace mode	FCMPOPT	TRACE=
Accumulation and Seasonality Options		
Specifies the accumulation frequency	ID	INTERVAL=
Specifies the length of seasonal cycle	PROC SIMILARITY	SEASONALITY=
Specifies the interval alignment	ID	ALIGN=
Specifies that the time ID variable values are not sorted	ID	NOTSORTED
Specifies the starting time ID value	ID	START=
Specifies the ending time ID value	ID	END=

Description	Statement	Option
Specifies the accumulation statistic	ID, INPUT, TARGET	ACCUMULATE=
Specifies the missing value interpretation	ID, INPUT, TARGET	SETMISS=
Specifies the zero value interpretation	ID, INPUT, TARGET	ZEROMISS=
Specifies the type of missing value trimming	INPUT, TARGET	TRIMMISS=
Time Series Transformation Options		
Specifies simple differencing	INPUT, TARGET	DIF=
Specifies seasonal differencing	INPUT, TARGET	SDIF=
Specifies the transformation	INPUT, TARGET	TRANSFORM=
Input Sequence Options		
Specifies normalization	INPUT	NORMALIZE=
Specifies scaling	INPUT	SCALE=
Target Sequence Options		
Specifies normalization	TARGET	NORMALIZE=
Similarity Measure Options		
Specifies the compression limits	TARGET	COMPRESS=
Specifies the expansion limits	TARGET	EXPAND=
Specifies the similarity measure	TARGET	MEASURE=
Specifies the similarity measure and path	TARGET	PATH=
Specifies the sequence slide	TARGET	SLIDE=
Printing and Graphical Control Options		
Specifies the time ID format	ID	FORMAT=
Specifies printed output	PROC SIMILARITY	PRINT=
Specifies detailed printed output	PROC SIMILARITY	PRINTDETAILS
Specifies graphical output	PROC SIMILARITY	PLOTS=
Miscellaneous Options		
Specifies that analysis variables are processed in ascending order	PROC SIMILARITY	SORTNAMES
Specifies the ordering of the processing of the input and target variables	PROC SIMILARITY	ORDER=

PROC SIMILARITY Statement

PROC SIMILARITY *options* **;**

The following options can be used in the PROC SIMILARITY statement.

DATA=*SAS-data-set*
names the SAS data set that contains the time series, transactional, or sequence input data for the procedure. If the DATA= option is not specified, the most recently created SAS data set is used.

ORDER=*order-option*
specifies the order in which the variables listed in the INPUT and TARGET statements are to be processed. This ordering affects the OUTSEQUENCE=, OUTPATH=, OUTMEASURE=, and OUTSUM= data sets, in addition to the printed and graphical output. The SORTNAMES option also affects the ordering of the analysis. You must specify one of the following *order-options*:

INPUT specifies that each INPUT variable be processed and then the TARGET variables be processed. The results are stored and printed based only on the INPUT variables.

INPUTTARGET specifies that each INPUT variable be processed and then the TARGET variables be processed. The results are stored and printed based on both the INPUT and TARGET variables. This is the default.

TARGET specifies that each TARGET variable be processed and then the INPUT variables be processed. The results are stored and printed based only on the TARGET variables.

TARGETINPUT specifies that each TARGET variable be processed and then the INPUT variables be processed. The results are stored and printed based on both the TARGET and INPUT variables.

OUT=*SAS-data-set*
names the output data set to contain the time series variables specified in the subsequent INPUT and TARGET statements. If an ID variable is specified in the ID statement, it is also included in the OUT= data set. The values are accumulated based on the ID statement INTERVAL= option or the ACCUMULATE= options or both. The values are transformed based on the INPUT or TARGET statement TRANSFORM=, DIF=, and SDIF= options in this order. The OUT= data set is particularly useful when you want to further analyze, model, or forecast the resulting time series with other SAS/ETS procedures.

OUTMEASURE=*SAS-data-set*
names the output data set to contain the detailed similarity measures by time ID value. The form of the OUTMEASURE= data set is determined by the PROC SIMILARITY statement SORTNAMES and ORDER= options.

OUTPATH=*SAS-data-set*
names the output data set to contain the path used to compute the similarity measures for each slide and warp. The form of the OUTPATH= data set is determined by the PROC SIMILARITY statement SORTNAMES and ORDER= options. If a user-defined similarity measure is specified, the path cannot be determined; therefore, the OUTPATH= data set does not contain information related to this measure.

OUTSEQUENCE=*SAS-data-set*
names the output data set to contain the sequences used to compute the similarity measures for each slide and warp. The form of the OUTSEQUENCE= data set is determined by the PROC SIMILARITY statement SORTNAMES and ORDER= options.

OUTSUM=*SAS-data-set*

names the output data set to contain the similarity measure summary. The OUTSUM= data set is particularly useful when analyzing large numbers of series and only the summary of the results are needed. The form of the OUTSUM= data set is determined by the PROC SIMILARITY statement SORTNAMES and ORDER= options.

PLOTS=*option*

PLOTS=(*options* **...)**

specifies the graphical output desired. To specify multiple *options*, separate them by spaces and enclose the group in parentheses. By default, the SIMILARITY procedure produces no graphical output. The following graphical *options* are available:

COSTS	plots graphics for time warp costs.
DISTANCES	plots graphics for similarity absolute and relative distances (OUTPATH= data set).
INPUTS	plots graphics for input variable time series (OUT= data set).
MAPS	plots graphics for time warp maps (OUTPATH= data set).
MEASURES	plots graphics for similarity measures (OUTMEASURE= data set).
NORMALIZED	plots graphics for both the input and target variable normalized sequence. These plots are displayed only when the INPUT or TARGET statement NORMALIZE= option is specified.
PATHS	plots time warp paths graphics (OUTPATH= data set).
SCALED	plots graphics for both the input variable scaled sequence. These plots are displayed only when the INPUT statement SCALE= option is specified.
SEQUENCES	plots graphics for both the input and target variable sequence (OUTSEQUENCE= data set).
TARGETS	plots graphics for the target variable time series (OUT= data set).
WARPS	plots graphics for time warps (OUTPATH= data set).
ALL	is the same as PLOTS=(INPUTS TARGETS SEQUENCES NORMALIZED SCALED DISTANCES PATHS MAPS WARPS COST MEASURES).

PRINT=*option*

PRINT=(*options* **...)**

specifies the printed output desired. To specify multiple *options*, separate them by spaces and enclose the group in parentheses. By default, the SIMILARITY procedure produces no printed output. The following printing options are available:

DESCSTATS	prints the descriptive statistics for the working time series.
PATHS	prints the path statistics table. If a user-defined similarity measure is specified, the path cannot be determined; therefore, the PRINT=PATHS table is not printed for this measure.
COSTS	prints the cost statistics table.
WARPS	prints the warp summary table.

SLIDES	prints the slides summary table.
SUMMARY	prints the similarity measure summary table.
ALL	is the same as PRINT=(DESCSTATS PATHS COSTS WARPS SLIDES SUMMARY).

PRINTDETAILS
specifies that the output requested with the PRINT= option be printed in greater detail.

SEASONALITY=*integer*
specifies the length of the seasonal cycle where *integer* ranges from one to 10,000. For example, SEASONALITY=3 means that every group of three time periods forms a seasonal cycle. By default, the length of the seasonal cycle is 1 (no seasonality) or the length implied by the INTERVAL= option specified in the ID statement. For example, INTERVAL=MONTH implies that the length of the seasonal cycle is 12.

SORTNAMES
specifies that the variables specified in the INPUT and TARGET statements be processed in alphabetical order of the variable names. By default, the SIMILARITY procedure processes the variables in the order in which they are listed. The ORDER= option also affects the ordering in which the analysis is performed.

BY Statement

A BY statement can be used with PROC SIMILARITY to obtain separate dummy variable definitions for groups of observations defined by the BY variables.

When a BY statement appears, the procedure expects the input data set to be sorted in order of the BY variables.

If your input data set is not sorted in ascending order, use one of the following alternatives:

- Sort the data by using the SORT procedure with a similar BY statement.
- Specify the option NOTSORTED or DESCENDING in the BY statement for the SIMILARITY procedure. The NOTSORTED option does not mean that the data are unsorted, but rather that the data are arranged in groups (according to values of the BY variables) and that these groups are not necessarily in alphabetical or increasing numeric order.
- Create an index on the BY variables by using the DATASETS procedure.

For more information about the BY-group processing, see *SAS Language Reference: Concepts*. For more information about the DATASETS procedure, see the discussion in the *Base SAS Procedures Guide.*

FCMPOPT Statement

FCMPOPT *options* **;**

The FCMPOPT statement specifies the following *options* that are related to user-defined functions and subroutines:

QUIET=ON | OFF
specifies whether the nonfatal errors and warnings that are generated by the user-defined SAS language functions and subroutines are printed to the log. Nonfatal errors are usually associated with operations with missing values. The default is QUIET=ON.

TRACE=ON | OFF
specifies whether the user-defined SAS language functions and subroutines tracings are printed to the log. Tracings are the results of every operation executed. This option is generally used for debugging. The default is TRACE=OFF.

ID Statement

ID *variable* **INTERVAL=** *interval options* **;**

The ID statement names a numeric variable that identifies observations in the input and output data sets. The ID variable's values are assumed to be SAS date, time, or datetime values. In addition, the ID statement specifies the (desired) frequency associated with the time series. The ID statement options also specify how the observations are accumulated and how the time ID values are aligned to form the time series. The options specified affect all variables listed in subsequent INPUT and TARGET statements. If an ID statement is specified, the INTERVAL= option must also be specified. The other ID statement options are optional. If an ID statement is not specified, the observation number, with respect to the BY group, is used as the time ID.

The following options can be used with the ID statement:

ACCUMULATE=*option*
specifies how the data set observations are accumulated within each time period. The frequency (width of each time interval) is specified by the INTERVAL= option. The ID variable contains the time ID values. Each time ID variable value corresponds to a specific time period. The accumulated values form the time series, which is used in subsequent analysis.

The ACCUMULATE= option is particularly useful when there are zero or more than one input observations that coincide with a particular time period (for example, time-stamped transactional data). The EXPAND procedure offers additional frequency conversions and transformations that can also be useful in creating a time series.

The following *options* determine how the observations are accumulated within each time period based on the ID variable and the frequency specified by the INTERVAL= option:

NONE	No accumulation occurs; the ID variable values must be equally spaced with respect to the frequency. This is the default option.
TOTAL	Observations are accumulated based on the total sum of their values.

AVERAGE | AVG	Observations are accumulated based on the average of their values.
MINIMUM | MIN	Observations are accumulated based on the minimum of their values.
MEDIAN | MED	Observations are accumulated based on the median of their values.
MAXIMUM | MAX	Observations are accumulated based on the maximum of their values.
N	Observations are accumulated based on the number of nonmissing observations.
NMISS	Observations are accumulated based on the number of missing observations.
NOBS	Observations are accumulated based on the number of observations.
FIRST	Observations are accumulated based on the first of their values.
LAST	Observations are accumulated based on the last of their values.
STDDEV |STD	Observations are accumulated based on the standard deviation of their values.
CSS	Observations are accumulated based on the corrected sum of squares of their values.
USS	Observations are accumulated based on the uncorrected sum of squares of their values.

If the ACCUMULATE= option is specified, the SETMISSING= option is useful for specifying how accumulated missing values are treated. If missing values should be interpreted as zero, then SETMISSING=0 should be used. The section "Details: SIMILARITY Procedure" on page 2255 describes accumulation in greater detail.

ALIGN=*option*

controls the alignment of SAS dates that are used to identify output observations. The ALIGN= option accepts the following values: BEGINNING | BEG | B, MIDDLE | MID | M, and ENDING | END | E. ALIGN=BEGINNING is the default.

END=*option*

specifies a SAS date, datetime, or time value that represents the end of the data. If the last time ID variable value is less than the END= value, the series is extended with missing values. If the last time ID variable value is greater than the END= value, the series is truncated. For example, END="&sysdate"D uses the automatic macro variable SYSDATE to extend or truncate the series to the current date. The START= and END= options can be used to ensure that data that are associated within each BY group contain the same number of observations.

FORMAT=*format*

specifies the SAS format for the time ID values. If the FORMAT= option is not specified, the default format is implied by the INTERVAL= option. For example, FORMAT=DATE9. specifies that the DATE9. SAS format be used. Notice that the terminating "." is required when specifying a SAS format.

INTERVAL= *interval*

specifies the frequency of the accumulated time series. For example, if the input data set consists of quarterly observations, then INTERVAL=QTR should be used. If the SEASONALITY= option is not specified, the length of the seasonal cycle is implied from the INTERVAL= option. For example, INTERVAL=QTR implies a seasonal cycle of length 4. If the ACCUMULATE= option is also specified, the INTERVAL= option determines the time periods for the accumulation of observations.

NOTSORTED

specifies that the time ID values are not in sorted order. The SIMILARITY procedure sorts the data with respect to the time ID prior to analysis if the NOTSORTED option is specified.

SETMISSING=*option* | *number*

specifies how missing values (either actual or accumulated) are interpreted in the accumulated time series. If a *number* is specified, missing values are set to that number. If a missing value indicates an unknown value, the SETMISSING= option should not be used. If a missing value indicates no value, then SETMISSING=0 should be used. You typically use SETMISSING=0 for transactional data, because no recorded data usually implies no activity. The following *options* can also be used to determine how missing values are assigned:

MISSING	Missing values are set to missing. This is the default option.
AVERAGE | AVG	Missing values are set to the accumulated average value.
MINIMUM | MIN	Missing values are set to the accumulated minimum value.
MEDIAN | MED	Missing values are set to the accumulated median value.
MAXIMUM | MAX	Missing values are set to the accumulated maximum value.
FIRST	Missing values are set to the accumulated first nonmissing value.
LAST	Missing values are set to the accumulated last nonmissing value.
PREVIOUS | PREV	Missing values are set to the previous period's accumulated nonmissing value. Missing values at the beginning of the accumulated series remain missing.
NEXT	Missing values are set to the next period's accumulated nonmissing value. Missing values at the end of the accumulated series remain missing.

START=*option*

specifies a SAS date, datetime, or time value that represents the beginning of the data. If the first time ID variable value is greater than the START= value, missing values are added to the beginning of the series. If the first time ID variable value is less than the START= value, the series is truncated. The START= and END= options can be used to ensure that data that are associated with each BY group contain the same number of observations.

ZEROMISS=*option*

specifies how beginning and ending zero values (either actual or accumulated) are interpreted in the accumulated time series. The following *options* can also be used to determine how beginning and ending zero values are assigned:

NONE	Beginning and ending zeros are unchanged. This is the default.
LEFT	Beginning zeros are set to missing.
RIGHT	Ending zeros are set to missing.
BOTH	Both beginning and ending zeros are set to missing.

If the accumulated series is all missing or zero, the series is not changed.

INPUT Statement

INPUT *variable-list* < */ options* > **;**

The INPUT statement lists the input numeric variables in the DATA= data set whose values are to be accumulated to form the time series or represent ordered numeric sequences (when no ID statement is specified).

An input data set variable can be specified in only one INPUT or TARGET statement. Any number of INPUT statements can be used. The following *options* can be used with an INPUT statement:

ACCUMULATE=*option*

specifies how the data set observations are accumulated within each time period for the variables listed in the INPUT statement. If the ACCUMULATE= option is not specified in the INPUT statement, accumulation is determined by the ACCUMULATE= option of the ID statement. If the ACCUMULATE= option is not specified in the ID statement or the INPUT statement, no accumulation is performed. See the ID statement ACCUMULATE= option for more details.

DIF=*(numlist)*

specifies the differencing to be applied to the accumulated time series. The list of differencing orders must be separated by spaces or commas. For example, DIF=(1,3) specifies first, then third order, differencing. Differencing is applied after time series transformation. The TRANSFORM= option is applied before the DIF= option. Simple differencing is useful when you want to detrend the time series before computing the similarity measures.

NORMALIZE=*option*

specifies the sequence normalization to be applied to the working input sequence. The following normalization *options* are provided:

NONE	No normalization is applied. This option is the default.
ABSOLUTE	Absolute normalization is applied.
STANDARD	Standard normalization is applied.
User-Defined	Normalization is computed by a user-defined subroutine that is created using the FCMP procedure, where *User-Defined* is the subroutine name.

Normalization is applied to the working input sequence, which can be a subset of the working input time series if the SLIDE=INDEX or SLIDE=SEASON option is specified.

SCALE=*option*

specifies the scaling of the working input sequence with respect to the working target sequence. Scaling is performed after normalization. The following scaling *options* are provided:

NONE	No scaling is applied. This option is the default.
ABSOLUTE	Absolute scaling is applied.
STANDARD	Standard scaling is applied.
User-Defined	Scaling is computed by a user-defined subroutine that is created using the FCMP procedure, where *User-Defined* is the subroutine name.

Scaling is applied to the working input sequence, which can be a subset of the working input time series if the SLIDE=INDEX or SLIDE=SEASON option is specified.

SDIF=*(numlist)*

specifies the seasonal differencing to be applied to the accumulated time series. The list of seasonal differencing orders must be separated by spaces or commas. For example, SDIF=(1,3) specifies first, then third, order seasonal differencing. Differencing is applied after time series transformation. The TRANSFORM= option is applied before the SDIF= option. Seasonal differencing is useful when you want to deseasonalize the time series before computing the similarity measures.

SETMISSING=*option* | *number*

SETMISS=*option* | *number*

specifies how missing values (either actual or accumulated) are interpreted in the accumulated time series or ordered sequence for variables listed in the INPUT statement. If the SETMISSING= option is not specified in the INPUT statement, missing values are set based on the SETMISSING= option in the ID statement. If the SETMISSING= option is not specified in the ID statement or the INPUT statement, no missing value interpretation is performed. See the ID statement SETMISSING= option for more details.

TRANSFORM=*option*

specifies the time series transformation to be applied to the accumulated time series. The following transformations are provided:

NONE	No transformation is applied. This option is the default.
LOG	Logarithmic transformation is applied.
SQRT	Square-root transformation is applied.
LOGISTIC	Logistic transformation is applied.
BOXCOX(*number*)	Box-Cox transformation with parameter is applied, where the real *number* is between –5 and 5.
User-Defined	Transformation is computed by a user-defined subroutine that is created using the FCMP procedure, where *User-Defined* is the subroutine name.

When the TRANSFORM= option is specified, the time series must be strictly positive unless a user-defined function is used.

TRIMMISSING=*option*

TRIMMISSING=*option*

specifies how missing values (either actual or accumulated) are trimmed from the accumulated time series or ordered sequence for variables that are listed in the INPUT statement. The following trimming options are provided:

NONE	No missing value trimming is applied.
LEFT	Beginning missing values are trimmed.
RIGHT	Ending missing values are trimmed.
BOTH	Both beginning and ending missing value are trimmed. This is the default.

ZEROMISS=*option*

specifies how beginning and ending zero values (either actual or accumulated) are interpreted in the accumulated time series or ordered sequence for variables listed in the INPUT statement. If the ZEROMISS= option is not specified in the INPUT statement, beginning and ending zero values are set based on the ZEROMISS= option of the ID statement. If the ZERO= option is not specified in the ID statement or the INPUT statement, no zero value interpretation is performed. See the ID statement ZEROMISS= option for more details.

TARGET Statement

TARGET *variable-list* < / *options* > **;**

The TARGET statement lists the numeric target variables in the DATA= data set whose values are to be accumulated to form the time series or represent ordered numeric sequences (when no ID statement is specified).

An input data set variable can be specified in only one INPUT or TARGET statement. Any number of TARGET statements can be used. The following *options* can be used with a TARGET statement:

ACCUMULATE=*option*

specifies how the data set observations are accumulated within each time period for the variables listed in the TARGET statement. If the ACCUMULATE= option is not specified in the TARGET statement, accumulation is determined by the ACCUMULATE= option in the ID statement. If the ACCUMULATE= option is not specified in the ID statement or the TARGET statement, no accumulation is performed. See the ID statement ACCUMULATE= option for more details.

COMPRESS=*option* | **(***options***)**

specifies the sliding sequence (global) and warping (local) compression range of the target sequence with respect to the input sequence. Compression of the target sequence is the same as expansion of the input sequence and vice versa. The compression limits are defined based on the length of the target sequence and are imposed on the target sequence. The following compression options are provided:

GLOBALABS=*integer* specifies the absolute global compression, where *integer* ranges from zero to 10,000. GLOBALABS=0 implies no global compression, which is the default unless the GLOBALPCT= option is specified.

GLOBALPCT=*number* specifies global compression as a percentage of the length of the target sequence, where *number* ranges from zero to 100. GLOBALPCT=0 implies no global compression, which is the default. GLOBALPCT=100 implies maximum allowable compression.

LOCALABS=*integer* specifies the absolute local compression, where *integer* ranges from zero to 10,000. The default is maximum allowable absolute local compression unless the LOCALPCT= option is specified.

LOCALPCT=*number* specifies local compression as a percentage of the length of the target sequence, where *number* ranges from zero to 100. The percentage specified by the LOCALPCT= option must be less than the GLOBALPCT= option. LOCALPCT=0 implies no local compression. LOCALPCT=100 implies maximum allowable local compression. The default is LOCALPCT=100.

If the SLIDE=NONE or the SLIDE=SEASON option is specified in the TARGET statement, the global compression options are ignored. To disallow local compression, use the option COMPRESS=(LOCALPCT=0 LOCALABS=0).

If the SLIDE=INDEX option is specified, the global compression options are not ignored. To completely disallow both global and local compression, use the option COMPRESS=(GLOBALPCT=0 LOCALPCT=0) or COMPRESS=(GLOBALABS=0 LOCALABS=0). To allow only local compression, use the option COMPRESS=(GLOBALPCT=0 GLOBALABS=0). These are the default compression options.

The preceding options can be used in combination to specify the desired amount of global and local compression as the following examples illustrate, where L_c denotes the global compression limit and l_c denotes the local compression limit:

- COMPRESS=(GLOBALPCT=20) allows the global and local compression to range from zero to $L_c = \min\left(\lfloor 0.2N_y \rfloor, \left(N_y - 1\right)\right)$.
- COMPRESS=(GLOBALPCT=20 GLOBALABS=10) allows the global and local compression to range from zero to $L_c = \min\left(\lfloor 0.2N_y \rfloor, \min\left(\left(N_y - 1\right), 10\right)\right)$.
- COMPRESS=(LOCALPCT=10) allows the local compression to range from zero to $l_c = \min\left(\lfloor 0.1N_y \rfloor, \left(N_y - 1\right)\right)$.
- COMPRESS=(LOCALPCT=20 LOCALABS=5) allows the local compression to range from zero to $l_c = \min\left(\lfloor 0.2N_y \rfloor, \min\left(\left(N_y - 1\right), 5\right)\right)$.
- COMPRESS=(GLOBALPCT=20 LOCALPCT=20) allows the global compression to range from zero to $L_c = \min\left(\lfloor 0.2N_y \rfloor, \left(N_y - 1\right)\right)$ and allows the local compression to range from zero to $l_c = \min\left(\lfloor 0.2N_y \rfloor, \left(N_y - 1\right)\right)$.
- COMPRESS=(GLOBALPCT=20 GLOBALABS=10 LOCALPCT=10 LOCALABS=5) allows the global compression to range from zero to $L_c = \min\left(\lfloor 0.2N_y \rfloor, \min\left(\left(N_y - 1\right), 10\right)\right)$ and allows the local compression to range from zero to $l_c = \min\left(\lfloor 0.1N_y \rfloor, \min\left(\left(N_y - 1\right), 5\right)\right)$.

Suppose T_z is the length of the input time series and N_y is the length of the target sequence. The *valid* global compression limit, L_c, is always limited by the length of the target sequence: $0 \leq L_c < N_y$.

Suppose N_x is the length of the input sequence and N_y is the length of the target sequence. The *valid* local compression limit, l_c, is always limited by the lengths of the input and target sequence: $\max\left(0, \left(N_y - N_x\right)\right) \leq l_c < N_y$.

DIF=*(numlist)*

specifies the differencing to be applied to the accumulated time series. The list of differencing orders must be separated by spaces or commas. For example, DIF=(1,3) specifies first, then third, order differencing. Differencing is applied after time series transformation. The TRANSFORM= option is applied before the DIF= option. Simple differencing is useful when you want to detrend the time series before computing the similarity measures.

EXPAND=*option* | **(***options***)**

specifies the sliding sequence (global) and warping (local) expansion range of the target sequence with respect to the input sequence. Expansion of the target sequence is the same as compression of the input sequence and vice versa. The expansion limits are defined based on the length of the input sequence, but are imposed on the target sequence. The following expansion options are provided:

GLOBALABS=*integer* specifies the absolute global expansion, where *integer* ranges from zero to 10,000. GLOBALABS=0 implies no global expansion, which is the default unless the GLOBALPCT= option is specified.

GLOBALPCT=*number* specifies global expansion as a percentage of the length of the target sequence, where *number* ranges from zero to 100. GLOBALPCT=0 implies no global expansion, which is the default unless the GLOBALABS= option is specified. GLOBALPCT=100 implies maximum allowable global expansion.

LOCALABS=*integer* specifies the absolute local expansion, where *integer* ranges from zero to 10,000. The default is the maximum allowable absolute local expansion unless the LOCALPCT= option is specified.

LOCALPCT=*number* specifies local expansion as a percentage of the length of the target sequence, where *number* ranges from zero to 100. LOCALPCT=0 implies no local expansion. LOCALPCT=100 implies maximum allowable local expansion. The default is LOCALPCT=100.

If the SLIDE=NONE or the SLIDE=SEASON option is specified in the TARGET statement, the global expansion options are ignored. To disallow local expansion, use the option EXPAND=(LOCALPCT=0 LOCALABS=0).

If the SLIDE=INDEX option is specified, the global expansion options are not ignored. To completely disallow both global and local expansion, use the option EXPAND=(GLOBALPCT=0 LOCALPCT=0) or EXPAND=(GLOBALABS=0 LOCALABS=0). To allow only local expansion, use the option EXPAND=(GLOBALPCT=0 GLOBALABS=0). These are the default expansion options.

The preceding options can be used in combination to specify the desired amount of global and local expansion as the following examples illustrate, where L_e denotes the global expansion limit and l_e denotes the local expansion limit:

- EXPAND=(GLOBALPCT=20) allows the global and local expansion to range from zero to $L_e = \min\left(\lfloor 0.2N_y \rfloor, (N_y - 1)\right)$.
- EXPAND=(GLOBALPCT=20 GLOBALABS=10) allows the global and local expansion to range from zero to $L_e = \min\left(\lfloor 0.2N_y \rfloor, \min\left((N_y - 1), 10\right)\right)$.
- EXPAND=(LOCALPCT=10) allows the local expansion to range from zero to $l_e = \min\left(\lfloor 0.1N_y \rfloor, (N_y - 1)\right)$.

- EXPAND=(LOCALPCT=10 LOCALABS=5) allows the local expansion to range from zero to $l_e = \min\left(\lfloor 0.1N_y \rfloor, \min\left(\left(N_y - 1\right), 5\right)\right)$.
- EXPAND=(GLOBALPCT=20 LOCALPCT=10) allows the global expansion to range from zero to $L_e = \min\left(\lfloor 0.2N_y \rfloor, \left(N_y - 1\right)\right)$ and allows the local expansion to range from zero to $l_e = \min\left(\lfloor 0.1N_y \rfloor, \left(N_y - 1\right)\right)$.
- EXPAND=(GLOBALPCT=20 GLOBALABS=10 LOCALPCT=10 LOCALABS=5) allows the global expansion to range from zero to $L_e = \min\left(\lfloor 0.2N_y \rfloor, \min\left(\left(N_y - 1\right), 10\right)\right)$ and allows the local expansion to range from zero to $l_e = \min\left(\lfloor 0.1N_y \rfloor, \min\left(\left(N_y - 1\right), 5\right)\right)$.

Suppose T_z is the length of the input time series and N_y is the length of the target sequence. The *valid* global expansion limit, L_e, is always limited by the length of the input time series: $0 \leq L_e < T_z$.

Suppose N_x is the length of the input sequence and N_y is the length of the target sequence. The *valid* local expansion limit, l_e, is always limited by the lengths of the input and target sequence: $\max\left(0, \left(N_x - N_y\right)\right) \leq l_e < N_x$.

MEASURE=*option*

specifies the similarity measure to be computed by using the working input and target sequences. The following similarity measures are provided:

SQRDEV	squared deviation. This option is the default.
ABSDEV	absolute deviation
MSQRDEV	mean squared deviation
MSQRDEVINP	mean squared deviation relative to the length of the input sequence
MSQRDEVTAR	mean squared deviation relative to the length of the target sequence
MSQRDEVMIN	mean squared deviation relative to the minimum valid path length
MSQRDEVMAX	mean squared deviation relative to the maximum valid path length
MABSDEV	mean absolute deviation
MABSDEVINP	mean absolute deviation relative to the length of the input sequence
MABSDEVTAR	mean absolute deviation relative to the length of the target sequence
MABSDEVMIN	mean absolute deviation relative to the minimum valid path length
MABSDEVMAX	mean absolute deviation relative to the maximum valid path length
User-Defined	The measure is computed by a user-defined function created by using the FCMP procedure, where *User-Defined* is the function name.

NORMALIZE=*option*

specifies the sequence normalization to be applied to the working target sequence. The following normalization options are provided:

NONE	No normalization is applied. This option is the default.
ABSOLUTE	Absolute normalization is applied.
STANDARD	Standard normalization is applied.
User-Defined	Normalization is computed by a user-defined subroutine that is created by using the FCMP procedure, where *User-Defined* is the subroutine name.

PATH=*option*

specifies the similarity measure and warping path information to be computed using the working input and target sequences. The following similarity measures and warping path are provided:

User-Defined The measure and path are computed by a user-defined subroutine that is created by using the FCMP procedure, where *User-Defined* is the subroutine name

For computational efficiency, the PATH= option should be only used when you want to compute both the similarity measure and the warping path information. If only the similarity measure is needed, use the MEASURE= option. If you specify both the MEASURE= and PATH= option in the TARGET statement, the PATH= option takes precedence.

SDIF=*(numlist)*

specifies the seasonal differencing to be applied to the accumulated time series. The list of seasonal differencing orders must be separated by spaces or commas. For example, SDIF=(1,3) specifies first, then third, order seasonal differencing. Differencing is applied after time series transformation. The TRANSFORM= option is applied before the SDIF= option. Seasonal differencing is useful when you want to deseasonalize the time series before computing the similarity measures.

SETMISSING=*option* | *number*

SETMISS=*option* | *number*

option specifies how missing values (either actual or accumulated) are interpreted in the accumulated time series for variables that are listed in the TARGET statement. If the SETMISSING= option is not specified in the TARGET statement, missing values are set based on the SETMISSING= option in the ID statement. If the SETMISSING= option is not specified in the ID statement or the TARGET statement, no missing value interpretation is performed. See the ID statement SETMISSING= option for more details.

SLIDE=*option*

specifies the sliding of the target sequence with respect to the input sequence. The following slides are provided:

NONE No sequence sliding. The input time series is compared with the target sequence directly with no sliding. This option is the default.

INDEX Slide by time index. The input time series is compared with the target sequence by observation index.

SEASON Slide by seasonal index. The input time series is compared with the target sequence by seasonal index.

The SLIDE= option takes precedence over the COMPRESS= and EXPAND= options.

TRANSFORM=*option*

specifies the time series transformation to be applied to the accumulated time series. The following transformations are provided:

NONE No transformation is applied. This option is the default.

LOG Logarithmic transformation is applied.

SQRT Square-root transformation is applied.

LOGISTIC Logistic transformation is applied.

BOXCOX(*number*) Box-Cox transformation with parameter is applied, where the real *number* is between –5 and 5

User-Defined Transformation is computed by a user-defined subroutine that is created by using the FCMP procedure, where *User-Defined* is the subroutine name.

When the TRANSFORM= option is specified, the time series must be strictly positive unless a user-defined function is used.

TRIMMISSING=*option*

TRIMMISS= *option*

specifies how missing values (either actual or accumulated) are trimmed from the accumulated time series or ordered sequence for variables that are listed in the TARGET statement. The following trimming options are provided:

NONE No missing value trimming is applied.

LEFT Beginning missing values are trimmed.

RIGHT Ending missing values are trimmed.

BOTH Both beginning and ending missing values are trimmed. This is the default.

ZEROMISS=*option*

specifies how beginning and ending zero values (either actual or accumulated) are interpreted in the accumulated time series or ordered sequence for variables listed in the TARGET statement. If the ZEROMISS= option is not specified in the TARGET statement, beginning and ending values are set based on the ZEROMISS= option in the ID statement. See the ID statement ZEROMISS= option for more details.

Details: SIMILARITY Procedure

You can use the SIMILARITY procedure to do the following functions, which are done in the order shown. First, you can form time series data from transactional data with the options shown:

1. accumulation ACCUMULATE= option
2. missing value interpretation SETMISSING= option
3. zero value interpretation ZEROMISS= option

Next, you can transform the accumulated time series to form the working time series with the following options. Transformations are useful when you want to stabilize the time series before computing the similarity measures. Simple and seasonal differencing are useful when you want to detrend or deseasonalize the time series before computing the similarity measures. Often, but not always, the TRANSFORM=, DIF=, and SDIF= options should be specified in the same way for both the target and input variables.

4. time series transformation TRANSFORM= option
5. time series differencing DIF= and SDIF= option
6. time series missing value trimming TRIMMISSING= option
7. time series descriptive statistics PRINT=DESCSTATS option

After the working series is formed, you can treat it as an ordered sequence that can be normalized or scaled. Normalizations are useful when you want to compare the "shape" or "profile" of the time series. Scaling is useful when you want to compare the input sequence to the target sequence while discounting the variation of the target sequence.

8. normalization NORMALIZE= option
9. scaling SCALE= option

After the working sequences are formed, you can compute similarity measures between input and target sequences:

10. sliding SLIDE= option
11. warping COMPRESS= and EXPAND= option
12. similarity measure MEASURE= and PATH= option

The SLIDE= option specifies observation-index sliding, seasonal-index sliding, or no sliding. The COMPRESS= and EXPAND= options specify the warping limits. The MEASURE= and PATH= options specify how the similarity measures are computed.

Accumulation

If the ACCUMULATE= option is specified in the ID, INPUT, or TARGET statement, data set observations are accumulated within each time period. The frequency (width of each time interval) is specified by the INTERVAL= option in the ID statement. The ID variable contains the time ID values. Each time ID value corresponds to a specific time period. Accumulation is particularly useful when the input data set contains transactional data, whose observations are not spaced with respect to any particular time interval. The accumulated values form the time series, which is used in subsequent analyses.

For example, suppose a data set contains the following observations:

```
19MAR1999    10
19MAR1999    30
11MAY1999    50
12MAY1999    20
23MAY1999    20
```

If the INTERVAL=MONTH is specified, all of the preceding observations fall within three time periods of March 1999, April 1999, and May 1999. The observations are accumulated within each time period as follows:

If the ACCUMULATE=NONE option is specified, an error is generated because the ID variable values are not equally spaced with respect to the specified frequency (MONTH).

If the ACCUMULATE=TOTAL option is specified, the data are accumulated as follows:

```
O1MAR1999    40
O1APR1999    .
O1MAY1999    90
```

If the ACCUMULATE=AVERAGE option is specified, the data are accumulated as follows:

```
O1MAR1999    20
O1APR1999    .
O1MAY1999    30
```

If the ACCUMULATE=MINIMUM option is specified, the data are accumulated as follows:

```
O1MAR1999    10
O1APR1999    .
O1MAY1999    20
```

If the ACCUMULATE=MEDIAN option is specified, the data are accumulated as follows:

```
O1MAR1999    20
01APR1999    .
O1MAY1999    20
```

If the ACCUMULATE=MAXIMUM option is specified, the data are accumulated as follows:

```
O1MAR1999    30
O1APR1999    .
O1MAY1999    50
```

If the ACCUMULATE=FIRST option is specified, the data are accumulated as follows:

```
O1MAR1999    10
O1APR1999    .
O1MAY1999    50
```

If the ACCUMULATE=LAST option is specified, the data are accumulated as follows:

```
O1MAR1999    30
O1APR1999    .
O1MAY1999    20
```

If the ACCUMULATE=STDDEV option is specified, the data are accumulated as follows:

```
O1MAR1999    14.14
O1APR1999    .
O1MAY1999    17.32
```

As can be seen from the preceding examples, even though the data set observations contain no missing values, the accumulated time series can have missing values.

Missing Value Interpretation

Sometimes missing values should be interpreted as unknown values. But sometimes missing values are known, such as when missing values are created from accumulation and no observations should be interpreted as no (zero) value. In the former case, the SETMISSING= option in the ID, INPUT, or TARGET statement can be used to interpret how missing values are treated. The SETMISSING=0 option should be used when missing observations are to be treated as no (zero) values. In other cases, missing values should be interpreted as global values, such as minimum or maximum values of the accumulated series. The accumulated and interpreted time series is used in subsequent analyses.

The SETMISSING=0 option should be used with missing observations are to be treated as a zero value. In other cases, missing values should be interpreted as global values, such as minimum or maximum values of the accumulated series. The accumulated and interpreted time series is then used in subsequent analyses.

Zero Value Interpretation

When querying certain databases for time-stamped data based on a particular time range, time periods that contain no data are sometimes assigned zero values. For certain analyses, it is more desirable to assign these values to missing. Often, these beginning or ending zero values need to be interpreted as missing values. The ZEROMISS= option in the ID, INPUT, or TARGET statement specifies that the beginning, ending, or both the beginning and ending values are to be interpreted as zero values.

Time Series Transformation

Transformations are useful when you want to stabilize the time series before computing the similarity measures. There are four transformations available, for strictly positive series only. Let $y_t > 0$ be the original time series, and let w_t be the transformed series. The transformations are defined as follows:

Log is the logarithmic transformation,

$$w_t = \ln(y_t)$$

Logistic is the logistic transformation,

$$w_t = \ln(cy_t/(1 - cy_t))$$

where the scaling factor c is

$$c = (1 - e^{-6})10^{-\text{ceil}(\log_{10}(max(y_t)))}$$

and $\text{ceil}(x)$ is the smallest integer greater than or equal to x.

Square root — is the square root transformation,

$$w_t = \sqrt{y_t}$$

Box-Cox — is the Box-Cox transformation,

$$w_t = \begin{cases} \frac{y_t^{\lambda}-1}{\lambda} & \lambda \neq 0 \\ \ln(y_t) & \lambda = 0 \end{cases}$$

User-Defined — is the transformation computed by a user-defined subroutine that is created by using the FCMP procedure, where *User-Defined* is the subroutine name.

Other time series transformations can be performed prior to invoking the SIMILARITY procedure by using the SAS/ETS EXPAND procedure or the DATA step.

Time Series Differencing

After optionally transforming the series, the accumulated series can be simply or seasonally differenced using the INPUT or TARGET statement DIF= and SDIF= options. Simple and seasonal differencing are useful when you want to detrend or deseasonalize the time series before computing the similarity measures.

For example, suppose y_t is a monthly time series. The following examples of the DIF= and SDIF= options demonstrate how to simply and seasonally difference the time series: DIF=(1,3) specifies first, then third, order differencing; SDIF=(1,3) specifies first, then third, order seasonal differencing.

Additionally, assuming that y_t is strictly positive, the INPUT or TARGET statement TRANSFORM= option and the DIF= and SDIF= options can be combined.

Time Series Missing Value Trimming

In some instances, missing values should be interpreted as an unknown observation, but other times, missing values are known and should be interpreted as a zero value. This is the case when missing values are created from accumulation, and a missing observation should be interpreted as having no value (meaning a value of zero). In the former case, the SETMISSING=option in the ID, INPUT, or TARGET, statement can be used to interpret how missing observations should be treated. By default, missing values, at the beginning and ending of the data set, are trimmed from the data set prior to analysis. This can be performed using TRIMMISS=both.

Time Series Descriptive Statistics

After a series has been optionally accumulated and transformed with missing values interpreted, descriptive statistics can be computed for the resulting working series by specifying the PRINT=DESCSTATS option. This option produces an ODS table that contains the sum, mean, minimum, maximum, and standard deviation of the working series.

Input and Target Sequences

After the input and target working series are formed, they can be treated as two ordered sequences. Given an input time sequence, x_i, for $i = 1$ to N_x, where i is the input sequence index, and a target time sequence, y_j, for $j = 1$ to N_y, where j is the target sequence index, these sequences are analyzed for similarity.

Sliding Sequences

Similarity measures can be computed between the target sequence and any contiguous subsequences of the input time series.

There are three types of sequence sliding:

- no sliding
- slide by time index
- slide by season index

For more information, see Leonard et al. (2008).

Time Warping

Time warping allows for the comparison between target and input sequences of differing lengths by compressing or expanding the input sequence with respect the target sequence while respecting the order of the sequence elements.

For more information, see Leonard et al. (2008).

Sequence Normalization

The working (input or target) sequence can be normalized prior to further analysis. Let q_i be the original sequence with mean μ_q and standard deviation σ_q, and let r_t be the normalized sequence. The normalizations are defined as follows:

- Standard is the standard normalization

$$r_i = (q_i - \mu_q)/\sigma_q$$

- Absolute is the absolute normalization

$$r_i = (q_i - min(q_i))/(max(q_i) - min(q_i))$$

- User-defined is a user-defined normalization created by the FCMP procedure.

Sequence Scaling

The working input sequence can be scaled to the working target sequence. Sequence scaling is applied after normalization. Let y_j be the working target sequence with mean μ_y and standard deviation σ_y. Let x_i be the working input sequence and let q_i be the scaled sequence. The scaling is defined as follows:

- Standard is the standard normalization

$$q_i = (x_i - \mu_y)/\sigma_y$$

- Absolute is the absolute scaling

$$q_i = (x_i - min(y_j))/(max(y_j) - min(y_j))$$

- User-defined is a user-defined scaling created by the FCMP procedure.

Similarity Measures

The working input sequence can be compared to the working target sequence to create a similarity.

For more information, see Leonard et al. (2008).

User-Defined Functions and Subroutines

A user-defined routine can be written in the SAS language by using the FCMP procedure or in the C language by using both the FCMP procedure and the PROTO procedure, respectively. The SIMILARITY procedure cannot use C language routines directly. The procedure can use only SAS language routines that might or might not call C language routines. Creating user-defined routines is more completely described in the FCMP procedure and the PROTO procedure documentation. The FCMP and PROTO procedures are part of Base SAS software.

The SAS language provides integrated memory management and exception handling such as operations on missing values. The C language provides flexibility and allows the integration of existing C language libraries. However, proper memory management and exception handling are solely the responsibility of

the user. Additionally, the support for standard C libraries is restricted. If you have a choice, it is highly recommended that you write user-defined functions and subroutines in the SAS language using the FCMP procedure.

For each of the tasks previously described, the following sections describe the required subroutine or function signature and provide examples of using a user-defined routine with the SIMILARITY procedure.

Time Series Transformations

A user-defined transformation subroutine has the following subroutine signature:

```
SUBROUTINE <SUBROUTINE-NAME> ( <ARRAY-NAME>[*] );
```

where the array-name is the time series to be transformed.

For example, to duplicate the functionality of the built-in TRANSFORM=LOG option in the INPUT and TARGET statement, the following SAS statements create a user-defined version of this transformation called MYTRANSFORM and store this subroutine in the catalog SASUSER.MYSIMILAR.

```
proc fcmp outlib=sasuser.mysimilar.package;

   subroutine mytransform( series[*] );

      outargs series;

      length = DIM(series);

      do i = 1 to length;
         value = series[i];
         if value > 0 then do;
            series[i] = log( value );
         end;
         else do;
            series[i] = .;
         end;
      end;

   endsub;

run;
```

This user-defined subroutine can be specified in the TRANSFORM= option in the INPUT or TARGET statement as follows:

```
options cmplib = sasuser.mysimilar;

proc similarity ...;
...
input  myinput  / transform=mytransform;
target mytarget / transform=mytransform;
...
run;
```

Sequence Normalizations

A user-defined normalization subroutine has the following signature:

```
SUBROUTINE <SUBROUTINE-NAME> ( <ARRAY-NAME>[*] );
```

where the array-name is the sequence to be normalized.

For example, to duplicate the functionality of the built-in NORMALIZE=ABSOLUTE option in the INPUT and TARGET statement, the following SAS statements create a user-defined version of this normalization called MYNORMALIZE and store this subroutine in the catalog SASUSER.MYSIMILAR.

```
proc fcmp outlib=sasuser.mysimilar.package;

   subroutine mynormalize( sequence[*] );

      outargs sequence;

      length = DIM(sequence);
      minimum = .; maximum = .;

      do i = 1 to length;
         value = sequence[i];
         if nmiss(minimum) | nmiss(maximum) then do;
            minimum = value;
            maximum = value;
         end;
         if nmiss(value) = 0 then do;
            if value < minimum then minimum = value;
            if value > maximum then maximum = value;
         end;
      end;

      do i = 1 to length;
         value = sequence[i];
         if nmiss( value ) | minimum > maximum then do;
            sequence[i] = .;
         end;
         else do;
            sequence[i] = (value - minimum) / (maximum - minimum);
         end;
      end;

   endsub;

run;
```

This user-defined subroutine can be specified in the NORMALIZE= option in the INPUT or TARGET statement as follows:

```
options cmplib = sasuser.mysimilar;

proc similarity ...;
   ...
   input  myinput  / normalize=mynormalize;
   target mytarget / normalize=mynormalize;
   ...
run;
```

Sequence Scaling

A user-defined scaling subroutine has the following signature:

```
SUBROUTINE <SUBROUTINE-NAME> ( <ARRAY-NAME>[*], <ARRAY-NAME>[*] );
```

where the first array-name is the target sequence and the second array-name is the input sequence to be scaled.

For example, to duplicate the functionality of the built-in SCALE=ABSOLUTE option in the INPUT statement, the following SAS statements create a user-defined version of this scaling called MYSCALE and store this subroutine in the catalog SASUSER.MYSIMILAR.

```
proc fcmp outlib=sasuser.mysimilar.package;

   subroutine myscale( target[*], input[*] );

      outargs input;

      length = DIM(target);
      minimum = .; maximum = .;

      do i = 1 to length;
         value = target[i];
         if nmiss(minimum) | nmiss(maximum) then do;
            minimum = value;
            maximum = value;
         end;
         if nmiss(value) = 0 then do;
            if value < minimum then minimum = value;
            if value > maximum then maximum = value;
         end;
      end;

      do i = 1 to length;
         value = input[i];
         if nmiss( value ) | minimum > maximum then do;
            input[i] = .;
         end;
         else do;
            input[i] = (value - minimum) / (maximum - minimum);
         end;
      end;

   endsub;

run;
```

This user-defined subroutine can be specified in the SCALE= option in the INPUT statement as follows:

```
options cmplib=sasuser.mysimilar;

proc similarity ...;
   ...
   input  myinput  / scale=myscale;
   ...
run;
```

Similarity Measures

A user-defined similarity measure function has the following signature:

```
FUNCTION <FUNCTION-NAME> ( <ARRAY-NAME>[*], <ARRAY-NAME>[*] );
```

where the first array-name is the target sequence and the second array-name is the input sequence. The return value of the function is the similarity measure associated with the target sequence and the input sequence.

For example, to duplicate the functionality of the built-in MEASURE=ABSDEV option in the TARGET statement with no warping, the following SAS statements create a user-defined version of this measure called MYMEASURE and store this subroutine in the catalog SASUSER.MYSIMILAR.

```
proc fcmp outlib=sasuser.mysimilar.package;

   function mymeasure( target[*], input[*] );

      length = min(DIM(target), DIM(input));
      sum = 0; num = 0;

      do i = 1 to length;
         x = input[i];
         w = target[i];
         if nmiss(x) = 0 & nmiss(w) = 0 then do;
            d = x - w;
            sum = sum + abs(d);
            num = num + 1;
         end;
      end;

      if num <= 0 then return(.);

      return(sum);

   endsub;

run;
```

This user-defined function can be specified in the MEASURE= option in the TARGET statement as follows:

```
options cmplib=sasuser.mysimilar;

proc similarity ...;
   ...
   target  mytarget  / measure=mymeasure;
   ...
run;
```

For another example, to duplicate the functionality of the built-in MEASURE=SQRDEV and MEASURE=ABSDEV options by using the C language, the following SAS statements create a user-defined C language version of these measures called DTW_SQRDEV_C and DTW_ABSDEV_C and store these functions in the catalog SASUSER.CSIMIL.CFUNCS. DTW refers to dynamic time warping. These C language functions can be then called by SAS language functions and subroutines.

```
proc proto package=sasuser.csimil.cfuncs;

mapmiss double = 999999999;

double dtw_sqrdev_c( double * target / iotype=input,
                     int      targetLength,
                     double * input / iotype=input,
                     int      inputLength );

externc dtw_sqrdev_c;
 double dtw_sqrdev_c( double * target,
                      int      targetLength,
                      double * input,
                      int      inputLength )
   {
      int      i,j;
      double   x,w,d;
      double * prev = (double *)malloc( sizeof(double)*targetLength);
      double * curr = (double *)malloc( sizeof(double)*inputLength);
      if ( prev == 0 || curr == 0 ) return 999999999;

      x = input[0];
      for ( j=0; j<targetLength; j++ ) {
         w = target[j];
         d = x - w;
         d = d*d;
         if ( j == 0 ) prev[j] = d;
         else          prev[j] = d + prev[j-1];
      }

      for (i=1; i<inputLength; i++ ) {
         x = input[i];

         j = 0;
         w = target[j];
         d = x - w;
```

```
         d = d*d;
         curr[j] = d + prev[j];

         for (j=1; j<targetLength; j++ ) {
            w = target[j];
            d = x - w;
            d = d*d;
            curr[j] = d + fmin( prev[j],
                                fmin( prev[j-1], curr[j]));
         }

         if ( i < targetLength ) {
           for( j=0; j<inputLength; j++ )
             prev[j] = curr[j];
         }
       }

       d = curr[inputLength-1];
       free( (char*) prev);
       free( (char*) curr);
       return( d );
  }
externcend;

double dtw_absdev_c( double * target / iotype=input,
                     int      targetLength,
                     double * input / iotype=input,
                     int      inputLength );
externc dtw_absdev_c;
 double dtw_absdev_c( double * target,
                      int      targetLength,
                      double * input,
                      int      inputLength )
   {
      int      i,j;
      double   x,w,d;
      double * prev = (double *)malloc( sizeof(double)*targetLength);
      double * curr = (double *)malloc( sizeof(double)*inputLength);
      if ( prev == 0 || curr == 0 ) return 999999999;

      x = input[0];
      for ( j=0; j<targetLength; j++ ) {
        w = target[j];
        d = x - w;
        d = fabs(d);
        if (j == 0) prev[j] = d;
        else prev[j] = d + prev[j-1];
      }

      for (i=1; i<inputLength; i++ ) {
        x = input[i];

        j = 0;
        w = target[j];
```

```
          d = x - w;
          d = fabs(d);
          curr[j] = d + prev[j];

          for (j=1; j<targetLength; j++) {
            w = target[j];
            d = x - w;
            d = fabs(d);
            curr[j] = d + fmin( prev[j],
                          fmin( prev[j-1], curr[j] ));
          }

          if ( i < inputLength) {
            for ( j=0; j<targetLength; j++ )
               prev[j] = curr[j];
          }

        }

        d = curr[inputLength-1];
        free( (char*) prev);
        free( (char*) curr);
        return( d );
      }
  externcend;

run;
```

The preceding SAS statements create two C language functions which can then be used in SAS language functions or subroutines or both. However, these functions cannot be directly used by the SIMILARITY procedure. In order to use these C language functions in the SIMILARITY procedure, two SAS language functions must be created that call these two C language functions. The following SAS statements create two user-defined SAS language versions of these measures called DTW_SQRDEV and DTW_ABSDEV and stores these functions in the catalog SASUSER.MYSIMILAR.FUNCS. These SAS language functions use the previously created C language function; the SAS language functions can then be used by the SIMILARITY procedure.

```
proc fcmp outlib=sasuser.mysimilar.funcs
          inlib=sasuser.cfuncs;

   function dtw_sqrdev( target[*], input[*] );
      dev = dtw_sqrdev_c(target,DIM(target),input,DIM(input));
      return( dev );
   endsub;

   function dtw_absdev( target[*], input[*] );
      dev = dtw_absdev_c(target,DIM(target),input,DIM(input));
      return( dev );
   endsub;

run;
```

This user-defined function can be specified in the MEASURE= option in the TARGET statement as follows:

```
options cmplib=sasuser.mysimilar;

proc similarity ...;
   ...
   target  mytarget    / measure=dtw_sqrdev;
   target  yourtarget  / measure=dtw_absdev;
   ...
run;
```

Similarity Measures and Warping Path

A user-defined similarity measure and warping path information function has the following signature:

```
FUNCTION <FUNCTION-NAME> ( <ARRAY-NAME>[*], <ARRAY-NAME>[*],
                           <ARRAY-NAME>[*], <ARRAY-NAME>[*],
                           <ARRAY-NAME>[*] );
```

where the first array-name is the target sequence, the second array-name is the input sequence, the third array-name is the returned target sequence indices, the fourth array-name is the returned input sequence indices, the fifth array-name is the returned path distances. The returned value of the function is the similarity measure. The last three returned arrays are used to compute the path and cost statistics.

The returned sequence indices must represent a valid warping path; that is, integers greater than zero and less than or equal to the sequence length and recorded in ascending order. The returned path distances must be nonnegative numbers.

Output Data Sets

The SIMILARITY procedure can create the OUT=, OUTMEASURE=, OUTPATH= , OUTSEQUENCE=, and OUTSUM= data sets. In general, these data sets contain the variables listed in the BY statement. The ID statement time ID variable is also included in the data sets when the time dimension is important. If an analysis step related to an output data step fails, then the values of this step are not recorded or are set to missing in the related output data set, and appropriate error and warning messages are recorded in the SAS log.

OUT= Data Set

The OUT= data set contains the variables that are specified in the BY, ID, INPUT, and TARGET statements. If the ID statement is specified, the ID variable values are aligned and extended based on the ALIGN=, INTERVAL=, START=, and END= options. The values of the variables specified in the INPUT and TARGET statements are accumulated based on the ACCUMULATE= option, missing values are interpreted based on the SETMISSING= option, and zero values are interpreted using the ZEROMISS= option. The accumulated time series is transformed based on the TRANSFORM=, DIF=, and SDIF= options.

OUTMEASURE= Data Set

The OUTMEASURE= data set records the similarity measures between each INPUT and TARGET statement variable with respect to each time ID value. The form of the OUTMEASURE= data set depends on the SORTNAMES and ORDER= options. The OUTMEASURE= data set contains the variables specified in the BY statement in addition to the variables listed below.

For ORDER=INPUTTARGET and ORDER=TARGETINPUT, the OUTMEASURE= data set has the following form:

INPUT	input variable name
TARGET	target variable name
TIMEID	time ID values
INPSEQ	input sequence values
TARSEQ	target sequence values
SIM	similarity measures

The OUTMEASURE= data set is ordered by the variables _INPUT_, then _TARGET_, then _TIMEID_ when ORDER=INPUTTARGET. The OUTMEASURE= data set is ordered by the variables _TARGET_, then _INPUT_, then _TIMEID_ when ORDER=TARGETINPUT.

For ORDER=INPUT, the OUTMEASURE= data set has the following form:

INPUT	input variable name
TIMEID	time ID values
INPSEQ	input sequence values
target-names	similarity measures that are associated with each TARGET statement variable name

The OUTMEASURE= data set is ordered by the variables _INPUT_, then _TIMEID_.

For ORDER=TARGET, the OUTMEASURE= data set has the following form:

TARGET	target variable name
TIMEID	time ID values
TARSEQ	target sequence values
input-names	similarity measures that are associated with each INPUT statement variable name

The OUTMEASURE= data set is ordered by the variables _TARGET_, then _TIMEID_.

OUTPATH= Data Set

The OUTPATH= data set records the path analysis between each INPUT and TARGET statement variable. This data set records the path sequences for each slide index and for each warp index associated with the slide

index. The sequence values recorded are normalized and scaled based on the NORMALIZE= and SCALE= options.

The OUTPATH= data set contains the variables specified in the BY statement and the following variables:

INPUT	input variable name
TARGET	target variable name
TIMEID	time ID values
SLIDE	slide index
WARP	warp index
INPSEQ	input sequence values
TARSEQ	target sequence values
INPPTH	input path index
TARPTH	target path index
METRIC	distance metric values

The Warp Index indicates the total amount of warping for each slide. A negative number represents compression of the target sequence. A positive number represents expansion of the target sequence. The Warp Index is always zero for SLIDE=NONE and SLIDE=SEASON.

The sorting of the OUTPATH= data set depends on the SORTNAMES and the ORDER= option.

The OUTPATH= data set is ordered by the variables _INPUT_, then _TARGET_, then _TIMEID_ when ORDER=INPUTTARGET or ORDER=INPUT. The OUTPATH= data set is ordered by the variables _TARGET_, then _INPUT_, then _TIMEID_ when ORDER=TARGETINPUT or ORDER=TARGET.

If there are a large number of slides or warps or both, this data set might be large.

OUTSEQUENCE= Data Set

The OUTSEQUENCE= data set records the input and target sequences that are associated with each INPUT and TARGET statement variable. This data set records the input and target sequence values for each slide index and for each warp index that is associated with the slide index. The sequence values that are recorded are normalized and scaled based on the NORMALIZE= and SCALE= options. This data set also contains the similarity measure associated with the two sequences.

The OUTSEQUENCE= data set contains the variables specified in the BY statement in addition to the following variables:

INPUT	input variable name
TARGET	target variable name
TIMEID	time ID values
SLIDE	slide index
WARP	warp index

INPSEQ	input sequence values
TARSEQ	target sequence values
SIM	similarity measure
STATUS	sequence status

The sorting of the OUTSEQUENCE= data set depends on the SORTNAMES and the ORDER= option.

The OUTSEQUENCE= data set is ordered by the variables _INPUT_, then _TARGET_, then _TIMEID_ when ORDER=INPUTTARGET or ORDER=INPUT. The OUTSEQUENCE= data set is ordered by the variables _TARGET_, then _INPUT_, then _TIMEID_ when ORDER=TARGETINPUT or ORDER=TARGET.

If there are a large number of slides or warps or both, this data set might be large.

OUTSUM= Data Set

The OUTSUM= data set summarizes the similarity measures between each INPUT and TARGET statement variable. The form of the OUTSUM= data set depends on the SORTNAMES and ORDER= option. If the SORTNAMES option is specified, each variable (INPUT or TARGET) is analyzed in ascending order. The OUTSUM= data set contains the variables specified in the BY statement in addition to the variables listed below.

For ORDER=INPUTTARGET and ORDER=TARGETINPUT, the OUTSUM= data set has the following form:

INPUT	input variable name
TARGET	target variable name
STATUS	status flag that indicates whether the requested analyses were successful
TIMEID	time ID values
SIM	similarity measure summary

The OUTSUM= data set is ordered by the variables _INPUT_, then _TARGET_ when ORDER=INPUTTARGET. The OUTSUM= data set is ordered by the variables _TARGET_, then _INPUT_ when ORDER=TARGETINPUT.

For ORDER=INPUT, the OUTSUM= data set has the following form:

INPUT	input variable name
STATUS	status flag that indicates whether the requested analyses were successful
target-names	similarity measure summary that is associated with each TARGET statement variable name

The OUTSUM= data set is ordered by the variable _INPUT_.

For ORDER=TARGET, the OUTSUM= data set has the following form:

TARGET	target variable name
STATUS	status flag that indicates whether the requested analyses were successful
input-names	similarity measure summary that is associated with each INPUT statement variable name

The OUTSUM= data set is ordered by the variable _TARGET_.

STATUS Variable Values

The _STATUS_ variable contains a code that specifies whether the similarity analysis has been successful or not. The _STATUS_ variable can take the following values:

0	Success
3000	Accumulation failure
4000	Missing value interpretation failure
6000	Series is all missing
7000	Transformation failure
8000	Differencing failure
9000	Unable to compute descriptive statistics
10000	Normalization failure
11000	Input contains imbedded missing values
12000	Target contains imbedded missing values
13000	Scaling failure
14000	Measure failure
15000	Path failure
16000	Slide summarization failure

Printed Output

The SIMILARITY procedure optionally produces printed output by using the Output Delivery System (ODS). By default, the procedure produces no printed output. All output is controlled by the PRINT= and PRINTDETAILS options in the PROC SIMILARITY statement.

The sort, order, and form of the printed output depends on both the SORTNAMES option and the ORDER= option. If the SORTNAMES option is specified, each variable (INPUT or TARGET) is analyzed in ascending order. For ORDER=INPUTTARGET, the printed output is ordered by the INPUT statement variables (row) and then by the TARGET statement variables (row). For ORDER=TARGETINPUT, the printed output is ordered by the TARGET statement variables (row) and then by the INPUT statement variables (row). For ORDER=INPUT, the printed output is ordered by the INPUT statement variables (row) and then by the TARGET statement variables (column). For ORDER=TARGET, the printed output is ordered by the TARGET statement variables (row) and then by the INPUT statement variables (column).

In general, if an analysis step related to printed output fails, the values of that step are not printed and appropriate error and warning messages are recorded in the SAS log. The printed output is similar to the output data set; these similarities are described as follows:

PRINT=COSTS

prints the costs statistics.

PRINT=DESCSTATS

prints the descriptive statistics.

PRINT=PATHS

prints the path statistics.

PRINT=SLIDES

prints the sliding sequence summary.

PRINT=SUMMARY

prints the summary of similarity measures similar to the OUTSUM= data set.

PRINT=WARPS

prints the warp summary.

PRINTDETAILS

prints each table with greater detail.

ODS Table Names

The following table relates the PRINT= options to ODS tables.

Table 31.2 ODS Tables Produced in PROC SIMILARITY

ODS Table Name	Description	Option
CostStatistics	Cost statistics	PRINT=COSTS
DescStats	Descriptive statistics	PRINT=DESCSTATS
PathLimits	Path limits	PRINT=PATHS
PathStatistics	Path statistics	PRINT=PATHS
SlideMeasuresSummary	Summary of measure per slide	PRINT=SLIDES
MeasuresSummary	Measures summary	PRINT=SUMMARY
InputMeasuresSummary	Measures summary	PRINT=SUMMARY
TargetMeasuresSummary	Measures summary	PRINT=SUMMARY
WarpMeasuresSummary	Summary of measure per warp	PRINT=WARPS

The tables are related to a single series within a BY group.

ODS Graphics

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*).

Before you create graphs, ODS Graphics must be enabled (for example, with the ODS GRAPHICS ON statement). For more information about enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" in that chapter.

The overall appearance of graphs is controlled by ODS styles. Styles and other aspects of using ODS Graphics are discussed in the section "A Primer on ODS Statistical Graphics" in that chapter.

This section describes the use of ODS for creating graphics with the SIMILARITY procedure.

ODS Graph Names

PROC SIMILARITY assigns a name to each graph it creates by using ODS. You can use these names to selectively reference the graphs. The names are listed in Table 31.3.

Table 31.3 ODS Graphics Produced by PROC SIMILARITY

ODS Graph Name	Plot Description	Statement	PLOTS= Option
CostsPlot	Costs plot	SIMILARITY	PLOTS=COSTS
NormalizedSequencePlot	Normalized Sequence Plot	SIMILARITY	PLOTS=NORMALIZED
PathDistancePlot	Path distances plot	SIMILARITY	PLOTS=DISTANCES
PathDistanceHistogram	Path distances histogram	SIMILARITY	PLOTS=DISTANCES
PathRelativeDistancePlot	Path relative distances plot	SIMILARITY	PLOTS=DISTANCES
PathRelativeDistanceHistogram	Path relative distances histogram	SIMILARITY	PLOTS=DISTANCES
PathPlot	Path plot	SIMILARITY	PLOTS=PATHS
PathSequencesPlot	Path sequences plot	SIMILARITY	PLOTS=MAPS
PathSequencesScaledPlot	Scaled path sequences map plot	SIMILARITY	PLOTS=MAPS
ScaledSequencePlot	Scaled Sequence plot	SIMILARITY	PLOTS=SCALED
SequencePlot	Sequence plot	SIMILARITY	PLOTS=SEQUENCES
SeriesPlot	Input time series plot	SIMILARITY	PLOTS=INPUTS
SimilarityPlot	Similarity measures plot	SIMILARITY	PLOTS=MEASURES
TargetSequencePlot	Target sequence plot	SIMILARITY	PLOTS=TARGETS
WarpPlot	Warping plot	SIMILARITY	PLOTS=WARPS
WarpScaledPlot	Scaled warping plot	SIMILARITY	PLOTS=WARPS

Time Series Plots

The time series plots (SeriesPlot) illustrate the input time series to be compared. The horizontal axis represents the input series time ID values, and the vertical axis represents the input series values.

Sequence Plots

The sequence plots (SequencePlot) illustrate the target and input sequences to be compared. The horizontal axis represents the (target or input) sequence index, and the vertical axis represents the (target or input) sequence values.

Path Plots

The path plot (PathPlot) and path limits plot (PathLimitsPlot) illustrate the path through the distance matrix. The horizontal axis represents the input sequence index, and the vertical axis represents the target sequence index. The dots represent the path coordinates. The upper parallel line represents the compression limit, and the lower parallel line represents the expansion limit. These plots visualize the path through the distance matrix. Vertical movements indicate compression, and horizontal movements represent expansion of the target sequence with respect to the input sequence. These plots are useful for visualizing the amount of expansion and compression along the path.

Time Warp Plots

The time warp plot (WarpPlot) and scaled time warp plot (WarpScaledPlot) illustrate the time warping. The horizontal axis represents the (input and target) sequence index. The upper line plot represents the target sequence. The lower line plot represents the input sequence. The lines that connect the input and target sequence values represent the mapping between the input and target sequence indices along the optimal path. These plots visualize the warping of the time index with respect to the input and target sequence values. Expansion of a single target sequence value occurs when it is mapped to more than one input sequence value. Expansion of a single input sequence value occurs when it is mapped to more than one target sequence value. The plots are useful for visualizing the mapping between the input and target sequence values along the path. The plots are useful for comparing the path sequences or input and target sequence after time warping.

Path Sequence Plots

The path sequence plot (PathSequencesPlot) and scaled path sequence plot (PathSequencesScaledPlot) illustrate the sequence mapping along the optimal path. The horizontal axis represents the path index. The dashed line represents the time warped input sequence. The solid line represents the time warped target sequence. These plots visualize the mapping between the input and target sequence values with respect to the path index. The scaled plot with the input and target sequence values are scaled and evenly separated for visual convenience.

Path Distance Plots

The path distance plots (PathDistancePlot) and path relative distance plots (PathRelativeDistancePlot) illustrate the path (relative) distances. The horizontal axis represents the path index. The vertical needles represent the (relative) distances. The horizontal reference lines indicate one and two standard deviations.

The path distance histogram (PathDistanceHistogram) and path relative distance histogram (PathDistanceRelativeHistogram) illustrate the distribution of the path (relative) distances. The bars represent the histogram, and the solid line represents a normal distribution with the same mean and variance.

Cost Plots

The cost plot (CostPlot) and cost limits plot (CostPlot) illustrate the cost of traversing the distance matrix. The horizontal axis represents the input sequence index, and the vertical axis represents the target sequence index. The colors and shading within the plot illustrate the incremental cost of traversing the distance matrix. The upper parallel line represents the compression limit, and the lower parallel line represents the expansion limit.

Examples: SIMILARITY Procedure

Example 31.1: Accumulating Transactional Data into Time Series Data

This example uses the SIMILARITY procedure to illustrate the accumulation of time-stamped transactional data that has been recorded at no particular frequency into time series data at a specific frequency. After the time series is created, the various SAS/ETS procedures related to time series analysis, similarity analysis, seasonal adjustment and decomposition, modeling, and forecasting can be used to further analyze the time series data.

Suppose that the input data set WORK.RETAIL contains variables STORE and TIMESTAMP and numerous other numeric transaction variables. The BY variable STORE contains values that break up the transactions into groups (BY groups). The time ID variable TIMESTAMP contains SAS date values recorded at no particular frequency. The other data set variables contain the numeric transaction values to be analyzed. It is further assumed that the input data set is sorted by the variables STORE and TIMESTAMP.

The following statements form monthly time series from the transactional data based on the median value (ACCUMULATE=MEDIAN) of the transactions recorded with each time period. The accumulated time series values for time periods with no transactions are set to zero instead of missing (SETMISS=0). Only transactions recorded between the first day of 1998 (START='01JAN1998'D) and last day of 2000 (END='31JAN2000'D) are considered and if needed are extended to include this range.

```
proc similarity data=work.retail out=mseries;
   by store;
   id timestamp interval=month
                accumulate=median
                setmiss=0
                start='01jan1998'd
                end  ='31dec2000'd;
   target _NUMERIC_;
run;
```

The monthly time series data are stored in the data WORK.MSERIES. Each BY group associated with the BY variable STORE contains an observation for each of the 36 months associated with the years 1998, 1999, and 2000. Each observation contains the variable STORE, TIMESTAMP, and each of the analysis variables in the input DATA= data set.

After each set of transactions has been accumulated to form the corresponding time series, the accumulated time series can be analyzed by using various time series analysis techniques. For example, exponentially weighted moving averages can be used to smooth each series. The following statements use the EXPAND procedure to smooth the analysis variable named STOREITEM.

```
proc expand data=mseries
            out=smoothed
            from=month;
   by store;
   id timestamp;
   convert storeitem=smooth / transform=(ewma 0.1);
run;
```

The smoothed series is stored in the data set WORK.SMOOTHED. The variable SMOOTH contains the smoothed series.

If the time ID variable TIMESTAMP contains SAS datetime values instead of SAS date values, the INTERVAL= , START=, and END= options in the SIMILARITY procedure must be changed accordingly, and the following statements could be used to accumulate the datetime transactions to a monthly interval:

```
proc similarity data=work.retail
                out=tseries;
   by store;
   id timestamp interval=dtmonth
                accumulate=median
                setmiss=0
                start='01jan1998:00:00:00'dt
                end  ='31dec2000:00:00:00'dt;
   target _NUMERIC_;
run;
```

The monthly time series data are stored in the data WORK.TSERIES, and the time ID values use a SAS datetime representation.

Example 31.2: Similarity Analysis

This simple example illustrates how to use similarity analysis to compare two time sequences. The following statements create an example data set that contains two time sequences of differing lengths:

```
data test;
input i y x;
datalines;
1   2  3
2   4  5
3   6  3
4   7  3
5   3  3
6   8  6
7   9  3
8   3  8
9  10  .
10 11  .
;
run;
```

The following statements perform similarity analysis on the example data set:

```
proc similarity data=test out=_null_
   print=all plot=all;
   input x;
   target y / measure=absdev;
run;
```

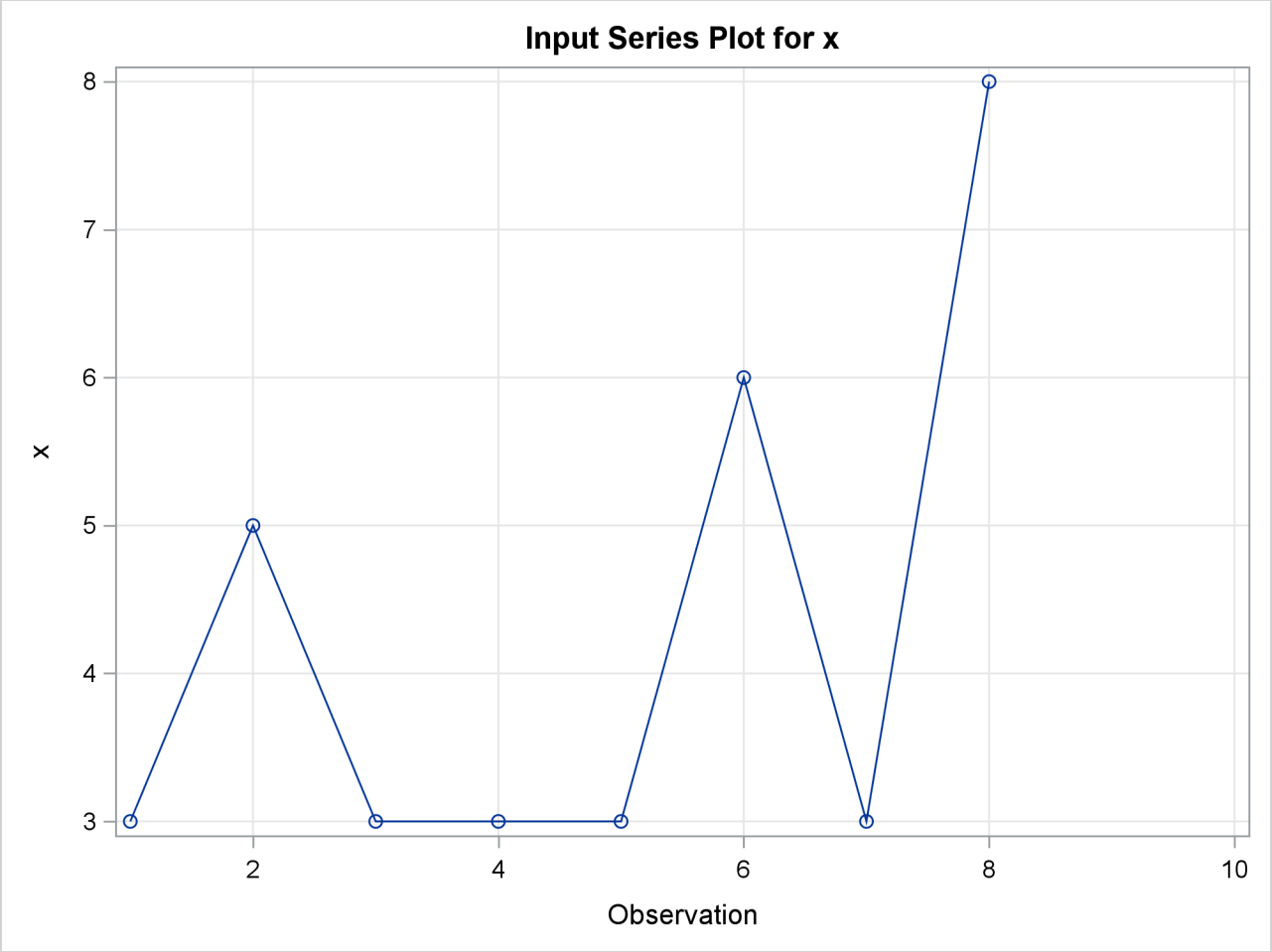
The DATA=TEST option specifies that the input data set WORK.TEST is to be used in the analysis. The OUT=_NULL_ option specifies that no output time series data set is to be created. The PRINT=ALL and PLOTS=ALL options specify that all ODS tables and graphs are to be produced. The INPUT statement specifies that the input variable is X. The TARGET statement specifies that the target variable is Y and that the similarity measure is computed using absolute deviation (MEASURE=ABSDEV).

Output 31.2.1 Description Statistics of the Input Variable, x

The SIMILARITY Procedure

Time Series Descriptive Statistics	
Variable	x
Number of Observations	10
Number of Missing Observations	2
Minimum	3
Maximum	8
Mean	4.25
Standard Deviation	1.908627

Output 31.2.2 Plot of Input Variable, x



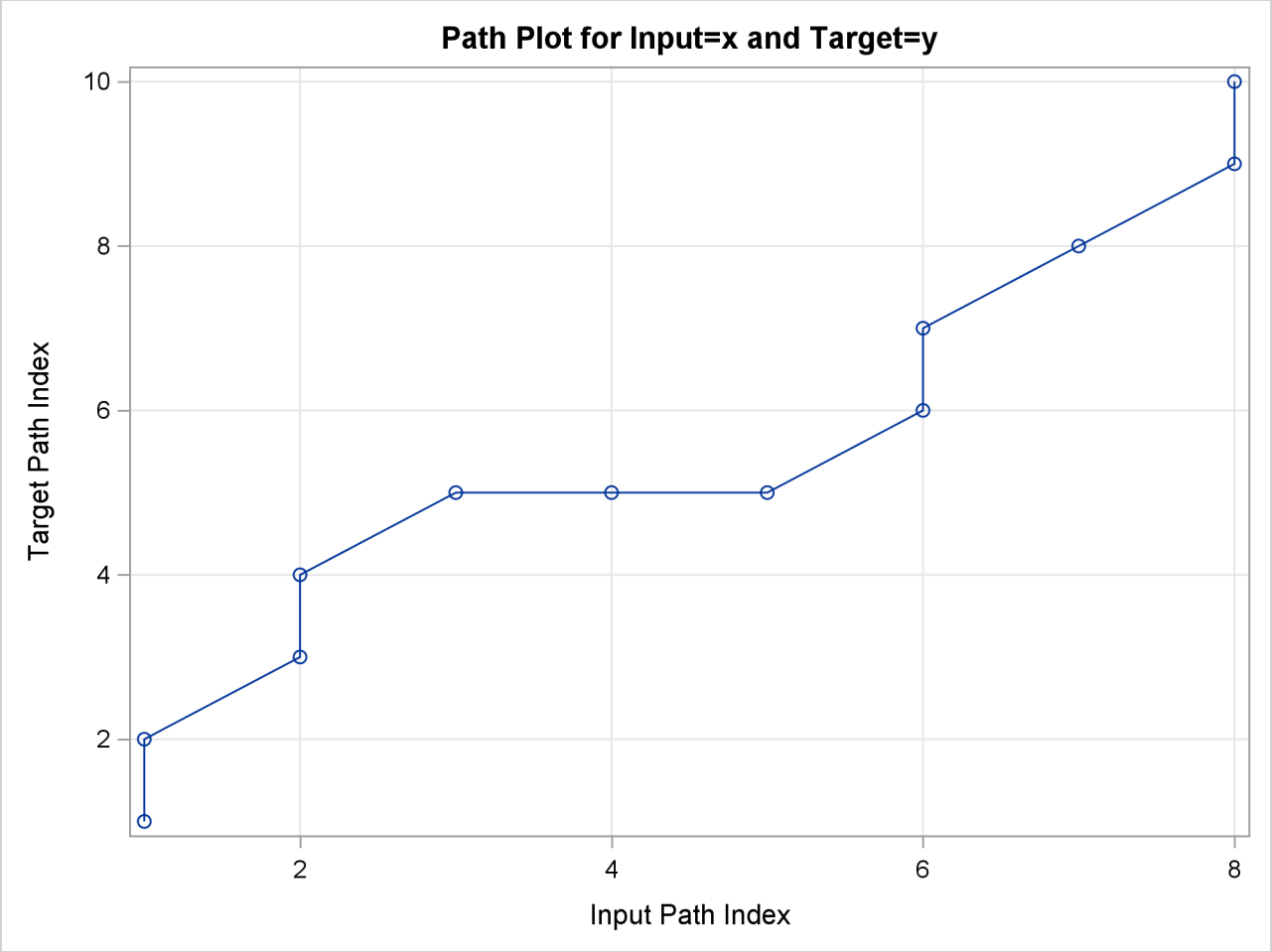
Output 31.2.3 Target Sequence Plot



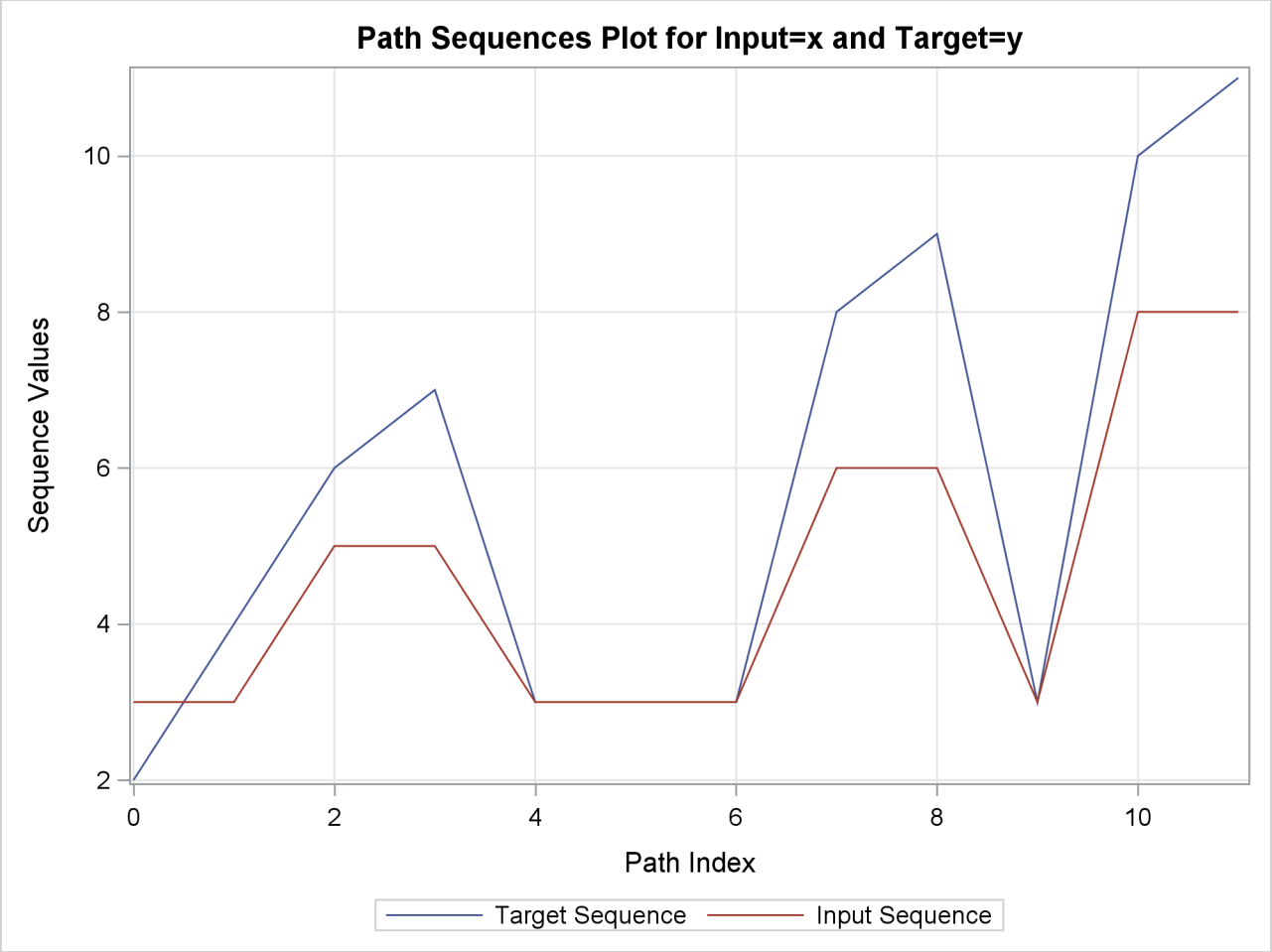
Output 31.2.4 Sequence Plot



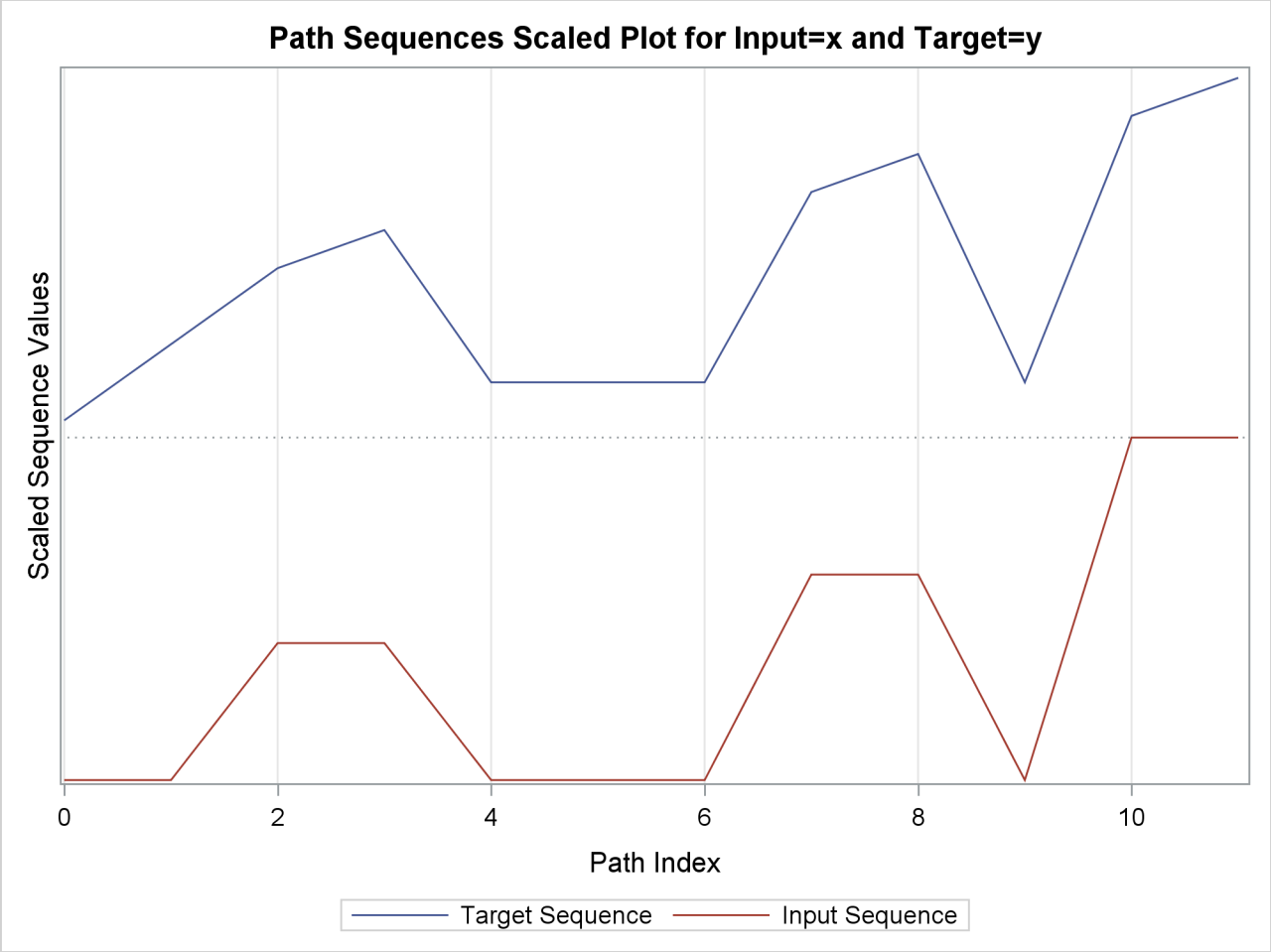
Output 31.2.5 Path Plot



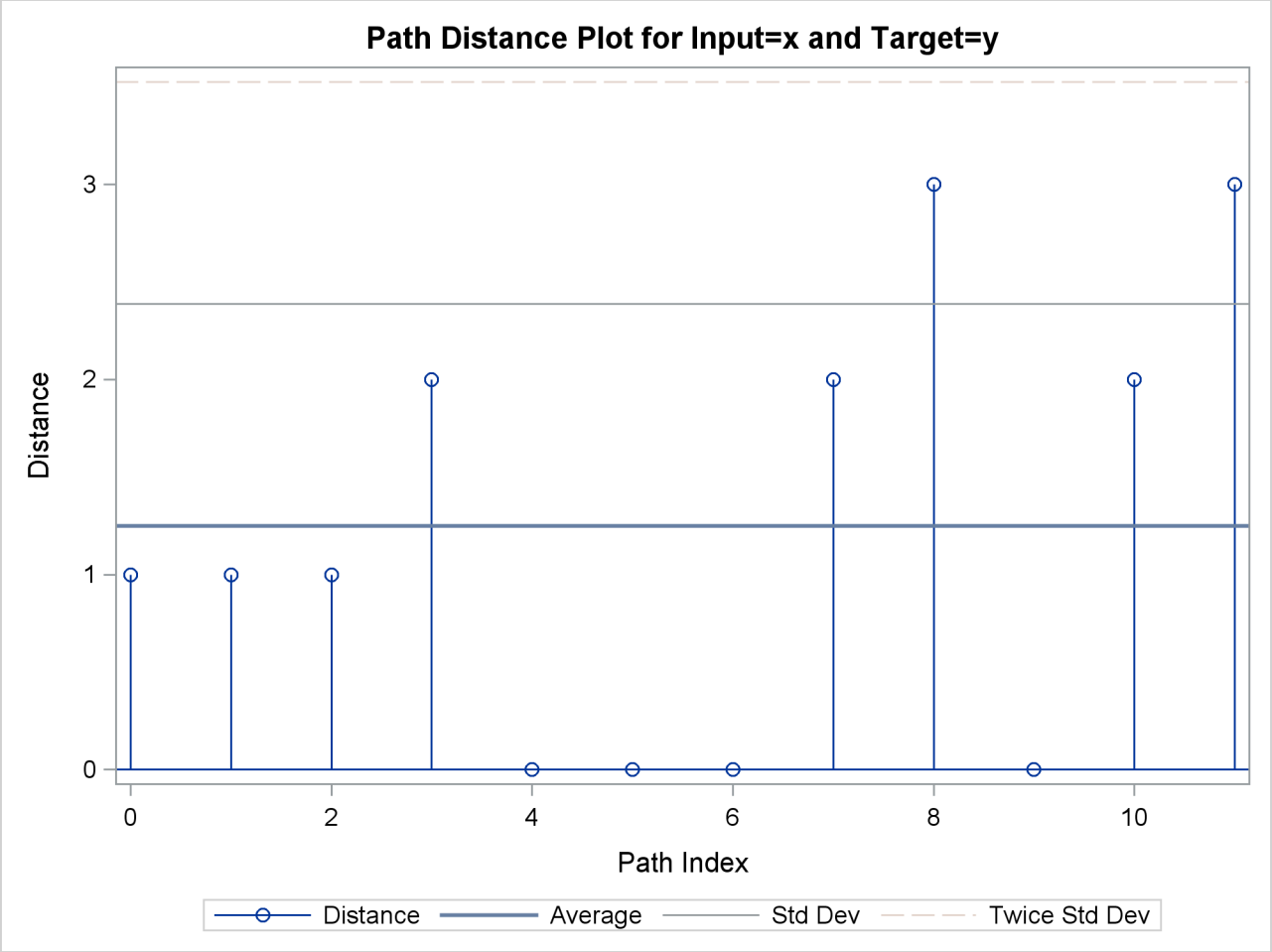
Output 31.2.6 Path Sequences Plot



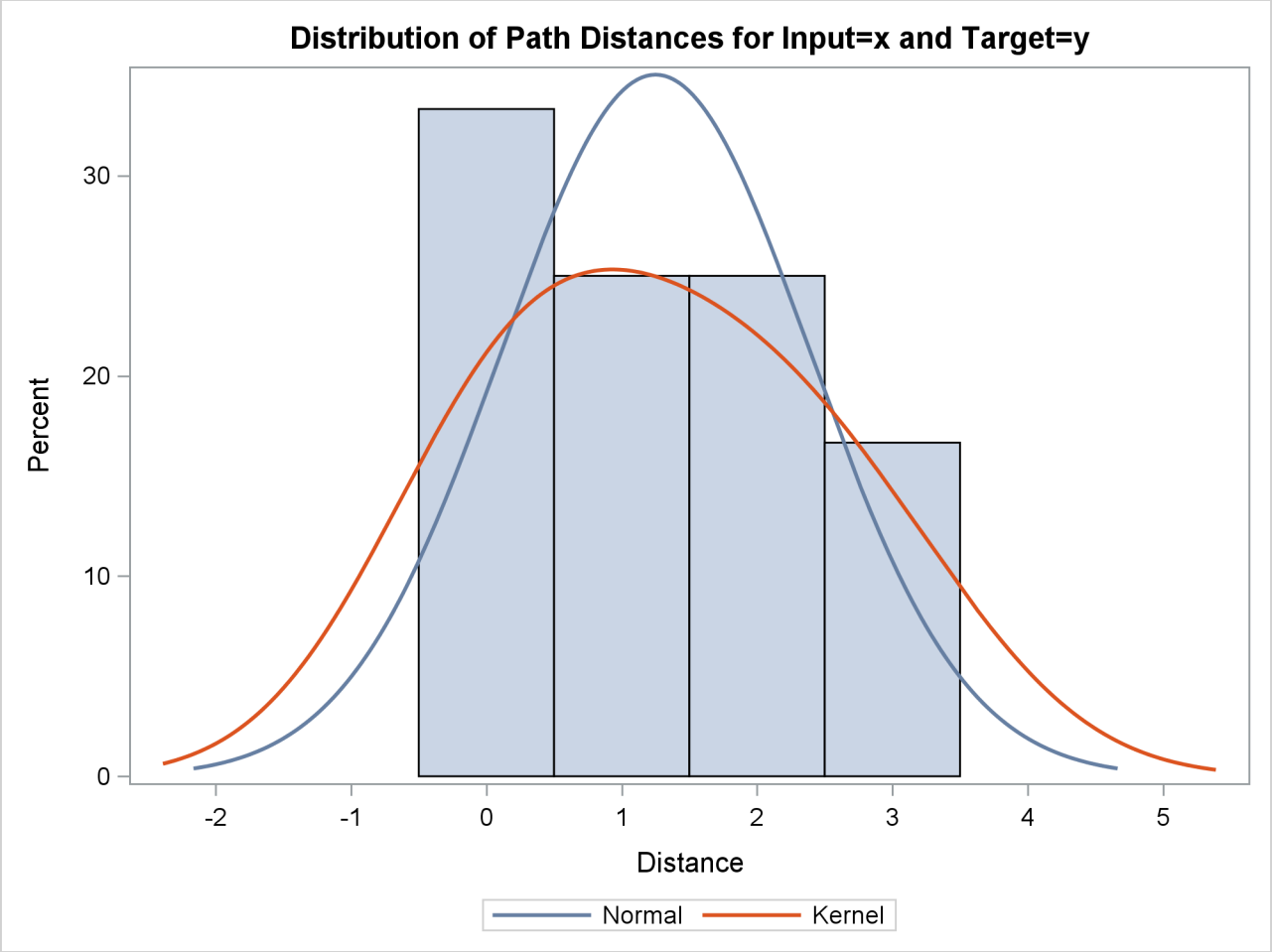
Output 31.2.7 Path Sequences Scaled Plot



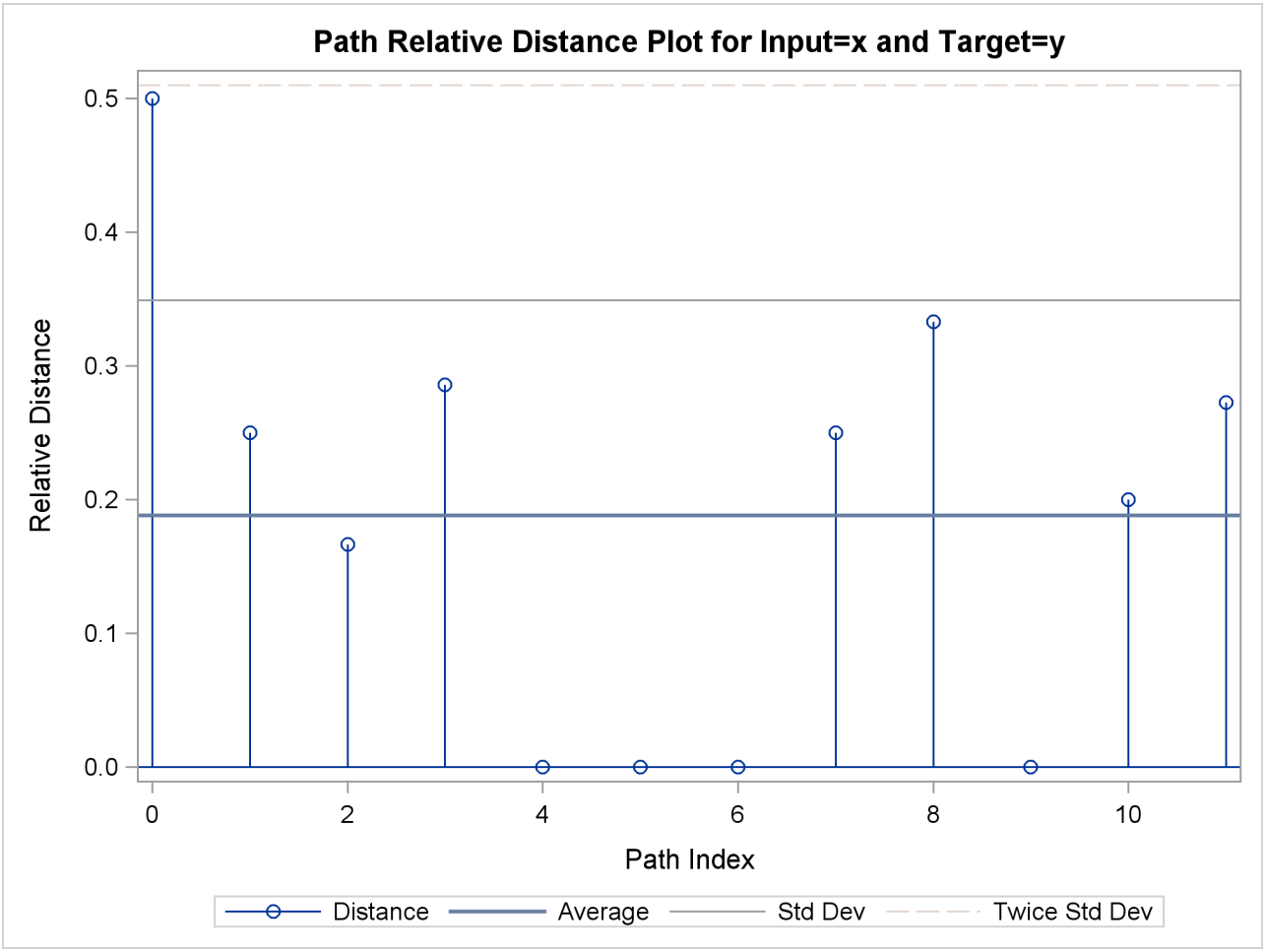
Output 31.2.8 Path Distance Plot



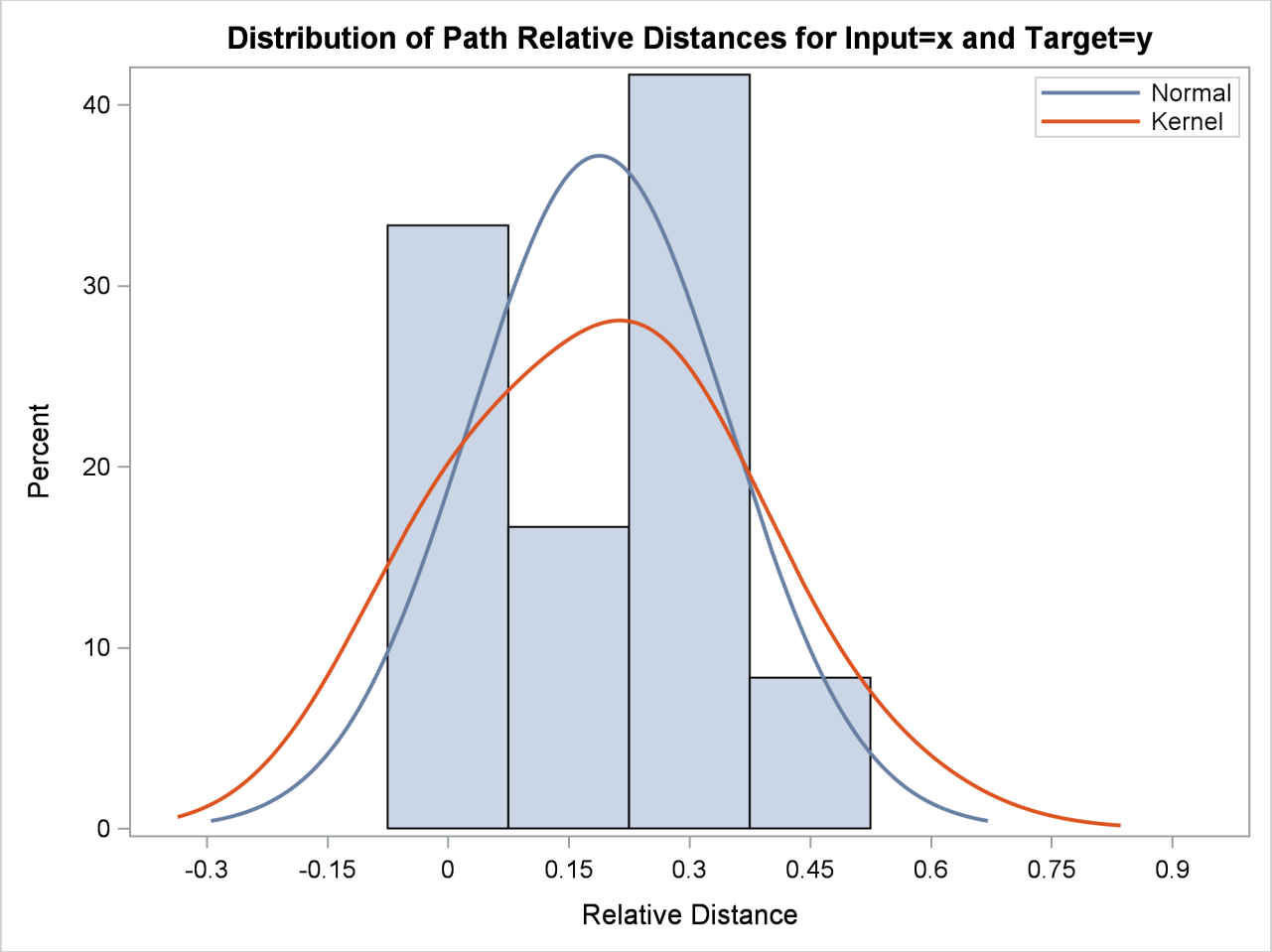
Output 31.2.9 Path Distance Histogram



Output 31.2.10 Path Relative Distance Plot



Output 31.2.11 Path Relative Distance Histogram



Output 31.2.12 Path Limits

Path Limits

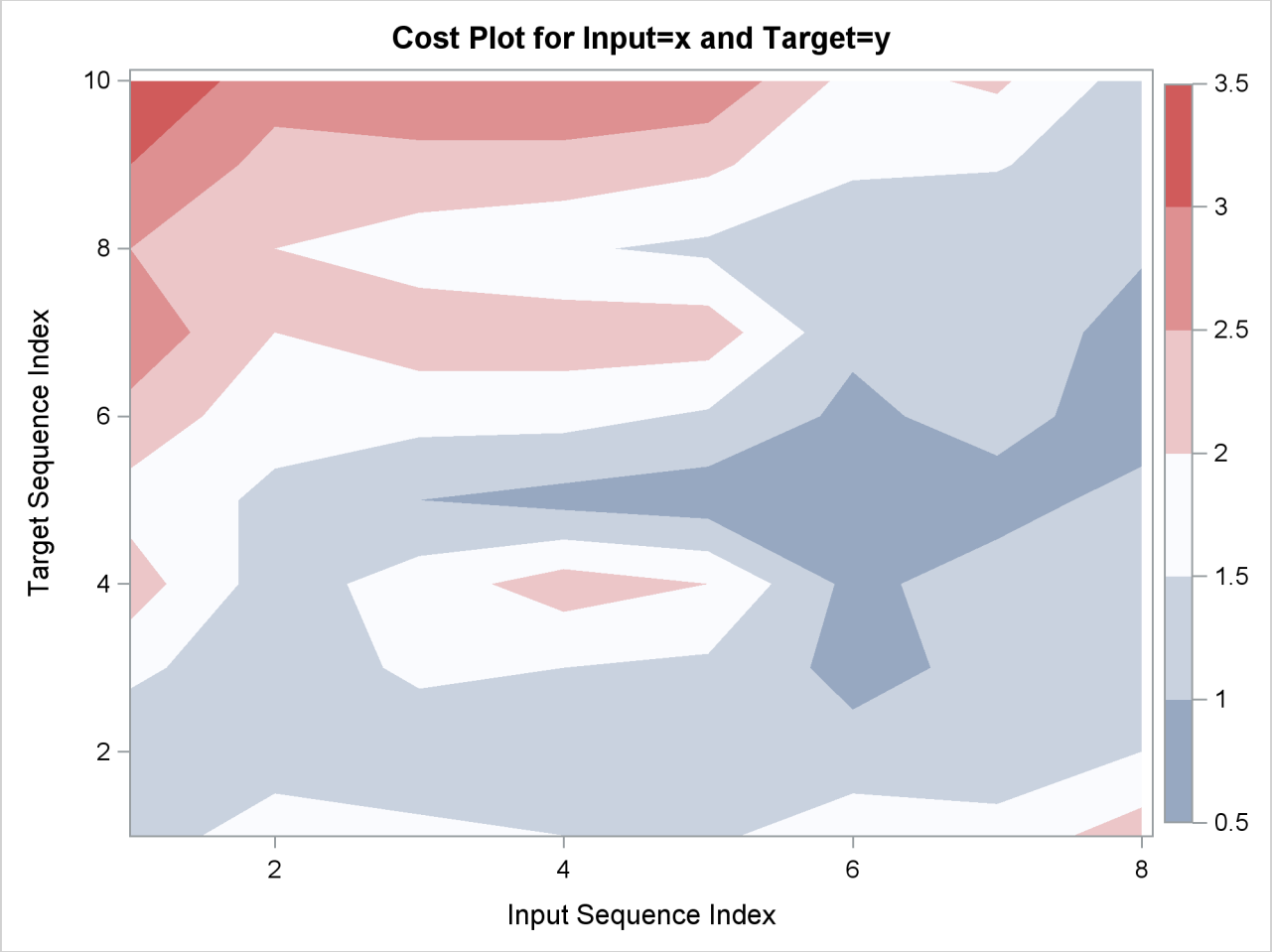
Limit	Specified Absolute	Specified Percentage	Minimum Allowed	Maximum Allowed	Applied
Compression	None	None	2	9	9
Expansion	None	None	0	7	7

Output 31.2.13 Path Statistics

Path Statistics

Path	Number	Path Percent	Input Percent	Target Percent	Maximum	Path Maximum Percent	Input Maximum Percent	Target Maximum Percent
Missing Map	0	0.000%	0.000%	0.000%	0	0.000%	0.000%	0.000%
Direct Maps	6	50.00%	75.00%	60.00%	2	16.67%	25.00%	20.00%
Compression	4	33.33%	50.00%	40.00%	1	8.333%	12.50%	10.00%
Expansion	2	16.67%	25.00%	20.00%	2	16.67%	25.00%	20.00%
Warps	6	50.00%	75.00%	60.00%	2	16.67%	25.00%	20.00%

Output 31.2.14 Cost Plot



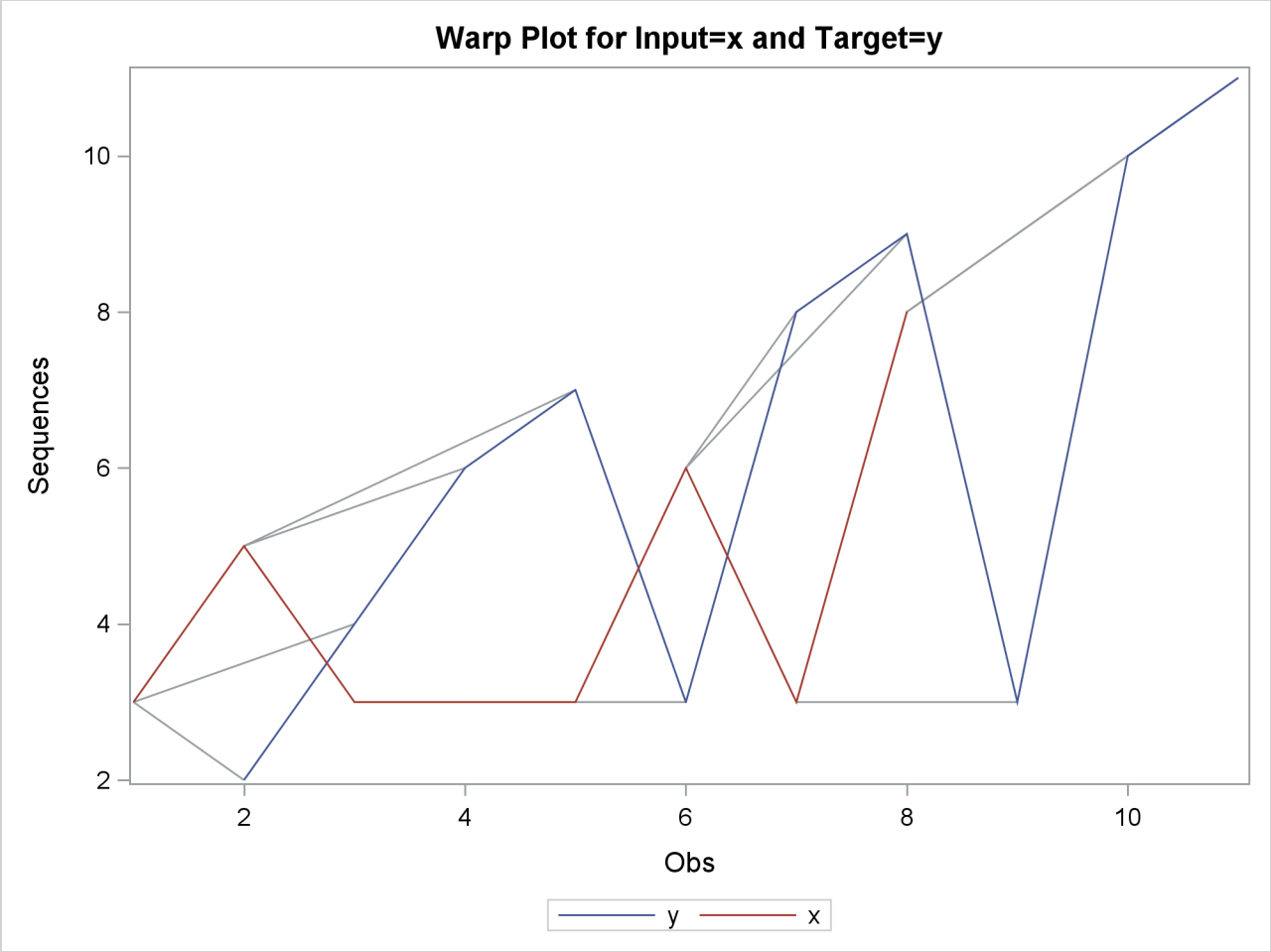
Output 31.2.15 Cost Statistics

Cost Statistics

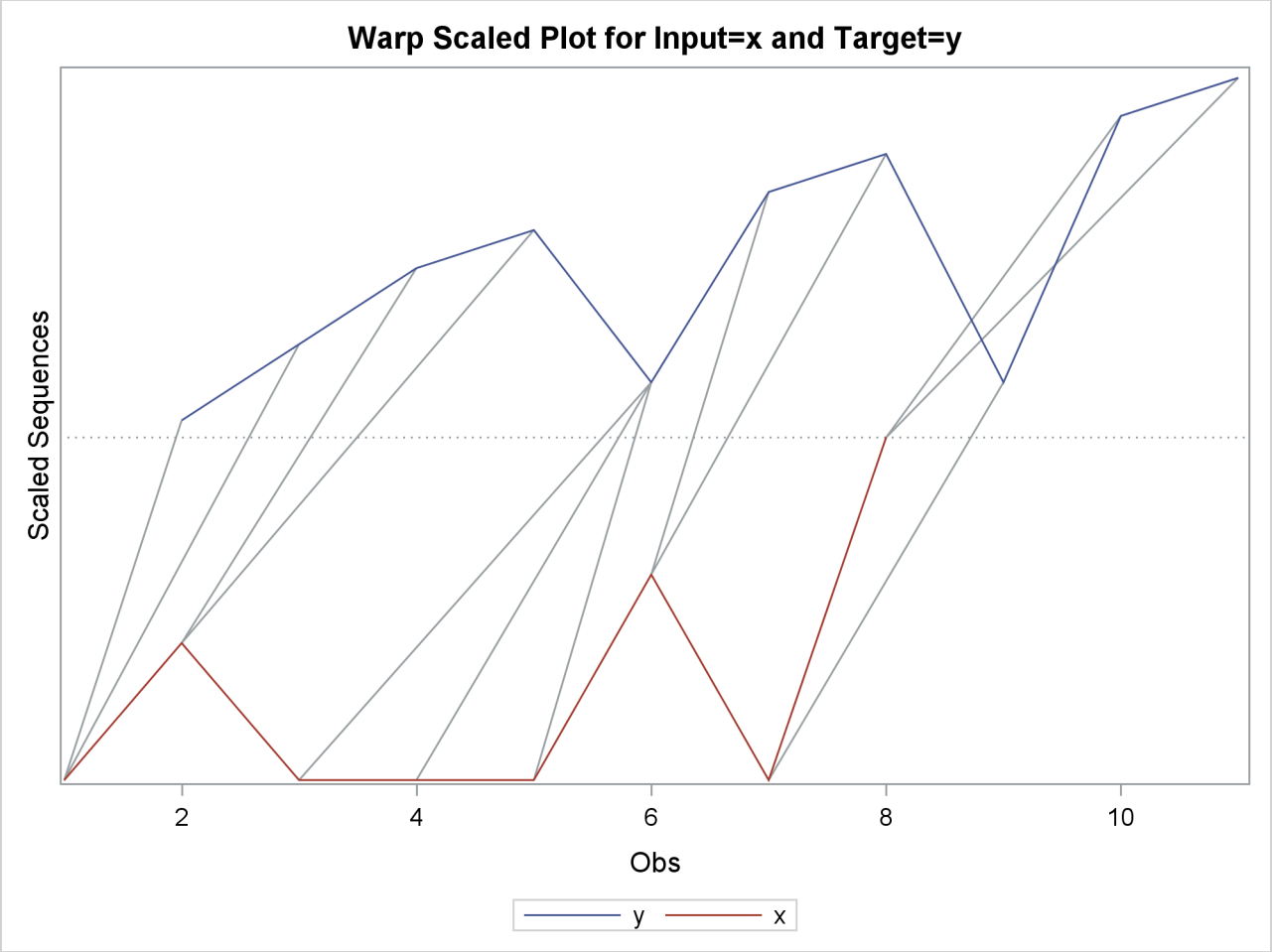
Cost	Number	Total	Average	Standard Deviation	Minimum	Maximum	Input Mean	Target Mean	Minimum Path Mean	Maximum Path Mean
Absolute	12	15.00000	1.250000	1.138180	0	3.000000	1.875000	1.500000	1.875000	0.8823529
Relative	12	2.25844	0.188203	0.160922	0	0.500000	0.282305	0.225844	0.282305	0.1328495

Relative Costs based on Target Sequence values

Output 31.2.16 Time Warp Plot



Output 31.2.17 Time Warp Scaled Plot

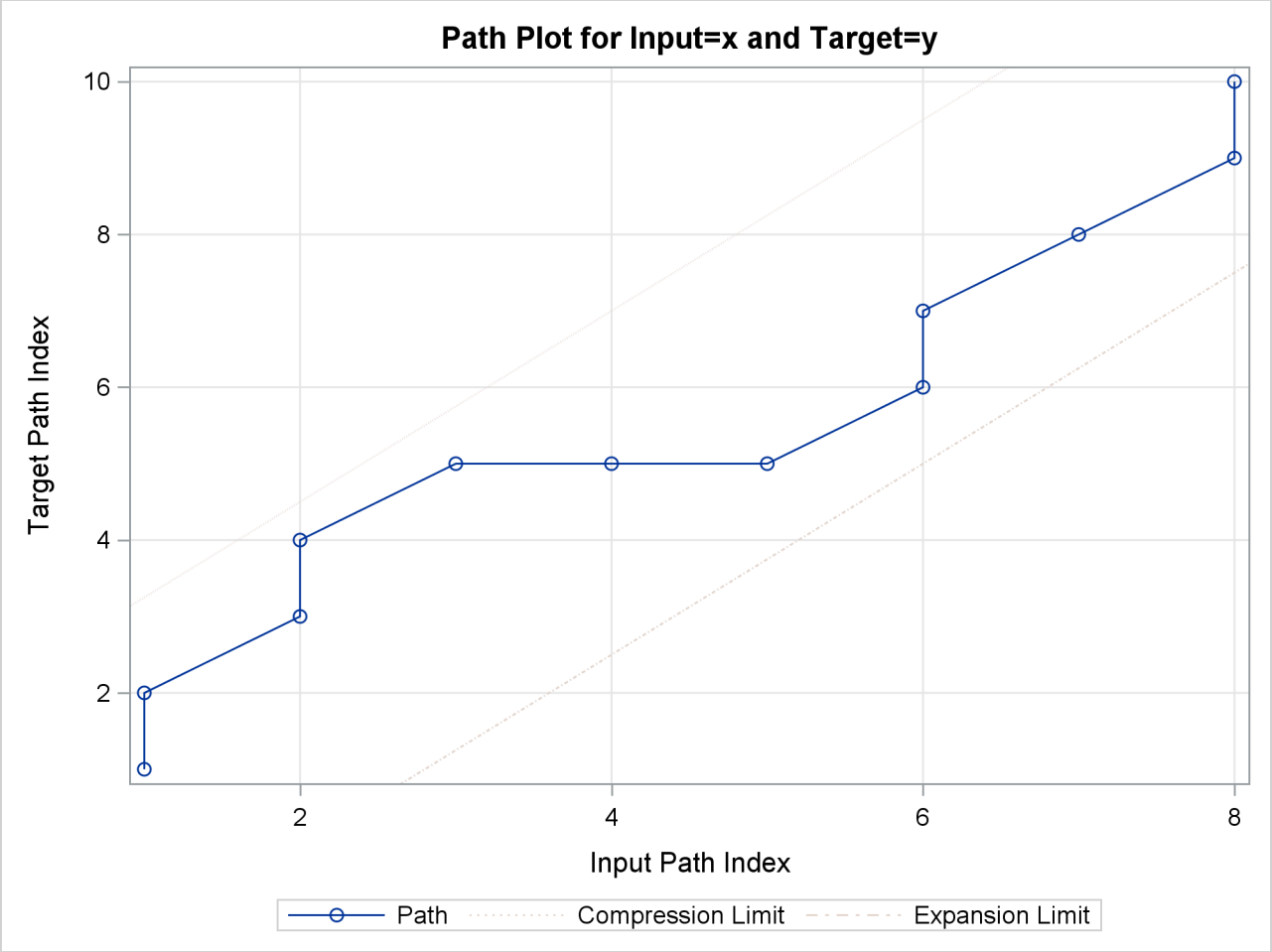


The following statements repeat the preceding similarity analysis on the example data set with warping limits:

```
proc similarity data=test out=_null_
   print=all plot=all;
   input x;
   target y / measure=absdev
              compress=(localabs=2)
              expand=(localabs=2);
run;
```

The COMPRESS=(LOCALABS=2) option limits local absolute compression to 2. The EXPAND=(LOCALABS=2) option limits local absolute expansion to 2.

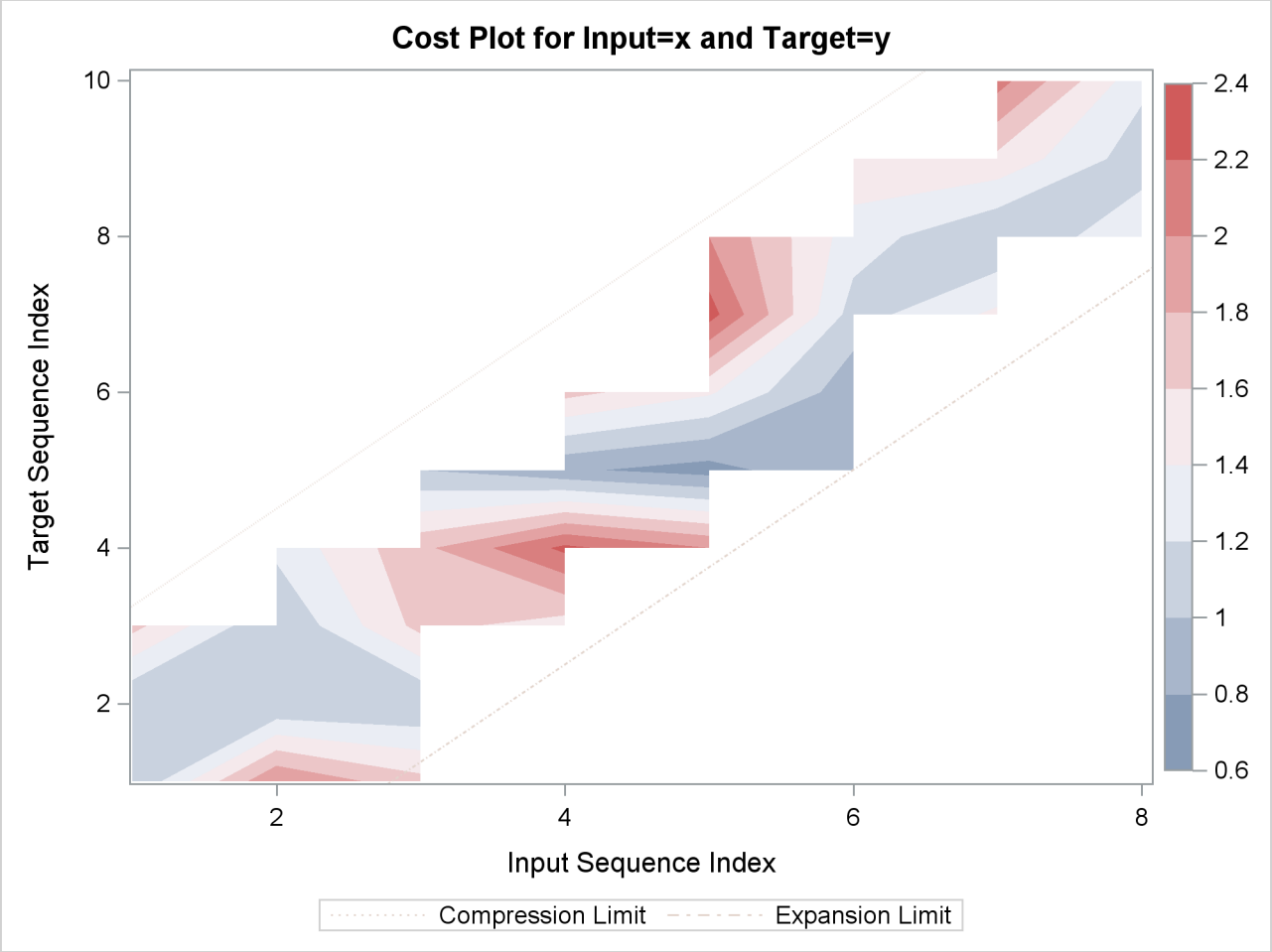
Output 31.2.18 Path Plot with Warping Limits



Output 31.2.19 Warped Path Limits

	Path Limits				
Limit	**Specified Absolute**	**Specified Percentage**	**Minimum Allowed**	**Maximum Allowed**	**Applied**
Compression	2	None	2	9	2
Expansion	2	None	0	7	2

Output 31.2.20 Cost Plot with Warping Limits



The following statements repeat the preceding similarity analysis on the example data set but store the results in output data sets:

```
proc similarity data=test out=series
   outsequence=sequences outpath=path outsum=summary;
   input x;
   target y / measure=absdev
              compress=(localabs=2)
              expand=(localabs=2);
run;
```

The OUT=SERIES, OUTSEQUENCE=SEQUENCES, OUTPATH=PATH, and OUTSUM=SUMMARY options specify that the output time series, time sequences, path analysis, and summary data sets be created, respectively.

Example 31.3: Sliding Similarity Analysis

This example illustrates how to use sliding similarity analysis to compare two time sequences. The SASHELP.WORKERS data set contains two similar time series variables (ELECTRIC and MASONRY), which represent employment over time. The following statements create an example data set that contains two time series of differing lengths, where the variable MASONRY has the first 12 and last 7 observations set to missing to simulate the lack of data associated with the target series:

```
data workers; set sashelp.workers;
   if '01JAN1978'D <= date < '01JAN1982'D then masonry = masonry;
   else masonry = .;
run;
```

The goal of sliding similarity measures analysis is find the slide index that corresponds to the most similar subsequence of the input series when compared to the target sequence. The following statements perform sliding similarity analysis on the example data set:

```
proc similarity data=workers out=_NULL_ print=(slides summary);
   id date interval=month;
   input  electric;
   target masonry / slide=index measure=msqrdev
                    expand=(localabs=3 globalabs=3)
                    compress=(localabs=3 globalabs=3);
run;
```

The DATA=WORKERS option specifies that the input data set WORK.WORKERS is to be used in the analysis. The OUT=_NULL_ option specifies that no output time series data set is to be created. The PRINT=(SLIDES SUMMARY) option specifies that the ODS tables related to the sliding similarity measures and their summary be produced. The INPUT statement specifies that the input variable is ELECTRIC. The TARGET statement specifies that the target variable is MASONRY and that the similarity measure be computed using mean squared deviation (MEASURE=MSQRDEV). The SLIDE=INDEX option specifies observation index sliding. The COMPRESS=(LOCALABS=3 GLOBALABS=3) option limits local and global absolute compression to 3. The EXPAND=(LOCALABS=3 GLOBALABS=3) option limits local and global absolute expansion to 3.

Output 31.3.1 Summary of the Slide Measures

The SIMILARITY Procedure

Slide Measures Summary for Input=ELECTRIC and Target=MASONRY

Slide Index	DATE	Slide Target Sequence Length	Slide Input Sequence Length	Slide Warping Amount	Slide Minimum Measure
0	JAN1977	48	51	3	497.6737
1	FEB1977	48	51	1	482.6777
2	MAR1977	48	51	0	474.1251
3	APR1977	48	51	0	490.7792
4	MAY1977	48	51	-2	533.0788
5	JUN1977	48	51	-3	605.8198
6	JUL1977	48	51	-3	701.7138
7	AUG1977	48	51	3	646.5918
8	SEP1977	48	51	3	616.3258
9	OCT1977	48	51	3	510.9836
10	NOV1977	48	51	3	382.1434
11	DEC1977	48	51	3	340.4702
12	JAN1978	48	51	2	327.0572
13	FEB1978	48	51	1	322.5460
14	MAR1978	48	51	0	325.2689
15	APR1978	48	51	-1	351.4161
16	MAY1978	48	51	-2	398.0490
17	JUN1978	48	50	-3	471.6931
18	JUL1978	48	49	-3	590.8089
19	AUG1978	48	48	0	595.2538
20	SEP1978	48	47	-1	689.2233
21	OCT1978	48	46	-2	745.8891
22	NOV1978	48	45	-3	679.1907

Output 31.3.2 Minimum Measure

Minimum Measure Summary

Input Variable	MASONRY
ELECTRIC	322.5460

This analysis results in 23 slides based on the observation index. The minimum measure (322.5460) occurs at slide index 13 which corresponds to the time value FEB1978. Note that the original data set SASHELP.WORKERS was modified beginning at the time value JAN1978. This similarity analysis justifies the belief the ELECTRIC lags MASONRY by one month based on the time series cross-correlation analysis despite the lack of target data (MASONRY).

The goal of seasonal sliding similarity measures is to find the seasonal slide index that corresponds to the most similar seasonal subsequence of the input series when compared to the target sequence. The following statements repeat the preceding similarity analysis on the example data set with seasonal sliding:

```
proc similarity data=workers out=_NULL_ print=(slides summary);
   id date interval=month;
   input  electric;
   target masonry / slide=season measure=msqrdev;
run;
```

Output 31.3.3 Summary of the Seasonal Slide Measures

The SIMILARITY Procedure

Slide Measures Summary for Input=ELECTRIC and Target=MASONRY

Slide Index	DATE	Slide Target Sequence Length	Slide Input Sequence Length	Slide Warping Amount	Slide Minimum Measure
0	JAN1977	48	48	0	1040.086
12	JAN1978	48	48	0	641.927

Output 31.3.4 Seasonal Minimum Measure

Minimum Measure Summary

Input Variable	MASONRY
ELECTRIC	641.9273

The analysis differs from the previous analysis in that the slides are performed based on the seasonal index (SLIDE=SEASON) with no warping. With a seasonality of 12, two seasonal slides are considered at slide indices 0 and 12 with the minimum measure (641.9273) occurring at slide index 12 which corresponds to the time value JAN1978. Note that the original data set SASHELP.WORKERS was modified beginning at the time value JAN1978. This similarity analysis justifies the belief that ELECTRIC and MASONRY have similar seasonal properties based on seasonal decomposition analysis despite the lack of target data (MASONRY).

Example 31.4: Searching for Historical Analogies

This example illustrates how to search for historical analogies by using seasonal sliding similarity analysis of transactional time-stamped data. The SASHELP.TIMEDATA data set contains the variable (VOLUME), which represents activity over time. The following statements create an example data set that contains two time series of differing lengths, where the variable HISTORY represents the historical activity and RECENT represents the more recent activity:

```
data timedata; set sashelp.timedata;
   drop volume;
   recent  = .;
   history = volume;
   if datetime >= '20AUG2000:00:00:00'DT then do;
      recent  = volume;
      history = .;
   end;
run;
```

The goal of seasonal sliding similarity measures is to find the seasonal slide index that corresponds to the most similar seasonal subsequence of the input series when compared to the target sequence. The following statements perform similarity analysis on the example data set with seasonal sliding:

```
proc similarity data=timedata out=_NULL_ outsequence=sequences
                outsum=summary;
   id datetime interval=dtday accumulate=total
               start='27JUL1997:00:00:00'dt
               end='21OCT2000:11:59:59'DT;
   input history / normalize=absolute;
   target recent / slide=season normalize=absolute measure=mabsdev;
run;
```

The DATA=TIMEDATA option specifies that the input data set WORK.TIMEDATA be used in the analysis. The OUT=_NULL_ option specifies that no output time series data set is to be created. The OUTSEQUENCE=SEQUENCES and OUTSUM=SUMMARY options specify the output sequences and summary data sets, respectively. The ID statement specifies that the time ID variable is DATETIME, which is to be accumulated on a daily basis (INTERVAL=DTDAY) by summing the transactions (ACCUMULATE=TOTAL). The ID statement also specifies that the data is accumulated on the weekly boundaries starting on the week of 27JUL1997 and ending on the week of 15OCT2000 (START='27JUL1997:00:00:00'DT END='21OCT2000:11:59:59'DT). The INPUT statement specifies that the input variable is HISTORY, which is to be normalized using absolute normalization (NORMALIZE=ABSOLUTE). The TARGET statement specifies that the target variable is RECENT, which is to be normalized by using absolute normalization (NORMALIZE=ABSOLUTE) and that the similarity measure be computed by using mean absolute deviation (MEASURE=MABSDEV). The SLIDE=SEASON options specifies season index sliding.

To illustrate the results of the similarity analysis, the output sequence data set must be subset by using the output summary data set.

```
data _NULL_; set summary;
   call symput('MEASURE', left(trim(putn(recent,'BEST20.'))));
run;

data result; set sequences;
```

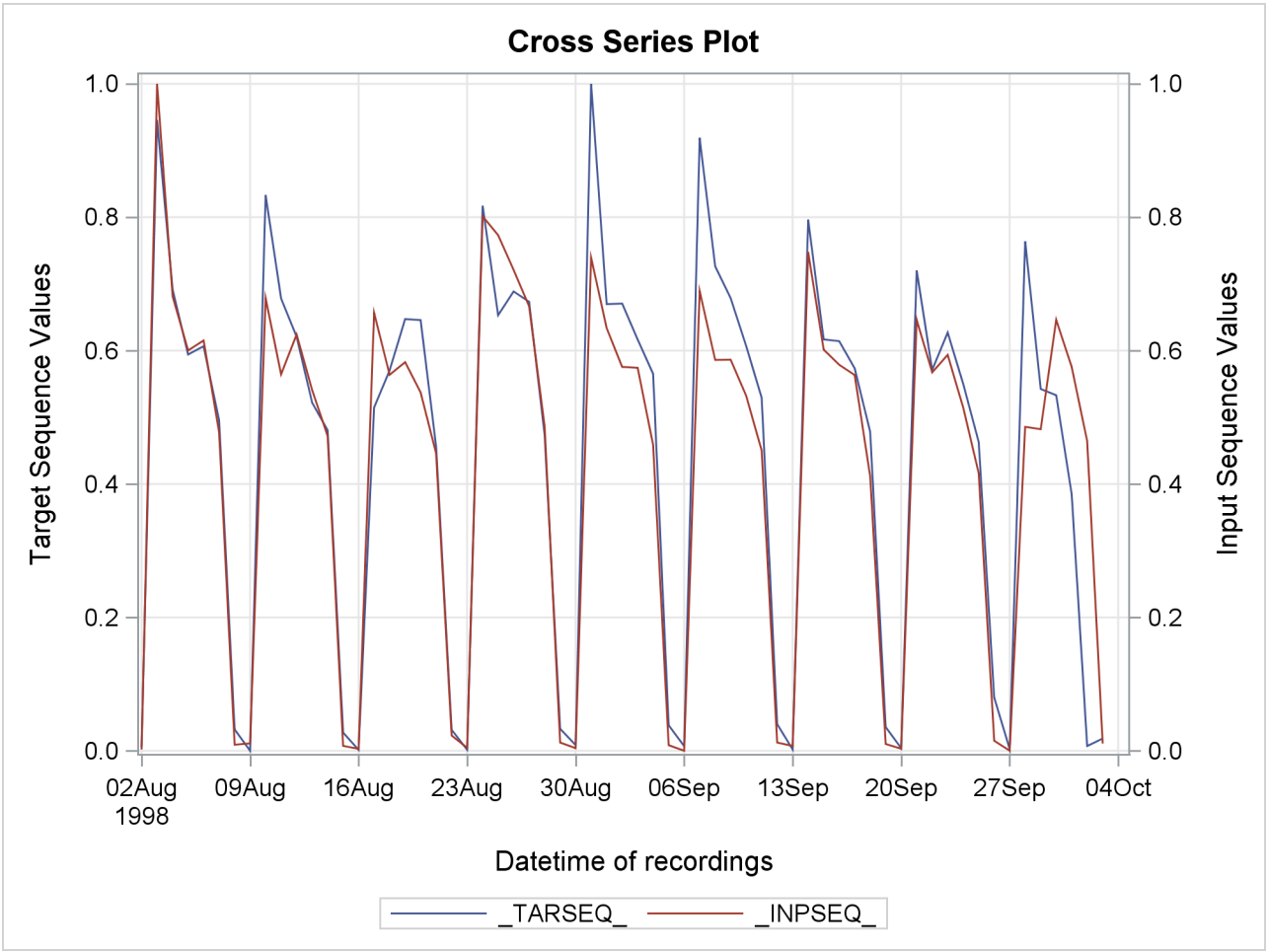
```
   by _SLIDE_;
   retain flag 0;
   if first._SLIDE_  then do;
      if (&measure - 0.00001 < _SIM_ < &measure + 0.00001)
      then flag = 1;
   end;
   if flag then output;
   if last._SLIDE_ then flag = 0;
run;
```

The following statements generate a cross series plot of the results:

```
proc timeseries data=result out=_NULL_ crossplot=series;
   id datetime interval=dtday;
   var _TARSEQ_;
   crossvar _INPSEQ_;
run;
```

The cross series plot illustrates that the historical time series analogy most similar to the most recent time series data that started on 20AUG2000 occurred on 02AUG1998.

Output 31.4.1 Cross Series Plot of the Historical Time Series



Example 31.5: Clustering Time Series

This example illustrates how to cluster time series using a similarity matrix. The WORK.APPLIANCE data set contains 24 variables that record sales histories. The following statements create a similarity matrix and store the matrix in the WORK.SIMMATRIX data set:

```
proc similarity data=sashelp.applianc out=_null_ outsum=simmatrix;
   target units_1--units_24 / measure=mabsdev normalize=absolute;
run;
```

The following statements cluster the rows of the similarity matrix.

```
proc cluster data=simmatrix(drop=_status_) outtree=tree method=ward noprint;
   id _input_;
run;
```

The following statements plot the dendrogram:

```
proc tree data=tree horizontal;
run;
```

References

Barry, M. J., and Linoff, G. S. (1997). *Data Mining Techniques: For Marketing, Sales, and Customer Support*. New York: John Wiley & Sons.

Han, J., and Kamber, M. (2001). *Data Mining: Concepts and Techniques*. San Francisco: Morgan Kaufmann.

Leonard, M. J., Lee, J. S., Lee, T., and Elsheimer, D. B. (2008). "An Introduction to Similarity Analysis Using SAS." In *Proceedings of the SAS Global Forum 2008 Conference*. Cary, NC: SAS Institute Inc. http://www2.sas.com/proceedings/forum2008/319-2008.pdf.

Leonard, M. J., and Wolfe, B. L. (2005). "Mining Transactional and Time Series Data." In *Proceedings of the Thirtieth Annual SAS Users Group International Conference*. Cary, NC: SAS Institute Inc. http://www2.sas.com/proceedings/sugi30/080-30.pdf.

Pyle, D. (1999). *Data Preparation for Data Mining*. San Francisco: Morgan Kaufmann.

Sankoff, D., and Kruskal, J. B. (2001). *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Stanford, CA: CSLI Publications.

Chapter 32
The SIMLIN Procedure

Contents

Overview: SIMLIN Procedure **2301**
Getting Started: SIMLIN Procedure **2302**
Prediction and Simulation 2303
Syntax: SIMLIN Procedure **2304**
Functional Summary 2304
PROC SIMLIN Statement 2305
BY Statement 2306
ENDOGENOUS Statement 2306
EXOGENOUS Statement 2306
ID Statement 2307
LAGGED Statement 2307
OUTPUT Statement 2307
Details: SIMLIN Procedure **2308**
Defining the Structural Form 2308
Computing the Reduced Form 2309
Dynamic Multipliers 2309
Multipliers for Higher Order Lags 2310
EST= Data Set 2310
DATA= Data Set 2311
OUTEST= Data Set 2311
OUT= Data Set 2312
Printed Output 2313
ODS Table Names 2314
Examples: SIMLIN Procedure **2315**
Example 32.1: Simulating Klein's Model I 2315
Example 32.2: Multipliers for a Third-Order System 2323
References **2328**

Overview: SIMLIN Procedure

The SIMLIN procedure reads the coefficients for a set of linear structural equations, which are usually produced by the SYSLIN procedure. PROC SIMLIN then computes the reduced form and, if input data are given, uses the reduced form equations to generate predicted values. PROC SIMLIN is especially useful

when dealing with sets of structural difference equations. The SIMLIN procedure can perform simulation or forecasting of the endogenous variables.

The SIMLIN procedure can be applied only to models that are:

- linear with respect to the parameters
- linear with respect to the variables
- square (as many equations as endogenous variables)
- nonsingular (the coefficients of the endogenous variables form an invertible matrix)

Getting Started: SIMLIN Procedure

The SIMLIN procedure processes the coefficients in a data set created by the SYSLIN procedure using the OUTEST= option or by another regression procedure such as PROC REG. To use PROC SIMLIN you must first produce the coefficient data set and then specify this data set on the EST= option of the PROC SIMLIN statement. You must also tell PROC SIMLIN which variables are endogenous and which variables are exogenous. List the endogenous variables in an ENDOGENOUS statement, and list the exogenous variables in an EXOGENOUS statement.

The following example illustrates the creation of an OUTEST= data set with PROC SYSLIN and the computation and printing of the reduced form coefficients for the model with PROC SIMLIN.

```
proc syslin data=in outest=e;
   model y1 = y2 x1;
   model y2 = y1 x2;
run;

proc simlin est=e;
   endogenous y1 y2;
   exogenous x1 x2;
run;
```

If the model contains lagged endogenous variables you must also use a LAGGED statement to tell PROC SIMLIN which variables contain lagged values, which endogenous variables they are lags of, and the number of periods of lagging. For dynamic models, the TOTAL and INTERIM= options can be used on the PROC SIMLIN statement to compute and print total and impact multipliers. (See "Dynamic Multipliers" later in this section for an explanation of multipliers.)

In the following example the variables Y1LAG1, Y2LAG1, and Y2LAG2 contain lagged values of the endogenous variables Y1 and Y2. Y1LAG1 and Y2LAG1 contain values of Y1 and Y2 for the previous observation, while Y2LAG2 contains 2 period lags of Y2. The LAGGED statement specifies the lagged relationships, and the TOTAL and INTERIM= options request multiplier analysis. The INTERIM=2 option prints matrices showing the impact that changes to the exogenous variables have on the endogenous variables after 1 and 2 periods.

```
data in; set in;
  y1lag1 = lag(y1);
  y2lag1 = lag(y2);
  y2lag2 = lag2(y2);
run;

proc syslin data=in outest=e;
   model y1 = y2 y1lag1 y2lag2 x1;
   model y2 = y1 y2lag1 x2;
run;

proc simlin est=e total interim=2;
   endogenous y1 y2;
   exogenous x1 x2;
   lagged y1lag1 y1 1 y2lag1 y2 1 y2lag2 y2 2;
run;
```

After the reduced form of the model is computed, the model can be simulated by specifying an input data set on the PROC SIMLIN statement and using an OUTPUT statement to write the simulation results to an output data set. The following example modifies the PROC SIMLIN step from the preceding example to simulate the model and stores the results in an output data set.

```
proc simlin est=e total interim=2 data=in;
   endogenous y1 y2;
   exogenous x1 x2;
   lagged y1lag1 y1 1 y2lag1 y2 1 y2lag2 y2 2;
   output out=sim predicted=y1hat y2hat
                  residual=y1resid y2resid;
run;
```

Prediction and Simulation

If an input data set is specified with the DATA= option in the PROC SIMLIN statement, the procedure reads the data and uses the reduced form equations to compute predicted and residual values for each of the endogenous variables. (If no data set is specified with the DATA= option, no simulation of the system is performed, and only the reduced form and multipliers are computed.)

The character of the prediction is based on the START= value. Until PROC SIMLIN encounters the START= observation, actual endogenous values are found and fed into the lagged endogenous terms. Once the START= observation is reached, dynamic simulation begins, where predicted values are fed into lagged endogenous terms until the end of the data set is reached.

The predicted and residual values generated here are different from those produced by the SYSLIN procedure since PROC SYSLIN uses the structural form with actual endogenous values. The predicted values computed by the SIMLIN procedure solve the simultaneous equation system. These reduced-form predicted values are functions only of the exogenous and lagged endogenous variables and do not depend on actual values of current period endogenous variables.

Syntax: SIMLIN Procedure

The following statements can be used with PROC SIMLIN:

PROC SIMLIN *options* ;
  BY *variables* ;
  ENDOGENOUS *variables* ;
  EXOGENOUS *variables* ;
  ID *variables* ;
  LAGGED *lag-var endogenous-var number ellipsis* ;
  OUTPUT *OUT=SAS-data-set options* ;

Functional Summary

The statements and options controlling the SIMLIN procedure are summarized in the following table.

Description	Statement	Option
Data Set Options		
specify input data set containing structural coefficients	PROC SIMLIN	EST=
specify type of estimates read from EST= data set	PROC SIMLIN	TYPE=
write reduced form coefficients and multipliers to an output data set	PROC SIMLIN	OUTEST=
specify the input data set for simulation	PROC SIMLIN	DATA=
write predicted and residual values to an output data set	OUTPUT	
Printing Control Options		
print the structural coefficients	PROC SIMLIN	ESTPRINT
suppress printing of reduced form coefficients	PROC SIMLIN	NORED
suppress all printed output	PROC SIMLIN	NOPRINT
Dynamic Multipliers		
compute interim multipliers	PROC SIMLIN	INTERIM=
compute total multipliers	PROC SIMLIN	TOTAL
Declaring the Role of Variables		
specify BY-group processing	BY	
specify the endogenous variables	ENDOGENOUS	
specify the exogenous variables	EXOGENOUS	
specify identifying variables	ID	
specify lagged endogenous variables	LAGGED	

Description	Statement	Option
Controlling the Simulation		
specify the starting observation for dynamic simulation	PROC SIMLIN	START=

PROC SIMLIN Statement

PROC SIMLIN *options* **;**

The following options can be used in the PROC SIMLIN statement:

DATA= *SAS-data-set*
specifies the SAS data set containing input data for the simulation. If the DATA= option is used, the data set specified must supply values for all exogenous variables throughout the simulation. If the DATA= option is not specified, no simulation of the system is performed, and only the reduced form and multipliers are computed.

EST= *SAS-data-set*
specifies the input data set containing the structural coefficients of the system. If EST= is omitted the most recently created SAS data set is used. The EST= data set is normally a "TYPE=EST" data set produced by the OUTEST= option of PROC SYSLIN. However, you can also build the EST= data set with a SAS DATA step. See "The EST= Data Set" later in this chapter for details.

ESTPRINT
prints the structural coefficients read from the EST= data set.

INTERIM= *n*
requests that interim multipliers be computed for interims 1 through *n*. If not specified, no interim multipliers are computed. This feature is available only if there are no lags greater than 1.

NOPRINT
suppresses all printed output.

NORED
suppresses the printing of the reduced form coefficients.

OUTEST= *SAS-data-set*
specifies an output SAS data set to contain the reduced form coefficients and multipliers, in addition to the structural coefficients read from the EST= data set. The OUTEST= data set has the same form as the EST= data set. If the OUTEST= option is not specified, the reduced form coefficients and multipliers are not written to a data set.

START= *n*
specifies the observation number in the DATA= data set where the dynamic simulation is to be started. By default, the dynamic simulation starts with the first observation in the DATA= data set for which all variables (including lags) are not missing.

TOTAL
requests that the total multipliers be computed. This feature is available only if there are no lags greater than 1.

TYPE= *value*
specifies the type of estimates to be read from the EST= data set. The TYPE= value must match the value of the _TYPE_ variable for the observations that you want to select from the EST= data set (TYPE=2SLS, for example).

BY Statement

BY *variables* **;**

A BY statement can be used with PROC SIMLIN to obtain separate analyses for groups of observations defined by the BY variables.

The BY statement can be applied to one or both of the EST= and the DATA= input data set. When a BY statement is used and both an EST= and a DATA= input data set are specified, PROC SIMLIN checks to see if one or both of the data sets contain the BY variables.

Thus, there are three ways of using the BY statement with PROC SIMLIN:

1. If the BY variables are found in the EST= data set only, PROC SIMLIN simulates over the entire DATA= data set once for each set of coefficients read from the BY groups in the EST= data set.
2. If the BY variables are found in the DATA= data set only, PROC SIMLIN performs separate simulations over each BY group in the DATA= data set, using the single set of coefficients in the EST= data set.
3. If the BY variables are found in both the EST= and the DATA= data sets, PROC SIMLIN performs separate simulations over each BY group in the DATA= data set using the coefficients from the corresponding BY group in the EST= data set.

ENDOGENOUS Statement

ENDOGENOUS *variables* **;**

List the names of the endogenous (jointly dependent) variables in the ENDOGENOUS statement. The ENDOGENOUS statement can be abbreviated as ENDOG or ENDO.

EXOGENOUS Statement

EXOGENOUS *variables* **;**

List the names of the exogenous (independent) variables in the EXOGENOUS statement. The EXOGENOUS statement can be abbreviated as EXOG or EXO.

ID Statement

ID *variables* **;**

The ID statement can be used to restrict the variables copied from the DATA= data set to the OUT= data set. Use the ID statement to list the variables you want copied to the OUT= data set besides the exogenous, endogenous, lagged endogenous, and BY variables. If the ID statement is omitted, all the variables in the DATA= data set are copied to the OUT= data set.

LAGGED Statement

LAGGED *lag-var endogenous-var number ellipsis* **;**

For each lagged endogenous variable, specify the name of the lagged variable, the name of the endogenous variable that was lagged, and the degree of the lag. Only one LAGGED statement is allowed.

The following is an example of the use of the LAGGED statement:

```
proc simlin est=e;
   endog y1 y2;
   lagged y1lag1 y1 1  y2lag1 y2 1  y2lag3 y2 3;
run;
```

This statement specifies that the variable Y1LAG1 contains the values of the endogenous variable Y1 lagged one period; the variable Y2LAG1 refers to the values of Y2 lagged one period; and the variable Y2LAG3 refers to the values of Y2 lagged three periods.

OUTPUT Statement

OUTPUT *OUT= SAS-data-set options* **;**

The OUTPUT statement specifies that predicted and residual values be put in an output data set. A DATA= input data set must be supplied if the OUTPUT statement is used, and only one OUTPUT statement is allowed. The following options can be used in the OUTPUT statement:

OUT= *SAS-data-set*
names the output SAS data set to contain the predicted values and residuals. If OUT= is not specified, the output data set is named using the DATA*n* convention.

PREDICTED= *names*

P= *names*
names the variables in the output data set that contain the predicted values of the simulation. These variables correspond to the endogenous variables in the order in which they are specified in the ENDOGENOUS statement. Specify up to as many names as there are endogenous variables. If you specify names on the PREDICTED= option for only some of the endogenous variables, predicted values for the remaining variables are not output. The names must not match any variable name in the input data set.

RESIDUAL= *names*

R= *names*

names the variables in the output data set that contain the residual values from the simulation. The residuals are the differences between the actual values of the endogenous variables from the DATA= data set and the predicted values from the simulation. These variables correspond to the endogenous variables in the order in which they are specified in the ENDOGENOUS statement. Specify up to as many names as there are endogenous variables. The names must not match any variable name in the input data set.

The following is an example of the use of the OUTPUT statement. This example outputs predicted values for Y1 and Y2 and outputs residuals for Y1.

```
proc simlin est=e;
   endog y1 y2;
   output out=b predicted=y1hat y2hat
                residual=y1resid;
run;
```

Details: SIMLIN Procedure

The following sections explain the structural and reduced forms, dynamic multipliers, input data sets, and the model simulation process in more detail.

Defining the Structural Form

An EST= input data set supplies the coefficients of the equation system. The data set containing the coefficients is normally a "TYPE=EST" data set created by the OUTEST= option of PROC SYSLIN or another regression procedure. The data set contains the special variables _TYPE_, _DEPVAR_, and INTERCEPT. You can also supply the structural coefficients of the system to PROC SIMLIN in a data set produced by a SAS DATA step as long as the data set is of the form TYPE=EST. Refer to SAS/STAT software documentation for a discussion of the special TYPE=EST type of SAS data set.

Suppose that there is a $g \times 1$ vector of endogenous variables $\boldsymbol{y}_t$, an $l \times 1$ vector of lagged endogenous variables $\mathbf{y}_t^L$, and a $k \times 1$ vector of exogenous variables $\boldsymbol{x}_t$, including the intercept. Then, there are g structural equations in the simultaneous system that can be written

$$\mathbf{G}\mathbf{y}_t = \mathbf{C}\mathbf{y}_t^L + \mathbf{B}\mathbf{x}_t$$

where $\mathbf{G}$ is the matrix of coefficients of current period endogenous variables, $\mathbf{C}$ is the matrix of coefficients of lagged endogenous variables, and $\mathbf{B}$ is the matrix of coefficients of exogenous variables. $\mathbf{G}$ is assumed to be nonsingular.

Computing the Reduced Form

First, the SIMLIN procedure computes reduced form coefficients by premultiplying by $\mathbf{G}^{-1}$:

$$\mathbf{y}_t = \mathbf{G}^{-1}\mathbf{C}\mathbf{y}_t^L + \mathbf{G}^{-1}\mathbf{B}\mathbf{x}_t$$

This can be written as

$$\mathbf{y}_t = \Pi_1\mathbf{y}_t^L + \Pi_2\mathbf{x}_t$$

where $\Pi_1 = \mathbf{G}^{-1}\mathbf{C}$ and $\Pi_2 = \mathbf{G}^{-1}\mathbf{B}$ are the reduced form coefficient matrices.

The reduced form matrices $\Pi_1 = \mathbf{G}^{-1}\mathbf{C}$ and $\Pi_2 = \mathbf{G}^{-1}\mathbf{B}$ are printed unless the NORED option is specified in the PROC SIMLIN statement. The structural coefficient matrices **G**, **C**, and **B** are printed when the ESTPRINT option is specified.

Dynamic Multipliers

For models that have only first-order lags, the equation of the reduced form of the system can be rewritten

$$\mathbf{y}_t = \mathbf{D}\mathbf{y}_{t-1} + \Pi_2\mathbf{x}_t$$

D is a matrix formed from the columns of Π_1 plus some columns of zeros, arranged in the order in which the variables meet the lags. The elements of Π_2 are called *impact multipliers* because they show the immediate effect of changes in each exogenous variable on the values of the endogenous variables. This equation can be rewritten as

$$\mathbf{y}_t = \mathbf{D}^2\mathbf{y}_{t-2} + \mathbf{D}\Pi_2\mathbf{x}_{t-1} + \Pi_2\mathbf{x}_t$$

The matrix formed by the product $\mathbf{D}\ \Pi_2$ shows the effect of the exogenous variables one lag back; the elements in this matrix are called *interim multipliers* and are computed and printed when the INTERIM= option is specified in the PROC SIMLIN statement. The *i*th period interim multipliers are formed by $\mathbf{D}^{\,i}\Pi_2$.

The series can be expanded as

$$\mathbf{y}_t = \mathbf{D}^{\infty}\mathbf{y}_{t-\infty} + \sum_{i=0}^{\infty}\mathbf{D}^i\,\Pi_2\mathbf{x}_{t-i}$$

A permanent and constant setting of a value for *x* has the following cumulative effect:

$$\left(\sum_{i=0}^{\infty}\mathbf{D}^i\right)\Pi_2\mathbf{x} = (\mathbf{I}-\mathbf{D})^{-1}\Pi_2\mathbf{x}$$

The elements of $(\mathbf{I}\text{-}\mathbf{D})^{-1}\Pi_2$ are called the *total multipliers*. Assuming that the sum converges and that $(\mathbf{I}\text{-}\mathbf{D})$ is invertible, PROC SIMLIN computes the total multipliers when the TOTAL option is specified in the PROC SIMLIN statement.

Multipliers for Higher Order Lags

The dynamic multiplier options require the system to have no lags of order greater than one. This limitation can be circumvented, since any system with lags greater than one can be rewritten as a system where no lag is greater than one by forming new endogenous variables that are single-period lags.

For example, suppose you have the third-order single equation

$$y_t = a y_{t-3} + b\mathbf{x}_t$$

This can be converted to a first-order three-equation system by introducing two additional endogenous variables, $y_{1,t}$ and $y_{2,t}$, and computing corresponding first-order lagged variables for each endogenous variable: y_{t-1}, $y_{1,t-1}$, and $y_{2,t-1}$. The higher order lag relations are then produced by adding identities to link the endogenous and identical lagged endogenous variables:

$$y_{1,t} = y_{t-1}$$

$$y_{2,t} = y_{1,t-1}$$

$$y_t = a y_{2,t-1} + b\mathbf{X}_t$$

This conversion using the SYSLIN and SIMLIN procedures requires three steps:

1. Add the extra endogenous and lagged endogenous variables to the input data set using a DATA step. Note that two copies of each lagged endogenous variable are needed for each lag reduced, one to serve as an endogenous variable and one to serve as a lagged endogenous variable in the reduced system.
2. Add IDENTITY statements to the PROC SYSLIN step to equate each added endogenous variable to its lagged endogenous variable copy.
3. In the PROC SIMLIN step, declare the added endogenous variables in the ENDOGENOUS statement and define the lag relations in the LAGGED statement.

See Example 32.2 for an illustration of how to convert an equation system with higher-order lags into a larger system with only first-order lags.

EST= Data Set

Normally, PROC SIMLIN uses an EST= data set produced by PROC SYSLIN with the OUTEST= option. This data set is in the form expected by PROC SIMLIN. If there is more than one set of estimates produced by PROC SYSLIN, you must use the TYPE= option in the PROC SIMLIN statement to select the set to be simulated. Then PROC SIMLIN reads from the EST= data set only those observations with a _TYPE_ value corresponding to the TYPE= option (for example, TYPE=2SLS) or with a _TYPE_ value of IDENTITY.

The SIMLIN procedure can only solve square, nonsingular systems. If you have fewer equations than endogenous variables, you must specify IDENTITY statements in the PROC SYSLIN step to bring the system up to full rank. If there are g endogenous variables and $m < g$ stochastic equations with unknown

parameters, then you use *m* MODEL statements to specify the equations with parameters to be estimated and you must use *g-m* IDENTITY statements to complete the system.

You can build your own EST= data set with a DATA step rather than use PROC SYSLIN. The EST= data set must contain the endogenous variables, the lagged endogenous variables (if any), and the exogenous variables in the system (if any). If any of the equations have intercept terms, the variable INTERCEPT must supply these coefficients. The EST= data set should also contain the special character variable comp _DEPVAR_ to label the equations.

The EST= data set must contain one observation for each equation in the system. The values of the lagged endogenous variables must contain the **C** coefficients. The values of the exogenous variables and the INTERCEPT variable must contain the **B** coefficients. The values of the endogenous variables, however, must contain the negatives of the **G** coefficients. This is because the SYSLIN procedure writes the coefficients to the OUTEST= data set in the form

$$0 = \mathbf{H}\mathbf{y}_t + \mathbf{C}\mathbf{y}_t^L + \mathbf{B}\mathbf{x}_t$$

where **H** =-**G**.

See "Multipliers for Higher Order Lags" and Example 32.2 later in this chapter for more information on building the EST= data set.

DATA= Data Set

The DATA= data set must contain all of the exogenous variables. Values for all of the exogenous variables are required for each observation for which predicted endogenous values are desired. To forecast past the end of the historical data, the DATA= data set should contain nonmissing values for all of the exogenous variables and missing values for the endogenous variables for the forecast periods, in addition to the historical data. (See Example 32.1 for an illustration.)

In order for PROC SIMLIN to output residuals and compute statistics of fit, the DATA= data set must also contain the endogenous variables with nonmissing actual values for each observation for which residuals and statistics are to be computed.

If the system contains lags, initial values must be supplied for the lagged variables. This can be done by including either the lagged variables or the endogenous variables, or both, in the DATA= data set. If the lagged variables are not in the DATA= data set or if they have missing values in the early observations, PROC SIMLIN prints a warning and uses the endogenous variable values from the early observations to initialize the lags.

OUTEST= Data Set

The OUTEST= data set contains all the variables read from the EST= data set. The variables in the OUTEST= data set are as follows.

- the BY statement variables, if any
- _TYPE_, a character variable that identifies the type of observation

- _DEPVAR_, a character variable containing the name of the dependent variable for the observation
- the endogenous variables
- the lagged endogenous variables
- the exogenous variables
- INTERCEPT, a numeric variable containing the intercept values
- _MODEL_, a character variable containing the name of the equation
- _SIGMA_, a numeric variable containing the estimated error variance of the equation (output only if present in the EST= data set)

The observations read from the EST= data set that supply the structural coefficients are copied to the OUTEST= data set, except that the signs of endogenous coefficients are reversed. For these observations, the _TYPE_ variable values are the same as in the EST= data set.

In addition, the OUTEST= data set contains observations with the following _TYPE_ values:

REDUCED — the reduced form coefficients. The endogenous variables for this group of observations contain the inverse of the endogenous coefficient matrix $\mathbf{G}$. The lagged endogenous variables contain the matrix Π_1=$\mathbf{G}^{-1}\mathbf{C}$. The exogenous variables contain the matrix Π_2=$\mathbf{G}^{-1}\mathbf{B}$.

IMULT*i* — the interim multipliers, if the INTERIM= option is specified. There are *gn* observations for the interim multipliers, where *g* is the number of endogenous variables and *n* is the value of the INTERIM=*n* option. For these observations the _TYPE_ variable has the value IMULT*i*, where the interim number *i* ranges from 1 to *n*.

The exogenous variables in groups of *g* observations that have a _TYPE_ value of IMULT*i* contain the matrix $\mathbf{D}^{i}\Pi_2$ of multipliers at interim *i*. The endogenous and lagged endogenous variables for this group of observations are set to missing.

TOTAL — the total multipliers, if the TOTAL option is specified. The exogenous variables in this group of observations contain the matrix $(\mathbf{I}\text{-}\mathbf{D})^{-1}\Pi_2$. The endogenous and lagged endogenous variables for this group of observations are set to missing.

OUT= Data Set

The OUT= data set normally contains all of the variables in the input DATA= data set, plus the variables named in the PREDICTED= and RESIDUAL= options in the OUTPUT statement.

You can use an ID statement to restrict the variables that are copied from the input data set. If an ID statement is used, the OUT= data set contains only the BY variables (if any), the ID variables, the endogenous and lagged endogenous variables (if any), the exogenous variables, plus the PREDICTED= and RESIDUAL= variables.

The OUT= data set contains an observation for each observation in the DATA= data set. When the actual value of an endogenous variable is missing in the DATA= data set, or when the DATA= data set does not contain the endogenous variable, the corresponding residual is missing.

Printed Output

Structural Form

The following items are printed as they are read from the EST= input data set. Structural zeros are printed as dots in the listing of these matrices.

1. Structural Coefficients for Endogenous Variables. This is the **G** matrix, with g rows and g columns.
2. Structural Coefficients for Lagged Endogenous Variables. These coefficients make up the **C** matrix, with g rows and l columns.
3. Structural Coefficients for Exogenous Variables. These coefficients make up the **B** matrix, with g rows and k columns.

Reduced Form

1. The reduced form coefficients are obtained by inverting **G** so that the endogenous variables can be directly expressed as functions of only lagged endogenous and exogenous variables.
2. Inverse Coefficient Matrix for Endogenous Variables. This is the inverse of the **G** matrix.
3. Reduced Form for Lagged Endogenous Variables. This is Π_1=$\mathbf{G}^{-1}\mathbf{C}$, with g rows and l columns. Each value is a dynamic multiplier that shows how past values of lagged endogenous variables affect values of each of the endogenous variables.
4. Reduced Form for Exogenous Variables. This is Π_2=$\mathbf{G}^{-1}\mathbf{B}$, with g rows and k columns. Its values are called *impact multipliers* because they show the immediate effect of each exogenous variable on the value of the endogenous variables.

Multipliers

Interim and total multipliers show the effect of a change in an exogenous variable over time.

1. Interim Multipliers. These are the interim multiplier matrices. They are formed by multiplying Π_2 by powers of **D**. The dth interim multiplier is $\mathbf{D}^{d}\Pi_2$. The interim multiplier of order d shows the effects of a change in the exogenous variables after d periods. Interim multipliers are only available if the maximum lag of the endogenous variables is 1.
2. Total Multipliers. This is the matrix of total multipliers, T=$(\mathbf{I}-\mathbf{D})^{-1}\Pi_2$. This matrix shows the cumulative effect of changes in the exogenous variables. Total multipliers are only available if the maximum lag is one.

Statistics of Fit

If the DATA= option is used and the DATA= data set contains endogenous variables, PROC SIMLIN prints a statistics-of-fit report for the simulation. The statistics printed include the following. (Summations are over the observations for which both y_t and $\hat{y}_t$ are nonmissing.)

1. the number of nonmissing errors. (Number of observations for which both y_t and $\hat{y}_t$ are nonmissing.)
2. the mean error: $\frac{1}{n}\sum(y_t - \hat{y}_t)$
3. the mean percent error: $\frac{100}{n}\sum\frac{(y_t-\hat{y}_t)}{y_t}$
4. the mean absolute error: $\frac{1}{n}\sum|y_t - \hat{y}_t|$
5. the mean absolute percent error $\frac{100}{n}\sum\frac{|y_t-\hat{y}_t|}{y_t}$
6. the root mean square error: $\sqrt{\frac{1}{n}\sum(y_t - \hat{y}_t)^2}$
7. the root mean square percent error: $\sqrt{\frac{100}{n}\sum(\frac{(y_t-\hat{y}_t)}{y_t})^2}$

ODS Table Names

PROC SIMLIN assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in the following table.

Table 32.2 ODS Tables Produced in PROC SIMLIN

ODS Table Name	Description	Option
Endogenous	Structural Coefficients for Endogenous Variables	default
LaggedEndogenous	Structural Coefficients for Lagged Endogenous Variables	default
Exogenous	Structural Coefficients for Exogenous Variables	default
InverseCoeff	Inverse Coefficient Matrix for Endogenous Variables	default
RedFormLagEndo	Reduced Form for Lagged Endogenous Variables	default
RedFormExog	Reduced Form for Exogenous Variables	default
InterimMult	Interim Multipliers	INTERIM= option
TotalMult	Total Multipliers	TOTAL= option
FitStatistics	Fit statistics	default

Examples: SIMLIN Procedure

Example 32.1: Simulating Klein's Model I

In this example, the SIMLIN procedure simulates a model of the U.S. economy called Klein's Model I. The SAS data set KLEIN is used as input to the SYSLIN and SIMLIN procedures.

```
data klein;
   input year c p w i x wp g t k wsum;
   date=mdy(1,1,year);
   format date year.;
   y   = c + i + g - t;
   yr  = year - 1931;
   klag = lag( k );
   plag = lag( p );
   xlag = lag( x );
   if year >= 1921;
   label c   ='consumption'
         p   ='profits'
         w   ='private wage bill'
         i   ='investment'
         k   ='capital stock'
         y   ='national income'
         x   ='private production'
         wsum='total wage bill'
         wp  ='govt wage bill'
         g   ='govt demand'
         t   ='taxes'
         klag='capital stock lagged'
         plag='profits lagged'
         xlag='private product lagged'
         yr  ='year-1931';
datalines;
1920     .  12.7     .    .  44.9    .     .     .  182.8     .
1921  41.9  12.4  25.5 -0.2  45.6  2.7   3.9   7.7  182.6  28.2

   ... more lines ...
```

First, the model is specified and estimated using the SYSLIN procedure, and the parameter estimates are written to an OUTEST= data set. The printed output produced by the SYSLIN procedure is not shown here; see Example 36.1 in Chapter 36 for the printed output of the PROC SYSLIN step.

```
title1 'Simulation of Klein''s Model I using SIMLIN';
proc syslin 3sls data=klein outest=a;

   instruments klag plag xlag wp g t yr;
   endogenous c p w i x wsum k y;

   consume: model    c = p plag wsum;
   invest:  model    i = p plag klag;
   labor:   model    w = x xlag yr;

   product: identity x = c + i + g;
   income:  identity y = c + i + g - t;
   profit:  identity p = x - w - t;
   stock:   identity k = klag + i;
   wage:    identity wsum = w + wp;
run;
```

The OUTEST= data set A created by the SYSLIN procedure contains parameter estimates to be used by the SIMLIN procedure. The OUTEST= data set is shown in Output 32.1.1.

Output 32.1.1 The OUTEST= Data Set Created by PROC SYSLIN

Simulation of Klein's Model I using SIMLIN

Obs	_TYPE_	_STATUS_	_MODEL_	_DEPVAR_	_SIGMA_	Intercept	klag	plag	xlag	wp
1	INST	0 Converged	FIRST	c	2.11403	58.3018	-0.14654	0.74803	0.23007	0.19327
2	INST	0 Converged	FIRST	p	2.18298	50.3844	-0.21610	0.80250	0.02200	-0.07961
3	INST	0 Converged	FIRST	w	1.75427	43.4356	-0.12295	0.87192	0.09533	-0.44373
4	INST	0 Converged	FIRST	i	1.72376	35.5182	-0.19251	0.92639	-0.11274	-0.71661
5	INST	0 Converged	FIRST	x	3.77347	93.8200	-0.33906	1.67442	0.11733	-0.52334
6	INST	0 Converged	FIRST	wsum	1.75427	43.4356	-0.12295	0.87192	0.09533	0.55627
7	INST	0 Converged	FIRST	k	1.72376	35.5182	0.80749	0.92639	-0.11274	-0.71661
8	INST	0 Converged	FIRST	y	3.77347	93.8200	-0.33906	1.67442	0.11733	-0.52334
9	3SLS	0 Converged	CONSUME	c	1.04956	16.4408	.	0.16314	.	.
10	3SLS	0 Converged	INVEST	i	1.60796	28.1778	-0.19485	0.75572	.	.
11	3SLS	0 Converged	LABOR	w	0.80149	1.7972	.	.	0.18129	.
12	IDENTITY	0 Converged	PRODUCT	x	.	0.0000	.	.	.	.
13	IDENTITY	0 Converged	INCOME	y	.	0.0000	.	.	.	.
14	IDENTITY	0 Converged	PROFIT	p	.	0.0000	.	.	.	.
15	IDENTITY	0 Converged	STOCK	k	.	0.0000	1.00000	.	.	.
16	IDENTITY	0 Converged	WAGE	wsum	.	0.0000	.	.	.	1.00000

Obs	g	t	yr	c	p	w	i	x	wsum	k	y
1	0.20501	-0.36573	0.70109	-1	.	.	.	.	.	.	.
2	0.43902	-0.92310	0.31941	.	-1.00000	.	.	.	.	.	.
3	0.86622	-0.60415	0.71358	.	.	-1	.	.	.	.	.
4	0.10023	-0.16152	0.33190	.	.	.	-1	.	.	.	.
5	1.30524	-0.52725	1.03299	.	.	.	.	-1.00000	.	.	.
6	0.86622	-0.60415	0.71358	.	.	.	.	.	-1.00000	.	.
7	0.10023	-0.16152	0.33190	.	.	.	.	.	.	-1	.
8	1.30524	-1.52725	1.03299	.	.	.	.	.	.	.	-1
9	.	.	.	-1	0.12489	.	.	.	0.79008	.	.
10	.	.	.	.	-0.01308	.	-1	.	.	.	.
11	.	.	0.14967	.	.	-1	.	0.40049	.	.	.
12	1.00000	.	.	1	.	.	1	-1.00000	.	.	.
13	1.00000	-1.00000	.	1	.	.	1	.	.	.	-1
14	.	-1.00000	.	.	-1.00000	-1	.	1.00000	.	.	.
15	.	.	.	.	.	.	1	.	.	-1	.
16	.	.	.	.	.	1	.	.	-1.00000	.	.

Using the OUTEST= data set A produced by the SYSLIN procedure, the SIMLIN procedure can now compute the reduced form and simulate the model. The following statements perform the simulation.

```
title1 'Simulation of Klein''s Model I using SIMLIN';
proc simlin data=klein
            est=a type=3sls
            estprint
            total interim=2
            outest=b;
   endogenous c p w i x wsum k y;
   exogenous  wp g t yr;
   lagged  klag k 1   plag p 1   xlag x 1;
   id year;
   output out=c p=chat phat what ihat xhat wsumhat khat yhat
                r=cres pres wres ires xres wsumres kres yres;
run;
```

The reduced form coefficients and multipliers are added to the information read from EST= data set A and written to the OUTEST= data set B. The predicted and residual values from the simulation are written to the OUT= data set C specified in the OUTPUT statement.

The SIMLIN procedure first prints the structural coefficient matrices read from the EST= data set, as shown in Output 32.1.2 through Output 32.1.4.

Output 32.1.2 SIMLIN Procedure Output – Endogenous Structural Coefficients

Simulation of Klein's Model I using SIMLIN

The SIMLIN Procedure

Structural Coefficients for Endogenous Variables								
Variable	c	p	w	i	x	wsum	k	y
c	1.0000	-0.1249	.	.	.	-0.7901	.	.
i	.	0.0131	.	1.0000	.	.	.	.
w	.	.	1.0000	.	-0.4005	.	.	.
x	-1.0000	.	.	-1.0000	1.0000	.	.	.
y	-1.0000	.	.	-1.0000	.	.	.	1.0000
p	.	1.0000	1.0000	.	-1.0000	.	.	.
k	.	.	.	-1.0000	.	.	1.0000	.
wsum	.	.	-1.0000	.	.	1.0000	.	.

Output 32.1.3 SIMLIN Procedure Output – Lagged Endogenous Structural Coefficients

Structural Coefficients for Lagged Endogenous Variables			
Variable	klag	plag	xlag
c	.	0.1631	.
i	-0.1948	0.7557	.
w	.	.	0.1813
x	.	.	.
y	.	.	.
p	.	.	.
k	1.0000	.	.
wsum	.	.	.

Output 32.1.4 SIMLIN Procedure Output – Exogenous Structural Coefficients

Structural Coefficients for Exogenous Variables					
Variable	**wp**	**g**	**t**	**yr**	**Intercept**
c	.	.	.	.	16.4408
i	.	.	.	.	28.1778
w	.	.	.	0.1497	1.7972
x	.	1.0000	.	.	0
y	.	1.0000	-1.0000	.	0
p	.	.	-1.0000	.	0
k	.	.	.	.	0
wsum	1.0000	.	.	.	0

The SIMLIN procedure then prints the inverse of the endogenous variables coefficient matrix, as shown in Output 32.1.5.

Output 32.1.5 SIMLIN Procedure Output – Inverse Coefficient Matrix

Inverse Coefficient Matrix for Endogenous Variables								
Variable	**c**	**i**	**w**	**x**	**y**	**p**	**k**	**wsum**
c	1.6347	0.6347	1.0957	0.6347	0	0.1959	0	1.2915
p	0.9724	0.9724	-0.3405	0.9724	0	1.1087	0	0.7682
w	0.6496	0.6496	1.4406	0.6496	0	0.0726	0	0.5132
i	-0.0127	0.9873	0.004453	-0.0127	0	-0.0145	0	-0.0100
x	1.6219	1.6219	1.1001	1.6219	0	0.1814	0	1.2815
wsum	0.6496	0.6496	1.4406	0.6496	0	0.0726	0	1.5132
k	-0.0127	0.9873	0.004453	-0.0127	0	-0.0145	1.0000	-0.0100
y	1.6219	1.6219	1.1001	0.6219	1.0000	0.1814	0	1.2815

The SIMLIN procedure next prints the reduced form coefficient matrices, as shown in Output 32.1.6.

Output 32.1.6 SIMLIN Procedure Output – Reduced Form Coefficients

Reduced Form for Lagged Endogenous Variables			
Variable	**klag**	**plag**	**xlag**
c	-0.1237	0.7463	0.1986
p	-0.1895	0.8935	-0.0617
w	-0.1266	0.5969	0.2612
i	-0.1924	0.7440	0.000807
x	-0.3160	1.4903	0.1994
wsum	-0.1266	0.5969	0.2612
k	0.8076	0.7440	0.000807
y	-0.3160	1.4903	0.1994

Output 32.1.6 *continued*

Reduced Form for Exogenous Variables					
Variable	wp	g	t	yr	Intercept
c	1.2915	0.6347	-0.1959	0.1640	46.7273
p	0.7682	0.9724	-1.1087	-0.0510	42.7736
w	0.5132	0.6496	-0.0726	0.2156	31.5721
i	-0.0100	-0.0127	0.0145	0.000667	27.6184
x	1.2815	1.6219	-0.1814	0.1647	74.3457
wsum	1.5132	0.6496	-0.0726	0.2156	31.5721
k	-0.0100	-0.0127	0.0145	0.000667	27.6184
y	1.2815	1.6219	-1.1814	0.1647	74.3457

The multiplier matrices (requested by the INTERIM=2 and TOTAL options) are printed next, as shown in Output 32.1.7 and Output 32.1.8.

Output 32.1.7 SIMLIN Procedure Output – Interim Multipliers

Interim Multipliers for Interim 1					
Variable	wp	g	t	yr	Intercept
c	0.829130	1.049424	-0.865262	-.0054080	43.27442
p	0.609213	0.771077	-0.982167	-.0558215	28.39545
w	0.794488	1.005578	-0.710961	0.0125018	41.45124
i	0.574572	0.727231	-0.827867	-.0379117	26.57227
x	1.403702	1.776655	-1.693129	-.0433197	69.84670
wsum	0.794488	1.005578	-0.710961	0.0125018	41.45124
k	0.564524	0.714514	-0.813366	-.0372452	54.19068
y	1.403702	1.776655	-1.693129	-.0433197	69.84670

Interim Multipliers for Interim 2					
Variable	wp	g	t	yr	Intercept
c	0.663671	0.840004	-0.968727	-.0456589	28.36428
p	0.350716	0.443899	-0.618929	-.0401446	10.79216
w	0.658769	0.833799	-0.925467	-.0399178	28.33114
i	0.345813	0.437694	-0.575669	-.0344035	10.75901
x	1.009485	1.277698	-1.544396	-.0800624	39.12330
wsum	0.658769	0.833799	-0.925467	-.0399178	28.33114
k	0.910337	1.152208	-1.389035	-.0716486	64.94969
y	1.009485	1.277698	-1.544396	-.0800624	39.12330

Output 32.1.8 SIMLIN Procedure Output – Total Multipliers

Total Multipliers					
Variable	**wp**	**g**	**t**	**yr**	**Intercept**
c	1.881667	1.381613	-0.685987	0.1789624	41.3045
p	0.786945	0.996031	-1.286891	-.0748290	15.4770
w	1.094722	1.385582	-0.399095	0.2537914	25.8275
i	0.000000	0.000000	-0.000000	0.0000000	0.0000
x	1.881667	2.381613	-0.685987	0.1789624	41.3045
wsum	2.094722	1.385582	-0.399095	0.2537914	25.8275
k	2.999365	3.796275	-4.904859	-.2852032	203.6035
y	1.881667	2.381613	-1.685987	0.1789624	41.3045

The last part of the SIMLIN procedure output is a table of statistics of fit for the simulation, as shown in Output 32.1.9.

Output 32.1.9 SIMLIN Procedure Output – Simulation Statistics

Fit Statistics								
Variable	**N**	**Mean Error**	**Mean Pct Error**	**Mean Abs Error**	**Mean Abs Pct Error**	**RMS Error**	**RMS Pct Error**	**Label**
c	21	0.1367	-0.3827	3.5011	6.69769	4.3155	8.1701	consumption
p	21	0.1422	-4.0671	2.9355	19.61400	3.4257	26.0265	profits
w	21	0.1282	-0.8939	3.1247	8.92110	4.0930	11.4709	private wage bill
i	21	0.1337	105.8529	2.4983	127.13736	2.9980	252.3497	investment
x	21	0.2704	-0.9553	5.9622	10.40057	7.1881	12.5653	private production
wsum	21	0.1282	-0.6669	3.1247	7.88988	4.0930	10.1724	total wage bill
k	21	-0.1424	-0.1506	3.8879	1.90614	5.0036	2.4209	capital stock
y	21	0.2704	-1.3476	5.9622	11.74177	7.1881	14.2214	national income

The OUTEST= output data set contains all the observations read from the EST= data set, and in addition contains observations for the reduced form and multiplier matrices. The following statements produce a partial listing of the OUTEST= data set, as shown in Output 32.1.10.

```
proc print data=b;
   where _type_ = 'REDUCED' | _type_ = 'IMULT1';
run;
```

Output 32.1.10 Partial Listing of OUTEST= Data Set

Simulation of Klein's Model I using SIMLIN

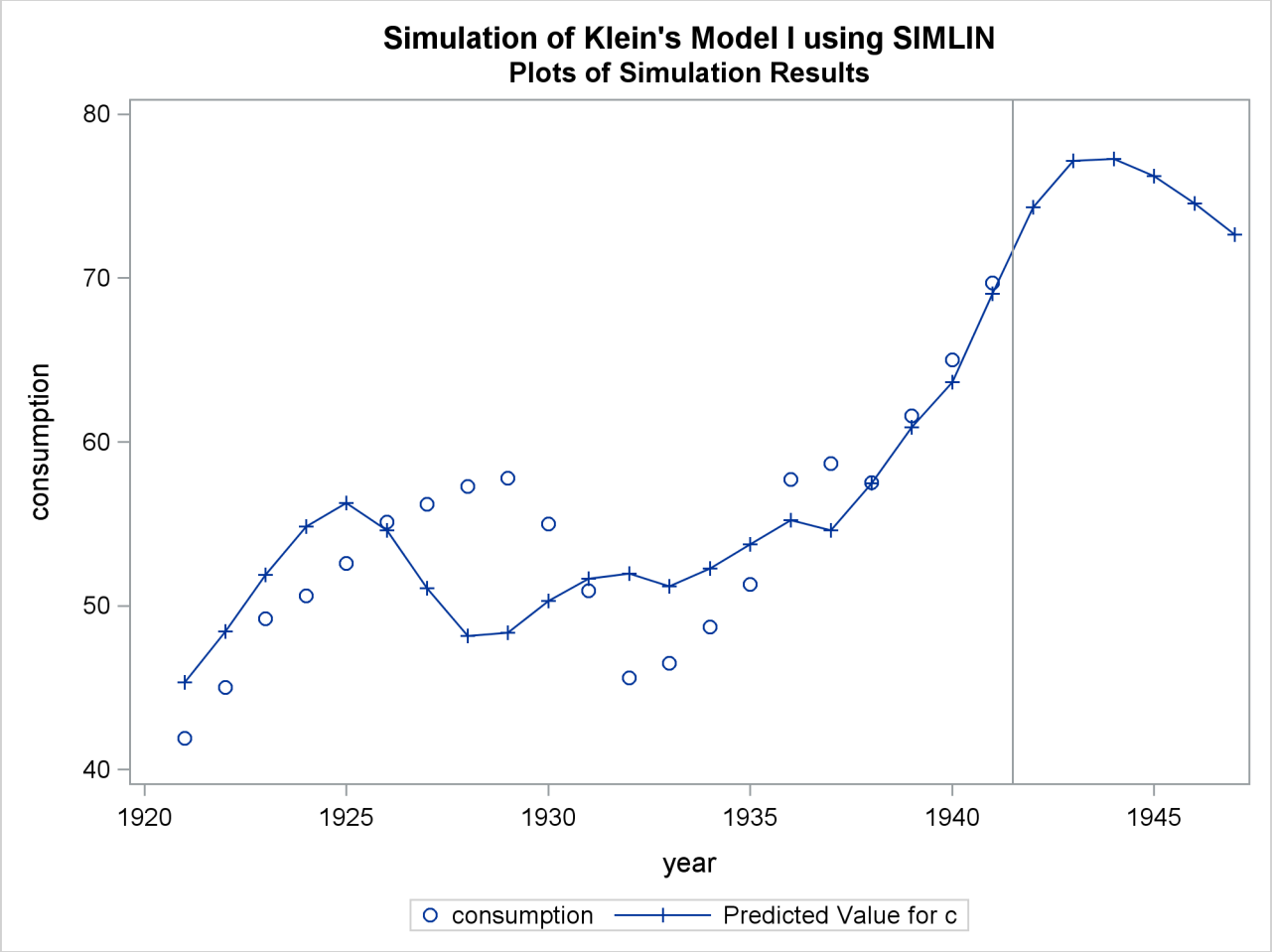
Obs	_TYPE_	_DEPVAR_	_MODEL_	_SIGMA_	c	p	w	i	x	wsum	k
9	REDUCED	c		.	1.63465	0.63465	1.09566	0.63465	0	0.19585	0
10	REDUCED	p		.	0.97236	0.97236	-0.34048	0.97236	0	1.10872	0
11	REDUCED	w		.	0.64957	0.64957	1.44059	0.64957	0	0.07263	0
12	REDUCED	i		.	-0.01272	0.98728	0.00445	-0.01272	0	-0.01450	0
13	REDUCED	x		.	1.62194	1.62194	1.10011	1.62194	0	0.18135	0
14	REDUCED	wsum		.	0.64957	0.64957	1.44059	0.64957	0	0.07263	0
15	REDUCED	k		.	-0.01272	0.98728	0.00445	-0.01272	0	-0.01450	1
16	REDUCED	y		.	1.62194	1.62194	1.10011	0.62194	1	0.18135	0
17	IMULT1	c		.	.	.	.	.	.	.	.
18	IMULT1	p		.	.	.	.	.	.	.	.
19	IMULT1	w		.	.	.	.	.	.	.	.
20	IMULT1	i		.	.	.	.	.	.	.	.
21	IMULT1	x		.	.	.	.	.	.	.	.
22	IMULT1	wsum		.	.	.	.	.	.	.	.
23	IMULT1	k		.	.	.	.	.	.	.	.
24	IMULT1	y		.	.	.	.	.	.	.	.

Obs	y	klag	plag	xlag	wp	g	t	yr	Intercept
9	1.29151	-0.12366	0.74631	0.19863	1.29151	0.63465	-0.19585	0.16399	46.7273
10	0.76825	-0.18946	0.89347	-0.06173	0.76825	0.97236	-1.10872	-0.05096	42.7736
11	0.51321	-0.12657	0.59687	0.26117	0.51321	0.64957	-0.07263	0.21562	31.5721
12	-0.01005	-0.19237	0.74404	0.00081	-0.01005	-0.01272	0.01450	0.00067	27.6184
13	1.28146	-0.31603	1.49034	0.19944	1.28146	1.62194	-0.18135	0.16466	74.3457
14	1.51321	-0.12657	0.59687	0.26117	1.51321	0.64957	-0.07263	0.21562	31.5721
15	-0.01005	0.80763	0.74404	0.00081	-0.01005	-0.01272	0.01450	0.00067	27.6184
16	1.28146	-0.31603	1.49034	0.19944	1.28146	1.62194	-1.18135	0.16466	74.3457
17	.	.	.	.	0.82913	1.04942	-0.86526	-0.00541	43.2744
18	.	.	.	.	0.60921	0.77108	-0.98217	-0.05582	28.3955
19	.	.	.	.	0.79449	1.00558	-0.71096	0.01250	41.4512
20	.	.	.	.	0.57457	0.72723	-0.82787	-0.03791	26.5723
21	.	.	.	.	1.40370	1.77666	-1.69313	-0.04332	69.8467
22	.	.	.	.	0.79449	1.00558	-0.71096	0.01250	41.4512
23	.	.	.	.	0.56452	0.71451	-0.81337	-0.03725	54.1907
24	.	.	.	.	1.40370	1.77666	-1.69313	-0.04332	69.8467

The actual and predicted values for the variable C are plotted in Output 32.1.11.

```
title2 'Plots of Simulation Results';
proc sgplot data=c;
   scatter x=year y=c;
   series x=year y=chat / markers markerattrs=(symbol=plus);
   refline 1941.5 / axis=x;
run;
```

Output 32.1.11 Plot of Actual and Predicted Consumption



Example 32.2: Multipliers for a Third-Order System

This example shows how to fit and simulate a single equation dynamic model with third-order lags. It then shows how to convert the third-order equation into a three equation system with only first-order lags, so that the SIMLIN procedure can compute multipliers. (See the section "Multipliers for Higher Order Lags" earlier in this chapter for more information.)

The input data set TEST is created from simulated data. A partial listing of the data set TEST produced by PROC PRINT is shown in Output 32.2.1.

Output 32.2.1 Partial Listing of Input Data Set

Simulate Equation with Third-Order Lags
Listing of Simulated Input Data

Obs	y	ylag1	ylag2	ylag3	x	n
1	8.2369	8.5191	6.9491	7.8800	-1.2593	1
2	8.6285	8.2369	8.5191	6.9491	-1.6805	2
3	10.2223	8.6285	8.2369	8.5191	-1.9844	3
4	10.1372	10.2223	8.6285	8.2369	-1.7855	4
5	10.0360	10.1372	10.2223	8.6285	-1.8092	5
6	10.3560	10.0360	10.1372	10.2223	-1.3921	6
7	11.4835	10.3560	10.0360	10.1372	-2.0987	7
8	10.8508	11.4835	10.3560	10.0360	-1.8788	8
9	11.2684	10.8508	11.4835	10.3560	-1.7154	9
10	12.6310	11.2684	10.8508	11.4835	-1.8418	10

The REG procedure processes the input data and writes the parameter estimates to the OUTEST= data set A.

```
title2 'Estimated Parameters';
proc reg data=test outest=a;
   model y=ylag3 x;
run;

title2 'Listing of OUTEST= Data Set';
proc print data=a;
run;
```

Output 32.2.2 shows the printed output produced by the REG procedure, and Output 32.2.3 displays the OUTEST= data set A produced.

Output 32.2.2 Estimates and Fit Information from PROC REG

Simulate Equation with Third-Order Lags
Estimated Parameters

The REG Procedure
Model: MODEL1
Dependent Variable: y

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	2	173.98377	86.99189	1691.98	<.0001
Error	27	1.38818	0.05141		
Corrected Total	29	175.37196			

Root MSE	0.22675	R-Square	0.9921
Dependent Mean	13.05234	Adj R-Sq	0.9915
Coeff Var	1.73721		

Output 32.2.2 *continued*

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	0.14239	0.23657	0.60	0.5523
ylag3	1	0.77121	0.01723	44.77	<.0001
x	1	-1.77668	0.10843	-16.39	<.0001

Output 32.2.3 The OUTEST= Data Set Created by PROC REG

Simulate Equation with Third-Order Lags
Listing of OUTEST= Data Set

Obs	_MODEL_	_TYPE_	_DEPVAR_	_RMSE_	Intercept	ylag3	x	y
1	MODEL1	PARMS	y	0.22675	0.14239	0.77121	-1.77668	-1

The SIMLIN procedure processes the TEST data set using the estimates from PROC REG. The following statements perform the simulation and write the results to the OUT= data set OUT2.

```
title2 'Simulation of Equation';
proc simlin est=a data=test nored;
   endogenous y;
   exogenous  x;
   lagged ylag3 y 3;
   id n;
   output out=out1 predicted=yhat residual=yresid;
run;
```

The printed output from the SIMLIN procedure is shown in Output 32.2.4.

Output 32.2.4 Output Produced by PROC SIMLIN

Simulate Equation with Third-Order Lags
Simulation of Equation

The SIMLIN Procedure

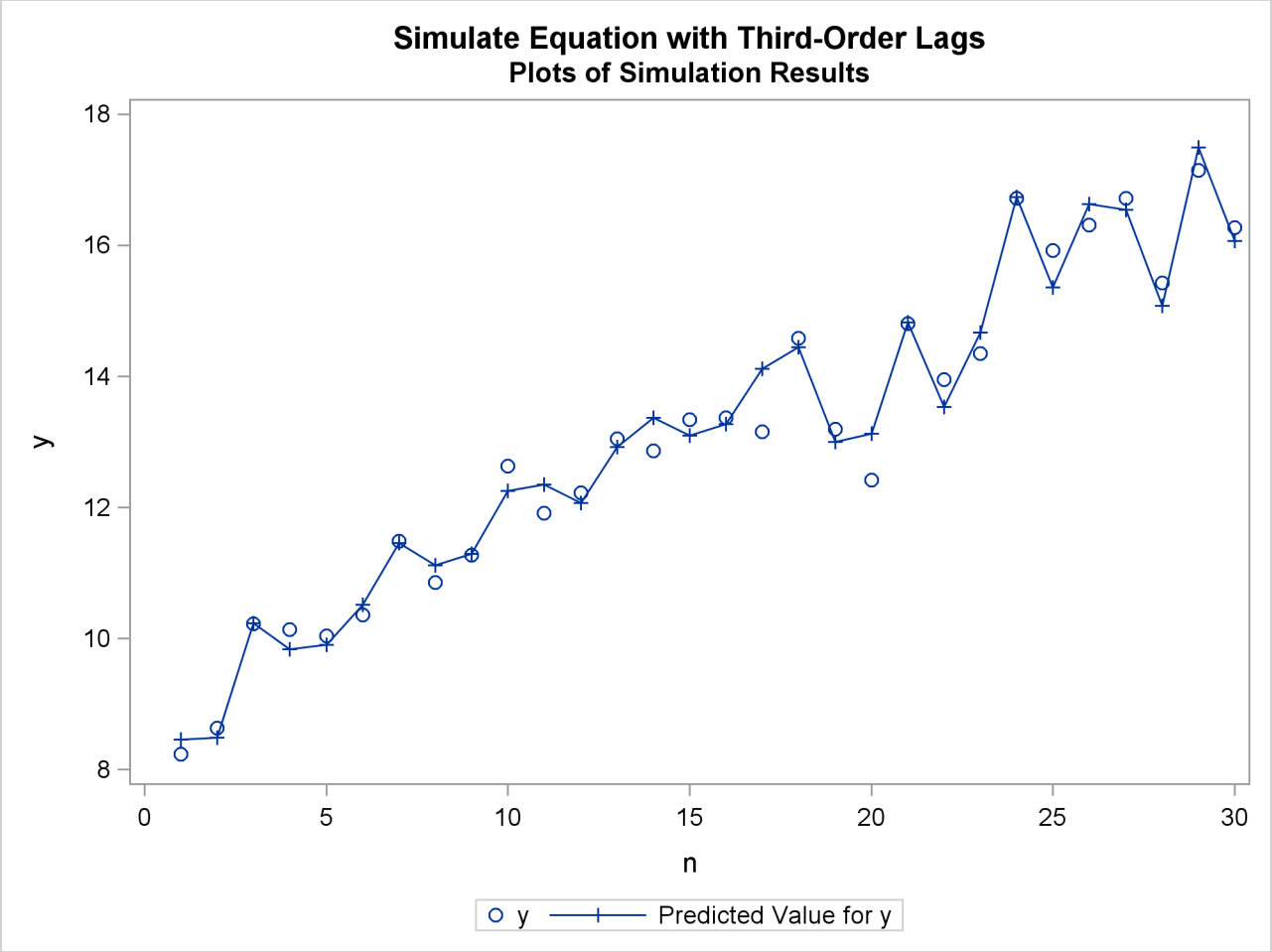
Fit Statistics

Variable	N	Mean Error	Mean Pct Error	Mean Abs Error	Mean Abs Pct Error	RMS Error	RMS Pct Error
y	30	-0.0233	-0.2268	0.2662	2.05684	0.3408	2.6159

The following statements plot the actual and predicted values, as shown in Output 32.2.5.

```
title2 'Plots of Simulation Results';
proc sgplot data=out1;
   scatter x=n y=y;
   series x=n y=yhat / markers markerattrs=(symbol=plus);
run;
```

Output 32.2.5 Plot of Predicted and Actual Values



Next, the input data set TEST is modified by creating two new variables, YLAG1X and YLAG2X, that are equal to YLAG1 and YLAG2. These variables are used in the SYSLIN procedure. (The estimates produced by PROC SYSLIN are the same as before and are not shown.) A listing of the OUTEST= data set B created by PROC SYSLIN is shown in Output 32.2.6.

```
data test2;
   set test;
   ylag1x=ylag1;
   ylag2x=ylag2;
run;

title2 'Estimation of parameters and definition of identities';
proc syslin data=test2 outest=b;
   endogenous y ylag1x ylag2x;
   model y=ylag3 x;
   identity ylag1x=ylag1;
   identity ylag2x=ylag2;
run;

title2 'Listing of OUTEST= data set from PROC SYSLIN';
proc print data=b;
run;
```

Output 32.2.6 Listing of OUTEST= Data Set Created from PROC SYSLIN

Simulate Equation with Third-Order Lags
Listing of OUTEST= data set from PROC SYSLIN

Obs	_TYPE_	_STATUS_	_MODEL_	_DEPVAR_	_SIGMA_	Intercept	ylag3	x	ylag1	ylag2	y	ylag1x	ylag2x
1	OLS	0 Converged	y	y	0.22675	0.14239	0.77121	-1.77668	.	.	-1	.	.
2	IDENTITY	0 Converged		ylag1x	.	0.00000	.	.	1	.	.	-1	.
3	IDENTITY	0 Converged		ylag2x	.	0.00000	.	.	.	1	.	.	-1

The SIMLIN procedure is used to compute the reduced form and multipliers. The OUTEST= data set B from PROC SYSLIN is used as the EST= data set for the SIMLIN procedure. The following statements perform the multiplier analysis.

```
title2 'Simulation of transformed first-order equation system';

proc simlin est=b data=test2 total interim=2;
   endogenous y ylag1x ylag2x;
   exogenous  x;
   lagged  ylag1 y 1  ylag2 ylag1x 1  ylag3 ylag2x 1;
   id n;
   output out=out2 predicted=yhat residual=yresid;
run;
```

Output 32.2.7 shows the interim 2 and total multipliers printed by the SIMLIN procedure.

Output 32.2.7 Interim 2 and Total Multipliers

Simulate Equation with Third-Order Lags
Simulation of transformed first-order equation system

The SIMLIN Procedure

Interim Multipliers for Interim 2

Variable	x	Intercept
y	0.000000	0.0000000
ylag1x	0.000000	0.0000000
ylag2x	-1.776682	0.1423865

Total Multipliers

Variable	x	Intercept
y	-7.765556	0.6223455
ylag1x	-7.765556	0.6223455
ylag2x	-7.765556	0.6223455

References

Maddala, G. S. (1977). *Econometrics*. New York: McGraw-Hill.

Pindyck, R. S., and Rubinfeld, D. L. (1991). *Econometric Models and Economic Forecasts*. 3rd ed. New York: McGraw-Hill.

Theil, H. (1971). *Principles of Econometrics*. New York: John Wiley & Sons.

Chapter 33
The SPECTRA Procedure

Contents

Overview: SPECTRA Procedure **2329**
Getting Started: SPECTRA Procedure **2331**
Syntax: SPECTRA Procedure **2332**
Functional Summary 2332
PROC SPECTRA Statement 2333
BY Statement 2334
VAR Statement 2334
WEIGHTS Statement 2335
Details: SPECTRA Procedure **2336**
Input Data 2336
Missing Values 2336
Computational Method 2336
Kernels 2336
White Noise Test 2339
Transforming Frequencies 2339
OUT= Data Set 2339
Printed Output 2341
ODS Table Names: SPECTRA procedure 2342
Examples: SPECTRA Procedure **2342**
Example 33.1: Spectral Analysis of Sunspot Activity 2342
Example 33.2: Cross-Spectral Analysis 2349
References **2352**

Overview: SPECTRA Procedure

The SPECTRA procedure performs spectral and cross-spectral analysis of time series. You can use spectral analysis techniques to look for periodicities or cyclical patterns in data.

The SPECTRA procedure produces estimates of the spectral and cross-spectral densities of a multivariate time series. Estimates of the spectral and cross-spectral densities of a multivariate time series are produced using a finite Fourier transform to obtain periodograms and cross-periodograms. The periodogram ordinates are smoothed by a moving average to produce estimated spectral and cross-spectral densities. PROC SPECTRA can also test whether or not the data are white noise.

PROC SPECTRA uses the finite Fourier transform to decompose data series into a sum of sine and cosine waves of different amplitudes and wavelengths. The finite Fourier transform decomposition of the series x_t is

$$x_t = \frac{a_0}{2} + \sum_{k=1}^{m-1} f_k (a_k \cos \omega_k t + b_k \sin \omega_k t)$$

$$f_k = \begin{cases} 1/2 & \text{if } n \text{ is even and } k = m-1 \\ 1 & \text{otherwise} \end{cases}$$

where

t	is the time subscript, $t = 0, 1, 2, \ldots, n-1$
x_t	are the equally spaced time series data
n	is the number of observations in the time series
m	is the number of frequencies in the Fourier decomposition: $m = \frac{n+2}{2}$ if n is even, $m = \frac{n+1}{2}$ if n is odd
k	is the frequency subscript, $k = 0, 1, 2, \ldots, m-1$
a_0	is the mean term: $a_0 = 2\overline{x}$
a_k	are the cosine coefficients
b_k	are the sine coefficients
ω_k	are the Fourier frequencies: $\omega_k = \frac{2\pi k}{n}$

Functions of the Fourier coefficients a_k and b_k can be plotted against frequency or against wave length to form *periodograms*. The amplitude periodogram J_k is defined as follows:

$$J_k = \frac{n}{2}(a_k^2 + b_k^2)$$

Several definitions of the term periodogram are used in the spectral analysis literature. The following discussion refers to the J_k sequence as the periodogram.

The periodogram can be interpreted as the contribution of the kth harmonic ω_k to the total sum of squares (in an analysis of variance sense) in the decomposition of the process into two-degree-of-freedom components for each of the m frequencies. When n is even, $sin(\omega_{\frac{n}{2}})$ is zero, and thus the last periodogram value is a one-degree-of-freedom component.

The periodogram is a volatile and inconsistent estimator of the spectrum. The spectral density estimate is produced by smoothing the periodogram. Smoothing reduces the variance of the estimator but introduces a bias. The weight function used for the smoothing process, W(), often called the kernel or spectral window, is specified with the WEIGHTS statement. It is related to another weight function, $w()$, the lag window, that is used in other methods to taper the correlogram rather than to smooth the periodogram. Many specific weighting functions have been suggested in the literature (Fuller 1976; Jenkins and Watts 1968; Priestley 1981). Table 33.3 later in this chapter gives the relevant formulas when the WEIGHTS statement is used.

Letting i represent the imaginary unit $\sqrt{-1}$, the cross-periodogram is defined as follows:

$$J_k^{xy} = \frac{n}{2}(a_k^x a_k^y + b_k^x b_k^y) + i\frac{n}{2}(a_k^x b_k^y - b_k^x a_k^y)$$

The cross-spectral density estimate is produced by smoothing the cross-periodogram in the same way as the periodograms are smoothed using the spectral window specified by the WEIGHTS statement.

The SPECTRA procedure creates an output SAS data set whose variables contain values of the periodograms, cross-periodograms, estimates of spectral densities, and estimates of cross-spectral densities. The form of the output data set is described in the section "OUT= Data Set" on page 2339.

Getting Started: SPECTRA Procedure

To use the SPECTRA procedure, specify the input and output data sets and options for the analysis you want in the PROC SPECTRA statement, and list the variables to analyze in the VAR statement. The procedure produces no printed output unless the WHITETEST option is specified in the PROC SPECTRA statement. The periodogram, spectral density, and other results are written to the OUT= data set, depending on the options used.

For example, to compute the Fourier transform of a variable X in a data set A, use the following statements:

```
proc spectra data=a out=b coef;
   var x;
run;
```

This PROC SPECTRA step writes the Fourier coefficients a_k and b_k to the variables COS_01 and SIN_01 in the output data set B.

When a WEIGHTS statement is specified, the periodogram is smoothed by a weighted moving average to produce an estimate of the spectral density of the series. The following statements write a spectral density estimate for X to the variable S_01 in the output data set B.

```
proc spectra data=a out=b s;
   var x;
   weights 1 2 3 4 3 2 1;
run;
```

When the VAR statement specifies more than one variable, you can perform cross-spectral analysis by specifying the CROSS option in the PROC SPECTRA statement. The CROSS option by itself produces the cross-periodograms for all two-way combinations of the variables listed in the VAR statement. For example, the following statements write the real and imaginary parts of the cross-periodogram of X and Y to the variables RP_01_02 and IP_01_02 in the output data set B.

```
proc spectra data=a out=b cross;
   var x y;
run;
```

To produce cross-spectral density estimates, specify both the CROSS option and the S option. The cross-periodogram is smoothed using the weights specified by the WEIGHTS statement in the same way as the spectral density. The squared coherency and phase estimates of the cross-spectrum are computed when the K and PH options are used.

The following example computes cross-spectral density estimates for the variables X and Y.

```
proc spectra data=a out=b cross s;
   var x y;
   weights 1 2 3 4 3 2 1;
run;
```

The real part and imaginary part of the cross-spectral density estimates are written to the variables CS_01_02 and QS_01_02, respectively.

Syntax: SPECTRA Procedure

The following statements are used with the SPECTRA procedure:

PROC SPECTRA *options* ;
  BY *variables* ;
  VAR *variables* ;
  WEIGHTS < *weights* > < *kernel* > ;

Functional Summary

Table 33.1 summarizes the statements and options that control the SPECTRA procedure.

Table 33.1 SPECTRA Functional Summary

Description	Statement	Option
Statements		
specify BY-group processing	BY	
specify the variables to be analyzed	VAR	
specify weights for spectral density estimates	WEIGHTS	
Data Set Options		
specify the input data set	PROC SPECTRA	DATA=
specify the output data set	PROC SPECTRA	OUT=
Output Control Options		
output the amplitudes of the cross-spectrum	PROC SPECTRA	A
output the Fourier coefficients	PROC SPECTRA	COEF
output the periodogram	PROC SPECTRA	P
output the spectral density estimates	PROC SPECTRA	S
output cross-spectral analysis results	PROC SPECTRA	CROSS
output squared coherency of the cross-spectrum	PROC SPECTRA	K
output the phase of the cross-spectrum	PROC SPECTRA	PH

Table 33.1 *continued*

Description	Statement	Option
Smoothing Options		
specify the Bartlett kernel	WEIGHTS	BART
specify the Parzen kernel	WEIGHTS	PARZEN
specify the quadratic spectral kernel	WEIGHTS	QS
specify the Tukey-Hanning kernel	WEIGHTS	TUKEY
specify the truncated kernel	WEIGHTS	TRUNCAT
Other Options		
subtract the series mean	PROC SPECTRA	ADJMEAN
specify an alternate quadrature spectrum estimate	PROC SPECTRA	ALTW
request tests for white noise	PROC SPECTRA	WHITETEST

PROC SPECTRA Statement

PROC SPECTRA *options* **;**

The following options can be used in the PROC SPECTRA statement:

A

outputs the amplitude variables (A_*nn* _*mm*) of the cross-spectrum.

ADJMEAN

CENTER

subtracts the series mean before performing the Fourier decomposition. This sets the first periodogram ordinate to 0 rather than $2n$ times the squared mean. This option is commonly used when the periodograms are to be plotted to prevent a large first periodogram ordinate from distorting the scale of the plot.

ALTW

specifies that the quadrature spectrum estimate is computed at the boundaries in the same way as the spectral density estimate and the cospectrum estimate are computed.

COEF

outputs the Fourier cosine and sine coefficients of each series.

CROSS

is used with the P and S options to output cross-periodograms and cross-spectral densities when more than one variable is listed in the VAR statement.

DATA=*SAS-data-set*
names the SAS data set that contains the input data. If the DATA= option is omitted, the most recently created SAS data set is used.

K
outputs the squared coherency variables (K_*nn* _*mm*) of the cross-spectrum. The K_*nn* _*mm* variables are identically 1 unless weights are given in the WEIGHTS statement and the S option is specified.

OUT=*SAS-data-set*
names the output data set created by PROC SPECTRA to store the results. If the OUT= option is omitted, the output data set is named by using the DATA*n* convention.

P
outputs the periodogram variables. The variables are named P_*nn*, where *nn* is an index of the original variable with which the periodogram variable is associated. When both the P and CROSS options are specified, the cross-periodogram variables RP_*nn*_*mm* and IP_*nn*_*mm* are also output.

PH
outputs the phase variables (PH_*nn* _*mm*) of the cross-spectrum.

S
outputs the spectral density estimates. The variables are named S_*nn*, where *nn* is an index of the original variable with which the estimate variable is associated. When both the S and CROSS options are specified, the cross-spectral variables CS_*nn* _*mm* and QS_*nn* _*mm* are also output.

WHITETEST
prints two tests of the hypothesis that the data are white noise. See the section "White Noise Test" on page 2339 for details.

Note that the CROSS, A, K, and PH options are meaningful only if more than one variable is listed in the VAR statement.

BY Statement

BY *variables* **;**

A BY statement can be used with PROC SPECTRA to obtain separate analyses for groups of observations defined by the BY variables.

VAR Statement

VAR *variables* **;**

The VAR statement specifies one or more numeric variables that contain the time series to analyze. The order of the variables in the VAR statement list determines the index, *nn*, used to name the output variables. The VAR statement is required.

WEIGHTS Statement

WEIGHTS *weight-constants | kernel-specification* **;**

The WEIGHTS statement specifies the relative weights used in the moving average applied to the periodogram ordinates to form the spectral density estimates. A WEIGHTS statement must be used to produce smoothed spectral density estimates. You can specify the relative weights in two ways: you can specify them explicitly as explained in the section "Using Weight Constants Specification" on page 2335, or you can specify them implicitly by using the kernel specification as explained in the section "Using Kernel Specifications" on page 2335. If the WEIGHTS statement is not used, only the periodogram is produced.

Using Weight Constants Specification

Any number of weighting constants can be specified. The constants should be positive and symmetric about the middle weight. The middle constant (or the constant to the right of the middle if an even number of weight constants are specified) is the relative weight of the current periodogram ordinate. The constant immediately following the middle one is the relative weight of the next periodogram ordinate, and so on. The actual weights used in the smoothing process are the weights specified in the WEIGHTS statement scaled so that they sum to $\frac{1}{4\pi}$.

The moving average reflects at each end of the periodogram. The first periodogram ordinate is not used; the second periodogram ordinate is used in its place.

For example, a simple triangular weighting can be specified using the following WEIGHTS statement:

```
weights 1 2 3 2 1;
```

Using Kernel Specifications

You can specify five different kernels in the WEIGHTS statement. The syntax for the statement is

WEIGHTS [PARZEN][BART][TUKEY][TRUNCAT][QS] [c e] ;

where $c >= 0$ and $e >= 0$ are used to compute the bandwidth parameter as

$$\mathrm{l}(q) = cq^e$$

and q is the number of periodogram ordinates +1:

$$q = \mathrm{floor}(n/2) + 1$$

To specify the bandwidth explicitly, set $c =$ to the desired bandwidth and $e = 0$.

For example, a Parzen kernel can be specified using the following WEIGHTS statement:

```
weights parzen 0.5 0;
```

For details, see the section "Kernels" on page 2336.

Details: SPECTRA Procedure

Input Data

Observations in the data set analyzed by the SPECTRA procedure should form ordered, equally spaced time series. No more than 99 variables can be included in the analysis.

Data are often detrended before analysis by the SPECTRA procedure. This can be done by using the residuals output by a SAS regression procedure. Optionally, the data can be centered using the ADJMEAN option in the PROC SPECTRA statement, since the zero periodogram ordinate corresponding to the mean is of little interest from the point of view of spectral analysis.

Missing Values

Missing values are excluded from the analysis by the SPECTRA procedure. If the SPECTRA procedure encounters missing values for any variable listed in the VAR statement, the procedure determines the longest contiguous span of data that has no missing values for the variables listed in the VAR statement and uses that span for the analysis.

Computational Method

If the number of observations n factors into prime integers that are less than or equal to 23, and the product of the square-free factors of n is less than 210, then PROC SPECTRA uses the fast Fourier transform developed by Cooley and Tukey and implemented by Singleton (1969). If n cannot be factored in this way, then PROC SPECTRA uses a Chirp-Z algorithm similar to that proposed by Monro and Branch (1977). To reduce memory requirements, when n is small, the Fourier coefficients are computed directly using the defining formulas.

Kernels

Kernels are used to smooth the periodogram by using a weighted moving average of nearby points. A smoothed periodogram is defined by the following equation.

$$\hat{J}_i(\mathrm{l}(q)) = \sum_{\tau=-\mathrm{l}(q)}^{\mathrm{l}(q)} \mathrm{w}\left(\frac{\tau}{\mathrm{l}(q)}\right) \tilde{J}_{i+\tau}$$

where $\mathrm{w}(x)$ is the kernel or weight function. At the endpoints, the moving average is computed cyclically; that is,

$$\tilde{J}_{i+\tau} = \begin{cases} J_{i+\tau} & 0 <= i+\tau <= q \\ J_{-(i+\tau)} & i+\tau < 0 \\ J_{q-(i+\tau)} & i+\tau > q \end{cases}$$

The SPECTRA procedure supports the following kernels. They are listed with their default bandwidth functions.

Bartlett: KERNEL BART

$$\begin{aligned} \mathrm{w}(x) &= \begin{cases} 1-|x| & |x| \leq 1 \\ 0 & \text{otherwise} \end{cases} \\ \mathrm{l}(q) &= \frac{1}{2} q^{1/3} \end{aligned}$$

Parzen: KERNEL PARZEN

$$\begin{aligned} \mathrm{w}(x) &= \begin{cases} 1-6|x|^2+6|x|^3 & 0 \leq |x| \leq \frac{1}{2} \\ 2(1-|x|)^3 & \frac{1}{2} \leq |x| \leq 1 \\ 0 & \text{otherwise} \end{cases} \\ \mathrm{l}(q) &= q^{1/5} \end{aligned}$$

Quadratic spectral: KERNEL QS

$$\begin{aligned} \mathrm{w}(x) &= \frac{25}{12\pi^2 x^2} \left(\frac{sin(6\pi x/5)}{6\pi x/5} - cos(6\pi x/5) \right) \\ \mathrm{l}(q) &= \frac{1}{2} q^{1/5} \end{aligned}$$

Tukey-Hanning: KERNEL TUKEY

$$\begin{aligned} \mathrm{w}(x) &= \begin{cases} (1+cos(\pi x))/2 & |x| \leq 1 \\ 0 & \text{otherwise} \end{cases} \\ \mathrm{l}(q) &= \frac{2}{3} q^{1/5} \end{aligned}$$

Truncated: KERNEL TRUNCAT

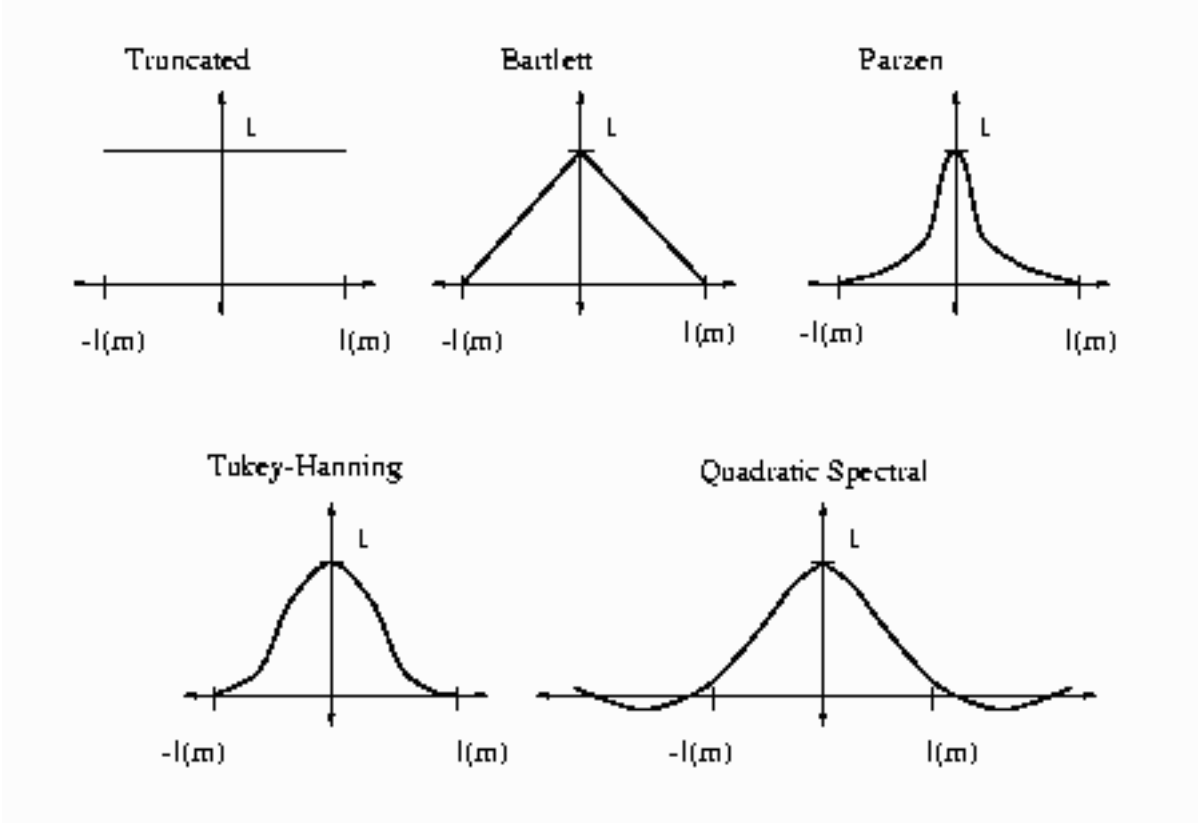
$$\begin{aligned} \mathrm{w}(x) &= \begin{cases} 1 & |x| \leq 1 \\ 0 & \text{otherwise} \end{cases} \\ \mathrm{l}(q) &= \frac{1}{4} q^{1/5} \end{aligned}$$

A summary of the default values of the bandwidth parameters, c and e, associated with the kernel smoothers in PROC SPECTRA are listed below in Table 33.2:

Table 33.2 Bandwidth Parameters

Kernel	c	e
Bartlett	1/2	1/3
Parzen	1	1/5
quadratic	1/2	1/5
Tukey-Hanning	2/3	1/5
truncated	1/4	1/5

Figure 33.1 Kernels for Smoothing



See Andrews (1991) for details about the properties of these kernels.

White Noise Test

PROC SPECTRA prints two test statistics for white noise when the WHITETEST option is specified: Fisher's Kappa (Davis 1941; Fuller 1976) and Bartlett's Kolmogorov-Smirnov statistic (Bartlett 1966; Fuller 1976; Durbin 1967).

If the time series is a sequence of independent random variables with mean 0 and variance σ^2, then the periodogram, J_k, will have the same expected value for all k. For a time series with nonzero autocorrelation, each ordinate of the periodogram, J_k, will have different expected values. The Fisher's Kappa statistic tests whether the largest J_k can be considered different from the mean of the J_k. Critical values for the Fisher's Kappa test can be found in Fuller 1976.

The Kolmogorov-Smirnov statistic reported by PROC SPECTRA has the same asymptotic distribution as Bartlett's test (Durbin 1967). The Kolmogorov-Smirnov statistic compares the normalized cumulative periodogram with the cumulative distribution function of a uniform(0,1) random variable. The normalized cumulative periodogram, F_j, of the series is

$$F_j = \frac{\sum_{k=1}^{j} J_k}{\sum_{k=1}^{m} J_k}, j = 1, 2 \ldots, m-1$$

where $m = \frac{n}{2}$ if n is even or $m = \frac{n-1}{2}$ if n is odd. The test statistic is the maximum absolute difference of the normalized cumulative periodogram and the uniform cumulative distribution function. Approximate p-values for Bartlett's Kolmogorov-Smirnov test statistics are provided with the test statistics. Small p-values cause you to reject the null-hypothesis that the series is white noise.

Transforming Frequencies

The variable FREQ in the data set created by the SPECTRA procedure ranges from 0 to π. Sometimes it is preferable to express frequencies in cycles per observation period, which is equal to $\frac{1}{2\pi}$FREQ.

To express frequencies in cycles per unit time (for example, in cycles per year), multiply FREQ by $\frac{d}{2\pi}$, where d is the number of observations per unit of time. For example, for monthly data, if the desired time unit is years then d is 12. The period of the cycle is $\frac{2\pi}{d \times \mathrm{FREQ}}$, which ranges from $\frac{2}{d}$ to infinity.

OUT= Data Set

The OUT= data set contains $\frac{n}{2} + 1$ observations, if n is even, or $\frac{n+1}{2}$ observations, if n is odd, where n is the number of observations in the time series or the span of data being analyzed if missing values are present in the data. See the section "Missing Values" on page 2336 for details.

The variables in the new data set are named according to the following conventions. Each variable to be analyzed is associated with an index. The first variable listed in the VAR statement is indexed as 01, the second variable as 02, and so on. Output variables are named by combining indexes with prefixes. The prefix always identifies the nature of the new variable, and the indices identify the original variables from which the statistics were obtained.

Variables that contain spectral analysis results have names that consist of a prefix, an underscore, and the index of the variable analyzed. For example, the variable S_01 contains spectral density estimates for the

first variable in the VAR statement. Variables that contain cross-spectral analysis results have names that consist of a prefix, an underscore, the index of the first variable, another underscore, and the index of the second variable. For example, the variable A_01_02 contains the amplitude of the cross-spectral density estimate for the first and second variables in the VAR statement.

Table 33.3 shows the formulas and naming conventions used for the variables in the OUT= data set. Let X be variable number *nn* in the VAR statement list and let Y be variable number *mm* in the VAR statement list. Table 33.3 shows the output variables that contain the results of the spectral and cross-spectral analysis of X and Y.

In Table 33.3 the following notation is used. Let W_j be the vector of $2p+1$ smoothing weights given by the WEIGHTS statement, normalized to sum to $\frac{1}{4\pi}$. Note that the weights are either explicitly provided using the constant specification or are implicitly determined by the kernel specification in the WEIGHTS statement.

The subscript of W_j runs from W_{-p} to W_p, so that W_0 is the middle weight in the list. Let $\omega_k = \frac{2\pi k}{n}$, where $k = 0, 1, \ldots, \text{floor}(\frac{n}{2})$.

Table 33.3 Variables Created by PROC SPECTRA

Variable	Description
FREQ	frequency in radians from 0 to π (Note: Cycles per observation is $\frac{\text{FREQ}}{2\pi}$.)
PERIOD	period or wavelength: $\frac{2\pi}{\text{FREQ}}$ (Note: PERIOD is missing for FREQ=0.)
COS_nn	cosine transform of X: $a_k^x = \frac{2}{n}\sum_{t=1}^{n} X_t \cos(\omega_k(t-1))$
SIN_nn	sine transform of X: $b_k^x = \frac{2}{n}\sum_{t=1}^{n} X_t \sin(\omega_k(t-1))$
P_*nn*	periodogram of X: $J_k^x = \frac{n}{2}[(a_k^x)^2 + (b_k^x)^2]$
S_*nn*	spectral density estimate of X: $F_k^x = \sum_{j=-p}^{p} W_j J_{k+j}^x$ (except across endpoints)
RP_*nn* _*mm*	real part of cross-periodogram X and Y: $\text{real}(J_k^{xy}) = \frac{n}{2}(a_k^x a_k^y + b_k^x b_k^y)$
IP_*nn* _*mm*	imaginary part of cross-periodogram of X and Y: $\text{imag}(J_k^{xy}) = \frac{n}{2}(a_k^x b_k^y - b_k^x a_k^y)$
CS_*nn* _*mm*	cospectrum estimate (real part of cross-spectrum) of X and Y: $C_k^{xy} = \sum_{j=-p}^{p} W_j \text{real}(J_{k+j}^{xy})$(except across endpoints)

Table 33.3 *continued*

Variable	Variable
QS_*nn* _*mm*	quadrature spectrum estimate (imaginary part of cross-spectrum) of X and Y: $Q_k^{xy} = \sum_{j=-p}^{p} W_j \text{imag}(J_{k+j}^{xy})$(except across end-points)
A_*nn* _*mm*	amplitude (modulus) of cross-spectrum of X and Y: $A_k^{xy} = \sqrt{(C_k^{xy})^2 + (Q_k^{xy})^2}$
K_*nn* _*mm*	coherency squared of X and Y: $K_k^{xy} = (A_k^{xy})^2/(F_k^x F_k^y)$
PH_*nn* _*mm*	phase spectrum in radians of X and Y: $\Phi_k^{xy} = \arctan(Q_k^{xy}/C_k^{xy})$

Printed Output

By default PROC SPECTRA produces no printed output.

When the WHITETEST option is specified, the SPECTRA procedure prints the following statistics for each variable in the VAR statement:

1. the name of the variable
2. M–1, the number of two-degree-of-freedom periodogram ordinates used in the test
3. MAX(P(*)), the maximum periodogram ordinate
4. SUM(P(*)), the sum of the periodogram ordinates
5. Fisher's Kappa statistic
6. Bartlett's Kolmogorov-Smirnov test statistic
7. Approximate *p*-value for Bartlett's Kolmogorov-Smirnov test statistic

See the section "White Noise Test" on page 2339 for details.

ODS Table Names: SPECTRA procedure

PROC SPECTRA assigns a name to each table it creates. You can use these names to reference the table when you use the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in the following table.

Table 33.4 ODS Tables Produced in PROC SPECTRA

ODS Table Name	Description	Option
WhiteNoiseTest	white noise test	WHITETEST
Kappa	Fishers Kappa statistic	WHITETEST
Bartlett	Bartletts Kolmogorov-Smirnov statistic	WHITETEST

Examples: SPECTRA Procedure

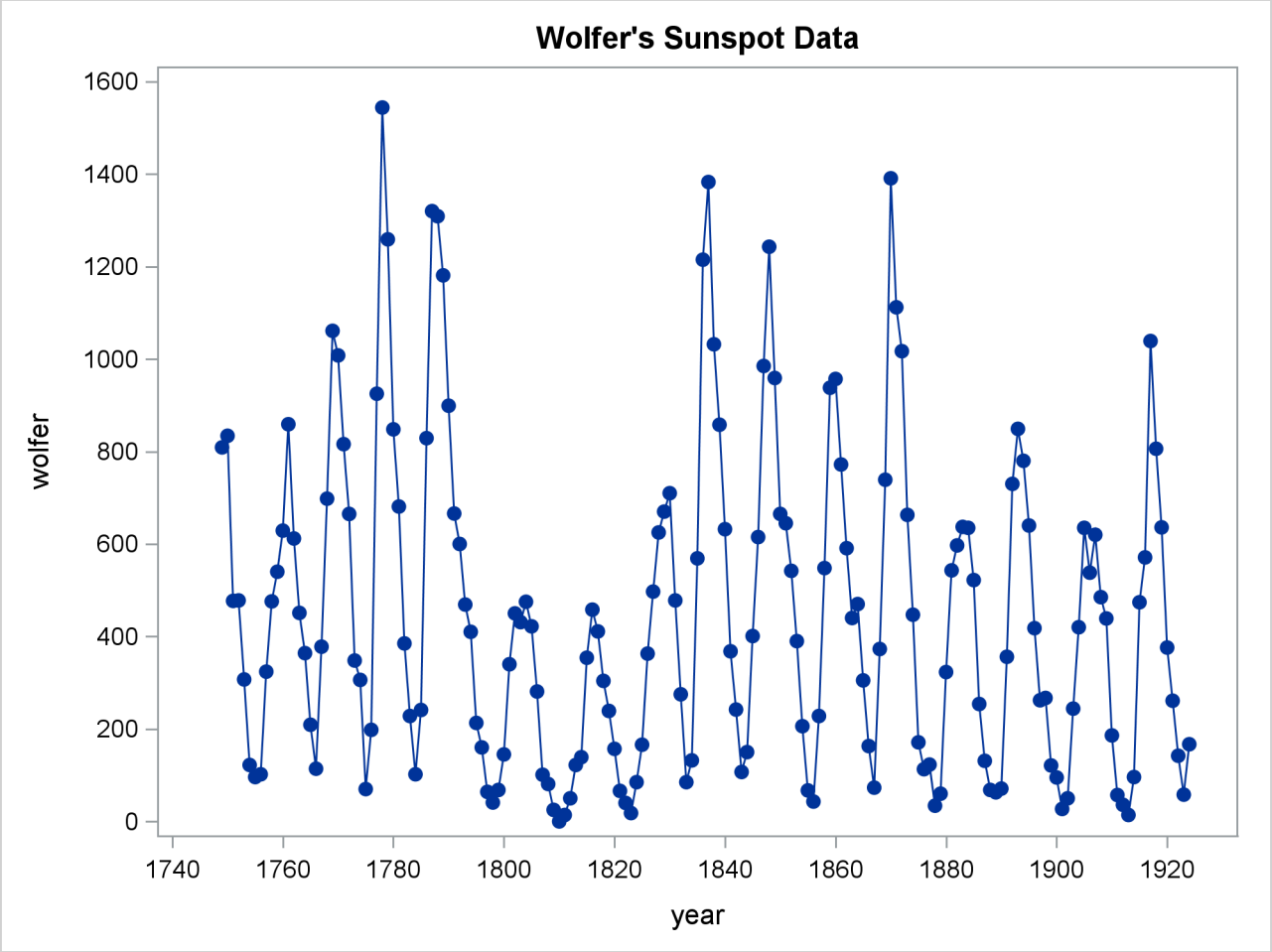
Example 33.1: Spectral Analysis of Sunspot Activity

This example analyzes Wolfer's sunspot data (Anderson 1971). The following statements read and plot the data.

```
title "Wolfer's Sunspot Data";
data sunspot;
   input year wolfer @@;
datalines;
1749  809 1750  834 1751  477 1752  478 1753  307 1754  122 1755   96

   ... more lines ...


proc sgplot data=sunspot;
   series x=year y=wolfer / markers markerattrs=(symbol=circlefilled);
   xaxis values=(1740 to 1930 by 10);
   yaxis values=(0 to 1600 by 200);
run;
```

The plot of the sunspot series is shown in Output 33.1.1.

Output 33.1.1 Plot of Original Sunspot Data



The spectral analysis of the sunspot series is performed by the following statements:

```
proc spectra data=sunspot out=b p s adjmean whitetest;
   var wolfer;
   weights 1 2 3 4 3 2 1;
run;

proc print data=b(obs=12);
run;
```

The PROC SPECTRA statement specifies the P and S options to write the periodogram and spectral density estimates to the OUT= data set B. The WEIGHTS statement specifies a triangular spectral window for smoothing the periodogram to produce the spectral density estimate. The ADJMEAN option zeros the frequency 0 value and avoids the need to exclude that observation from the plots. The WHITETEST option prints tests for white noise.

The Fisher's Kappa test statistic of 16.070 is larger than the 5% critical value of 7.2, so the null hypothesis that the sunspot series is white noise is rejected (see the table of critical values in Fuller (1976)).

The Bartlett's Kolmogorov-Smirnov statistic is 0.6501, and its approximate p-value is < 0.0001. The small p-value associated with this test leads to the rejection of the null hypothesis that the spectrum represents white noise.

The printed output produced by PROC SPECTRA is shown in Output 33.1.2. The output data set B created by PROC SPECTRA is shown in part in Output 33.1.3.

Output 33.1.2 White Noise Test Results

Wolfer's Sunspot Data

The SPECTRA Procedure

Test for White Noise for Variable wolfer	
M-1	87
Max(P(*))	4062267
Sum(P(*))	21156512

Fisher's Kappa: (M-1)*Max(P(*))/Sum(P(*))	
Kappa	16.70489

Bartlett's Kolmogorov-Smirnov Statistic: Maximum absolute difference of the standardized partial sums of the periodogram and the CDF of a uniform(0,1) random variable.	
Test Statistic	0.650055
Approximate P-Value	<.0001

Output 33.1.3 First 12 Observations of the OUT= Data Set

Wolfer's Sunspot Data

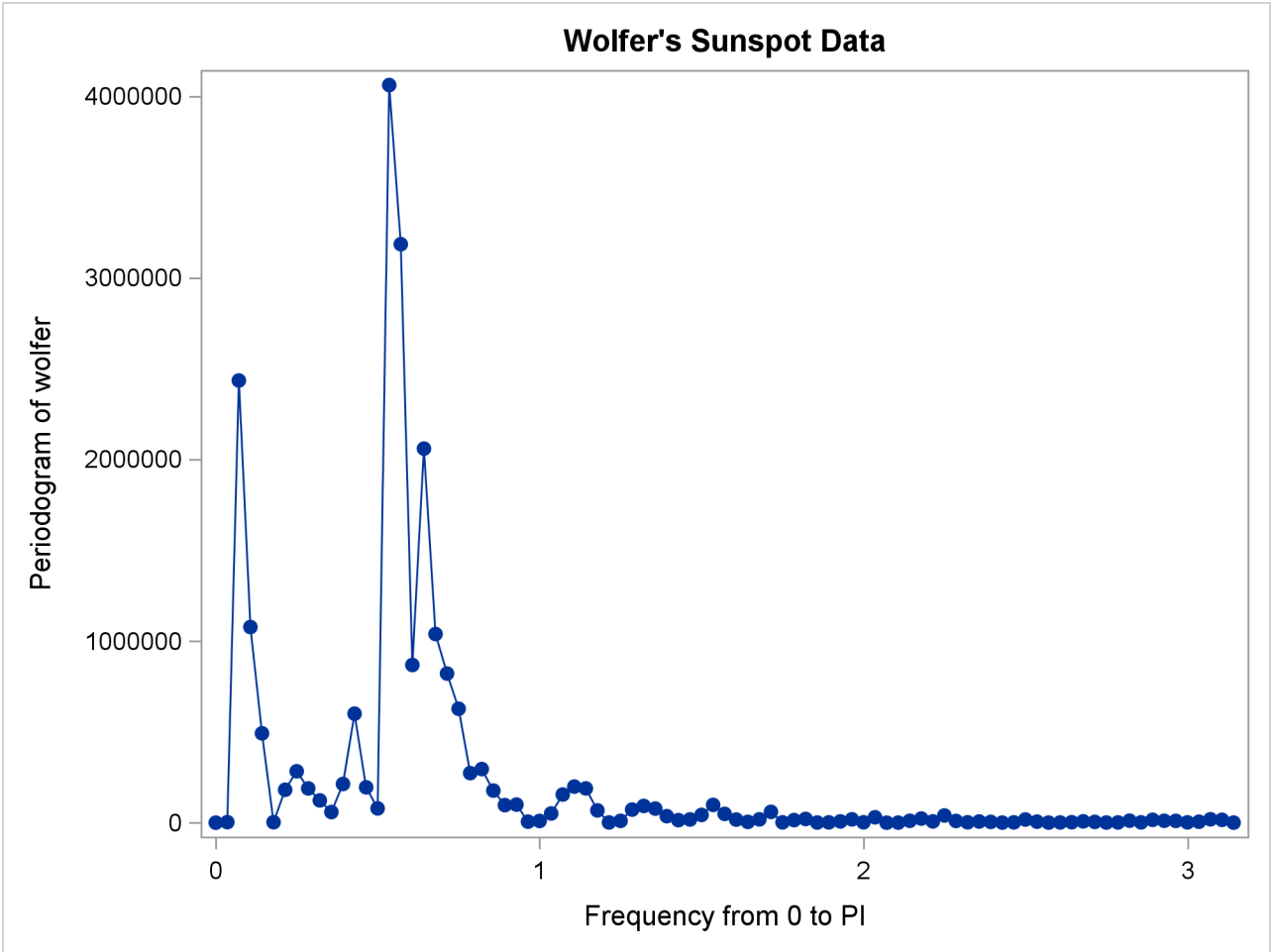
Obs	FREQ	PERIOD	P_01	S_01
1	0.00000	.	0.00	59327.52
2	0.03570	176.000	3178.15	61757.98
3	0.07140	88.000	2435433.22	69528.68
4	0.10710	58.667	1077495.76	66087.57
5	0.14280	44.000	491850.36	53352.02
6	0.17850	35.200	2581.12	36678.14
7	0.21420	29.333	181163.15	20604.52
8	0.24990	25.143	283057.60	15132.81
9	0.28560	22.000	188672.97	13265.89
10	0.32130	19.556	122673.94	14953.32
11	0.35700	17.600	58532.93	16402.84
12	0.39270	16.000	213405.16	18562.13

The following statements plot the periodogram and spectral density estimate by the frequency and period.

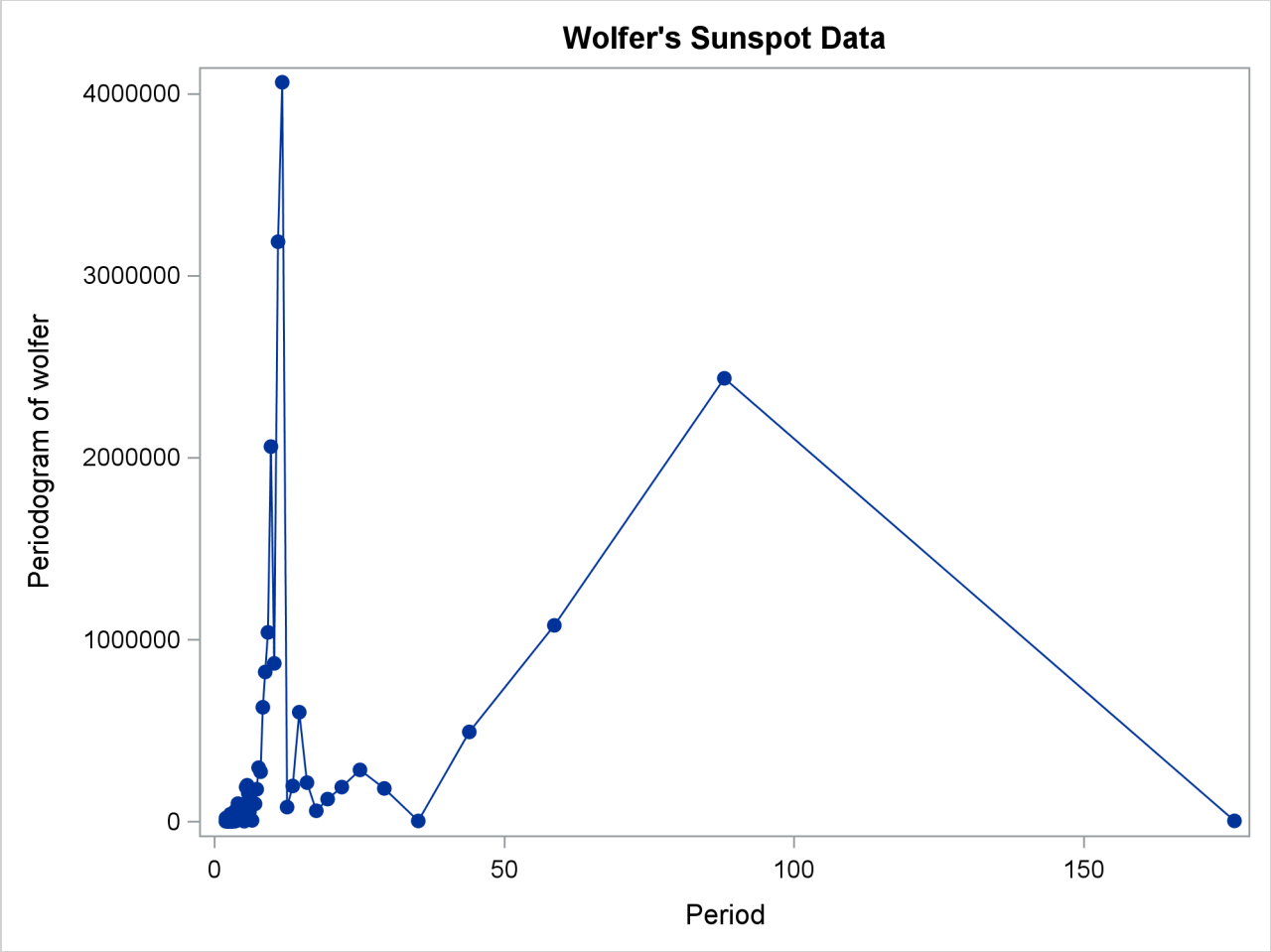
```
proc sgplot data=b;
   series x=freq y=p_01 / markers markerattrs=(symbol=circlefilled);
run;

proc sgplot data=b;
   series x=period y=p_01 / markers markerattrs=(symbol=circlefilled);
run;

proc sgplot data=b;
   series x=freq y=s_01 / markers markerattrs=(symbol=circlefilled);
run;

proc sgplot data=b;
   series x=period y=s_01 / markers markerattrs=(symbol=circlefilled);
run;
```

The periodogram is plotted against the frequency in Output 33.1.4 and plotted against the period in Output 33.1.5. The spectral density estimate is plotted against the frequency in Output 33.1.6 and plotted against the period in Output 33.1.7.

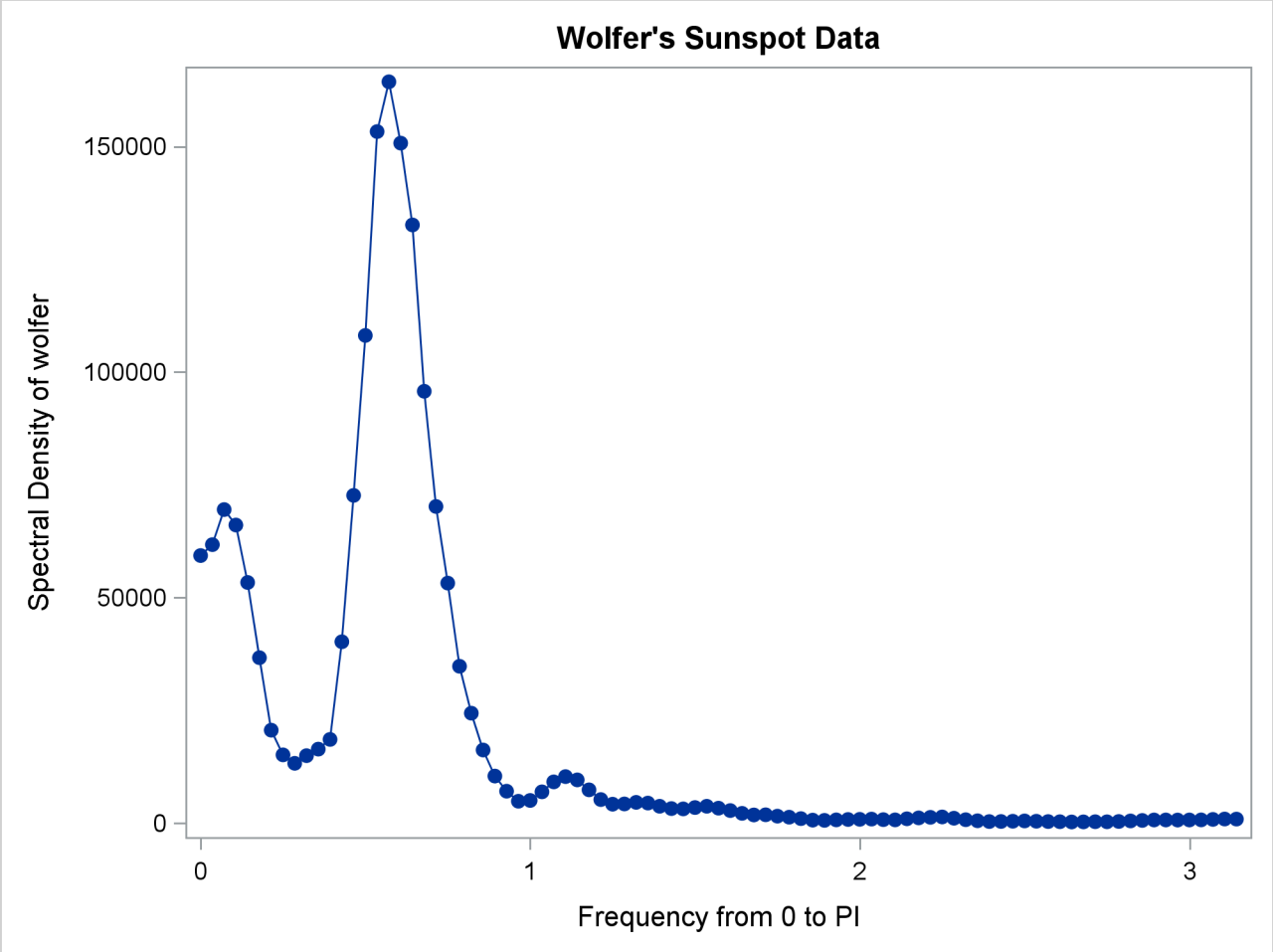
Output 33.1.4 Plot of Periodogram by Frequency



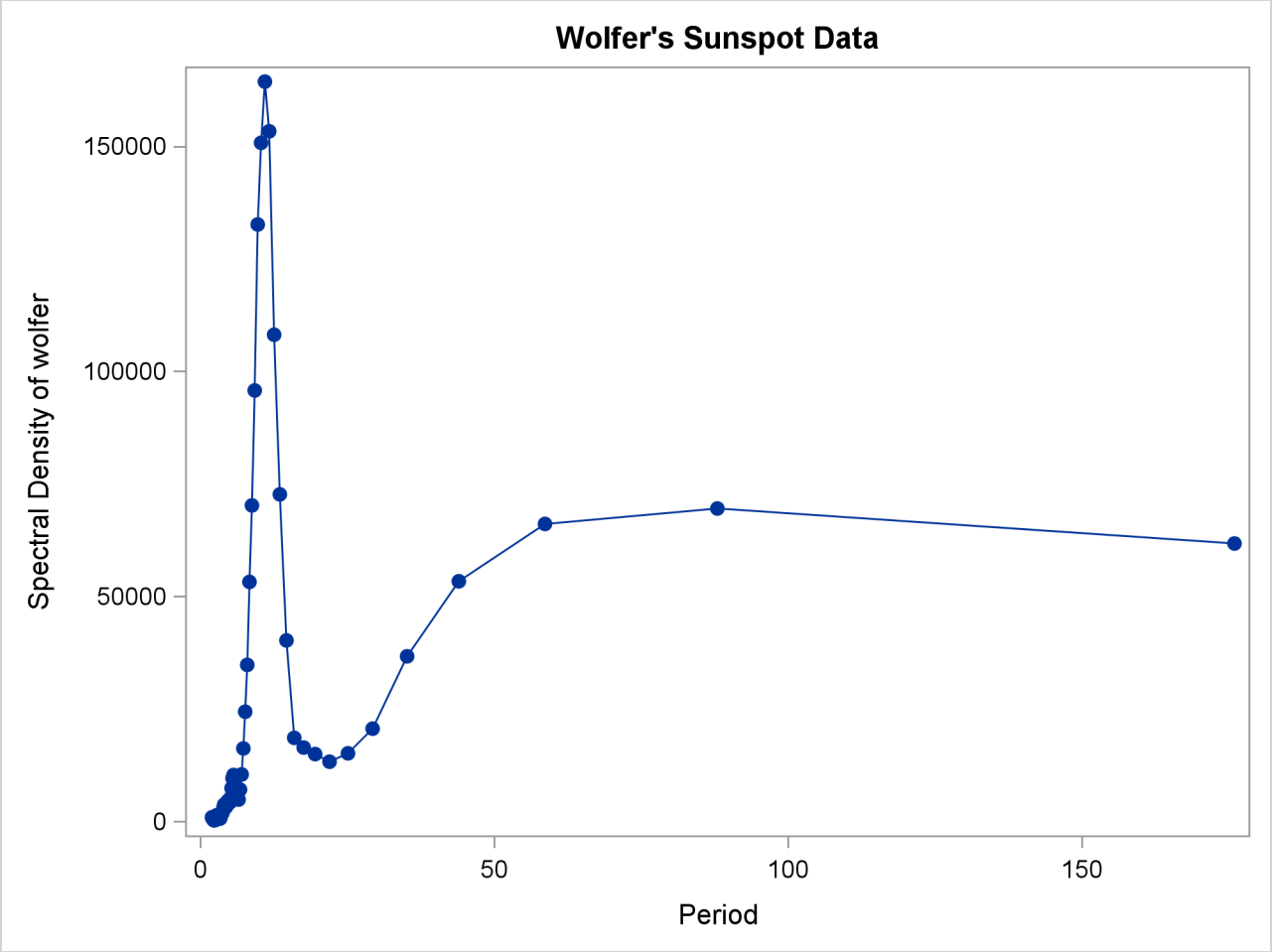
Output 33.1.5 Plot of Periodogram by Period



Output 33.1.6 Plot of Spectral Density Estimate by Frequency



Output 33.1.7 Plot of Spectral Density Estimate by Period



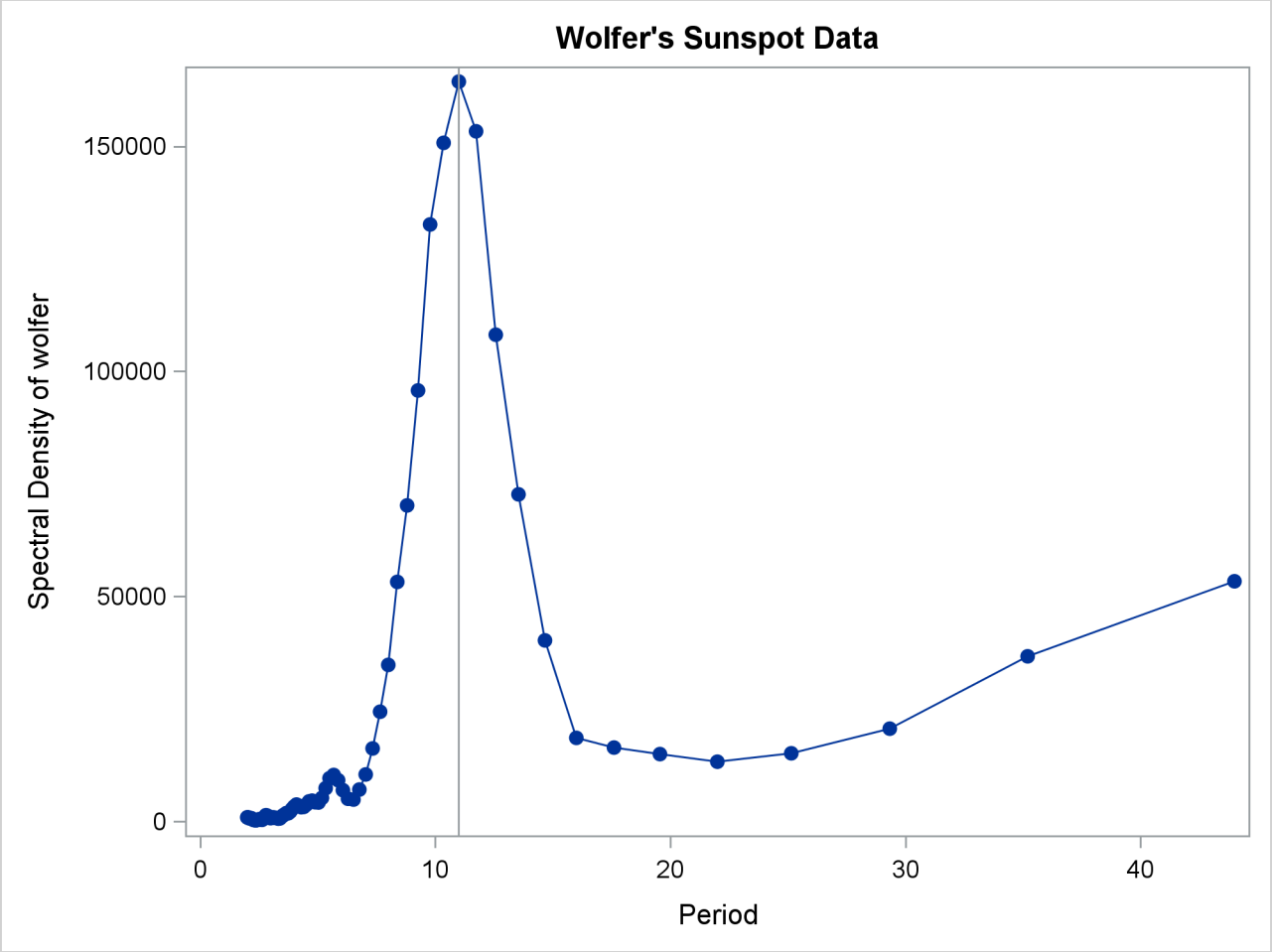
Since PERIOD is the reciprocal of frequency, the plot axis for PERIOD is stretched for low frequencies and compressed at high frequencies. One way to correct for this is to use a WHERE statement to restrict the plots and exclude the low frequency components. The following statements plot the spectral density for periods less than 50.

```
proc sgplot data=b;
   where period < 50;
   series x=period y=s_01 / markers markerattrs=(symbol=circlefilled);
   refline 11 / axis=x;
run;
title;
```

The spectral analysis of the sunspot series confirms a strong 11-year cycle of sunspot activity. The plot makes this clear by drawing a reference line at the 11 year period, which highlights the position of the main peak in the spectral density.

Output 33.1.8 shows the plot. Contrast Output 33.1.8 with Output 33.1.7.

Output 33.1.8 Plot of Spectral Density Estimate by Period to 50 Years



Example 33.2: Cross-Spectral Analysis

This example uses simulated data to show cross-spectral analysis for two variables X and Y. X is generated by an AR(1) process; Y is generated as white noise plus an input from X lagged 2 periods. All output options are specified in the PROC SPECTRA statement. PROC CONTENTS shows the contents of the OUT= data set.

```
data a;
   xl = 0; xll = 0;
   do i = - 10 to 100;
      x = .4 * xl  + rannor(123);
      y = .5 * xll + rannor(123);
      if i > 0 then output;
      xll = xl; xl = x;
   end;
run;

proc spectra data=a out=b cross coef a k p ph s;
   var x y;
   weights 1 1.5 2 4 8 9 8 4 2 1.5 1;
run;

proc contents data=b position;
run;
```

The PROC CONTENTS report for the output data set B is shown in Output 33.2.1.

Output 33.2.1 Contents of PROC SPECTRA OUT= Data Set

The CONTENTS Procedure

Alphabetic List of Variables and Attributes

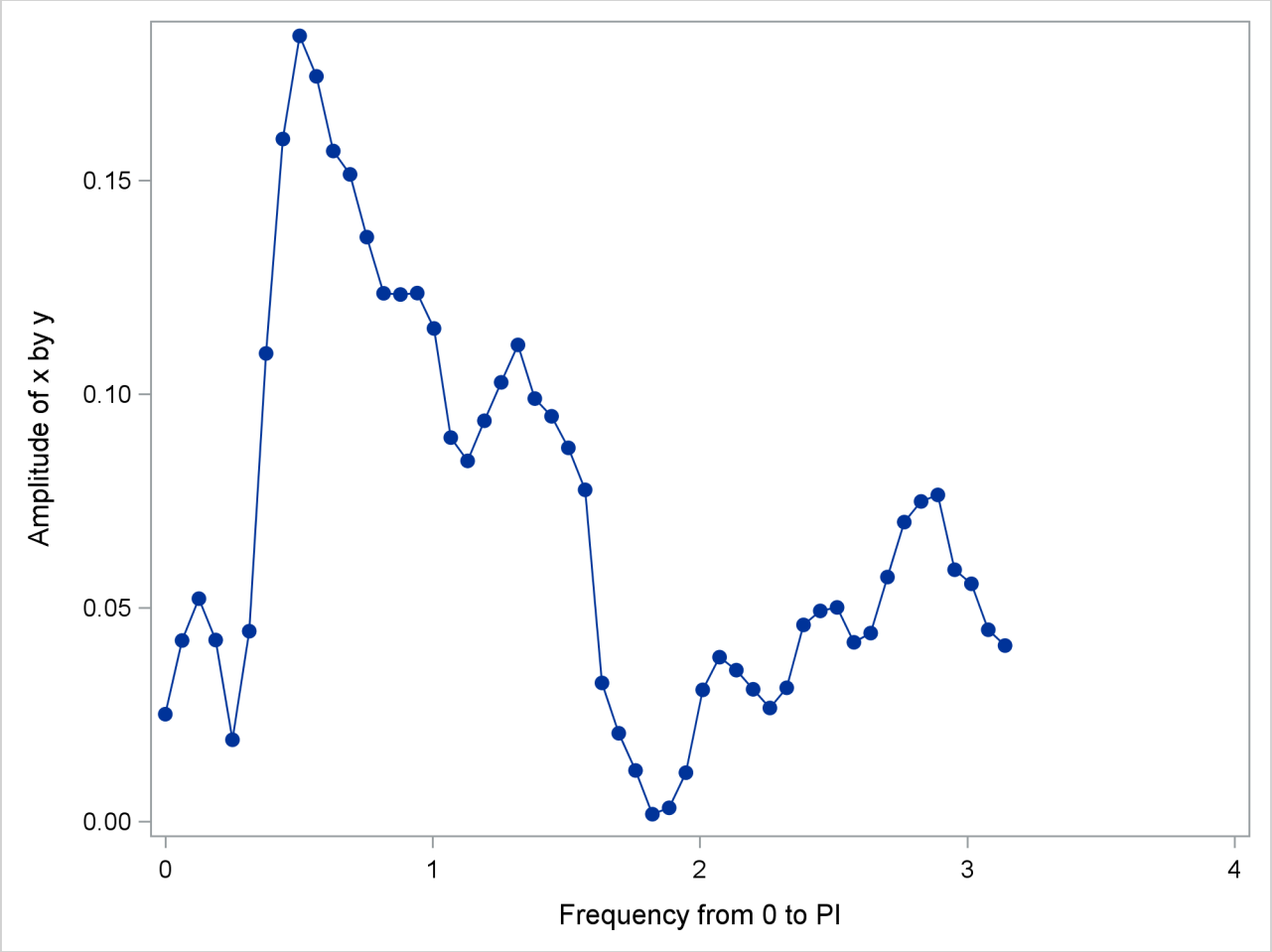
#	Variable	Type	Len	Label
16	A_01_02	Num	8	Amplitude of x by y
3	COS_01	Num	8	Cosine Transform of x
5	COS_02	Num	8	Cosine Transform of y
13	CS_01_02	Num	8	Cospectra of x by y
1	FREQ	Num	8	Frequency from 0 to PI
12	IP_01_02	Num	8	Imag Periodogram of x by y
15	K_01_02	Num	8	Coherency**2 of x by y
2	PERIOD	Num	8	Period
17	PH_01_02	Num	8	Phase of x by y
7	P_01	Num	8	Periodogram of x
8	P_02	Num	8	Periodogram of y
14	QS_01_02	Num	8	Quadrature of x by y
11	RP_01_02	Num	8	Real Periodogram of x by y
4	SIN_01	Num	8	Sine Transform of x
6	SIN_02	Num	8	Sine Transform of y
9	S_01	Num	8	Spectral Density of x
10	S_02	Num	8	Spectral Density of y

The following statements plot the amplitude of the cross-spectrum estimate against frequency and against period for periods less than 25.

```
proc sgplot data=b;
   series x=freq y=a_01_02 / markers markerattrs=(symbol=circlefilled);
   xaxis values=(0 to 4 by 1);
run;
```

The plot of the amplitude of the cross-spectrum estimate against frequency is shown in Output 33.2.2.

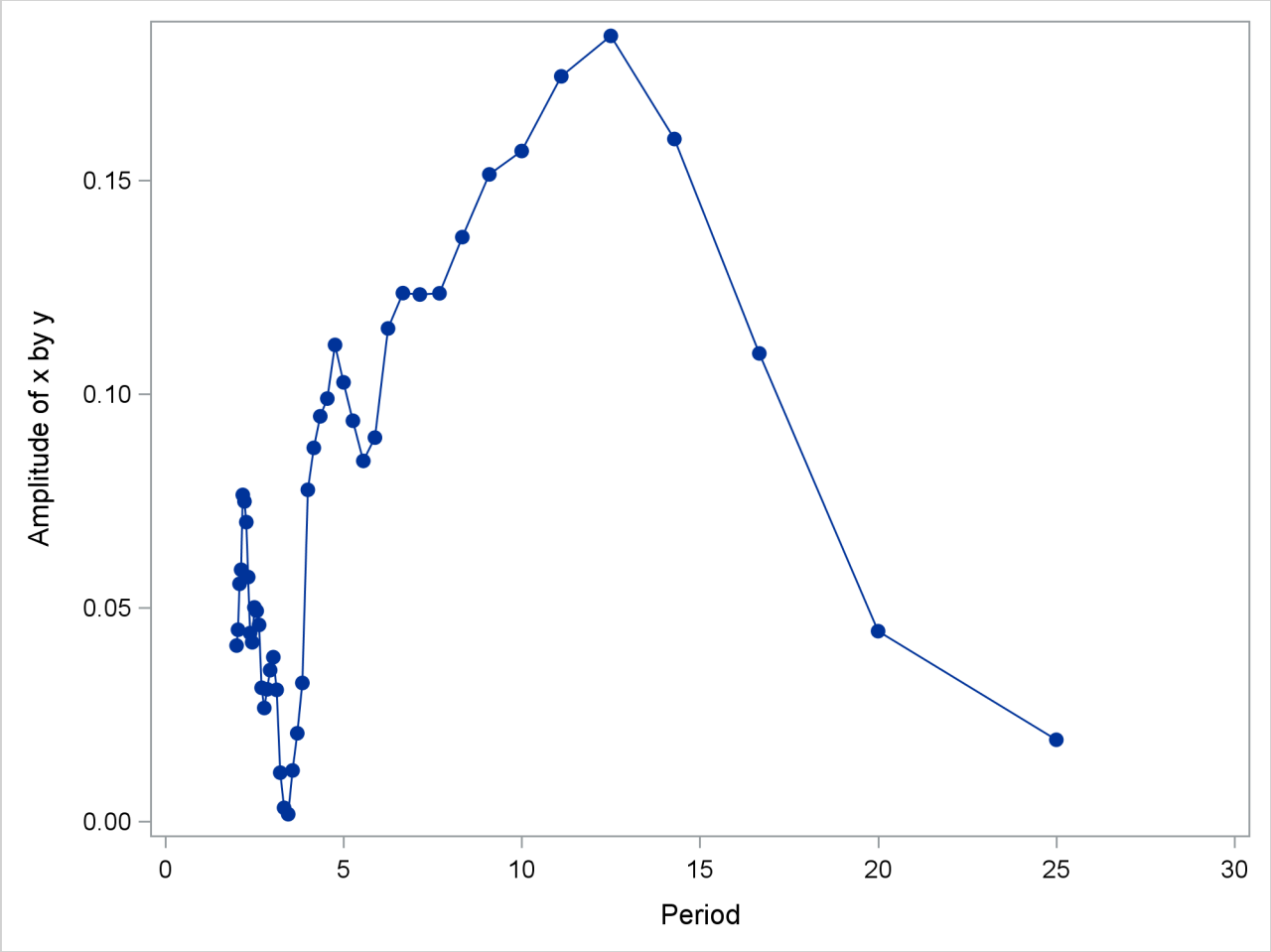
Output 33.2.2 Plot of Cross-Spectrum Amplitude by Frequency



The plot of the cross-spectrum amplitude against period for periods less than 25 observations is shown in Output 33.2.3.

```
proc sgplot data=b;
   where period < 25;
   series x=period y=a_01_02 / markers markerattrs=(symbol=circlefilled);
   xaxis values=(0 to 30 by 5);
run;
```

Output 33.2.3 Plot of Cross-Spectrum Amplitude by Period



References

Anderson, T. W. (1971). *The Statistical Analysis of Time Series*. New York: John Wiley & Sons.

Andrews, D. W. K. (1991). "Heteroscedasticity and Autocorrelation Consistent Covariance Matrix Estimation." *Econometrica* 59:817–858.

Bartlett, M. S. (1966). *An Introduction to Stochastic Processes*. 2nd ed. Cambridge: Cambridge University Press.

Brillinger, D. R. (1975). *Time Series: Data Analysis and Theory*. New York: Holt, Rinehart & Winston.

Davis, H. T. (1941). *The Analysis of Economic Time Series*. Bloomington, IN: Principia Press.

Durbin, J. (1967). "Tests of Serial Independence Based on the Cumulated Periodogram." *Bulletin of the International Statistical Institute* 42:1039–1049.

Fuller, W. A. (1976). *Introduction to Statistical Time Series*. New York: John Wiley & Sons.

Gentleman, W. M., and Sande, G. (1966). "Fast Fourier Transforms for Fun and Profit." *AFIPS Proceedings of the Fall Joint Computer Conference* 19:563–578.

Jenkins, G. M., and Watts, D. G. (1968). *Spectral Analysis and Its Applications*. San Francisco: Holden-Day.

Miller, L. H. (1956). "Tables of Percentage Points of Kolmogorov Statistics." *Journal of the American Statistical Association* 51:111.

Monro, D. M., and Branch, J. L. (1977). "Algorithm AS 117: The Chirp Discrete Fourier Transform of General Length." *Journal of the Royal Statistical Society, Series C* 26:351–361.

Nussbaumer, H. J. (1982). *Fast Fourier Transform and Convolution Algorithms*. 2nd ed. New York: Springer-Verlag.

Owen, D. B. (1962). *Handbook of Statistical Tables*. Reading, MA: Addison-Wesley.

Parzen, E. (1957). "On Consistent Estimates of the Spectrum of a Stationary Time Series." *Annals of Mathematical Statistics* 28:329–348.

Priestley, M. B. (1981). *Spectral Analysis and Time Series*. London: Academic Press.

Singleton, R. C. (1969). "An Algorithm for Computing the Mixed Radix Fast Fourier Transform." *IEEE Transactions on Audio and Electroacoustics* 17:93–103.

Chapter 34
The SSM Procedure

Contents

Overview: SSM Procedure **2357**
Background 2357
Getting Started: SSM Procedure **2358**
Syntax: SSM Procedure **2367**
Functional Summary 2367
PROC SSM Statement 2370
BY Statement 2372
COMPONENT Statement 2372
DEPLAG Statement (Experimental) 2373
EVAL Statement 2375
ID Statement 2376
IRREGULAR Statement 2376
MODEL Statement 2376
OUTPUT Statement 2377
PARMS Statement 2378
Programming Statements 2379
STATE Statement 2379
TREND Statement 2384
Details **2389**
State Space Model and Notation 2389
Types of Sequence Data 2391
Overview of Model Specification Syntax 2392
Building a Complex Model Specification 2392
Model Specification Steps 2393
Sparse Transition Matrix Specification 2394
Regression Variable Specification in Multivariate Models 2395
Filtering, Smoothing, Likelihood, and Structural Break Detection 2396
Filtering Pass 2397
Likelihood Computation and Model Fitting Phase 2398
Forecasting Phase 2400
Smoothing Phase 2400
Delete-One Cross Validation and Structural Breaks 2400
Estimation of User-Specified Linear Combination of State Elements 2402
Contrasting PROC SSM with Other SAS Procedures 2402
Predefined Trend Models 2403
Trend Models for Regular Data 2404

Trend Models for Irregular Data . . . 2405
Predefined Structural Models . . . 2407
Multivariate White Noise . . . 2408
Multivariate Random Walk Trend . . . 2409
Multivariate Local Linear Trend . . . 2409
Multivariate Cycle . . . 2409
Multivariate Season . . . 2410
Multivariate ARMA . . . 2411
Continuous-Time Cycle . . . 2412
Models with Dependent Lags (Experimental) . . . 2413
Covariance Parameterization . . . 2414
Missing Values . . . 2415
Computational Issues . . . 2415
A Well-Behaved Model . . . 2415
Convergence Problems . . . 2416
Computer Resource Requirements . . . 2416
Displayed Output . . . 2417
ODS Table Names . . . 2417
ODS Graph Names . . . 2419
OUT= Data Set . . . 2420
Examples: SSM Procedure . . . **2421**
Example 34.1: Bivariate Basic Structural Model . . . 2421
Example 34.2: Panel Data: Random-Effects and Autoregressive Models . . . 2427
Example 34.3: Backcasting, Forecasting, and Interpolation . . . 2431
Example 34.4: Longitudinal Data: Smoothing of Repeated Measures . . . 2432
Example 34.5: A User-Defined Trend Model . . . 2441
Example 34.6: Model with Multiple ARIMA Components . . . 2444
Example 34.7: Dynamic Factor Modeling . . . 2447
Example 34.8: Diagnostic Plots and Structural Break Analysis . . . 2459
Example 34.9: Longitudinal Data: Variable Bandwidth Smoothing . . . 2465
Example 34.10: A Transfer Function Model for the Gas Furnace Data . . . 2470
Example 34.11: Panel Data: Dynamic Panel Model for the Cigar Data . . . 2475
Example 34.12: Multivariate Modeling: Long-Term Temperature Trends . . . 2480
Example 34.13: Bivariate Model: Sales of Mink and Muskrat Furs . . . 2487
Example 34.14: Factor Model: Now-Casting the US Economy . . . 2492
Example 34.15: Longitudinal Data: Lung Function Analysis . . . 2497
References . . . **2500**

Overview: SSM Procedure

State space models (SSMs) are used for analyzing continuous response variables that are recorded sequentially according to a numeric indexing variable. In many cases, the indexing variable is time and the observations are collected at regular time intervals—for example, hourly, weekly, or monthly. In such cases, the resulting data are called time series data. In other cases, the indexing variable might not be time or the observations might not be equally spaced according to the indexing variable. These more general types of sequential data are called longitudinal data. Because of their sequential nature, these types of data exhibit some characteristic features. For example, chronologically closer measurements tend to be highly correlated while measurements farther apart are essentially uncorrelated. Data can be trending in a particular direction and can have seasonal or other periodic patterns. SSMs are specially designed to model such sequential data. They apply to both univariate and multivariate response situations and can easily incorporate predictor (independent variable) information when it is available.

The SSM procedure performs state space modeling of univariate and multivariate time series and longitudinal data. You can do the following with the SSM procedure:

- analyze quite general linear state space models.
- use an expressive language to specify an SSM. An SSM specification consists of specifying a variety of matrices—for example, the state transition matrix and the covariance matrices of the state and observation disturbances. The SSM procedure provides language similar to a DATA step for specifying the elements of these matrices. The matrix elements can be user-defined functions of data variables and unknown parameters.
- easily specify several commonly needed univariate and multivariate SSMs by using only a few keywords. These SSMs include the principal univariate and multivariate structural models for regularly spaced data and a variety of trend and cycle models for the longitudinal data.
- estimate unknown model parameters by (restricted) maximum likelihood. The likelihood function is computed by using the (diffuse) Kalman filter algorithm.
- print, or output to a data set, the series forecasts, residuals, and the full-sample estimates of any linear combination of the underlying state variables. These estimates are obtained by using the (diffuse) Kalman filter and smoother algorithm.
- generate residual diagnostic plots and plots useful for detecting structural breaks.

Background

State space models are widely used in a variety of fields such as engineering, statistics, econometrics, and agriculture. There are numerous references that deal with state space modeling, particularly with the state space modeling of time series data. State space modeling of longitudinal data has received a little less attention. The primary reference for the modeling techniques implemented in the SSM procedure is Harvey (1989). It contains treatment of both the time series and longitudinal data. Other useful books about this subject are: Durbin and Koopman (2001); Jones (1993); Anderson and Moore (1979). In addition,

informative articles about state space modeling of longitudinal data include Wecker and Ansley (1983); Kohn and Ansley (1991); De Jong and Mazzi (2001); Eubank, Huang, and Wang (2003); Selukar (2015). For the implementation details of the diffuse Kalman filter and smoother (the main computational tool used by the SSM procedure), the main references are a series of articles (De Jong 1989, 1991; De Jong and Chu-Chun-Lin 2003) and the references therein.

Getting Started: SSM Procedure

This example illustrates how you can use the SSM procedure to analyze a panel of time series. The following data set, Cigar, contains information about yearly per capita cigarette sales for 46 geographic regions in the United States over the period 1963–1992. The variables lsales, lprice, lndi, and lpimin denote the per capita cigarette sales, price per pack of cigarettes, per capita disposable income, and minimum price in adjoining regions per pack of cigarettes, respectively (all in the natural log scale). The variable year contains the observation year, and the variable region contains an integer between 1 to 46 that serves as the unique identifier for the region. For additional data description see Baltagi and Levin (1992); Baltagi (1995). The data are sorted by year.

```
data cigar;
    input year region lsales lprice lndi lpimin;
    label lsales = 'Log cigarette sales in packs per capita';
    label lprice = 'Log price per pack of cigarettes';
    label lndi = 'Log per capita disposable income';
    label lpimin = 'Log minimum price in adjoining regions
                    per pack of cigarettes';
    year = intnx( 'year', '1jan63'd, year-63 );
    format year year.;
datalines;
63 1 4.54223 3.35341 7.3514 3.26194
63 2 4.82831 3.17388 7.5729 3.21487
63 3 4.63860 3.29584 7.3000 3.25037
63 4 4.95583 3.23080 7.9288 3.17388
63 5 5.05114 3.28840 7.9772 3.26576

   ... more lines ...
```

The goal of the analysis is to study the impact of the regressors on the smoking behavior and to understand the changes in the smoking patterns in different regions over the years. Consider the following model for lsales:

$$\text{lsales}_{i,t} = \boldsymbol{\mu}_{i,t} + \text{lprice}\,\boldsymbol{\beta}_1 + \text{lndi}\,\boldsymbol{\beta}_2 + \text{lpimin}\,\boldsymbol{\beta}_3 + \boldsymbol{\epsilon}_{i,t}$$

This model represents lsales in a region i and in a year t as a sum of region-specific trend components $\boldsymbol{\mu}_{i,t}$, the regression effects due to lprice, lndi, and lpimin, and the observation noise $\boldsymbol{\epsilon}_{i,t}$. Different variations of this model are obtained by considering different models for the trend component $\boldsymbol{\mu}_{i,t}$. Proper modeling of the trend component is important because it captures differences between the regions because of unrecorded factors such as demographic changes over time, results of anti-smoking campaigns, and so on. The following statements specify and fit one such model:

```
proc ssm data=Cigar plots=residual;
   id year interval=year;
   array RegionArray{46} region1-region46;
   do i=1 to 46;
      RegionArray[i] = (region=i);
   end;
   trend IrwTrend(ll) cross(matchparm)=(RegionArray) levelvar=0;
   irregular wn;
   model lsales = lprice lndi lpimin IrwTrend wn;
   eval TrendPlusReg = IrwTrend + lprice + lndi + lpimin;
   output out=forCigar pdv press;
run;
```

The PROC SSM statement specifies the input data set, Cigar, which contains analysis variables such as the response variable, lsales, and the predictor variables, lprice, lndi, and lpimin. The PLOTS=RESIDUAL option in the PROC SSM statement produces residual diagnostic plots. The optional ID statement specifies a numeric index variable (often a SAS date or datetime variable), which is year in this case. The INTERVAL=YEAR option in the ID statement indicates that the measurements are collected on a yearly basis. The next few statements define a 46-dimensional array of dummy variables, RegionArray, such that RegionArray[i] is 1 if region is i and is 0 otherwise. The next three statements, TREND, IRREGULAR, and MODEL, constitute the model specification part of the program:

- **`trend IrwTrend(ll) cross(matchparm)=(RegionArray) levelvar=0;`** defines a trend, named IrwTrend, of local linear type (which is signified by the keyword *ll* used within the parenthesis after the name). A local linear trend—a trend with time-varying level and time-varying slope—depends on two parameters: the disturbance variance of the level equation and the disturbance variance of the slope equation (see the section "Local Linear Trend" on page 2404 for more information). The LEVELVAR=0 specification fixes the disturbance variance of the level equation to 0, which results in a trend model called an *integrated random walk* (IRW). An IRW model tends to produce a smoother trend than a general local linear trend. In the limiting case, if the disturbance variance of the slope equation is also 0, the IRW trend reduces to a straight line (with a fixed intercept and slope). In addition, because of the use of the 46-dimensional array, RegionArray, in the CROSS= option (**`cross(matchparm)=(RegionArray)`**), this trend specification amounts to fitting a separate IRW trend for each region. This is because, as a result of the CROSS= option, IrwTrend is treated as a linear combination of 46 (the number of variables in RegionArray) stochastically independent, integrated random walks,

  $$\text{IrwTrend}_t = \sum_{i=1}^{46} \text{RegionArray}[i]\, \boldsymbol{\mu}_{i,t}$$

  where each $\boldsymbol{\mu}_{i,t}$ is an integrated random walk. Note that since RegionArray[i] is a binary variable, IrwTrend equals $\boldsymbol{\mu}_{i,t}$ when region is i. Lastly, the use of MATCHPARM option specifies that the different IRW trends $\boldsymbol{\mu}_{i,t}$ use the same disturbance variance parameter for their slope equation. This is done mainly for parsimony. Based on the model diagnostics shown later, this appears to be a reasonable model simplification.

- **`irregular wn;`** defines the observation noise $\boldsymbol{\epsilon}_{i,t}$, named wn, as a sequence of independent, identically distributed, zero-mean, Gaussian variables—a white noise sequence.

- **model lsales = lprice lndi lpimin IrwTrend wn;** defines the model for lsales as a sum of regression effects that involve lprice, lndi, and lpimin, a trend term, lrwTrend, and the observation noise wn.

The last two statements, EVAL and OUTPUT, control certain aspects of the procedure output. The following EVAL statement defines a linear combination, named TrendPlusReg, of selected terms in the MODEL statement.

```
eval TrendPlusReg = IrwTrend + lprice + lndi + lpimin;
```

This EVAL statement causes the SSM procedure to produce an estimate of TrendPlusReg (and its standard error), which can then be printed or output to a data set. TrendPlusReg contains all the terms in the model except for the observation noise and thus can be regarded as the *explanatory* part of the model. In the OUTPUT statement, you can specify an output data set that stores all the component estimates that are produced by the procedure. The following OUTPUT statement specifies forCigar as the output data set:

```
output out=forCigar pdv press;
```

The PDV option causes variables such as region1–region46, which are defined by the DATA step statements within the SSM procedure, also to be included in the output data set. The PRESS option causes the printing of fit measures that are based on the delete-one cross validation errors (see the section "Delete-One Cross Validation and Structural Breaks" on page 2400 for more information).

All the models that are specified in the SSM procedure possess a state space representation. See the section "State Space Model and Notation" on page 2389 for more information. The SSM procedure output begins with a table (not shown here) of the input data set that provides the name and other information. Next, the "Model Summary" table, shown in Figure 34.1, provides basic model information, such as the following:

- the dimension of the underlying state equation, 92 (because each of the 46 IRW trends $\boldsymbol{\mu}_{i,t}$ contributes two elements to the state)
- the diffuse dimension of the model, 95 (which is equal to the three regressors plus the 92 diffuse initial states of $\boldsymbol{\mu}_{i,t}$)
- the number of model parameters, 2 (which is the common disturbance variance of the slope equation in lrwTrend and the variance of the noise term wn)

This information is very useful in determining the computational complexity of the model (the larger state size, 92, explains the relatively long computing time—as much as two minutes on some desktops—for this example).

Figure 34.1 Summary of the Underlying State Space Model

The SSM Procedure

Model Summary	
Model Property	**Value**
Number of Model Equations	1
State Dimension	92
Dimension of the Diffuse Initial Condition	95
Number of Parameters	2

The index variable information is shown in Figure 34.2. Among other things, it categorizes the data to be of the type *Regular with Replication*, which implies that the data are regularly spaced with respect to the ID variable and at least some observations have the same ID value. This is clearly true in this example: the data are yearly without any gaps, and there are 46 observations in each year—one per region.

Figure 34.2 Index Variable Information

ID Variable Information					
Name	**Start**	**End**	**Max Delta**	**NDistinct**	**Type**
year	1963	1992	1	30	Regular with Replication

Figure 34.3 provides simple summary information about the response variable. It shows that lsales has no missing values and no induced missing values because the predictors in the model, lprice, lndi, and lpimin, do not have any missing values either.

Figure 34.3 Response Variable Summary

Response Variable Information							
	Number of Observations						
Name	**Total**	**Missing**	**Induced Missing**	**Minimum**	**Maximum**	**Mean**	**Std Deviation**
lsales	1380	0	0	3.98	5.7	4.79	0.225

The regression coefficients of lprice, lndi, and lpimin are shown in Figure 34.4. As expected, the coefficient of lprice is negative and the coefficients of lndi and lpimin are positive, all being statistically significant. This is consistent with the expectation that the cigarette sales are adversely affected by the price and are positively correlated with the disposable income. The estimated effect of lpimin, called bootlegging effect by Baltagi and Levin (1992), is statistically significant but smaller than the effects of lprice and lndi.

Figure 34.4 Estimated Regression Coefficients

Regression Parameter Estimates					
Response Variable	**Regression Variable**	**Estimate**	**Standard Error**	**t Value**	**Pr > |t|**
lsales	**lprice**	-0.3480	0.0232	-15.01	<.0001
lsales	**lndi**	0.1425	0.0344	4.15	<.0001
lsales	**lpimin**	0.0619	0.0269	2.30	0.0214

Figure 34.5 Estimated Model Parameters

Model Parameter Estimates

Component	Type	Parameter	Estimate	Standard Error	t Value
lrwTrend	LL Trend	Slope Variance	0.000169	0.0000219	7.72
wn	Irregular	Variance	0.000592	0.0000342	17.29

Figure 34.5 shows the estimates of the disturbance variance of the slope equation in lrwTrend and the variance of the noise term wn.

Figure 34.6 shows a panel of residual normality diagnostic plots. These plots show that the residuals are symmetrically distributed but contain slightly larger than expected number of extreme residuals. Figure 34.7 shows the plot of residuals versus time. There the residuals do not exhibit any obvious pattern; however, the plot does show that more extreme residuals appear before 1970 and after 1989. On the whole, however, these plots do not exhibit serious violations of model assumptions.

Figure 34.6 Residual Normality Check

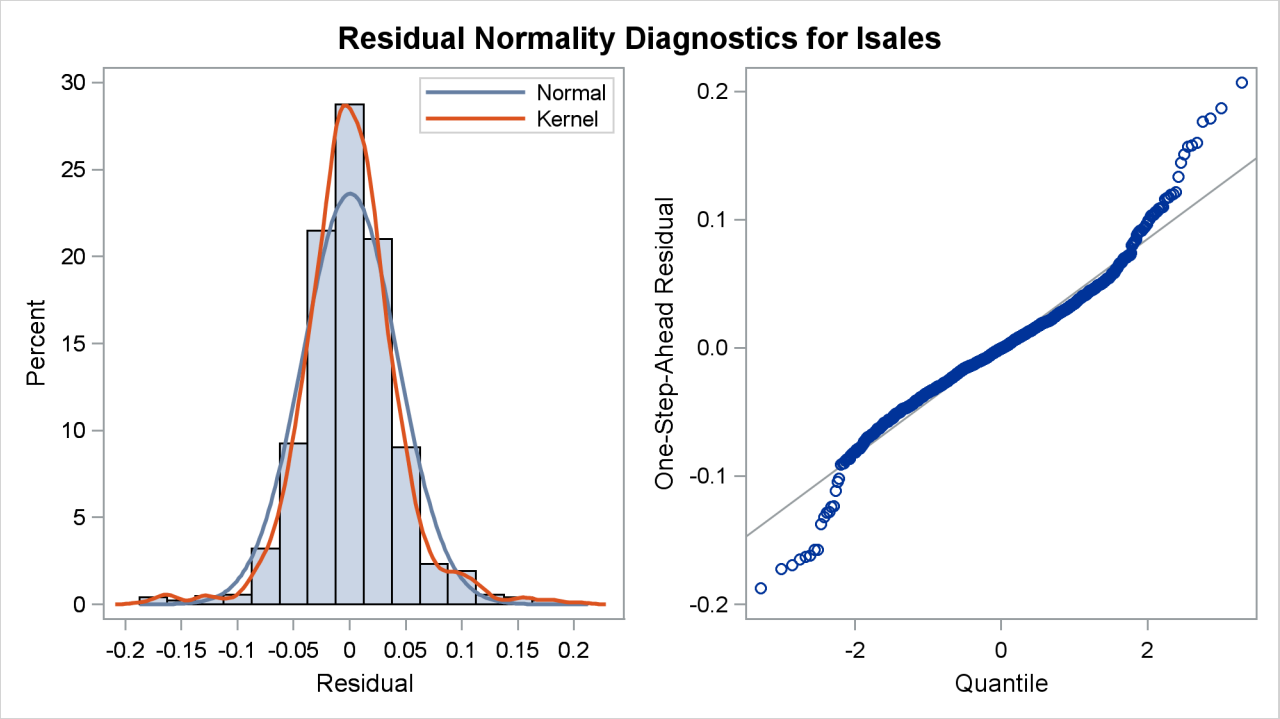


Figure 34.7 Standardized Residuals Plotted against Time

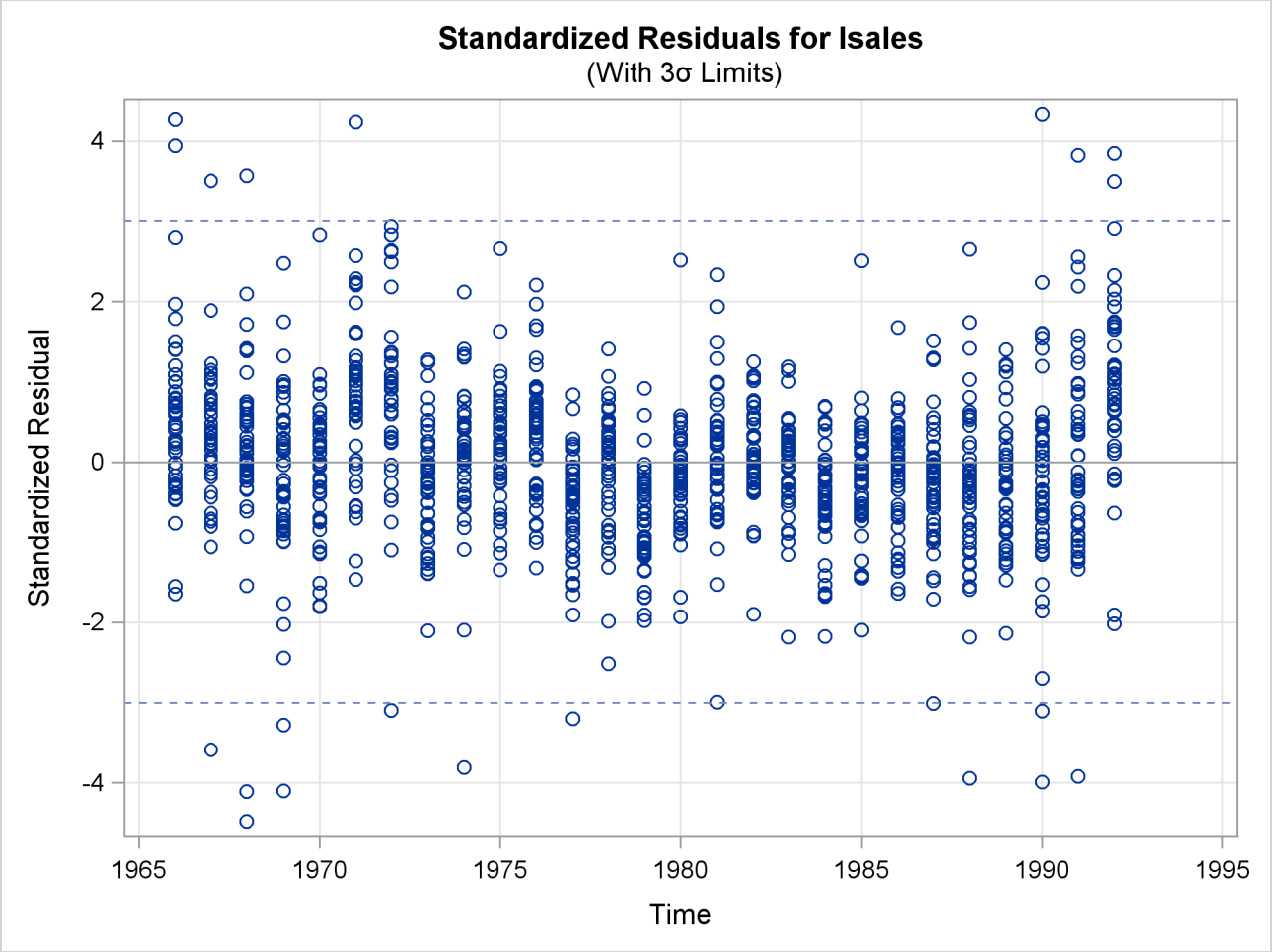


Figure 34.8 shows the details of the likelihood computations such as the number of nonmissing response values used and the likelihood of the fitted model. See the section "Likelihood Computation and Model Fitting Phase" on page 2398 for more information. Figure 34.8 shows the likelihood-based information criteria in lower-is-better format, which are useful for model comparison.

Figure 34.8 Likelihood Computation Details

Likelihood Computation Summary	
Statistic	**Value**
Nonmissing Response Values Used	1380
Estimated Parameters	2
Initialized Diffuse State Elements	95
Normalized Residual Sum of Squares	1285.0002
Diffuse Log Likelihood	2246.0466
Profile Log Likelihood	2169.6232

Figure 34.9 Information Criteria

Information Criteria		
Statistic	Diffuse Likelihood Based	Profile Likelihood Based
AIC (lower is better)	-4488.093	-4145.246
BIC (lower is better)	-4477.776	-3637.952
AICC (lower is better)	-4488.084	-4130.417
HQIC (lower is better)	-4484.220	-3955.472
CAIC (lower is better)	-4475.776	-3540.952

In addition to the regression estimates, it is useful to analyze the estimates of different model components such as the trend component lrwTrend and the linear combination TrendPlusReg. These estimates can be printed by using the PRINT= option provided in the TREND and EVAL statements, or they can be output to a data set (as it is done in this illustration). This latter option is particularly useful for graphical exploration of these components by standard graphical procedures such as SGPLOT and SGPANEL procedures. The following statements produce a panel of plots that shows how well the proposed model fits the observed cigarette sales in the first three regions, which correspond to Alabama, Arizona, and Arkansas. The output data set, forCigar, contains all the needed information: Smoothed_TrendPlusReg contains the smoothed (full-sample) estimate of TrendPlusReg, and Smoothed_Lower_TrendPlusReg and Smoothed_Upper_TrendPlusReg contain its 95% lower and upper confidence limits. In addition, for easy readability, a user-defined format (RegionFormat), which is created by using the FORMAT procedure (not shown), is used to associate the region names to region values.

```
proc sgpanel data=forCigar noautolegend;
   where region <= 3;
   format region RegionFormat.;
   title 'Region-Specific Sales Patterns with 95% Confidence Band';
   panelby region / columns=3;
   band x=year lower=Smoothed_Lower_TrendPlusReg
   upper=Smoothed_Upper_TrendPlusReg;
   scatter x=year y=lsales;
   series x=year y= Smoothed_TrendPlusReg;
run;
```

Figure 34.10 Cigarette Sales Patterns for the First Three Regions

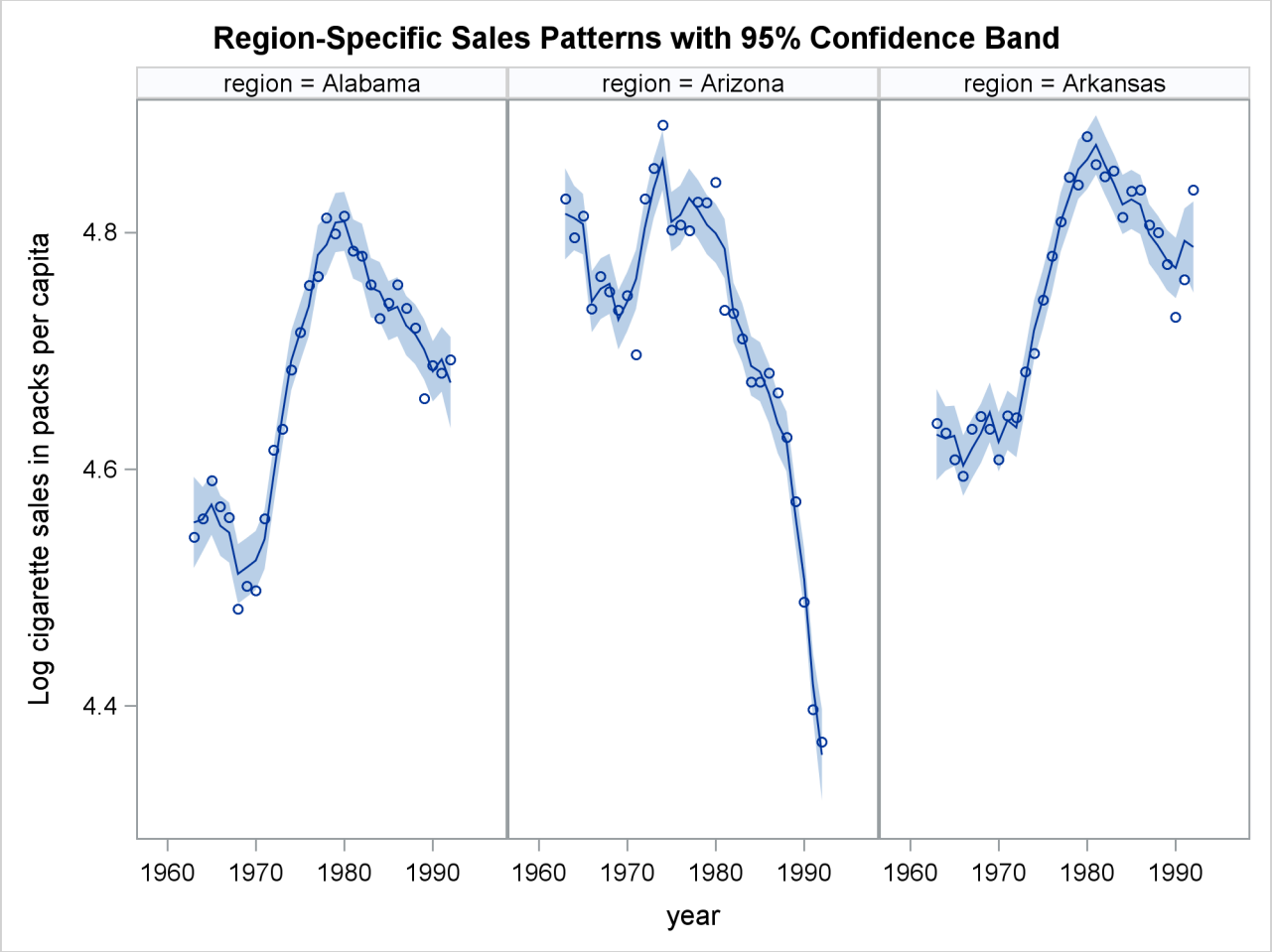
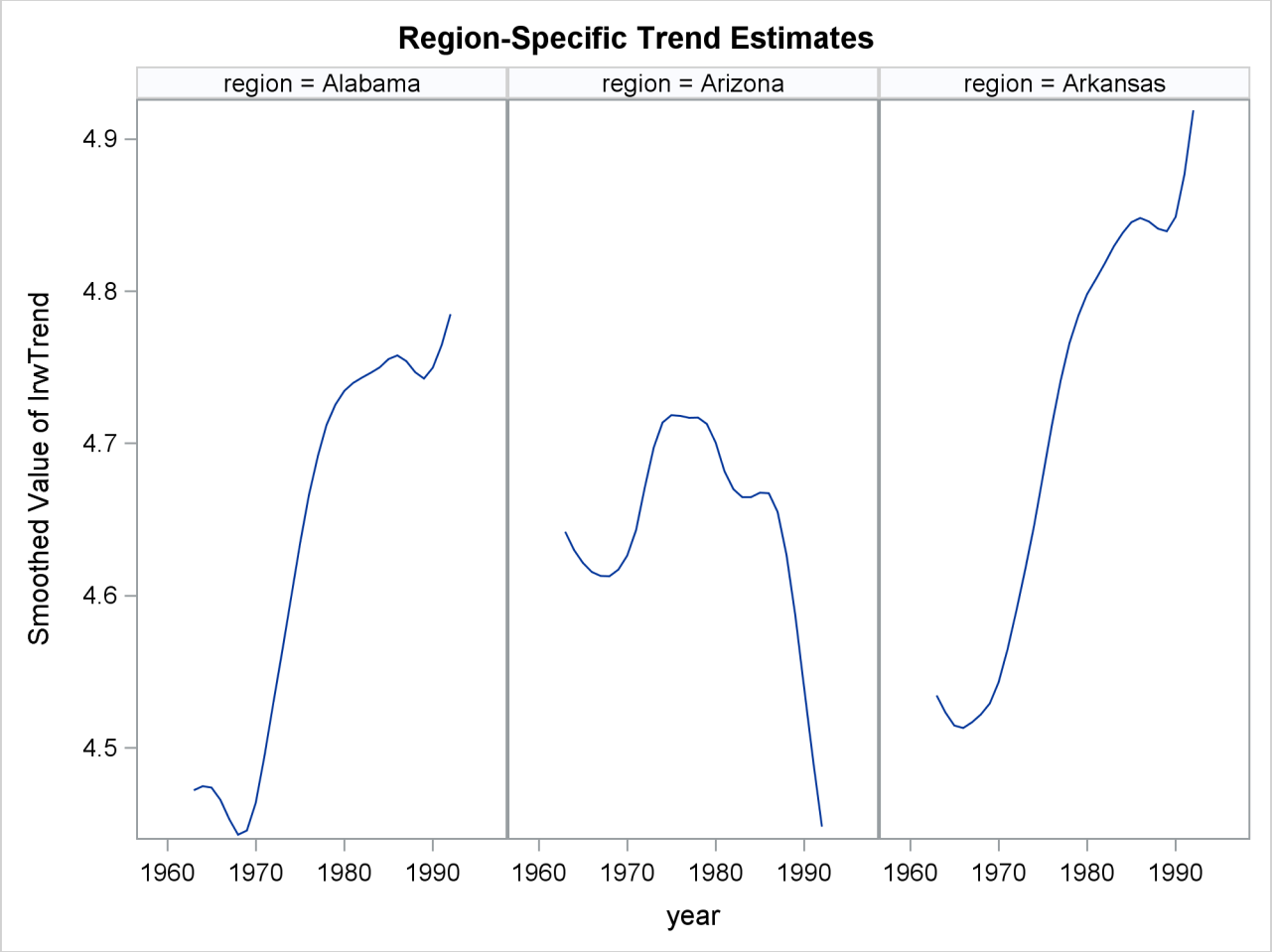


Figure 34.10 seems to indicate that the model fits the data reasonably well. It also shows that Arizona differs markedly from Alabama and Arkansas in its cigarette sales pattern over the years. The following statements produce a similar panel of plots that show the estimate of trend without the regression effects:

```
proc sgpanel data=forCigar noautolegend;
   where region <= 3;
   format region RegionFormat.;
   title 'Region-Specific Trend Estimates';
   panelby region / columns=3;
   series x=year y= smoothed_IrwTrend;
run;
```

Figure 34.11 Estimate of IrwTrend for the First Three Regions



The trend patterns, shown in Figure 34.11, seem to suggest that after accounting for the regression effects, per capita cigarette sales were on the rise in Alabama and Arkansas while they were declining in Arizona.

Syntax: SSM Procedure

The following statements are available in the SSM procedure:

PROC SSM < *options* > **;**
BY *variables* **;**
COMPONENT *name* **=** *(variables) * state* < */ *options* > **;**
DEPLAG *name(response-variable) lag-term1* < *lag-term2 . . .* > **;**
EVAL *name* **=** *expression* < */ *options* > **;**
ID *variable* < *option* > **;**
IRREGULAR *name* < *options* > **;**
MODEL *response* **=** *variables* < */ *options* > **;**
OUTPUT < *options* > **;**
PARMS *variables* < */ *options* > **;**
Programming statements **;**
STATE *name(dim)* < *options* > **;**
TREND *name(type)* < *options* > **;**

You can specify all statements except the BY, ID, and the OUTPUT statements multiple times. The PROC SSM statement and at least one MODEL statement are required. In addition to these statements, you can use most DATA step programming statements to define new variables that are needed for specifying different parts of the state space model.

NOTE: In the statement options described throughout this section, whenever you use a list to specify the elements of the system matrices, the list elements must all be of the same type: either all of them must be variables or all of them must be numbers. In addition, if the list contains more than one variable, then they cannot be of the array type. These are not serious restrictions. When the list contains mix of variables and numbers, you can redefine the numbers as constant variables. Similarly, you can reformulate a list that contains a mix of variables of array and non-array types as just one array by combining all its elements in a new array.

Functional Summary

Table 34.1 summarizes the statements and options that control the SSM procedure. Most commonly needed scenarios are listed; see the individual statements for additional details.

Table 34.1 Functional Summary

Description	Statement	Option
Data Set Options		
Specifies the input data set	PROC SSM	DATA=
Writes series and component forecasts to an output data set	OUTPUT	OUT=
Model Specification Options		
Specifies the index variable	ID	

Table 34.1 *continued*

Description	Statement	Option
Defines variables as model parameters	PARMS	
Specifies a response variable and the associated observation equation	MODEL	
Specifies a state subsection	STATE	
Specifies the transition matrix of a state subsection	STATE	T
Specifies the disturbance covariance matrix of a state subsection	STATE	COV
Specifies the size of the diffuse initial condition of a state subsection	STATE	A1
Specifies the initial covariance matrix of a state subsection	STATE	COV1
Specifies a state subsection for a predefined structural model	STATE	TYPE=
Specifies the regressors in a state equation	STATE	W
Specifies the input vector in a state equation	STATE	SINPUT=
Specifies a component	COMPONENT	
Specifies a predefined trend component	TREND	
Likelihood Optimization Process Control Options		
Specifies the optimization technique	PROC SSM	OPTIMIZER(TECH=)
Limits the number of iterations	PROC SSM	OPTIMIZER(MAXITER=)
Outlier and Structural Break Detection Options		
Turns on the search for additive outliers (AO)		Default
Turns on the search for structural breaks in a state subsection	STATE	CHECKBREAK
Turns on the search for structural breaks in a state subsection associated with a trend	TREND	CHECKBREAK
Specifies the significance level for additive outlier tests	OUTPUT	AO(ALPHA=)
Limits the reported number of additive outliers	OUTPUT	AO(MAXNUM=)
Limits the reported number of additive outliers to a percentage of the series length	OUTPUT	AO(MAXPCT=)
Specifies the significance level for structural break tests	OUTPUT	BREAK(ALPHA=)
Limits the reported number of structural breaks	OUTPUT	BREAK(MAXNUM=)
Limits the reported number of structural breaks to a percentage of the series length	OUTPUT	BREAK(MAXPCT=)
Turns on the search for maximal state shock	OUTPUT	MAXSHOCK
Graphical Residual and Outlier Analysis Options		
Creates a panel of plots that consists of residual normality plots	PROC SSM	PLOTS=RESIDUAL(NORMAL)

Table 34.1 *continued*

Description	Statement	Option
Creates the standardized residual plot against time	PROC SSM	PLOTS=RESIDUAL(STD)
Creates a panel of plots that consists of prediction error normality plots	PROC SSM	PLOTS=AO(NORMAL)
Creates the standardized prediction error plot against time	PROC SSM	PLOTS=AO(STD)
Creates the plot of maximal state shock chi-square statistics against time	PROC SSM	PLOTS=MAXSHOCK
Output Control Options		
Specifies the significance level of the forecast confidence limits	OUTPUT	ALPHA=
Prints the prediction error sum of squares table	OUTPUT	PRESS
Specifies a linear combination of components to be output	EVAL	
Global Printing and Plotting Options		
Turns off all printing for the procedure	PROC SSM	NOPRINT
Turns on all printing options for the procedure	PROC SSM	PRINTALL
Turns off all plotting for the procedure	PROC SSM	PLOTS=NONE
Turns on all plotting options for the procedure	PROC SSM	PLOTS=ALL
Printing State Equation System Matrix Options		
Prints the transition matrix that is associated with a state subsection	STATE	PRINT=T
Prints the disturbance covariance matrix that is associated with a state subsection	STATE	PRINT=COV
Prints the initial covariance matrix that is associated with a state subsection	STATE	PRINT=COV1
Prints the autoregressive coefficient matrix that is associated with a state subsection	STATE	PRINT=AR
Prints the moving average coefficient matrix that is associated with a state subsection	STATE	PRINT=MA
Printing Component, Series Forecast, and Smoothed Estimate Options		
Prints the series forecasts	MODEL	PRINT=FILTER
Prints the full-sample estimates of missing series values	MODEL	PRINT=SMOOTH
Prints the smoothed trend estimate	TREND	PRINT=SMOOTH
Prints the filtered trend estimate	TREND	PRINT=FILTER
Prints the smoothed component estimate	COMPONENT	PRINT=SMOOTH
Prints the filtered component estimate	COMPONENT	PRINT=FILTER
Prints the smoothed component estimate	EVAL	PRINT=SMOOTH
Prints the filtered component estimate	EVAL	PRINT=FILTER

Table 34.1 *continued*

Description	Statement	Option
BY Groups		
Specifies BY-group processing	BY	

PROC SSM Statement

PROC SSM < *options* > **;**

The PROC SSM statement is required. You can specify the following *options* in the PROC SSM statement:

DATA=*SAS-data-set*
specifies the name of the SAS data set that contains the variables needed for the analysis. If you do not specify this option, PROC SSM uses the most recently created SAS data set.

NOPRINT
turns off all the printing and plotting for the procedure. Any subsequent print options are ignored.

PLOTS < (*global-plot-options***) > = ***plot-request*** < (***options***) >**

PLOTS< (*global-plot-options***) > = (***plot-request*** < (***options***) > < ... ***plot-request*** < (***options***) > >)**
controls the plots produced with ODS Graphics. When you specify only one *plot-request*, you can omit the parentheses around it. Here are some examples:

```
plots=none
plots=all
plots=residual
plots=residual(normal)
plots=(maxshock residual(normal))
plots(unpack)=residual
```

If you do not specify any specific *plot-request*, then by default PROC SSM produces the plot of standardized residuals against time. For general information about ODS Graphics, see Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*).

Global Plot Options:
The *global-plot-options* apply to all relevant plots generated by the SSM procedure. The following *global-plot-option* is supported:

UNPACK
displays each graph separately. (By default, some graphs can appear together in a single panel.)

Specific Plot Options:
The following list describes the specific *plot-requests* and their *options*:

ALL
produces all plots appropriate for the particular analysis.

AO< (*prediction-error-plot-options***) >**
produces the prediction error plots—one for each response variable. You can specify the following *prediction-error-plot-options*:

NORMAL
produces a summary panel of the prediction error diagnostics, which consist of the following:

- histogram of prediction errors
- normal quantile plot of prediction errors

STD
produces a scatter plot of standardized prediction errors against time.

MAXSHOCK
produces a scatter plot of maximal state shock statistics against time.

NONE
suppresses all plots.

RESIDUAL < (*residual-plot-options***) >**
produces the residuals plots—one for each response variable. You can specify the following *residual-plot-options*:

NORMAL
produces a summary panel of the residual diagnostics, which consist of the following:

- histogram of residuals
- normal quantile plot of residuals

STD
produces a scatter plot of standardized residuals against time.

See the section "Delete-One Cross Validation and Structural Breaks" on page 2400 for more information about the precise meaning of the terms *maximal state shock statistics* and *prediction errors*.

PRINTALL
turns on all the printing options for the procedure. All subsequent NOPRINT options in the procedure are ignored.

STATEINFO
prints two tables that provide information about the composition of the state vector in terms of the components specified in the model. One table describes the composition of state $\boldsymbol{\alpha}_t$, and the other table describes the diffuse vector $\boldsymbol{\delta}$ and the regressors, which are part of the initial condition specification $\boldsymbol{\alpha}_1$. See the section "State Space Model and Notation" on page 2389 for more information about the state space model notation.

OPTIMIZER(< TECHNIQUE=*technique* > < MAXITER=*integer* >)

specifies options that are associated with the optimizer used in the maximum likelihood parameter estimation. The default settings of the optimization process are adequate in most problems. However, in some cases it might be useful to change the optimization technique or to change the maximum number of iterations. You can specify one of the following *techniques*:

ACTIVESET corresponds to the active-set method.

DBLDOG corresponds to the double-dogleg method.

INTERIORPOINT corresponds to the primal-dual interior point method.

NEWRAP corresponds to the Newton-Raphson method.

QUANEW corresponds to the (dual) quasi-Newton method.

TRUREG corresponds to the trust region method.

The default technique is TRUREG. The INTERIORPOINT and ACTIVESET techniques are documented in Chapter 10, "The Nonlinear Programming Solver" (*SAS/OR User's Guide: Mathematical Programming*), and the remaining techniques are documented in Chapter 7, "Nonlinear Optimization Methods." You can alter the maximum number of iterations setting in the nonlinear optimization search by specifying a nonnegative *integer* as the MAXITER= value.

BY Statement

BY *variables* **;**

A BY statement can be used in the SSM procedure to process a data set in groups of observations that are defined by the BY variables. The model specified by using the MODEL and other statements is applied to all the groups defined by the BY variables. When a BY statement appears, the procedure expects the input data set to be sorted in order of the BY variables. The BY variables are one or more variables in the input data set. The BY variables cannot be used in the model specification; in particular, they cannot be used as response variables or regressors in a MODEL statement.

COMPONENT Statement

COMPONENT *name* **=** *(var1 var2 ... | number1 number2 ...) * state* < */ options* > **;**

COMPONENT *name* **=** *state[integer]* < */ options* > **;**

COMPONENT *name* **=** *(Variable | Number) * state[integer]* < */ options* > **;**

The COMPONENT statement specifies a component (a linear combination of state elements), named *name*. You can use *name* later as a term in the right-hand side of the MODEL statement, which defines the observation equation. The estimate of *name* is output to the OUT= data set that is specified in the OUTPUT statement. In addition, you can print the component estimate by using the PRINT= option.

The first form of the COMPONENT statement defines a component as a dot product of a state subsection *state* and a row vector *(var1 var2 ...)*. The value of *state* can be the name of a state subsection that is defined by using a STATE statement elsewhere in the program, or it can be the name of the state that is associated with

a trend component defined by using a TREND statement elsewhere in the program (see the section "TREND Statement" on page 2384 for more information about the naming of the state that is associated with a trend component). The row vector *(var1 var2 ...)*, which can be either a list of numbers or a list of variables, must be of the same dimension as the actual dimension of the state subsection. The dot product form—also called the explicit dot product form—of the component specification is unambiguous; however, it requires detailed knowledge of the state vector underlying the *state* specification. Suppose that mystate is a two-dimensional state defined by a STATE statement elsewhere in the program and that X1 and X2 are (numeric) predictor variables. The following are valid examples of the dot product form of the COMPONENT statement:

```
component c1 = (x1 x2) * mystate;
component c2 = (1 1) * mystate;
```

The second and the third forms of the COMPONENT statement are a shortened version of the first form. The second form defines the component as a particular element of state—for example, state[3] defines the component as the third element of *state*. The specified *integer* must lie between 1 and *dim*, the nominal dimension of state. The second form of component specification has another important use when the STATE statement that defines *state* uses the TYPE= option to set its type or when *state* is associated with a trend component. In these cases, the second form of the component specification assumes additional meaning when the nominal state dimension and the actual state dimensions differ (specifically the state types LL, SEASON, CYCLE, and VARMA and the states associated with all the trend types). For example, if *state* is a three-dimensional seasonal component, state[2] signifies an appropriate linear combination of *state* that results in the second of the three seasonals that constitute the three-dimensional seasonal. Similar interpretation holds for the CYCLE type. See the sections "Multivariate Season" on page 2410 and "Predefined Structural Models" on page 2407 for more information. The third form extends the second form by permitting multiplication by a variable or a number.

NOTE: A component that is based on a state associated with a trend component cannot be used as a right-hand side term in any MODEL statement. That is, it is defined purely for output purposes (either printed or output to a data set). However, it can be used as a term in the expression that is specified in an EVAL statement to build more complex linear combinations for output.

You can specify the following *options* to print the filtered or smoothed estimate of the component:

PRINT=FILTER | SMOOTH

PRINT=(< FILTER > < SMOOTH >)

requests printing of the filtered or smoothed estimate of the specified component.

DEPLAG Statement (Experimental)

DEPLAG *name(response-variable) lag-term1 < lag-term2 . . . >* **;**

The DEPLAG statement defines a term, named *name*, that consists of a linear combination of lagged response variables. You can use *name* later as a right-hand-side term in the MODEL statement for the response variable, as specified in *name(response-variable)*. For a multivariate model, a separate DEPLAG statement is needed for each MODEL statement that has a right-hand-side term that involves lagged response variables. The linear combination of lagged response variables is specified by using one or more *lag-terms*. Each *lag-term* specifies the lags that are associated with one of the response variables.

A *lag-term* is specified in one of the following forms:

lag-response-variable(**LAGS=***maximum-lag*)
lag-response-variable(**LAGS=(***integer1 integer2 . . .*)**)**
lag-response-variable(**LAGS=***maximum-lag* **COEFF=(***number1 number2 . . .*) | (*variable1 variable2 . . .*)**)**

lag-response-variable(**LAGS=(***integer1 integer2 . . .*) **COEFF=(***number1 number2 . . .*) | (*variable1 variable2 . . .*)**)**

> The *lag-response-variable* in the lag term specification can be the same as the response variable that corresponds to the model equation (which is specified in *name(response-variable)*), or it can be a different response variable.

The first form of specification is useful when all lags up to the *maximum-lag*, which must be a positive integer, are present in the lag term. The second form is useful when only certain lags, which are specified as a list of positive integers in parentheses, are present. In these two cases, the lag coefficients are not specified and they are treated as unknown parameters to be estimated from the data.

The COEFF= option in the last two forms enable you to specify lag coefficients. The COEFF= option must follow the LAGS= option. You can use the COEFF=(*number1 number2 . . .*) option to specify the lag coefficients as known values. Similarly, you can use the COEFF=(*variable1 variable2 . . .*) option to specify user-defined variables as lag coefficients; the user-defined variables can be functions of parameters (which are defined by using the PARMS statement) and input variables. However, the lag coefficients cannot depend on any of the response variables. The number of coefficients specified in the COEFF= option must exactly equal the number of lags specified in the LAGS= option.

There can be at most one DEPLAG statement associated with a particular MODEL statement (you can specify all the needed lag terms in a single DEPLAG statement).

As an illustration, let lagsFORy1 and lagsFORy2 represent the following linear combinations of lagged response variables Y1, Y2, and Y3:

$$\text{lagsFORy1} = \theta_{11} Y1_{t-1} + \theta_{12} Y1_{t-2} + \theta_{22} Y2_{t-2} + \theta_{23} Y2_{t-3} + 1.2 Y3_{t-1} - 2.1 Y3_{t-2}$$

$$\text{lagsFORy2} = \text{Phi1} Y1_{t-1} + \text{Phi2} Y1_{t-2} + \theta_{21} Y2_{t-1}$$

where Phi1 and Phi2 denote user-defined variables and θ_{ij} denote generic parameters. You can specify lagsFORy1 (which is used in the model equation for Y1) and lagsFORy2 (which is used in the model equation for Y2) as follows:

```
deplag lagsFORy1(y1) y1(lags=2) y2(lags=(2 3)) y3(lags=2 coeff=(1.2 -2.1));
deplag lagsFORy2(y2) y1(lags=2 coeff=(phi1 phi2)) y2(lags=1);
... more statements ....;
model y1 = lagsFORy1 ...;
model y2 = lagsFORy2 ...;
model y3 = ...;
... more statements ....;
```

assuming that the right-hand-side of the MODEL equation for Y3 does not have a term that involves lags of response variables.

The DEPLAG statement in PROC SSM has the same purpose as the DEPLAG statement in PROC UCM (see Chapter 41, "The UCM Procedure"). However, there are many differences in the syntax of the two statements, mainly because PROC SSM supports much more complex models. The syntax difference between the two DEPLAG statements can be illustrated by considering the differencing specification— $(1 - B)(1 - B^{12}) = (1 - B - B^{12} + B^{13})$— in the well-known airline model (ARIMA(0, 1, 1)(0, 1, 1)12

model). You can specify the lag-term that is implied by the differencing in the airline model in PROC UCM as follows:

```
deplag lags=(1)(12) phi=1 1 noest;
```

In PROC SSM the same specification has the following form:

```
deplag airLags(y) y(lags=(1 12 13) coeff=(1 1 -1));
```

Both these specifications define the same lag-term: $(y_{t-1} + y_{t-12} - y_{t-13})$.

For an example of the use of lagged response variables in a model specification, see Example 34.13. For more information about models that have dependent lags, see the section "Models with Dependent Lags (Experimental)" on page 2413.

NOTE: Models that have lagged response variables are permitted only if the data form a time series (either univariate or multivariate). The SSM procedure adds one more restriction on the models that use lagged response variables: the variables in the list that define a component in any of the COMPONENT statements must be free of unknown parameters. This restriction is artificial and is made primarily to reduce the overall complexity of the model. In future versions of the SSM procedure, this restriction might go away.

EVAL Statement

EVAL *name* **=** *number1*variable1 + number2*variable2 + . . .* **<** */ options* **>** **;**

The EVAL statement defines a linear combination, named *name*, of the terms used in the right-hand side of a MODEL statement. You can specify any variables (for example, predictor variables and names of components) in the expression of the EVAL statement; however, you cannot specify in this expression any observation disturbances that are specified by the IRREGULAR statement and any model terms that are specified by the DEPLAG statement. Suppose C1 and C2 are two components (defined by COMPONENT statements elsewhere in the program), T1 is a trend component, and X1 is a regression variable used in a model. The following are valid examples of the EVAL statement:

```
eval e1 = c1 - c2;
eval e2 = t1 + c1 + x1;
eval e2 = t1 + 2*c1 - 1.5*x1;
```

The estimates of linear combinations defined by the EVAL statement (for example, E1, E2, and E3) are output to the OUT= data set that is specified in the OUTPUT statement.

The components used in a given EVAL expression must correspond to distinct state subsections. This requirement is imposed only to simplify the overall readability of the program and does not limit the type of linear combinations that can be specified; if two components in the right hand side of an EVAL expression share the same state subsection, a new component that combines the effect of these two components can always be defined.

In addition, you can print these estimates by using the following PRINT= options:

PRINT=FILTER | SMOOTH

PRINT=(< FILTER > < SMOOTH >)

requests printing of the filtered or smoothed estimate of the specified linear combination.

NOTE: The expression builder in the EVAL statement is primitive. For example, you cannot use parentheses to group terms.

ID Statement

ID *variable* < *option* > **;**

The ID statement names a numeric variable to associate a sequence value—usually related to a time stamp—to the observations in the input data set. The observations within a BY group must be ordered in ascending order by the ID variable. Often the ID variable's values are SAS date, time, or datetime values, and each observation within a BY group has a unique ID value. Generally, however, the ID variable can be any numeric variable, and there can be multiple observations with the same ID value. If the ID values are SAS date, time, or datetime values, you can specify the associated unit of time—for example, day, week or month—by using the INTERVAL= option. If an ID statement is not specified, the observation number, with respect to the BY group, is used as the time ID. Whenever an ID variable is specified, a variable, _ID_DELTA_, is automatically created that can be used as any input data set variable in the programming statements. _ID_DELTA_ contains the distance between two successive ID values. The first _ID_DELTA_ value is arbitrarily taken as one. If the INTERVAL= option is specified, the distance between the ID values is measured in terms of the number of intervals; therefore, for regularly spaced data, _ID_DELTA_ is identically equal to one. You can specify the following *option* in the ID statement:

INTERVAL=*value*

specifies the unit of time interval that is used for measuring the ID values. INTERVAL=*value* is used in conjunction with the ID variable to check that the input data are in the proper order. For a complete discussion of the supported intervals, see Chapter 5, "Date Intervals, Formats, and Functions."

IRREGULAR Statement

IRREGULAR *name* < *options* > **;**

The IRREGULAR statement specifies a one-dimensional white noise component, which can be used to specify the observation error in a MODEL statement. You can specify the following *options* in the IRREGULAR statement:

PRINT=SMOOTH

requests printing of the smoothed estimate of the specified irregular component.

VARIANCE=*variable* | *number*

specifies the variance of the white noise. Any nonnegative value, including 0, is permissible. If the *variable* contains unknown parameters, they are estimated from the data. Similarly, if the VARIANCE= option is not specified, the variance is estimated from the data.

MODEL Statement

MODEL *response* **=** *variables* < / *options* > **;**

A MODEL statement specifies an observation equation that describes a response variable as a sum of regression effects and components that are defined in the program. The response variable must be a numeric variable from the input data set. The variables used in the right-hand side of the model expression can be

numeric variables from the input data set, numeric variables defined by using programming statements, or names of components that are specified in the COMPONENT, DEPLAG, TREND, or IRREGULAR statements.

For a multivariate model, a separate MODEL statement is needed for each of the response variables. In this case, the observation errors, which are specified in an IRREGULAR statement, must be different in each MODEL statement.

The components that are specified in a given MODEL statement must correspond to distinct state subsections. This requirement is imposed only to simplify the overall readability of the program and does not limit the type of models that can be specified; if two components on the right-hand side of a MODEL statement share the same state subsection, a new component that combines the effect of these two components can always be defined.

You can specify the following *options* to print the filtered or smoothed estimate of the response variable:

PRINT=FILTER | SMOOTH

PRINT=(< FILTER > < SMOOTH >)

requests printing of the filtered or smoothed estimate of the specified response variable. The filtered estimate is produced during the filtering phase, and the smoothed estimate is produced by the smoothing phase of the Kalman filter and smoother algorithm. The filtered estimate is also called the one-step-ahead forecast of the response variable. The smoothed estimate corresponds to the full-sample prediction of the response variable. Since the full-sample prediction of a nonmissing response value is that value itself, full-sample predictions are printed only for the missing response values.

OUTPUT Statement

OUTPUT < *options* > **;**

The OUTPUT statement creates an optional output data set and also provides options to control certain aspects of the procedure output. If the OUT= option is specified, then an output data set is created to store estimates of the model components and series forecasts. If the OUT= option is omitted, then no data set is created by the OUTPUT statement. Other options in the OUTPUT statement produce additional information in the printed output generated by the procedure. For example, the AO and BREAK options control the search for additive outliers and structural breaks in the data, respectively.

AO(< ALPHA=*number* **> < MAXNUM=***number* **> < MAXPCT=***number* **>)**

controls the additive outlier search (see the section "Delete-One Cross Validation and the Additive Outlier Detection" on page 2401 for more information). The ALPHA= suboption specifies the significance level for reporting the outliers. The default is ALPHA=0.05. The MAXNUM= suboption limits the number of outliers to search. The default is MAXNUM=5. The MAXPCT= suboption is similar to the MAXNUM= suboption. In the MAXPCT= option you can limit the number of outliers to search for according to a percentage of the series length. The default is MAXPCT=1. When you specify both of these options, the lesser of the two search numbers is used.

ALPHA=*number*

specifies the significance level of the forecast confidence intervals. For example, ALPHA=0.05, which is the default, results in a 95% confidence interval.

BREAK(< ALPHA=*number* **> < MAXNUM=***number* **> < MAXPCT=***number* **>)**
controls the structural break search (for more information, see the section "Structural Breaks in the State Evolution" on page 2401). In order for this option to have any effect, the CHECKBREAK option in one of the STATE or TREND statements, or the MAXSHOCK option in the OUTPUT statement, must be turned on. The ALPHA= suboption specifies the significance level for reporting the breaks. The default is ALPHA=0.05. The MAXNUM= suboption limits the number of breaks to search. The default is MAXNUM=5. The MAXPCT= suboption is similar to the MAXNUM= suboption. In the MAXPCT= option, you can limit the number of breaks to search for according to a percentage of the number of distinct time points in the data. The default is MAXPCT=1. When you specify both of these options, the lesser of the two search numbers is used.

MAXSHOCK
causes the computation of the maximal state shock chi-square statistic at each distinct time point in the input data set. These statistics are output to the data set that is specified in the OUT= option. A time series plot of these statistics is produced if the PLOTS=MAXSHOCK option is specified in the PROC SSM statement. These statistics are useful for detecting structural breaks in the state evolution process. This option can be computationally expensive for a model with large state size. See the section "Structural Breaks in the State Evolution" on page 2401 for more information.

OUT=*SAS-data-set*
specifies an output data set for the forecasts. The output data set contains the ID variable (if specified), the response variables, the one-step-ahead and out-of-sample response variable forecasts, the forecast confidence intervals, the smoothed values of the response series, and the one-step-ahead and smoothed estimates of the model components—including expressions that are defined by using the EVAL statement. See the section "OUT= Data Set" on page 2420 for more information.

PDV
causes the inclusion of the variables (variables in the program data vector) that are defined by using the programming statements in the SSM procedure in the OUT= data set. The parameters defined by the PARMS statement are also included. The output data set contains the values of these variables evaluated for all the rows in the input data set that is specified in the DATA= option. The parameters in the PARMS statement contain their estimated values.

PRESS
prints the prediction error sum of squares (PRESS) and the generalized cross validation error sum of squares (GCV). The PRESS table also reports the number of summands that are used in these sums of squares. See the section "Delete-One Cross Validation and the Additive Outlier Detection" on page 2401 for more information.

PARMS Statement

PARMS *variable*< =*number*> *variable*< =*number*> < / *options*> **;**

The PARMS statement declares the parameters of a model and optionally sets their initial values. You can also specify the lower and upper limits of their validity range. The parameters declared by using the PARMS statement are called *named parameters* throughout this chapter. A model can have additional parameters: any unspecified quantity in the model specification becomes part of the parameter vector. You can specify the following *options*:

LOWER=(*number1 number2* **...)**

LOWER=(*number***)**

specifies the lower bounds for the specified parameters. The list can contain exactly one number, which is taken to be the lower bound for all the listed parameters in the statement, or it must contain as many values as the number of parameters specified. A missing value, denoted by ., is a permissible value, which signifies that the parameter has no lower bound.

UPPER=(*number1 number2* **...)**

UPPER=(*number***)**

specifies the upper bounds for the specified parameters. The list can contain exactly one number, which is taken to be the upper bound for all the listed parameters in the statement, or it must contain as many values as the number of parameters specified. A missing value, denoted by ., is a permissible value, which signifies that the parameter has no upper bound.

Programming Statements

To define the model, you can use most of the programming statements that are allowed in the SAS DATA step. See the *SAS Language Reference: Dictionary* for more information. For the most part, the syntax of programming statements used in PROC SSM is identical to that used in the MODEL procedure (see Chapter 26, "The MODEL Procedure") and the NLMIXED procedure (see Chapter 82, "The NLMIXED Procedure" (*SAS/STAT User's Guide*)). However, there are some restrictions: the DATA step lagging and differencing functions are not allowed, and the use of character variables in the DATA step expressions is not permitted. These are not serious restrictions; usually you can overcome them by adding the variables that are created by such operations to the input data set before its use in the SSM procedure.

STATE Statement

STATE *name (dim)* < *options* > **;**

The STATE statement specifies a subsection of $\boldsymbol{\alpha}_t$, the overall state vector at time t (for more information, see the section "State Space Model and Notation" on page 2389). Consider the state equations that define the state space model:

$$\begin{aligned} \boldsymbol{\alpha}_{t+1} &= \mathbf{T}_t\boldsymbol{\alpha}_t + \mathbf{W}_{t+1}\boldsymbol{\gamma} + \mathbf{c}_{t+1} + \boldsymbol{\eta}_{t+1} \\ \boldsymbol{\alpha}_1 &= \mathbf{c}_1 + \mathbf{A}_1\boldsymbol{\delta} + \mathbf{W}_1\boldsymbol{\gamma} + \boldsymbol{\eta}_1 \end{aligned}$$

You can specify multiple STATE statements, each specifying a separate subsection. It is assumed that the subsections that are specified by using different STATE statements are mutually independent. This independence assumption implies a block-diagonal structure for the transition matrices $\mathbf{T}_t$ and the disturbance covariances $\mathbf{Q}_t$ for all $t \geq 1$. An appropriate block structure also applies to $\mathbf{W}_t$ and $\mathbf{A}_1$. The *options* in the STATE statement provide complete control over the description of the relevant blocks of $\mathbf{T}_t$, $\mathbf{Q}_t$, $\mathbf{W}_t$, and $\mathbf{A}_1$. The argument *dim* (a positive integer in *name* (*dim*)) specifies the nominal dimension of this subsection. In most situations, the nominal dimension and the actual dimension of the state subsection are the same. However, when you specify the TYPE= option, the actual dimension of the state subsection can be different

from the nominal dimension. The TYPE= option simplifies the state specification task for some commonly needed models.

NOTE: The T, COV, W, and COV1 options, described later in this section, specify the relevant blocks of $\mathbf{T}_t$, $\mathbf{Q}_t$, $\mathbf{W}_t$, and $\mathbf{Q}_1$, respectively. The structure of these matrix blocks is described in a similar way in the option descriptions. For example, the specification COV(I) corresponds to the identity form, COV(D) corresponds to the diagonal form, and COV(G) corresponds to the general form of the $\mathbf{Q}_t$ block.

You can use the following *options* in the STATE statement to specify the system matrices $\mathbf{T}_t$, $\mathbf{Q}_t$, $\mathbf{W}_t$, and $\mathbf{A}_1$ and to request printing of their estimates when they contain unknown parameters. You can also request the checking of unexpected changes—structural breaks—in the evolution of this state subsection by using the CHECKBREAK option.

A1(*nd*)
specifies that the last *nd* elements of the state subsection be treated as diffuse. This becomes the dimension of the relevant subsection of the diffuse vector $\boldsymbol{\delta}$. The $\mathbf{A}_1$ block is created by using appropriate columns of the identity matrix. The value of *nd* must lie between 1 and the nominal dimension, *dim*. The absence of this option signifies that this subsection of $\boldsymbol{\alpha}_t$ is nondiffuse. If both the COV1 and A1 options are specified, the last *nd* rows and columns of the matrix specified in the COV1 option are taken to be 0. This option cannot be used together with the RANK= option of the COV1 option.

CHECKBREAK< (ELEMENTWISE | OVERALL) >
turns on the checking of breaks for this state subsection. The ELEMENTWISE suboption requests the elementwise checking of any unexpected change in the state subsection as it evolves from one time point to the next. The OVERALL suboption requests a similar check for the entire state subsection—that is, in this case the change is measured as a multidimensional change. The ELEMENTWISE suboption is the default. Unless the PRINT=BREAKDETAIL option is specified, only a summary of the most significant breaks is produced. If the PRINT=BREAKDETAIL is specified, tables that contain the break significance statistics at every distinct time point are produced—one for the ELEMENTWISE suboption and one for the OVERALL suboption. For more information about the structural break detection process, see the section "Structural Breaks in the State Evolution" on page 2401. For an example of the use of the CHECKBREAK option, see Example 34.8.

COV(D) < = (*var1 var2 ...*) | (*number1 number2 ...*) >
COV(G) < = (*var1 var2 ...*) | (*number1 number2 ...*) >
COV(I) < = (*variable*) | (*number*) >
COV(RANK=*integer*)
specifies the relevant block of the disturbance covariance Q_t (for $t \geq 2$) in the transition equation. As with the T option, the absence of this option signifies that this Q-block consists of only zeros. The structure of the Q-block is also similarly specified. However, the following differences exist:

- The list that is specified to form the covariance must result in a symmetric, positive semidefinite matrix. For an example, see Example 34.5.
- You can specify a rank constraint on the Q-block by specifying COV(RANK=*integer*), where the specified integer must lie between 1 and *dim*. A rank constraint is permissible only for the general form and only when its elements are not specified by using a list.
- The convention of treating unset variables as structural zeros, which is used in specifying sparsity of the T-block, is not used in the Q-block specification. Whenever you explicitly specify the

entries of the Q-block by specifying a list of variables in parentheses, all variables in the list must evaluate to nonmissing values.

The following examples illustrate different ways of specifying a Q-block. It is assumed that *dim* = 2.

- COV(G) specifies a general-form Q-block, which contributes $(2 * (2 + 1))/2 = 3$ unspecified elements to the parameter vector $\boldsymbol{\theta}$.
- COV(RANK=1) specifies a rank-one Q-block.

COV1(D) < = (*var1 var2 ...***) | (***number1 number2 ...***) >**

COV1(G) < = (*var1 var2 ...***) | (***number1 number2 ...***) >**

COV1(I) < = (*variable***) | (***number***) >**

COV1(RANK=*integer* **)**

specifies the relevant block of the initial state covariance Q_1. The different options in this case have the same meaning as the options of the COV option. However, the following differences exist:

- If the elements of Q_1 are specified by a list of variables in parentheses, then these variables must evaluate to constant values. In particular, they can depend on parameters that are specified by the PARMS statements; however, they cannot depend on any of the input data columns.
- If the initial condition is partially diffuse (that is, the diffuse dimension *nd* specified in the A1 option is nonzero), the last *nd* rows and columns of the matrix specified in COV1 are taken to be zero. Moreover, if the elements of Q_1 are specified by a list, its number of elements must correspond to a matrix of dimension (*dim* – *nd*).

PRINT=AR | BREAKDETAIL | COV | COV1 | MA | T

PRINT=(< AR > < BREAKDETAIL > < COV > < COV1 > < MA > < T >)

requests printing of the respective system matrices and the printing of the break statistics at each distinct time point. You can specify PRINT=AR or PRINT=MA only if you specify the TYPE=VARMA option. If any of these matrices are time-varying, the matrix that corresponds to the first time instance is printed. For the BREAKDETAIL suboption to have any effect, the CHECKBREAK option must be turned on. If TYPE= option is used, the result of PRINT=COV can be different than the matrix supplied in the COV= option.

SINPUT = (*var1 var2 ...***) | (***number1 number2 ...***)**

specifies the relevant *dim*-dimensional block of the state input vector $\mathbf{c}_t$. The absence of this option signifies that this block of the $\mathbf{c}_t$ vector consists of only zeros. If the elements of $\mathbf{c}_t$ are specified by a list of variables in parentheses, then these variables must be independent of unknown parameters. In particular, they cannot be functions of parameters that are defined by the PARMS statements.

T(D) < = (*var1 var2 ...***) | (***number1 number2 ...***) >**

T(G) < = (*var1 var2 ...***) | (***number1 number2 ...***) >**

T(I) < = (*variable***) | (***number***) >**

specifies the relevant block of the transition matrix T_t. The absence of this option signifies that this block consists of only zeros. You can specify the structure of the T-block by specifying T(I) for the identity form, T(D) for the diagonal form, and T(G) for a general unstructured form. In addition, you can explicitly specify the entries of the T-block by specifying a list of numbers in parentheses, or by specifying in parentheses a list of variables that are defined by using the programming statements. The

unspecified elements of the T-block are included in the list of parameters to be estimated from the data. If the elements of the T-block are supplied by a list in parentheses, the number of elements in the list depends on its structure. For the diagonal form, the list must contain exactly *dim* elements. In the case of the identity form—T(I)—the block is already fully specified; however, a specification T(I)=(*variable*) is understood to mean that the identity block is scaled by the specified *variable* (or a *number*). In the general case—T(G)—the list must consist of *dim* $*$ *dim* elements, specified in a rowwise fashion. An inappropriate number of elements in the list results in a syntax error.

The following examples illustrate different ways of specifying the transition matrix. It is assumed that *dim* = 2.

- T(I) specifies that the T-block is a two-dimensional identity matrix.
- T(D) specifies that the T-block is a two-dimensional diagonal matrix. The two unspecified diagonal entries become part of the parameter vector $\boldsymbol{\theta}$.
- T(D)=(1.1 2) fully specifies the two-dimensional diagonal T-block.
- T(D)=($X_1 X_2$) specifies a two-dimensional diagonal T-block where the diagonal elements are dynamically calculated based on the values of the variables X_1 and X_2. In this case the T-block can change with time if X_1 or X_2 changes with time.
- T(G) specifies a general form T-block (with $2^2 = 4$ unspecified elements).
- T(G)=($X_1 X_2 X_3 X_4$) specifies a general form T-block where the first row is formed by X_1 and X_2, and the second row is formed by X_3 and X_4.

In practice the transition matrix is often sparse—that is, many of its elements are 0. The algorithms in the SSM procedure exploit this sparsity structure for computational efficiency. Whenever you explicitly specify the entries of the T-block by specifying a list of variables in parentheses, you can leave the variables that correspond to the zero elements *unset*. These unset variables are treated as structural zeros by the SSM procedure. The section "Sparse Transition Matrix Specification" on page 2394 further explains how to use this sparsity convention.

TYPE=WN

TYPE=RW

TYPE=LL < (SLOPECOV(I | D | G) < = (*var1, var2, ...***) | (***number1, number2, ...***) >) >**

TYPE=LL (SLOPECOV(RANK=*integer***))**

TYPE=SEASON (LENGTH=*integer* **< DROPH=***number-list* **> < KEEPH=***number-list* **>)**

TYPE=CYCLE < (< CT > < RHO=*variable* **|** *number* **> < PERIOD=***variable* **|** *number* **>) >**

TYPE=VARMA (< p < (I | D) > =*integer* **> < q< (D) >=***integer* **>)**

specifies a state subsection that corresponds to the specified type. You can specify either a number or a variable for the RHO= and PERIOD= suboptions. When TYPE=VARMA, the autoregressive and moving average orders can be at most 1 ($0 \leq p \leq 1$ and $0 \leq q \leq 1$). Moreover, by using the D and I flags with the order specification, you can impose additional structure on the autoregressive and moving average coefficient matrices—for example, specifying TYPE=VARMA(P=1) implies a VAR(1) model with general autoregressive coefficient matrix, whereas specifying TYPE=VARMA(P(D)=1) implies a VAR(1) model with diagonal autoregressive coefficient matrix. If you specify the TYPE= option, the T, COV1, SINPUT, and A1 options are not needed. In fact they are ignored, since the transition matrix $\mathbf{T}_t$ and the matrices in the initial condition ($\mathbf{Q}_1$ and $\mathbf{A}_1$) are implicitly defined by the choice of the type. However, the COV and W options can be useful. In fact, the specification of the

COV option does play a key role in the eventual form of $\mathbf{Q}_t$—the covariance of the disturbance term in the transition equation. For the types LL, CYCLE, SEASON, and VARMA, the dimension of the resulting state subsection is a certain multiple of *dim*, the nominal dimension in the STATE statement. For example, the following specification results in a state subsection, named cycleState, of dimension 2**dim*:

```
state cycleState(dim) cov(g) type=cycle;
```

The name cycleState corresponds to the state underlying a *dim*-dimensional cycle component. All of these special state types require that the data be regular (replication is permissible); the only exception is TYPE=CYCLE(CT), which defines a continuous-time cycle and is applicable to any data type. Table 34.2 summarizes some of this information for easy reference. For more information about these state types, see the section "Predefined Structural Models" on page 2407.

The TYPE=LL specification results in a state that corresponds to a multivariate local linear trend. It is governed by two covariance matrices: the COV option specifies the covariance that corresponds to the level equation, and the SLOPECOV suboption specifies the covariance used in the slope equation. The omission of the SLOPECOV suboption signifies that the covariance used in the slope equation is zero. The form of the SLOPECOV suboption is exactly the same as that of the COV option.

The TYPE=CYCLE option results in a state that corresponds to a (stochastic) cycle. By default, this cycle is assumed to be for the regular data type. If TYPE=CYCLE(CT), the resulting cycle is applicable to any data type. The CT option is available only for *dim = 1*; that is, only a univariate cycle is available for the irregular data type. The cycle specification depends on a covariance matrix and two numbers: the damping factor RHO and the cycle period PERIOD. The covariance can be specified by the COV option. The damping factor is specified by the RHO= suboption; its value must lie between 0.0 and 1.0. The cycle period can be specified by the PERIOD= suboption. If the CT suboption is not included, the period value must be larger than 2.0. On the other hand, if the CT suboption is included, its value must be strictly positive. If these parameters are not specified, they are estimated from the data.

The TYPE=SEASON(LENGTH=*integer*) specifies a multivariate trigonometric season that contains the full set of harmonics (for more information, see "Multivariate Season" on page 2410). In some cases, you might want to drop some of the harmonics from this complete set to obtain a more parsimonious trigonometric season specification. You can use the DROPH= (to drop) or KEEPH= (to keep) suboption to control the harmonics that are included in the season specification as follows:

TYPE=SEASON(
LENGTH=*integer*
<DROPH=*number-list* | *n* **TO** *m* **BY** *p***>**
<KEEPH=*number-list* | *n* **TO** *m* **BY** *p***>**
)

The DROPH= and KEEPH= lists can include any integer between 1 and *LENGTH*/2 if the season length is even and any integer between 1 and (*LENGTH* – 1)/2 if the season length is odd. For example, the following specification results in a specification of a trigonometric season with a season length 12 that consists of only the first four harmonics $\boldsymbol{\zeta}_j$, $j = 1, 2, 3, 4$:

```
type=season(length=12  DROPH=5 6) ...;
```

The last two *high*-frequency harmonics, $\boldsymbol{\zeta}_5$ and $\boldsymbol{\zeta}_6$, are dropped. The DROPH= suboption cannot be used with the KEEPH= suboption.

Table 34.2 Summary of Predefined State Types

Type	Description	Parameters	State Dimension
WN	*dim*-variate white noise	COV	*dim*
RW	*dim*-variate random walk	COV	*dim*
LL	*dim*-variate local linear	COV, SLOPECOV	2**dim*
SEASON(LENGTH=*length*)	*dim*-variate season	COV	(*length*–1)**dim*
CYCLE	*dim*-variate cycle	COV, RHO, PERIOD	2**dim*
VARMA(P=*p* Q=*q*)	*dim*-variate VARMA(*p*, *q*)	COV, AR, MA	*dim**max(*p*, *q*+1)

W(D)= (*var1 var2 ...***) | (***number1 number2 ...***)**

W(G)= (*var1 var2 ...***) | (***number1 number2 ...***)**

W(I) < = (*variable***) | (***number***) >**

specifies the relevant block of the design matrix $\mathbf{W}_t$ in the transition equation. The W-block is of dimension $sdim \times sg$, where *sdim* denotes the actual dimension of the state subsection (which can be the same as *dim*, the nominal dimension, or different if the TYPE= option is used) and *sg* denotes the desired size of the subsection of the overall state regression vector $\boldsymbol{\gamma}$. The absence of this option signifies that the state equation does not contain any regression effects. The number of variables supplied in the W(G)= list option must be a multiple of *sdim*. For example, if *sdim* = 4 and the W(G)= list contains 8 variables, then the implied size of $\boldsymbol{\gamma}$ subsection is 2. If the W(D)= or W(I)= option is used, then the W-block is assumed to be an $sdim$-dimensional diagonal matrix and the W(D)= list must contain exactly *sdim* variables. For examples of the use of this option, see Example 34.8, Example 34.10, and Example 34.11.

TREND Statement

TREND *name (type)* < *options* > **;**

The TREND statement defines a term in the model that follows a stochastic pattern of a certain predefined type. The *options* in the TREND statement enable you to specify a wide variety of commonly used stochastic patterns. Each TREND statement in effect stands for a special pair of STATE and COMPONENT statements. You can specify more than one TREND statement. Each separate TREND statement defines a component that is assumed to be independent of all other component specifications in the model. Very often the TREND statement is used to specify a component that captures the time-varying level of the data. However, in many cases it is also used to define components of a more general nature; for example, it can be used to define a noise component that follows a stationary ARMA model.

You can refer to the state that is associated with a TREND statement by appending the string "_state_" to the end of its name. For example, name_state_ is the state that is associated with a trend named name. You can use name_state_ in a COMPONENT statement to define a linear combination of its elements. The estimate of this linear combination can then be printed or output to a data set. The nominal dimension of name_state_ is taken to be 1, or the number of variables in the list that is specified in the CROSS= option in the TREND statement that is used to define name (see Example 34.4 for an example of such use of the COMPONENT statement).

Some of these trend specifications are applicable to all the data types—that is, they can be used for both regular data types and irregular data types, whereas the others require that the data be regular or regular with

replication. Of course, the trend specification is only part of the overall model specification. Therefore, the other parts of the model can imply additional constraints on the data type.

Table 34.3 lists the available trend models and their data requirements. The *type* column shows the admissible keywords that signify the particular trend type. For brevity, the Data Type column groups the data types regular and regular with replication into one category: regular. For more information about these trend models, see the section "Predefined Trend Models" on page 2403.

Table 34.3 Summary of Trend Types

type	**Data Type**	**Description**	**Parameters**
ARIMA(P=*integer* D=*integer* ...)	Regular	ARIMA model specification	AR and MA coefficients, and the error variance σ^2
DLL	Regular	Damped local linear	Level and slope σ_1^2, σ_2^2, damping factor ϕ
LL	Regular	Local linear	Level and slope σ_1^2, σ_2^2
RW	Regular	Random walk	Level σ^2
DECAY	Irregular	A type of decay pattern	Level σ^2, decay rate ϕ
DECAY(OU)	Irregular	Ornstein-Uhlenbeck decay pattern	Level σ^2, decay rate ϕ
GROWTH	Irregular	A type of growth pattern	Level σ^2, growth rate ϕ
GROWTH(OU)	Irregular	Ornstein-Uhlenbeck growth pattern	Level σ^2, growth rate ϕ
PS(*order*)	Irregular	Polynomial spline of a given order	Level σ^2

The keyword specification of different trend types, except possibly the ARIMA trend, is quite simple. For example, the following statement specifies polySpline as a trend of the type second-order polynomial spline:

```
trend polySpline(ps(2));
```

Similarly, the following statement defines dampedTrend as a damped local linear trend:

```
trend dampedTrend(dll) slopevar=x;
```

The variance parameter that governs the slope equation of this trend type is given by a variable x, which must be defined elsewhere in the program. The other parameters that define dampedTrend are left unspecified (and are estimated by using the data).

The ARIMA trend specification permits specification of trends that follow an ARIMA$(p,d,q)\times(P,D,Q)_s$ model. The specification of ARIMA models requires some notation, which is explained first.

Let B denote the backshift operator—that is, for any sequence ζ_t, $B\zeta_t = \zeta_{t-1}$. The higher powers of B represent larger shifts (for example, $B^3\zeta_t = \zeta_{t-3}$). A random sequence ζ_t follows an ARIMA$(p,d,q)\times(P,D,Q)_s$ model with nonseasonal autoregressive order p, seasonal autoregressive order P, nonseasonal differencing order d, seasonal differencing order D, nonseasonal moving average order q, and seasonal moving average order Q if it satisfies the following difference equation, which is specified in terms of the polynomials in the backshift operator, where a_t is a white noise sequence and s is the season length:

$$\phi(B)\Phi(B^s)(1-B)^d(1-B^s)^D\zeta_t = \theta(B)\Theta(B^s)a_t$$

The polynomials ϕ, Φ, θ, and Θ are of orders p, P, q, and Q, respectively, which can be any nonnegative integers. The season length s must be a positive integer. For example, ζ_t satisfies an ARIMA(1,0,1) model (that is, $p = 1, d = 0, q = 1, P = 0, D = 0$, and $Q = 0$) if

$$\zeta_t = \phi_1 \zeta_{t-1} + a_t - \theta_1 a_{t-1}$$

for some coefficients ϕ_1 and θ_1 and a white noise sequence a_t. Similarly, ζ_t satisfies an ARIMA(0,1,1)×(0,1,1)$_{12}$ model if

$$\zeta_t = \zeta_{t-1} + \zeta_{t-12} - \zeta_{t-13} + a_t - \theta_1 a_{t-1} - \Theta_1 a_{t-12} + \theta_1 \Theta_1 a_{t-13}$$

for some coefficients θ_1 and Θ_1 and a white noise sequence a_t. An ARIMA process is zero-mean, stationary, and invertible if $d = 0$, $D = 0$, and the defining polynomials ϕ, Φ, θ, and Θ have all their roots outside the unit circle—that is, their absolute values are strictly larger than 1.0. It is assumed that the coefficients of the polynomials ϕ, Φ, θ, and Θ are constrained so that the stationarity and invertibility conditions are satisfied. The unknown coefficients of these polynomials become part of the model parameter vector that is estimated by using the data. The general form of the ARIMA trend specification is as follows:

ARIMA(< P=*integer* **> < D=***integer* **> < Q=***integer* **> < SP=***integer* **> < SD=***integer* **> < SQ=***integer* **> < S=***integer* **>)**

By default, the different orders are equal to 0 and the season length is equal to 1. The following examples illustrate a few different ARIMA trend specifications. The following statement defines ima as an integrated moving average trend:

```
trend ima(arima(d=1 q=1));
```

The following statement defines airTrend as a trend that satisfies the well-known airline model (ARIMA(0,1,1)(0,1,1)12 model) for monthly seasonal data:

```
trend airTrend(arima(d=1 q=1 sd=1 sq=1 s=12));
```

The following statement defines arma11 as a zero-mean ARMA(1,1) trend with autoregressive parameter fixed to 0.1:

```
trend arma11(arima(p=1 q=1)) ar=0.1;
```

For an example of the use of the ARIMA trend specification, see Example 34.6.

You can use the following *options* in the TREND statement to specify the trend parameters and to request printing of the trend estimates. In addition, you can create a custom combination of a given trend type by specifying the CROSS= option to create a more general trend. For an example of using the CROSS= option, see the section "Getting Started: SSM Procedure" on page 2358 and the discussion of the second model in Example 34.4. You can also check for the unexpected changes in the trend component by using the CHECKBREAK option.

AR=ϕ_1 ϕ_2 **...** ϕ_p
lists the values of the coefficients of the nonseasonal autoregressive polynomial

$$\phi(B) = 1 - \phi_1 B - \ldots - \phi_p B^p$$

where the order p is specified in the ARIMA trend specification. The coefficients ϕ_i must define a stationary autoregressive polynomial.

CHECKBREAK<(ELEMENTWISE | OVERALL)>

turns on the checking of breaks for this trend component. The ELEMENTWISE suboption requests the elementwise checking of any unexpected change in the state subsection that is associated with the trend component. The OVERALL suboption requests a similar check for the entire state subsection—that is, in this case the change is measured as a multidimensional change. The ELEMENTWISE suboption is the default. Unless the PRINT=BREAKDETAIL option is specified, only a summary of the most significant breaks is produced. If the PRINT=BREAKDETAIL is specified, tables that contain the break significance statistics at every distinct time point are produced—one for the ELEMENTWISE suboption and one for the OVERALL suboption. If the CROSS= option is specified and the CROSS= list contains more than one variable, the OVERALL suboption considers subsections that are associated with each CROSS= variable separately. For more information about the structural break detection process, see the section "Structural Breaks in the State Evolution" on page 2401.

CROSS=(*var1, var2, ...***)**

CROSS(MATCHPARM)=(*var1, var2, ...***)**

creates a linear combination of one or more independent trend components that is based on the variables in the list. If the parameters of the trend are specified by options such as the LEVELVAR= option or the PHI= option, these parameters are shared by these constituent trends. For example, suppose that the CROSS= list contains two variables (X_1 and X_2) and the trend specification is of the type RW. The effect of CROSS=(X_1, X_2) is to create a component $\mu_t = X_1\mu_{1,t} + X_2\mu_{2,t}$, where $\mu_{1,t}$ and $\mu_{2,t}$ are two independent random walk trends. Moreover, if the random walk trend specification uses the LEVELVAR= option to specify the variance parameter, $\mu_{1,t}$ and $\mu_{2,t}$ share the same variance parameter; otherwise, two separate variance parameters are assigned to these random walks. If the second form of the CROSS option, CROSS(MATCHPARM)=, is used, then the constituent trends share all the relevant parameters no matter how they are specified. The CROSS= option is useful for a variety of situations. For example, suppose X is an indicator variable that is 1 before a certain time point t_0 and 0 thereafter. Then CROSS=(X) has the effect of turning off the trend component after time t_0. Similarly, suppose G_1 and G_2 are indicators for gender—for example, G_1 = (GENDER=1) and G_2 = (GENDER=0) for male and female cases, respectively. Then CROSS=(G_1, G_2) results in separate trends according to the gender. The variables in the CROSS= list must be free of unknown parameters.

The CROSS= option can be computationally expensive; computationally it is equivalent to specifying as many separate trends as the number of variables in the specified list.

LEVELVAR=*variable* | *number*

specifies the disturbance variance parameter for all the trend types. For trend types LL and DLL, this option specifies σ_1^2. Any nonnegative value, including 0, is permissible. If *variable* contains unknown parameters, they are estimated from the data. Similarly, if the LEVELVAR= option is not specified, σ^2 is estimated from the data.

MA=$\theta_1\ \theta_2\ \ldots\ \theta_q$

lists the values of the coefficients of the nonseasonal moving average polynomial,

$$\theta(B) = 1 - \theta_1 B - \ldots - \theta_q B^q$$

where the order q is specified in the ARIMA trend specification. The coefficients θ_i must define an invertible moving average polynomial.

NODIFFUSE

causes the diffuse elements in the initial state of the state subsection underlying the trend component to be treated as nondiffuse. This option is applicable to all trend types except ARIMA. For the ARIMA trend type, this option is ignored even if the nonseasonal or seasonal differencing orders are nonzero. The diffuse elements are assumed to be independent, zero-mean, Gaussian variables. Their variances become part of the parameter vector and are estimated by using the data. This option is useful for creating a trend component that can be interpreted as a deviation from an overall trend component (with diffuse initialization), which is defined separately.

PHI=*variable* | *number*

specifies the value of ϕ for trend types DLL, DECAY, DECAY(OU), GROWTH, and GROWTH(OU). For the type DLL, the specified value must be between 0.0 and 1.0. For types DECAY and DECAY(OU), ϕ must be strictly negative. For types GROWTH and GROWTH(OU), ϕ must be strictly positive. If *variable* contains unknown parameters, they are estimated from the data. Similarly, if the PHI= option is not specified, ϕ is estimated from the data.

PRINT=BREAKDETAIL | COV | COV1 | FILTER | SMOOTH | T

PRINT= (< BREAKDETAIL > < COV > < COV1 > < FILTER > < SMOOTH > < T >)

requests printing of the respective system matrices of the state equation that underlies the specified trend, the printing of its filtered and smoothed estimates, and the printing of the break statistics at each distinct time point. For the BREAKDETAIL suboption to have any effect, the CHECKBREAK option must be turned on. If any of these matrices are time-varying, the matrix that corresponds to the first time instance is printed.

SAR=$\Phi_1\ \Phi_2 \ldots \Phi_P$

lists the values of the coefficients of the seasonal autoregressive polynomial

$$\Phi(B^s) = 1 - \Phi_1 B^s - \ldots - \Phi_P B^{sP}$$

where the order P is specified by using the SP= option in the ARIMA trend specification and the season length s is specified in the S= option. The coefficients Φ_i must define a stationary autoregressive polynomial.

SMA=$\Theta_1\ \Theta_2 \ldots \Theta_Q$

lists the values of the coefficients of the seasonal moving average polynomial

$$\Theta(B^s) = 1 - \Theta_1 B^s - \ldots - \Theta_Q B^{sQ}$$

where the order Q is specified by using the SQ= option in the ARIMA trend specification and the season length s is specified in the S= option. The coefficients Θ_i must define an invertible moving average polynomial.

SLOPEVAR=*variable* | *number*

specifies the second disturbance variance parameter, σ_2^2, for trend types LL and DLL. Any nonnegative value, including 0, is permissible. If *variable* contains unknown parameters, they are estimated from the data. Similarly, if the SLOPEVAR= option is not specified, σ_2^2 is estimated from the data.

Details

Throughout this section, vectors and matrices are denoted by boldface letters. Generally, Greek letters (such as $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and $\boldsymbol{\epsilon}$) denote unobserved or latent quantities—often estimated from the data—that represent model parameters, latent states, or noise variables. Capital letters such as X, Y, and Z are used to denote the observed data variables. Whenever there is no ambiguity, it is assumed that the matrices have appropriate dimensions when they are being multiplied—in particular, the vectors behave as column vectors or row vectors as the need arises. On many occasions, matrices are described inline—that is, they are described as parenthesized lists, in a rowwise fashion, with the rows separated by a comma. The term "dot product" is used to describe the scalar that results from the product of a row vector with a (conforming) column vector.

State Space Model and Notation

The (linear) state space model is described in the literature in a few different ways and with varying degree of generality. The description given in this section loosely follows the description given in Durbin and Koopman (2001, chap. 6, sec. 4). This formulation of SSM is quite general; in particular, it includes nonstationary SSMs with time-varying system matrices and state equations with a diffuse initial condition (these terms are defined later in this subsection).

Suppose that observations are collected in a sequential fashion (indexed by a numeric variable τ) on some variables: the vector $\mathbf{y} = (y_1, y_2, \ldots, y_q)$, which denotes the q-variate response values, and the k-dimensional vector $\mathbf{x}$, which denotes the predictors. Suppose that the observation instances are $\tau_1 < \tau_2 < \ldots < \tau_n$. The possibility that multiple observations are taken at a particular instance τ_i is not ruled out, and the successive observation instances do not need to be regularly spaced—that is, $(\tau_2 - \tau_1)$ does not need to equal $(\tau_3 - \tau_2)$. For $t = 1, 2, \ldots, n$, suppose p_t (≥ 1) denotes the number of observations recorded at instance τ_t. For notational simplicity, an integer-valued secondary index t is used to index the data so that $t = 1$ corresponds to $\tau = \tau_1$, $t = 2$ corresponds to $\tau = \tau_2$, and so on. Consider the following model:

$$\begin{aligned}
\mathbf{Y}_t &= \mathbf{Z}_t\boldsymbol{\alpha}_t + \mathbf{X}_t\boldsymbol{\beta} + \boldsymbol{\epsilon}_t && \text{Observation equation} \\
\boldsymbol{\alpha}_{t+1} &= \mathbf{T}_t\boldsymbol{\alpha}_t + \mathbf{W}_{t+1}\boldsymbol{\gamma} + \mathbf{c}_{t+1} + \boldsymbol{\eta}_{t+1} && \text{State transition equation} \\
\boldsymbol{\alpha}_1 &= \mathbf{c}_1 + \mathbf{A}_1\boldsymbol{\delta} + \mathbf{W}_1\boldsymbol{\gamma} + \boldsymbol{\eta}_1 && \text{Initial condition}
\end{aligned}$$

The following list describes these equations:

- The *observation equation* describes the relationship between the $(p_t * q)$-dimensional response vector $\mathbf{Y}_t$ and the unobserved vectors $\boldsymbol{\alpha}_t$, $\boldsymbol{\beta}$, and $\boldsymbol{\epsilon}_t$. The q-variate responses are vertically stacked in a column to form this $(p_t * q)$-dimensional response vector $\mathbf{Y}_t$. The m-dimensional vectors $\boldsymbol{\alpha}_t$ are called *states*, the k-dimensional vector $\boldsymbol{\beta}$ is the regression coefficient vector associated with predictors $\mathbf{x}$, and the $(p_t * q)$-dimensional vectors $\boldsymbol{\epsilon}_t$ are called the *observation disturbances*. The matrices $\mathbf{Z}_t$ (of dimension $(q * p_t) \times m$) and $\mathbf{X}_t$ (of dimension $(q * p_t) \times k$) correspond to the *state effect* and the *regression effect*, respectively. The elements of $\mathbf{X}_t$ are assumed to be fully known. The states $\boldsymbol{\alpha}_t$ and the disturbances $\boldsymbol{\epsilon}_t$ are *random* sequences. It is assumed that $\boldsymbol{\epsilon}_t$ is a sequence of independent, zero-mean, Gaussian random vectors with diagonal covariances, with the diagonal elements denoted by $\sigma^2_{t,i}, i = 1, 2, \ldots, q * p_t$.
- The state sequence $\boldsymbol{\alpha}_t$ is assumed to follow a Markovian structure described by the state transition equation and the associated initial condition.

- The *state transition equation* postulates that a new instance of the state, $\boldsymbol{\alpha}_{t+1}$, is obtained by multiplying its previous instance, $\boldsymbol{\alpha}_t$, by an m-dimensional square matrix $\mathbf{T}_t$ (called the state transition matrix) and by adding three more terms: a known nonrandom vector $\mathbf{c}_{t+1}$ (called the state input); a regression term $\mathbf{W}_{t+1}\boldsymbol{\gamma}$, where $\mathbf{W}_{t+1}$ is an $m \times g$-dimensional design matrix with fully known elements and γ is the g-dimensional regression vector; and a random disturbance vector $\boldsymbol{\eta}_{t+1}$. The m-dimensional state disturbance vectors $\boldsymbol{\eta}_t$ are assumed to be independent, zero-mean, Gaussian random vectors with covariances Q_t (not necessarily diagonal).

- The *initial condition* describes the starting condition of the state evolution equation. The starting state vector $\boldsymbol{\alpha}_1$ is assumed to be partially diffuse: it is the sum of a known nonrandom vector $\mathbf{c}_1$, a mean-zero Gaussian vector $\boldsymbol{\eta}_1$, and the terms $\mathbf{A}_1\boldsymbol{\delta}$ and $\mathbf{W}_1\boldsymbol{\gamma}$. $\mathbf{A}_1\boldsymbol{\delta}$ represents the contribution from a d-dimensional diffuse vector $\boldsymbol{\delta}$ (a diffuse vector is a Gaussian vector with infinite covariance). The observation and state regression vectors $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are also assumed to be diffuse. The $m \times d$ matrix $\mathbf{A}_1$ is assumed to be completely known.

- The observation disturbances $\boldsymbol{\epsilon}_t$ and the state disturbances $\boldsymbol{\eta}_t$ (for $t \geq 1$) are assumed to be mutually independent. Either the elements of the matrices $\mathbf{Z}_t$, $\mathbf{T}_t$, and Q_t and the diagonal elements of the observation disturbance covariances $\sigma^2_{t,i}$ are assumed to be completely known, or some of them can be functions of a small set of unknown parameters (to be estimated from the data). Suppose that this unknown set of parameters is denoted by $\boldsymbol{\theta}$.

- The d-dimensional diffuse vector $\boldsymbol{\delta}$ from the state initial condition together with the observation and state regression vectors $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ constitute the overall $(d + k + g)$-dimensional diffuse initial condition of the model. See the section "Filtering, Smoothing, Likelihood, and Structural Break Detection" on page 2396 for more information.

Although this description of the state space model might appear involved, it conveniently covers many variants of the SSMs that are encountered in practice and precisely describes the most general case that can be handled by the SSM procedure. An important restriction about the preceding description of the model formulation is that it assumes that the matrices $\mathbf{X}_t$ and $\mathbf{W}_t$ that appear in the observation equation and the state equation respectively are free of unknown parameters and that the covariances of the observation disturbances $\boldsymbol{\epsilon}_t$ are diagonal. In most practical situations, the model under consideration can be easily reformulated to a statistically equivalent form that conforms to this restriction.

NOTE: The transition matrix $\mathbf{T}_t$ in the state equation relates the state $\boldsymbol{\alpha}_t$ at time t with the state $\boldsymbol{\alpha}_{t+1}$ at time $t + 1$. In many situations, such as when the observations are taken at irregular time intervals, $\mathbf{T}_t$ depends on information at both t and $t + 1$. Therefore, it is more appropriate to denote the transition matrix as $\mathbf{T}_t^{t+1}$. However, for simplicity, the former notation is used throughout this chapter. The same comment applies to the covariance matrix $\mathbf{Q}_t$ of the disturbance term $\boldsymbol{\eta}_t$.

For easy reference, Table 34.4 summarizes the information contained in the SSM equations.

Table 34.4 State Space Model: Notation

Notation	Description
$\tau_1, \tau_2, \ldots, \tau_n$	Distinct index values at which the observations are recorded
n	Number of distinct index instances
p_t	Number of observations recorded at index τ_t, $t = 1, 2, \ldots, n$
q	Number of response variables in the model

Table 34.4 *continued*

Notation	Description
$\mathbf{Y}_t = (y_{t,1}, y_{t,2}, \ldots, y_{t,p_t*q})$	Vertically stacked vector of response values recorded at τ_t
$N = q * \sum_{t=1}^{n} p_t$	Total number of response values in the data set
k	Number of predictor (regressor) variables in the observation equation
$\mathbf{X}_t$	$(p_t * q) \times k$ matrix of predictor values recorded at τ_t
$\boldsymbol{\beta}$	k-dimensional regression vector that is associated with the predictors
$\boldsymbol{\epsilon}_t \sim N(0, (\sigma^2_{t,1}, \ldots))$	$(q * p_t)$-dimensional observation disturbance vector with diagonal covariance
m	Dimension of the state vectors $\boldsymbol{\alpha}_t$
$\boldsymbol{\alpha}_t$	m-dimensional state vector
$\mathbf{Z}_t$	$(q * p_t) \times m$ matrix that is associated with $\boldsymbol{\alpha}_t$ in the observation equation
$\mathbf{T}_t$	$m \times m$ state transition matrix
$\mathbf{c}_t$	m-dimensional state input vector
$\mathbf{W}_t$	$m \times g$ design matrix associated with $\boldsymbol{\gamma}$, the state regression vector
$\boldsymbol{\gamma}$	g-dimensional state regression vector
$\boldsymbol{\eta}_t \sim N(0, Q_t)$	m-dimensional state disturbance vector
d	Dimension of the diffuse vector $\boldsymbol{\delta}$ in the state initial condition
$\boldsymbol{\delta} \sim N(0, \kappa\Sigma), \kappa \to \infty$	Diffuse vector in the state initial condition
$\mathbf{A_1}$	$m \times d$ constant matrix associated with $\boldsymbol{\delta}$
$\boldsymbol{\eta}_1 \sim N(0, Q_1)$	m-dimensional state disturbance vector in the initial condition
$\boldsymbol{\theta}$	Parameter vector

Types of Sequence Data

The state space model specification in the SSM procedure requires proper understanding of both the data organization and the form of the model. The SSMs that are appropriate for time series data might not be appropriate for irregularly spaced longitudinal data. The SSM procedure distinguishes three types of data organization based on the way the observations are sequenced by the index variable. If an index variable is not specified, it is assumed that the observations are sequenced according to the observation number.

Regular: The observations are recorded at regularly spaced intervals; that is, $\tau_1, \tau_2, \ldots, \tau_n$ are regularly spaced. Moreover, at each observation instance τ_i a single observation is recorded; that is, $p_t = 1$ for all t. The standard time series data (both univariate and multivariate) fall in this category.

Regular with Replication: The observations are recorded at regularly spaced intervals, but $p_t > 1$ for at least one t. Here the word replication is used loosely—it does not mean that the multiple observations at τ_t are replications in the precise statistical sense. The panel or cross-sectional data types fall into this category. In the panel data case with p cross-sections, $p_t = p$ for all t.

Irregular: The observations are not recorded at regular intervals, and the number of observations p_t at each index instance can be different. The longitudinal data fall into this category.

It is not always easy to decide whether the specified model is appropriate for the given data type. Whenever possible, the SSM procedure issues a note regarding the possible mismatch between the specified model and the data type being analyzed.

Overview of Model Specification Syntax

An SSM specification involves the description of the terms in the observation equation, the state transition equation, and the initial condition. For example, the response variables, the predictor variables, and the elements of the state transition matrix $\mathbf{T}_t$ must somehow be specified. The SSM procedure syntax is designed so that little effort is needed to specify the more commonly needed models, while a highly flexible language is available for specifying more complex models. Two syntax features help achieve this goal: the ability to build a complex specification by combining simpler subspecifications, and a programming language for creating lists of variables to be used later in the model specification.

The SSM procedure statements can be divided into two classes:

- programming statements, which are used to create lists of variables that can be used for a variety purposes (for example, as the elements of the model system matrices)
- statements specific to the SSM procedure that formulate the state space model and control its other aspects such as the input data specification and the resulting output

Since the matrices involved in the model specification can be specified as lists of variables, which you separately create by using the programming statements, you can finely control all aspects of the model specification. These programming statements permit the use of most DATA step language features such as the conditional logic (IF-THEN-ELSE), array type variables, and all the mathematical functions available in the DATA step. You can also use programming statements to define predictor variables on the fly.

Building a Complex Model Specification

In addition to being able to specify the system matrices in a flexible way, you can also build a complex model specification in a modular way by combining simpler subspecifications. Suppose that the state vector for the model to be specified is composed of subsections that are statistically independent, which is a common scenario in practical modeling situations. For example, suppose that $\boldsymbol{\alpha}_t$ can be divided into two disjoint subsections $\boldsymbol{\alpha}_t^a$ and $\boldsymbol{\alpha}_t^b$, which are statistically independent. This entails a corresponding block-diagonal structure to the system matrices $\mathbf{T}_t$, $\mathbf{W}_t$, and $\mathbf{Q}_t$ that govern the state equations. In this case the term $\mathbf{Z}_t\boldsymbol{\alpha}_t$ that appears in the observation equation also splits into the sum $\mathbf{Z}_t^a\boldsymbol{\alpha}_t^a + \mathbf{Z}_t^b\boldsymbol{\alpha}_t^b$ for appropriately partitioned matrices $\mathbf{Z}_t^a$ and $\mathbf{Z}_t^b$. The model specification syntax of the SSM procedure makes building an SSM from such smaller pieces easy. Throughout this chapter, the linear combinations of the state subsections (such as $\mathbf{Z}_t^a\boldsymbol{\alpha}_t^a$) that appear in the observation equation are called *components*. An SSM specification in the SSM procedure is created by combining separate component specifications. In general, you specify a component in two steps: first you define a state subsection $\boldsymbol{\alpha}_t^a$, and then you define a matching linear combination $\mathbf{Z}_t^a\boldsymbol{\alpha}_t^a$. For some special components, such as some commonly needed *trend* components, you can combine these two steps into one keyword specification.

The following list summarizes the (nonprogramming) SSM procedure syntax statements used for model specification:

- The ID statement specifies the index variable (τ). It is assumed that the data within each BY group are ordered (in ascending order) according to the ID variable. The SSM procedure automatically creates a variable, _ID_DELTA_, which contains the difference between the successive ID values. This variable is available for use by the programming statements to define time-varying system matrices. For example,

in the case of SSMs used for modeling the longitudinal data, the $\mathbf{T}_t$ and $\mathbf{Q}_t$ matrices often depend on _ID_DELTA_ (see Example 34.5).

- The PARMS statement specifies variables that serve as the parameters of the model. That is, it partially defines the model parameter vector $\boldsymbol{\theta}$. Other elements of $\boldsymbol{\theta}$ are implicitly defined if your specification of the system matrices is not fully complete.
- The STATE statement specifies a subsection of the model state vector. Multiple STATE statements can be used in the model specification; each one defines a statistically independent subsection of the model state vector. For full customization, $\mathbf{T}_t$, $\mathbf{W}_t$, and $\mathbf{Q}_t$ blocks that govern this subsection can be specified as lists of variables that are created by programming statements. However, you can obtain many commonly needed state subsection types simply by using the TYPE= option in this statement. For example, the use of TYPE=SEASON(LENGTH=12) results in a state subsection that can be used to define a monthly seasonal component.
- The COMPONENT statement specifies a linear combination that matches a state subsection that is previously defined in a STATE statement. Thus, a matching pair of STATE and COMPONENT statements define a component.
- The TREND statement is used for easy specification of some commonly needed components that follow stochastic patterns of certain predefined types.
- The IRREGULAR statement specifies the observation disturbance for a particular response variable.
- The DEPLAG statement specifies the terms in the model that involve lagged response variables.
- The MODEL statement specifies the observation equation for one of the response variables. A separate MODEL statement is needed for each response variable in the multivariate case. The MODEL statement specifies an equation in which the left-hand side is the response variable and the right-hand side is a list that contains components and regression variables.

Model Specification Steps

To illustrate the model specification steps, suppose y is a response variable and variables x1 and x2 are predictors. The following statements specify a model for y that includes two components named cycle and randomWalk, predictors x1 and x2, and an observation disturbance named whiteNoise:

```
parms lambda / lower=(1.e-6) upper=(3.14);
parms cycleVar / lower=(1.e-6);
array cycleT{4} c1-c4;
c1 = cos(lambda);
c2 = sin(lambda);
c3 = -c2;
c4 = c1;
state s_cycle(2) T(g)=(cycleT) cov(I)=(cycleVar) a1(2);
component cycle=(1 0)*s_cycle;
trend randomWalk(rw);
irregular whiteNoise;
model y = x1 x2 randomWalk cycle whiteNoise;
```

The specification begins with a PARMS statement that defines two parameters, lambda and cycleVar, along with their lower and upper bounds (essentially 0 and π for lambda, and 0 and infinity for cycleVar). Next, programming statements define an array of variables, cycleT, which contains four variables, c1–c4; these variables are used later for defining the elements of the transition matrix of a state subsection. The STATE statement specifies the two-dimensional subsection s_cycle; the dimension appears within the parentheses after the name (`s_cycle(2)`). The T= option specifies the transition matrix for the s_cycle (`T(g)=(cycleT)`); the `g` in `T(g)` signifies that the form of the T matrix is *general*. The COV= option (`cov(I)=(cycleVar)`) specifies the covariance of the state disturbance ($\mathbf{Q}_t$ for $t \geq 2$); because of the use of `I` in `cov(I)`, the covariance is of the form scaled identity, essentially a two-dimensional diagonal matrix with both diagonal elements equal to cycleVar. The initial condition for s_cycle is completely diffuse because the A1= option, which specifies $\mathbf{A}_1$, specifies that the dimension of the diffuse vector $\boldsymbol{\delta}$ is 2: `a1(2)`. In this case there is no need to specify the covariance $\mathbf{Q}_1$ in the initial condition. The COMPONENT statement specifies the component cycle. It specifies cycle as a dot product of two vectors—(1 0) and s_cycle, which merely selects the first element of s_cycle: `component cycle=(1 0)*s_cycle`. The TREND statement defines a component named randomWalk; its type is *rw*, which signifies random walk. The IRREGULAR statement defines an observation disturbance named whiteNoise. Both the randomWalk and whiteNoise specifications are only partially complete—for example, the disturbance variance of whiteNoise is not specified. These unspecified variances, trendVar, which corresponds to randomWalk, and wnVar, which corresponds to whiteNoise, are automatically included in the list of unknown parameters, $\boldsymbol{\theta}$, along with the parameters that are defined by the PARMS statements. Thus, the parameter vector for this model is $\boldsymbol{\theta}$ = (lambda cycleVar trendVar wnVar). Finally, the model specification is completed by the MODEL statement, which specifies the components of the observation equation: the response variable y, the predictors x1 and x2, the components randomWalk and cycle, and the irregular term whiteNoise.

The preceding statements result in an SSM with a three-dimensional state vector, which is the result of combining the two-dimensional state subsection, s_cycle, and a one-dimensional subsection underlying the trend, randomWalk. In this specification, the initial state is completely diffuse with $\mathbf{Q}_1$ a null matrix, and $\mathbf{A}_1$ equal to the three-dimensional identity. The other state system matrices $\mathbf{T}_t$ and $\mathbf{Q}_t$ are time-invariant:

$$\mathbf{T} = \begin{bmatrix} \cos(\text{lambda}) & \sin(\text{lambda}) & 0 \\ -\sin(\text{lambda}) & \cos(\text{lambda}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{Q} = \begin{bmatrix} \text{cycleVar} & 0 & 0 \\ 0 & \text{cycleVar} & 0 \\ 0 & 0 & \text{trendVar} \end{bmatrix}$$

The observation equation is obvious with $\mathbf{Z} = [1\ 0\ 1]$.

Sparse Transition Matrix Specification

It often happens that the transition matrix $\mathbf{T}$ (or $\mathbf{T}_t$ in the time-varying case) specified in a STATE statement is sparse—that is, many of its elements are zero. The algorithms in the SSM procedure exploit this sparsity for computational efficiency. In most cases the sparsity of a T-block can be inferred from the context. However, if the elements of the T-block are supplied by a list of variables in parentheses, it can be difficult to recognize elements that are structurally zero (this is because of the generality of the DATA step language used for defining the variables). To simplify the specification of such sparse transition matrix, SSM procedure has adopted a convention: the variables that correspond to structural zeros can (and should) be left *unset*—that is, these variables are declared, but no value is assigned to them. As an example, suppose that a three-dimensional state subsection has the following form of transition matrix for some variables X1, X2, and X3 defined elsewhere in the program:

$$\mathbf{T} = \begin{bmatrix} \text{X1} & 0 & 0 \\ \text{X2} & \text{X1} & 0 \\ \text{X3} & 0 & \text{X1} \end{bmatrix}$$

The following (incomplete) statements show how to specify such a T-block:

```
array tMat{3,3};
do i=1 to 3;
   tMat[i, i] = x1;
end;
tMat[2,1] = x2;
tMat[3,1] = x3;
state foo(3) T(g)=(tMat) ...;
```

In this specification only the nonzero elements of the tMat array, which contains $3 \times 3 = 9$ elements, are assigned a value. On the other hand, the following statements show an alternate way of specifying the same T-block. This specification explicitly sets the zeros in the T-block (the elements above the diagonal and tMat[3,2]) to 0.

```
array tMat{3,3};
do i=1 to 3;
   do j=1 to 3;
       if i=j then tMat[i, j] = x1;
       else if j > i then tMat[i, j] = 0;
   end;
end;
tMat[2,1] = x2;
tMat[3,1] = x3;
tMat[3,2] = 0;
state foo(3) T(g)=(tMat) ...;
```

The first specification is simpler, and is preferred. The second specification is mathematically equivalent (and generates the same output) but is computationally less efficient since its sparsity structure cannot always be reliably inferred due to the generality of the DATA step language. In the first specification, the unset elements are recognized to be *structural* zeros while the set elements are treated as nonzero for sparsity purposes. See Example 34.5 for a simple illustration. Proper sparsity specification can lead to significant computational savings for larger matrices.

Regression Variable Specification in Multivariate Models

Suppose that a regression variable in a multivariate model affects two or more response variables. For example, suppose that response variables y1 and y2 depend on a regression variable x. This dependence can be categorized as one of two types:

- In the more common case, the regression coefficient of x for y1 and the regression coefficient of x for y2 are different. The relationship can be described as follows:

$$\begin{aligned} y1 &= \beta_1 x + \text{other terms} \\ y2 &= \beta_2 x + \text{other terms} \end{aligned}$$

  In the SSM procedure you can specify this type of relationship in two equivalent ways:

  – You can specify the variable x in the MODEL statement for y1 and specify the variable x_copy (a copy of x) in the MODEL statement for y2 as follows:

```
x_copy = x;             /* create a copy of x */
model y1 = x ...;
model y2 = x_copy ...;
```

– You can specify the variable x in MODEL statements for both y1 and y2 as follows:

```
model y1 = x ...;
model y2 = x ...;
```

This specification avoids creating x_copy.

Of these two alternate ways, the first is preferred because x and x_copy can then be unambiguously used in an EVAL statement to refer to the terms $\beta_1 x$ and $\beta_2 x$, respectively.

- In the less common case, y1 and y2 share a common regression coefficient.The relationship can be described as follows:

$$\begin{aligned} y1 &= \beta x + \text{other terms} \\ y2 &= \beta x + \text{other terms} \end{aligned}$$

You can specify this type of relationship by placing the regression coefficient in the model state vector as follows:

```
state beta(1) T(I) A1(1) ;       /* beta is a constant state */
comp xeffect = beta*(x) ;
model y1 =  xeffect ...;
model y2 =  xeffect ...;
```

Here the STATE statement defines beta as a one-dimensional, time-invariant constant (because the transition matrix is identity, the disturbance covariance is 0 and the initial state is diffuse). Next, the COMP statement defines xeffect as the product between beta and the variable x. Subsequently, both y1 and y2 use xeffect in their respective MODEL statements.

Filtering, Smoothing, Likelihood, and Structural Break Detection

The Kalman filter and smoother (KFS) algorithm is the main computational tool for using SSM for data analysis. This subsection briefly describes the basic quantities generated by this algorithm and their relationship to the output generated by the SSM procedure. For proper treatment of SSMs with a diffuse initial condition or when regression variables are present, a modified version of the traditional KFS, called diffuse Kalman filter and smoother (DKFS), is needed. A good discussion of the different variants of the traditional and diffuse KFS can be found in Durbin and Koopman (2001). The DKFS implemented in the SSM procedure closely follows the treatment in De Jong and Chu-Chun-Lin (2003). Additional details can be found in these references.

The state space model equations (see the section "State Space Model and Notation" on page 2389) imply that the combined response data vector $\mathbf{Y} = (\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_n)$ has a Gaussian probability distribution. This probability distribution is *proper* if d, the dimension of the diffuse vector $\boldsymbol{\delta}$ in the initial condition, is 0 and if $(k + g)$, the total number of regression variables in the observation and state equations, is also 0 (the regression vectors $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are also treated as a diffuse vectors). Otherwise, this probability distribution is *improper*. The KFS algorithm is a combination of two iterative phases: a forward pass through the data,

called *filtering*, and a backward pass through the data, called *smoothing*, that uses the quantities generated during filtering. One of the advantages of using the SSM formulation to analyze the time series data is its ability to handle the missing values in the response variables. The KFS algorithm appropriately handles the missing values in $\mathbf{Y}$. For additional information about how PROC SSM handles missing values, see the section "Missing Values" on page 2415.

Filtering Pass

The filtering pass sequentially computes the quantities shown in Table 34.5 for $t = 1, 2, \ldots, n$ and $i = 1, 2, \ldots, q * p_t$.

Table 34.5 KFS: Filtering Phase

Quantity	Description
$\hat{y}_{t,i} = \mathrm{E}(y_{t,i} \mid y_{t,i-1}, \ldots y_{t,1}, \mathbf{Y}_{t-1}, \ldots, \mathbf{Y}_1)$	One-step-ahead prediction of the response values
$v_{t,i} = y_{t,i} - \hat{y}_{t,i}$	One-step-ahead prediction residuals
$F_{t,i} = \mathrm{Var}(y_{t,i} \mid y_{t,i-1}, \ldots y_{t,1}, \mathbf{Y}_{t-1}, \ldots, \mathbf{Y}_1)$	Variance of the one-step-ahead prediction
$\hat{\boldsymbol{\alpha}}_{t,i} = \mathrm{E}(\boldsymbol{\alpha}_t \mid y_{t,i-1}, \ldots y_{t,1}, \mathbf{Y}_{t-1}, \ldots, \mathbf{Y}_1)$	One-step-ahead prediction of the state vector
$\mathbf{P}_{t,i} = \mathrm{Cov}(\boldsymbol{\alpha}_t \mid y_{t,i-1}, \ldots y_{t,1}, \mathbf{Y}_{t-1}, \ldots, \mathbf{Y}_1)$	Covariance of $\hat{\boldsymbol{\alpha}}_{t,i}$
$\mathbf{b}_{t,i}$	$(d + k + g)$-dimensional vector
$\mathbf{S}_{t,i}$	$(d + k + g)$-dimensional symmetric matrix
$\left(\hat{\boldsymbol{\delta}}' \ \hat{\boldsymbol{\beta}}' \ \hat{\boldsymbol{\gamma}}'\right)'_{t,i} = \mathbf{S}_{t,i}^{-1}\mathbf{b}_{t,i}$	Estimates of $\boldsymbol{\delta}$, $\boldsymbol{\beta}$, and $\boldsymbol{\gamma}$ by using the data up to (t, i)
$\mathbf{S}_{t,i}^{-1}$	Covariance of $\left(\hat{\boldsymbol{\delta}}' \ \hat{\boldsymbol{\beta}}' \ \hat{\boldsymbol{\gamma}}'\right)'_{t,i}$

Here the notation $\mathrm{E}(y_{t,i} \mid y_{t,i-1}, \ldots y_{t,1}, \mathbf{Y}_{t-1}, \ldots, \mathbf{Y}_1)$ denotes the *conditional expectation* of $y_{t,i}$ given the history up to the index $(t, i-1)$: $(y_{t,i-1}, \ldots y_{t,1}, \mathbf{Y}_{t-1}, \ldots, \mathbf{Y}_1)$. Similarly $\mathrm{Var}(y_{t,i} \mid y_{t,i-1}, \ldots y_{t,1}, \mathbf{Y}_{t-1}, \ldots, \mathbf{Y}_1)$ denotes the corresponding conditional variance. The quantity $v_{t,i} = y_{t,i} - \hat{y}_{t,i}$ is set to missing whenever $y_{t,i}$ is missing. Note that $\hat{y}_{t,i}$ are *one-step-ahead* forecasts only when the model has only one response variable and the data are a time series; in all other cases it is more appropriate to call them *one-measurement-ahead* forecasts (since the next measurement might be at the same time point). Despite this, $\hat{y}_{t,i}$ are called one-step-ahead predictions (and $v_{t,i}$ are called one-step-ahead residuals) throughout this document. In the diffuse case, the conditional expectations must be appropriately interpreted. The vector $\mathbf{b}_{t,i}$ and the matrix $\mathbf{S}_{t,i}$ contain some accumulated quantities that are needed for the estimation of $\boldsymbol{\delta}$, $\boldsymbol{\beta}$, and $\boldsymbol{\gamma}$. Of course, when $(d + k + g) = 0$ (the nondiffuse case), these quantities are not needed. In the diffuse case, because the matrix $\mathbf{S}_{t,i}$ is sequentially accumulated (starting at $t = 1, i = 1$), it might not be invertible until some $t = t_*, i = i_*$. The filtering process is called *initialized* after $t = t_*, i = i_*$. In some situations, this initialization might not happen even after the entire sample is processed—that is, the filtering process remains *uninitialized*. This can happen if the regression variables are collinear or if the data are not sufficient to estimate the initial condition $\boldsymbol{\delta}$ for some other reason.

The filtering process is used for a variety of purposes. One important use of filtering is to compute the likelihood of the data. In the model-fitting phase, the unknown model parameters $\boldsymbol{\theta}$ are estimated by maximum likelihood. This requires repeated evaluation of the likelihood at different trial values of $\boldsymbol{\theta}$. After $\boldsymbol{\theta}$ is estimated, it is treated as a known vector. The filtering process is used again with the fitted model in the forecasting phase, when the one-step-ahead forecasts and residuals based on the fitted model are provided.

In addition, this filtering output is needed by the smoothing phase to produce the full-sample component estimates and for the structural break analysis.

Likelihood Computation and Model Fitting Phase

In view of the Gaussian nature of the response vector, the likelihood of $\mathbf{Y}$ can be computed by using the prediction-error decomposition. In the diffuse case the definition of the likelihood depends on the treatment of the diffuse quantities—$\boldsymbol{\delta}$, $\boldsymbol{\beta}$, and $\boldsymbol{\gamma}$. In the SSM procedure a likelihood called the *diffuse*-likelihood, $\mathbf{L}_d(\mathbf{Y}, \boldsymbol{\theta})$, is used for parameter estimation. In the literature the diffuse likelihood is also called the *restricted*-likelihood. The diffuse likelihood is computed by treating the diffuse quantities as zero-mean, Gaussian, random variables with infinite variance (that is, they have diffuse distribution). In terms of the quantities described in Table 34.5 the diffuse likelihood is defined as follows:

$$-2\log \mathbf{L}_d(\mathbf{Y}, \boldsymbol{\theta}) = N_0 \log 2\pi + \sum_{t=1}^{n}\sum_{i=1}^{q*p_t}(\log F_{t,i} + \frac{v_{t,i}^2}{F_{t,i}}) - \log(|\mathbf{S}_{n,p_n}^{-1}|) - \mathbf{b}_{n,p_n}^{'}\mathbf{S}_{n,p_n}^{-1}\mathbf{b}_{n,p_n}$$

where $N_0 = (N - k - g - d)$, $|\mathbf{S}_{n,p_n}^{-1}|$ denotes the determinant of $\mathbf{S}_{n,p_n}^{-1}$, and $\mathbf{b}_{n,p_n}^{'}$ denotes the transpose of the column vector $\mathbf{b}_{n,p_n}$. In the preceding formula, the terms that are associated with the missing response values $y_{t,i}$ are excluded and N denotes the total number of nonmissing response values in the sample. If $\mathbf{S}_{n,p_n}$ is not invertible, then a generalized inverse is used in place of $\mathbf{S}_{n,p_n}^{-1}$, and $|\mathbf{S}_{n,p_n}^{-1}|$ is computed based on the nonzero eigenvalues of $\mathbf{S}_{n,p_n}$. Moreover, in this case $N_0 = N - \mathrm{Rank}(\mathbf{S}_{n,p_n})$. When $(d + k + g) = 0$, the terms that involve $\mathbf{S}_{n,p_n}$ and $\mathbf{b}_{n,p_n}$ are absent.

In addition to reporting the diffuse likelihood, the SSM procedure reports a variant of the likelihood called the *profile* likelihood. The profile likelihood is computed by treating the diffuse quantities—$\boldsymbol{\delta}$, $\boldsymbol{\beta}$, and $\boldsymbol{\gamma}$—as unknown parameters (similar to $\boldsymbol{\theta}$). Interestingly, the quantities that are described in Table 34.5 play a key role in the computation of this variant of the likelihood also. It turns out that the filtering process yields the maximum likelihood (ML) estimates of $\boldsymbol{\delta}$, $\boldsymbol{\beta}$, and $\boldsymbol{\gamma}$ conditional on the remaining parameters of the model—$\boldsymbol{\theta}$. Moreover, the likelihood that is evaluated at the ML estimates of $\boldsymbol{\delta}$, $\boldsymbol{\beta}$, and $\boldsymbol{\gamma}$—that is, the likelihood from which these parameters are *profiled* out—has the following expression:

$$-2\log \mathbf{L}_p(\mathbf{Y}, \boldsymbol{\theta}) = N \log 2\pi + \sum_{t=1}^{n}\sum_{i=1}^{q*p_t}(\log F_{t,i} + \frac{v_{t,i}^2}{F_{t,i}}) - \mathbf{b}_{n,p_n}^{'}\mathbf{S}_{n,p_n}^{-1}\mathbf{b}_{n,p_n}$$

Note that, computationally, the profile likelihood differs from the diffuse likelihood in only two respects: the constant term involves N—the total number of nonmissing response values—rather than N_0, and the log-determinant term $\log(|\mathbf{S}_{n,p_n}^{-1}|)$ is absent. However, in terms of theoretical considerations, the diffuse likelihood and the profile likelihood differ in an important way. It can be shown that the diffuse likelihood corresponds to the (nondiffuse) likelihood of a suitable transformation of $\mathbf{Y}$. The transformation is chosen in such a way that the distribution of the transformed data no longer depends on the initial condition $\boldsymbol{\delta}$ and the regression vectors $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$. In this sense, the diffuse likelihood is a pseudo-likelihood of the original data $\mathbf{Y}$. The profile likelihood, on the other hand, does not involve any data transformation and can be considered as the likelihood of the original data $\mathbf{Y}$. Of course, if the state space model for $\mathbf{Y}$ does not involve any diffuse quantities, then the two likelihoods are the same.

As noted earlier, the SSM procedure does not use the profile likelihood for parameter estimation. When the model specification contains any unknown parameters $\boldsymbol{\theta}$, they are estimated by maximizing the diffuse likelihood function. This is done by using a nonlinear optimization process that involves repeated evaluations of $\mathbf{L}_d(\mathbf{Y}, \boldsymbol{\theta})$ at different values of $\boldsymbol{\theta}$. The maximum likelihood (ML) estimate of $\boldsymbol{\theta}$ is denoted by $\hat{\boldsymbol{\theta}}$. Because

the diffuse likelihood is also called the restricted likelihood, $\hat{\boldsymbol{\theta}}$ is sometimes called the restricted maximum likelihood (REML) estimate. Approximate standard errors of $\hat{\boldsymbol{\theta}}$ are computed by taking the square root of the diagonal elements of its (approximate) covariance matrix. This covariance is computed as $-\mathbf{H}^{-1}$, where $\mathbf{H}$ is the Hessian (the matrix of the second-order partials) of $\log \mathbf{L}_d(\mathbf{Y}, \boldsymbol{\theta})$ evaluated at the optimum $\hat{\boldsymbol{\theta}}$. It is known that the ML (or REML) estimate of $\boldsymbol{\theta}$ based on the diffuse likelihood as well as the profile likelihood is consistent and efficient under mild regularity assumptions (as the number of distinct time points tend toward infinity). In addition, it is known that the estimate based on the diffuse likelihood is better in terms of having smaller bias. For good discussions about diffuse and profile likelihoods, see Laird (2004); Francke, Koopman, and de Vos (2010).

Let $\dim(\theta)$ denote the dimension of the parameter vector $\boldsymbol{\theta}$. After the parameter estimation is completed, PROC SSM prints the "Likelihood Computation Summary" table, which summarizes the likelihood calculations at $\hat{\boldsymbol{\theta}}$, as shown in Table 34.6.

Table 34.6 Likelihood Computation Summary

Quantity	Formula
Nonmissing response values used	N
Estimated parameters	$\dim(\theta)$
Initialized diffuse state elements	$\operatorname{rank}(\mathbf{S}_{n,p_n})$
Normalized residual sum of squares	$\sum_{t=1}^{n}\sum_{i=1}^{q*p_t}(\frac{v_{t,i}^2}{F_{t,i}}) - \mathbf{b}_{n,p_n}^{'}\mathbf{S}_{n,p_n}^{-1}\mathbf{b}_{n,p_n}$
Diffuse log likelihood	$\log \mathbf{L}_d(\mathbf{Y}, \hat{\boldsymbol{\theta}})$
Profile log likelihood	$\log \mathbf{L}_p(\mathbf{Y}, \hat{\boldsymbol{\theta}})$

In addition, the information criteria based on the diffuse likelihood and the profile likelihood are also reported. A variety of information criteria are reported. All these criteria are functions of twice the negative likelihood, $-2\log \mathbf{L}$ (the likelihood can be either diffuse or profile); N_*, the effective sample size; and $nparm$, the effective number of model parameters. For the information criteria based on the diffuse likelihood, the effective sample size $N_* = N_0$ and the effective number of model parameters $nparm = \dim(\theta)$. For the information criteria based on the profile likelihood, the effective sample size $N_* = N$ and the effective number of model parameters $nparm = \dim(\theta) + d + k + g$. Table 34.7 summarizes the reported information criteria in smaller-is-better form.

Table 34.7 Information Criteria

Criterion	Formula	Reference
AIC	$-2\log \mathbf{L} + 2nparm$	Akaike (1974)
AICC	$-2\log \mathbf{L} + 2nparm N_*/(N_* - nparm - 1)$	Hurvich and Tsai (1989) Burnham and Anderson (1998)
HQIC	$-2\log \mathbf{L} + 2nparm \log\log(N_*)$	Hannan and Quinn (1979)
BIC	$-2\log \mathbf{L} + nparm \log(N_*)$	Schwarz (1978)
CAIC	$-2\log \mathbf{L} + nparm(\log(N_*) + 1)$	Bozdogan (1987)

Forecasting Phase

After the model-fitting phase, the filtering process is repeated again to produce the model-based one-step-ahead response variable forecasts ($\hat{y}_{t,i}$), residuals ($v_{t,i}$), and their standard errors ($\sqrt{F_{t,i}}$). In addition, one-step-ahead forecasts of the components that are specified in the MODEL statements, and any other user-defined linear combinations of $\boldsymbol{\alpha}_t$, are also produced. These forecasts are set to missing as long as the index $t < t_*$ (that is, until the filtering process is initialized). If the filtering process remains uninitialized, then all the quantities that are related to the one-step-ahead forecast (such as $\hat{y}_{t,i}$ and $v_{t,i}$) are reported as missing. When the fitted model is appropriate, the one-step-ahead residuals $v_{t,i}$ form a sequence of uncorrelated normal variates. This fact can be used during model diagnostic process.

Smoothing Phase

After the filtering phase of KFS produces the one-step-ahead predictions of the response variables and the underlying state vectors, the smoothing phase of KFS produces the full-sample versions of these quantities—that is, rather than using the history up to $(t, i-1)$, the entire sample $\mathbf{Y}$ is used. The smoothing phase of KFS is a backward algorithm, which begins at $t = n$ and $i = q * p_n$ and goes back toward $t = 1$ and $i = 1$. It produces the following quantities:

Table 34.8 KFS: Smoothing Phase

Quantity	Description
$\tilde{y}_{t,i} = \mathrm{E}(y_{t,i}\|\mathbf{Y})$	Interpolated response value
$\tilde{F}_{t,i} = \mathrm{Var}(y_{t,i}\|\mathbf{Y})$	Variance of the interpolated response value
$\tilde{\boldsymbol{\alpha}}_t = \mathrm{E}(\boldsymbol{\alpha}_t\|\mathbf{Y})$	Full-sample estimate of the state vector
$\tilde{\mathbf{P}}_t = \mathrm{Cov}(\boldsymbol{\alpha}_t\|\mathbf{Y})$	Covariance of $\tilde{\boldsymbol{\alpha}}_t$
$\left(\hat{\boldsymbol{\delta}}\ \ \hat{\boldsymbol{\beta}}\ \ \hat{\boldsymbol{\gamma}}\right)' = \mathbf{S}_{n,p_n}^{-1}\mathbf{b}_{n,p_n}$	Full-sample estimates of $\boldsymbol{\delta}$, $\boldsymbol{\beta}$, and $\boldsymbol{\gamma}$
$\mathbf{S}_{n,p_n}^{-1}$	Covariance of $\left(\hat{\boldsymbol{\delta}}\ \ \hat{\boldsymbol{\beta}}\ \ \hat{\boldsymbol{\gamma}}\right)'$

Note that if $y_{t,i}$ is not missing, then $\tilde{y}_{t,i} = \mathrm{E}(y_{t,i}|\mathbf{Y}) = y_{t,i}$ and $\tilde{F}_{t,i} = \mathrm{Var}(y_{t,i}|\mathbf{Y}) = 0$ because $y_{t,i}$ is completely known, given $\mathbf{Y}$. Therefore, $\tilde{y}_{t,i}$ provides nontrivial information only when $y_{t,i}$ is missing—in which case $\tilde{y}_{t,i}$ represents the best estimate of $y_{t,i}$ based on the available data. The full-sample estimates of components that are specified in the model equations are based on the corresponding linear combinations of $\tilde{\boldsymbol{\alpha}}_t$. Similarly, their standard errors are computed by using appropriate functions of $\tilde{\mathbf{P}}_t$.

If the filtering process remains uninitialized until the end of the sample (that is, if $\mathbf{S}_{n,p_n}$ is not invertible), some linear combinations of $\boldsymbol{\delta}$, $\boldsymbol{\beta}$, and $\boldsymbol{\gamma}$ are not estimable. This, in turn, implies that some linear combinations of $\boldsymbol{\alpha}_t$ are also inestimable. These inestimable quantities are reported as missing. For more information about the estimability of the state effects, see Selukar (2010).

Delete-One Cross Validation and Structural Breaks

In addition to the interpolation of missing response values and the full-sample estimation of components in the model, the smoothing phase can also produce several useful diagnostic measures that can indicate outlying observations and breaks in the state evolution process. The treatment of additive outliers and structural breaks that is described in this section is based on De Jong and Penzer (1998).

Delete-One Cross Validation and the Additive Outlier Detection

Let $\mathrm{AO}_{t,i} = y_{t,i} - \mathrm{E}(y_{t,i}|\mathbf{Y}^{t,i})$ denote the difference between the observed response value $y_{t,i}$ and its estimate or prediction by using all the data except $y_{t,i}$, which is denoted by $\mathbf{Y}^{t,i}$. The smoothing phase of DKFS can generate $\mathrm{AO}_{t,i}$ (and its variance) at all (t, i). A large value of $\mathrm{AO}_{t,i}$ signifies that the observed response value $(y_{t,i})$ is unusual relative to the rest of the sample (according to the postulated model). Such values are called additive outliers. In the literature, $\mathrm{AO}_{t,i}$ are referred by a few different names. Sometimes they are called *delete-one cross validation errors* or simply *prediction errors*. In this chapter, these names are used interchangeably. Like the one-step-ahead residuals, $v_{t,i}$, the prediction errors can be used in checking the adequacy of the model. The prediction errors are normally distributed; however, unlike $v_{t,i}$, they are not serially uncorrelated. $\mathrm{AO}_{t,i}$ is set to missing when $y_{t,i}$ is missing. The SSM procedure prints a summary table of extreme additive outliers by default. In addition, you can request the plotting of the standardized prediction errors, and they can be output to a data set.

The prediction error sum of squares (PRESS)

$$\sum_{t,i} AO_{t,i}^2$$

can be a useful measure of fit to compare different models. It is also called the *cross validation error sum of squares*. An additional measure of fit based on the prediction errors is called the *generalized* cross validation error sum of squares (GCV). Denoting the variance of $\mathrm{AO}_{t,i}$ by $\mathrm{VAR\_AO}_{t,i}$, it is defined as

$$\frac{\sum_{t,i}(AO_{t,i}^2/VAR\_AO_{t,i}^2)}{[\,\sum_{t,i}(1/VAR\_AO_{t,i})\,]^2}$$

You can request the printing of PRESS and GCV by specifying the PRESS option in the OUTPUT statement.

After inspecting the reported additive outliers, you can adjust the model to account for the effects of some of the extreme outlying observations. This can be done by including appropriate dummy variables in the observation equation.

Structural Breaks in the State Evolution

The additive outliers that are discussed in the preceding section are diagnostic measures associated with the measurement equation. The smoothing phase of DKFS can generate diagnostic measures that are also associated with the state equation.

For simplicity of notation and exposition, initially assume that the state equation has the following form:

$$\boldsymbol{\alpha}_{t+1} = \mathbf{T}_t\boldsymbol{\alpha}_t + \mathbf{c}_{t+1} + \boldsymbol{\eta}_{t+1}$$

That is, the state regression term $\mathbf{W}_{t+1}\boldsymbol{\gamma}$ is absent in the postulated model. Suppose that an unanticipated change of unknown size takes place in the i_0th element of the state at time $(t_0 + 1)$. The model can then be adjusted to account for this change by including a suitable dummy regressor in the state equation as follows:

$$\boldsymbol{\alpha}_{t+1} = \mathbf{T}_t\boldsymbol{\alpha}_t + \mathbf{W}_{t+1}\boldsymbol{\gamma} + \mathbf{c}_{t+1} + \boldsymbol{\eta}_{t+1}$$

Here $\mathbf{W}_t$ is a sequence of m-dimensional column vectors such that $\mathbf{W}_{t_0+1}[i_0] = 1$ and $\mathbf{W}_t[i] = 0$ for all other t and i. The estimate of the regression coefficient $\boldsymbol{\gamma}$ provides information about the size of the unanticipated change in the i_0th element of $\boldsymbol{\alpha}_t$ at time $t = t_0 + 1$. Similarly, an unanticipated change in a subsection of $\boldsymbol{\alpha}_t$ at a time $t = t_0 + 1$ can be estimated by using a set of appropriate dummies (the number of dummies equals the number of elements in the state subsection) in the state equation. The algorithm

of De Jong and Penzer (1998) efficiently generates the estimates of such one-time changes in the state at all distinct time points in the sample in one smoothing pass. A statistically significant value of $\boldsymbol{\gamma}$ at a time point t_0 indicates an unanticipated change in the relevant element (or the subsection) of $\boldsymbol{\alpha}_{t_0}$. Note that the change associated with an additive outlier is temporary: the previous or the subsequent measurements are not affected. On the other hand, because of the evolutionary nature of the state equation, a one-time change in the state affects all the subsequent states, which in turn affect the subsequent observations. In this sense, a significant unanticipated change in the state is a *structural break*.

In the preceding discussion, the absence of the state regression variables in the postulated model was assumed only for notational simplicity. If the postulated model does contain some state regression variables, the dummy variable that is associated with the one-time state change is simply added to the existing set of state regression variables, and the interpretation of its regression coefficient as the measure of unanticipated change in the state remains unaffected.

In the SSM procedure, you can request the computation of significance statistics that are associated with one-time changes in the state subsections specified by using the STATE statement in addition to the state subsections that are associated with the components specified by using the TREND statements. This is done by using the CHECKBREAK option in these statements. In addition, you can request the computation of such statistics for the entire state $\boldsymbol{\alpha}_t$ by using the MAXSHOCK option in the OUTPUT statement. The significance statistics can be computed for both elementwise change and subsectionwise change. The computation of subsectionwise change statistics can be computationally expensive for large subsections (an inversion of a $p \times p$-dimensional matrix at each distinct time point in the sample is needed for the computation of significance statistics for a state subsection of size p). For an example of structural break analysis, see Example 34.8.

Estimation of User-Specified Linear Combination of State Elements

By default, the SSM procedure computes the estimates of all the components that are specified in the MODEL statements (you can print these estimates by using the PRINT= option in the respective TREND and COMPONENT statements, or you can output these estimates to a data set by specifying it in the OUT= option in the OUTPUT statement). However, in many cases it is desirable to obtain the estimates of additional linear combinations of the state elements and the regression effects. The SSM procedure provides two statements, the COMPONENT statement and the EVAL statement, that are useful for specifying virtually any desired linear combination of the elements of the state vector and the regression effects in the observation equation. After a desired linear combination is specified, you can print or output its estimate as you would for a component that is used in the MODEL statement. This feature of the SSM procedure is illustrated in many examples in the section "Examples: SSM Procedure" on page 2421. For example, in the second part of Example 34.4, the COMPONENT and EVAL statements are used to define the contrasts between the growth profiles of cows that are receiving different treatments. Similarly, in Example 34.7, the EVAL statement is used to define the yield curve as a sum of the components that are used in the MODEL statement.

Contrasting PROC SSM with Other SAS Procedures

The SSM procedure complements several SAS/ETS procedures and the MIXED procedure in SAS/STAT software (see Chapter 77, "The MIXED Procedure" (*SAS/STAT User's Guide*)). The statistical models underlying all these procedures can be formulated as state space models; however, in many cases this formulation

effort can be considerable. Generally speaking, when a problem can be formulated and satisfactorily solved either by using the SSM procedure or by using one of these other procedures, the other procedures are likely to be more efficient. However, in many instances, the SSM procedure can solve more general problems or offer more detailed analysis, or both. Throughout this discussion, it is assumed that the problem being solved can be modeled as a linear statistical model with Gaussian response variables. In particular, situations that require models such as autoregressive conditional heteroscedasticity (ARCH) models, and models with categorical response variables are not considered. The following list provides a more specific comparison of the SSM procedure with different procedures:

- All the SAS/ETS time series analysis procedures (the ARIMA, ESM, UCM, VARMAX, STATESPACE, and PANEL procedures) require time series data and are not applicable to the longitudinal data.
- For univariate time series analysis, the modeling facilities provided by the ARIMA, ESM, and UCM procedures are adequate in most cases. The SSM procedure can handle cases that do not fit neatly into one of these categories.
- For multivariate time series data analysis, you can use the VARMAX procedure for vector ARIMA modeling and the STATESPACE procedure for state space modeling. The capabilities of the SSM procedure are complementary to these procedures. In particular, the predefined multivariate structural models available in the SSM procedure cannot be specified by either of these procedures. In addition, you can formulate a much wider range of multivariate models—for example, models for series with different frequencies, by using the SSM procedure.
- When the $\mathbf{R}$ side effects are not too complicated (for example, if $\mathbf{R}$ is diagonal), the model considered by the MIXED procedure is a special case of the model considered by the SSM procedure. In the case of diagonal $\mathbf{R}$, it is easy to see that the state vector $\boldsymbol{\alpha}_t$ is equal to $\boldsymbol{\gamma}$, the MIXED random-effects vector, for all $t \geq 1$ (that is, $\boldsymbol{\alpha}_t$ is *time invariant*). Therefore, the random-effects MIXED model is obtained by setting $\mathbf{T} = \text{Identity}$, $\mathbf{Q}_t = \mathbf{0}, t \geq 2$, $\mathbf{Q}_1 = \mathbf{G}$ (the MIXED $\mathbf{G}$ matrix), and $\mathbf{A}_1 = \mathbf{0}$.
- For the analysis of cross-sectional data, you can use the PANEL procedure. In this case, the SSM procedure capabilities are complementary. PROC SSM can provide alternate models, REML estimates, richer missing value support, and the estimates of the unobserved components (see the section "Getting Started: SSM Procedure" on page 2358 and the examples Example 34.2 and Example 34.11 for more information). In some situations the cross-sectional studies contain many panels but very few distinct time points. The PANEL procedure based analysis is better suited in such settings. In order for the analysis based on PROC SSM to be valid, the cross-sectional study must contain an adequate number of distinct time points.

Predefined Trend Models

The statistical models that govern the predefined trend components available in the SSM procedure are divided into two groups: models that are applicable to equally spaced data (possibly with replication), and models that are applicable more generally (the irregular data type). Each trend component can be described as a dot product $\mathbf{Z}\boldsymbol{\alpha}_t$ for some (time-invariant) vector $\mathbf{Z}$ and a state vector $\boldsymbol{\alpha}_t$. The component specification is complete after the vector $\mathbf{Z}$ is specified and the system matrices that govern the equations of $\boldsymbol{\alpha}_t$ are specified. For trend models for regular data, all the system matrices are time-invariant. For irregular data, $\mathbf{T}_t$ and $\mathbf{Q}_t$ depend on the spacing between the distinct time points: $(\tau_{t+1} - \tau_t)$.

Trend Models for Regular Data

These models are applicable when the data type is either regular or regular with replication. A good reference for these models is Harvey (1989).

Random Walk Trend

This model provides a trend pattern in which the level of the curve changes with time. The rapidity of this change is inversely proportional to the disturbance variance σ^2 that governs the underlying state. It can be described as $\mathbf{Z}\alpha_t$, where $Z = (1)$ and the (one-dimensional) state α_t follows a random walk:

$$\alpha_{t+1} = \alpha_t + \eta_{t+1}, \quad \eta_t \sim N(0, \sigma^2)$$

Here $\mathbf{T} = 1$ and $\mathbf{Q} = \sigma^2$. The initial condition is fully diffuse. Note that if $\sigma^2 = 0$, the resulting trend is a fixed constant.

Local Linear Trend

This model provides a trend pattern in which both the level and the slope of the curve change with time. This variation in the level and the slope is controlled by two parameters: σ_1^2 controls the level variation, and σ_2^2 controls the slope variation. If $\sigma_1^2 = 0$, the resulting trend is called an *integrated random walk*. If both $\sigma_1^2 = 0$ and $\sigma_2^2 = 0$, then the resulting model is the deterministic linear time trend. Here $\mathbf{Z} = (1\ 0)$, $\mathbf{T} = (1\ 1,\ 0\ 1)$, and $\mathbf{Q} = \mathrm{Diag}(\sigma_1^2,\ \sigma_2^2)$. The initial condition is fully diffuse.

Damped Local Linear Trend

This trend pattern is similar to the local linear trend pattern. However, in the DLL trend the slope follows a first-order autoregressive model, whereas in the LL trend the slope follows a random walk. The autoregressive parameter or the damping factor, ϕ, must lie between 0.0 and 1.0, which implies that the long-run forecast according to this pattern has a slope that tends to 0. Here $\mathbf{Z} = (1\ 0)$, $\mathbf{T} = (1\ 1,\ 0\ \phi)$, and $\mathbf{Q} = \mathrm{Diag}(\sigma_1^2,\ \sigma_2^2)$. The initial condition is partially diffuse with $\mathbf{Q_1} = \mathrm{Diag}(0,\ \sigma_2^2/(1 - \phi * \phi))$.

ARIMA Trend

This section describes the state space form for a component that follows an ARIMA$(p,d,q)\times(P,D,Q)_s$ model. The notation for ARIMA models is explained in the TREND statement.

First the state space form for the stationary case—that is, when $d = 0$ and $D = 0$, is explained. A number of alternate state space forms are possible in this case; the one described here is based on Jones (1980). With slight abuse of notation, let $p = p + s * P$ denote the effective autoregressive order, and let $q = q + sQ$ denote the effective moving average order of the model. Similarly, let ϕ be the effective autoregressive polynomial, and let θ be the effective moving average polynomial in the backshift operator with coefficients $\phi_1,\ \ldots,\ \phi_p$ and $\theta_1,\ \ldots,\ \theta_q$, obtained by multiplying the respective nonseasonal and seasonal factors. Then, a random sequence ξ_t that follows an ARMA$(p,q)\times(P,Q)_s$ model with a white noise sequence a_t has a state space form with state vector of size $m = \max(p, q + 1)$. The system matrices are as follows: $\mathbf{Z} = [1\ 0\ \ldots\ 0]$, and the transition matrix $\mathbf{T}$, in a blocked form, is given by

$$\mathbf{T} = \begin{bmatrix} 0 & I_{m-1} \\ \phi_m\ \ldots & \phi_1 \end{bmatrix}$$

where $\phi_i = 0$ if $i > p$ and I_{m-1} is an $(m-1)$ dimensional identity matrix. The covariance of the state disturbance matrix $\mathbf{Q} = \sigma^2 \psi \psi'$, where σ^2 is the variance of the white noise sequence a_t and the

vector $\psi = [\psi_0 \ldots \psi_{m-1}]'$ contains the first m values of the impulse response function—that is, the first m coefficients in the expansion of the ratio θ/ϕ. The covariance matrix of the initial state, $\mathbf{Q}_1$, is computed as

$$vec(\mathbf{Q}_1) = (\mathbf{I} - \mathbf{T} \bigotimes \mathbf{T})^{-1} vec(\mathbf{Q})$$

where $\bigotimes$ denotes the Kronecker product and the vec operation on a matrix creates a vector formed by vertically stacking the rows of that matrix.

A number of alternate state space forms are possible in the nonstationary case also. The form used by the SSM procedure utilizes the state space form for the stationary case as a building block. Suppose that a random sequence ξ_t follows an ARIMA(*p,d,q*)×(*P,D,Q*)$_s$ model with a white noise sequence a_t. As in the notation for the stationary case, with slight abuse of notation, let $d = d + s * D$ denote the effective differencing order, and let Δ be the effective differencing polynomial in the backshift operator with coefficients $\Delta_1, \ldots, \Delta_d$. It can be shown that ξ_t has a state space form with state vector size $m^\dagger = m + d$. In what follows, the system matrices and related quantities in the nonstationary case are described in terms of similar entities in the stationary case. A superscript dagger (†) has been added to distinguish the entities from the nonstationary case. $\mathbf{Z}^\dagger = [0\ 0\ \ldots\ 1\ \ldots\ 0]$ where the only nonzero value, 1, is at the index $m + 1$, and the transition matrix, $\mathbf{T}^\dagger$, in a blocked form, is given by

$$\mathbf{T}^\dagger = \begin{bmatrix} \mathbf{T} & 0 & 0 \\ \mathbf{ZT} & \Delta_1 \ldots & \Delta_d \\ 0 & I_{d-1} & 0 \end{bmatrix}$$

The state disturbance matrix $\mathbf{Q}^\dagger$ is given by

$$\mathbf{Q}^\dagger = \begin{bmatrix} \mathbf{Q} & \mathbf{QZ}' & 0 \\ \mathbf{ZQ} & \mathbf{ZQZ}' & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Finally, the initial state is partially diffuse: the first m elements are nondiffuse and the last d elements are diffuse. The covariance matrix of the first m elements is $\mathbf{Q}_1$.

Trend Models for Irregular Data

A good reference for these models is De Jong and Mazzi (2001). Throughout this section $h_t = (\tau_{t+1} - \tau_t)$ denotes the difference between the successive time points. The system matrices $\mathbf{T}_t$ and $\mathbf{Q}_t$ that govern these models depend on h_t. However, whenever the notation is unambiguous, the subscript t is omitted.

Polynomial Spline Trend

This model is a general-purpose tool for extracting a smooth trend from the noisy data. The order of the spline governs the order of the local polynomial that defines the spline. The order-1 spline corresponds to Brownian motion (continuous-time random walk), the order-2 spline corresponds to integrated Brownian motion (continuous-time integrated random walk), and the order-3 spline provides a locally quadratic trend; the default order is 1. The dimension of the state underlying this component is the same as the order of the spline. The system matrices for the orders up to 3 are described as follows (in all the cases the initial condition is fully diffuse):

- order-1 spline: $\mathbf{Z} = (1)$, $\mathbf{T} = (1)$, and $\mathbf{Q} = \sigma^2(h)$
- order-2 spline: $\mathbf{Z} = (1\ 0)$, $\mathbf{T} = (1\ h,\ 0\ 1)$, and $\mathbf{Q} = \sigma^2 \begin{pmatrix} \frac{h^3}{3} & \frac{h^2}{2}, \frac{h^2}{2} & h \end{pmatrix}$

- order-3 spline: $\mathbf{Z} = (1\ 0\ 0)$, $\mathbf{T} = \begin{pmatrix} 1 & h & \frac{h^2}{2}, 0 & 1 & h, 0 & 0 & 1 \end{pmatrix}$, and

$$\mathbf{Q}[i,j] = \sigma^2 * \frac{h^{6-i-j+1}}{(6-i-j+1)(3-i)!(3-j)!} \quad 1 \le i,j \le 3$$

The system matrices for higher orders are similarly defined (for more information, see De Jong and Mazzi (2001)).

Note that, in addition to providing an estimate of the trend, this methodology can provide estimates of the higher-order derivatives of the trend. If $\boldsymbol{\alpha}$ denotes the k-dimensional subsection that is associated with a polynomial spline of order k, then its jth element ($1 \le j \le k$), $\boldsymbol{\alpha}[j]$, corresponds to the derivative of order $(j-1)$ of this polynomial spline. For an example of the estimation of the first derivative of a trend component, see Example 34.12. For additional information about using these types of trend patterns in data analysis, see Eubank, Huang, and Wang (2003); Kohn, Ansley, and Tharm (1991); Selukar (2015).

Decay and Growth Trends

There are two choices for the decay trend: DECAY and DECAY(OU). Similarly, there are two choices for the growth trend: GROWTH and GROWTH(OU). The "OU" stands for the Ornstein-Uhlenbeck form of these models. The decay trend is a sum of two correlated components: one component is a random walk, and the other component is a stationary autoregression. In its Ornstein-Uhlenbeck form, the random walk component is replaced by a constant. The growth trend (and its Ornstein-Uhlenbeck variant) has the same form as the decay trend except that the autoregression is nonstationary (in fact, it is explosive). For growth trend models, floating-point errors can result for even moderately long forecast horizons because of the explosive growth in the trend values.

The system matrices for the decay and the growth types in their respective cases are identical, except for the sign of the rate parameter ϕ: $\phi < 0.0$ for the decay type, and $\phi > 0.0$ for the growth type. In addition, the initial conditions for the growth models are fully diffuse; they are only partially diffuse for the decay models. The underlying state vector for all these models is two-dimensional.

The system matrices for the DECAY type are:

$$\begin{aligned} \mathbf{Z} &= (1\ 1) \\ \mathbf{T} &= \mathrm{Diag}(1,\ \exp(h\phi)) \\ \mathbf{Q} &= \frac{\sigma^2}{\phi^3}\begin{pmatrix} h\phi & 1-\exp(h\phi), & 1-\exp(h\phi) & (\exp(2h\phi)-1)/2 \end{pmatrix} \end{aligned}$$

The initial condition is partially diffuse with $\mathbf{Q_1} = \mathrm{Diag}(0,\ \frac{-\sigma^2}{2\phi^3})$. The system matrices for the GROWTH type are the same (with $\phi > 0.0$), except that the initial condition is fully diffuse; so $\mathbf{Q_1} = 0$.

For the DECAY(OU) type, $\mathbf{Z}$ and $\mathbf{T}$ are the same as DECAY, whereas

$$\mathbf{Q} = \mathrm{Diag}\left(0,\ \sigma^2\frac{(\exp(2h\phi)-1)}{2\phi}\right) \quad \text{and} \quad \mathbf{Q_1} = \mathrm{Diag}(0,\ \frac{-\sigma^2}{2\phi})$$

The system matrices for the GROWTH(OU) type are the same (with $\phi > 0.0$), except that the initial condition is fully diffuse; so $\mathbf{Q_1} = 0$.

Predefined Structural Models

A set of predefined models is available in the SSM procedure for models called structural models in the time series literature. These predefined models can be used to model trend, seasonal, and cyclical patterns in the univariate and multivariate time series. For the most part, the multivariate models are straightforward generalizations of the corresponding univariate models—for example, the multivariate random walk trend described later in this section generalizes the univariate random walk trend that is described in the section "Random Walk Trend" on page 2404. All of these models, with the exception of the continuous-time cycle model, are applicable only to the regular data type. The continuous-time cycle model is applicable to all the data types; however, it is available for the univariate case only.

To specify these models, you must first use the STATE statement with the correct TYPE= option. When you specify the TYPE=option, you do not need to specify other options of the STATE statement (for example, the T option, the COV1 option, and the A1 option). However, you must specify the COV option, which describes the covariance of the disturbance term that drives the state equation. Throughout this section, the symmetric matrix specified by using the COV option is denoted by $\mathbf{\Sigma}$. For TYPE= LL, an additional matrix, specified by using the SLOPECOV suboption, also plays a role; it is denoted by $\mathbf{\Sigma}_{slope}$. Subsequently you must specify one or more COMPONENT statements to define the (univariate) components that are based on this state subsection for their inclusion in the MODEL statement. These univariate components exhibit interesting behavior based on the structure of $\mathbf{\Sigma}$ (and $\mathbf{\Sigma}_{slope}$, whenever applicable)—for example, imposing rank restrictions on $\mathbf{\Sigma}$ in the multivariate random walk results in these univariate trends moving together. For additional information about these models, see Harvey (1989).

The following example summarizes the steps needed to define a multivariate structural model by using a sequence of STATE and COMPONENT statements. For a full example, see Example 34.1. Suppose that a three-dimensional time series is being studied with response variables y1, y2, and y3. Suppose you want to specify the trivariate structural model

$$\mathbf{y}_t = \boldsymbol{\mu}_t + \boldsymbol{\psi}_t + \boldsymbol{\epsilon}_t$$

where $\mathbf{y}_t = (y_{1,t},\ y_{2,t},\ y_{3,t})$ denotes the response series, and $\boldsymbol{\mu}_t$, $\boldsymbol{\psi}_t$, and $\boldsymbol{\epsilon}_t$ denote the trivariate components, trend, cycle, and white noise, respectively. The three components of $\boldsymbol{\epsilon}_t$, the observation noise in the model, are not assumed to be independent. Therefore, you cannot specify them by using three IRREGULAR statements; you must include them in the state specification. The following (incomplete) statements show how to specify this model:

```
state whiteNoise(3)  type=wn ...;
component wn1 =  whiteNoise[1];
component wn2 =  whiteNoise[2];
component wn3 =  whiteNoise[3];

state randomWalk(3)  type=rw ...;
component rw1 =  randomWalk[1];
component rw2 =  randomWalk[2];
component rw3 =  randomWalk[3];

state cycleState(3)  type=cycle ...;
component c1 = cycleState[1];
component c2 = cycleState[2];
component c3 = cycleState[3];
```

```
model y1 = rw1 c1 wn1;
model y2 = rw2 c2 wn2;
model y3 = rw3 c3 wn3;
```

The first STATE statement defines whiteNoise, a state subsection that is needed for defining a three-dimensional white noise component. In turn, whiteNoise is used to define the three univariate white noise components: wn1, wn2, and wn3. The components wn1, wn2, and wn3 are correlated—their correlation structure is controlled by the covariance specification of whiteNoise. The second set of STATE and COMPONENT statements result in three correlated random walk trend components: rw1, rw2, and rw3. Finally, the last set of STATE and COMPONENT statements result in three correlated cycle components: c1, c2, and c3. In the end, the desired multivariate model is defined by including these (univariate) components in the appropriate MODEL statements.

In the preceding example, it is important to note the relationship between the nominal dimension (denoted by *dim* throughout this section) that is specified in the STATE statement and the actual dimension of the resulting state subsection. Note that the three state subsections, whiteNoise, randomWalk, and cycleState, are defined by using the same *dim* specification: 3. However, the actual dimensions of these state subsections depend on their type; they do not need to equal this specified dimension. Here, whiteNoise and randomWalk do have the same size, 3, as the specified *dim*. However, the size of cycleState, which is of TYPE=CYCLE, is $2 * dim = 6$. Another important point to note: no matter what the underlying size of the state subsection, the desired univariate components were obtained by using an identical specification scheme in the COMPONENT statement. This happens because the component specification style that is based on the element operator—[]—in the COMPONENT statement behaves differently when the TYPE= option is used to define the state subsection (see the section "Multivariate Season" on page 2410 for an illustration).

The system matrices for all these models are time-invariant, with the exception of the continuous-time cycle model. In this section, $\boldsymbol{\alpha}_t$ denotes the subsection of the overall model state $\boldsymbol{\alpha}_t$, and $\mathbf{T}$, $\mathbf{Q}$, and $\mathbf{A}_1$ denote the corresponding blocks of the larger system matrices.

For the multivariate cycle system matrices described in the section "Multivariate Cycle" on page 2409, the Kronecker product notation is useful: if $\mathbf{A}$ is an $m \times n$ matrix and $\mathbf{B}$ is a $p \times q$ matrix, then the Kronecker product $\mathbf{A} \bigotimes \mathbf{B}$ is an $mp \times nq$ block matrix:

$$\mathbf{A} \bigotimes \mathbf{B} = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{bmatrix}$$

Multivariate White Noise

The STATE statement option TYPE=WN specifies white noise of dimension *dim*—that is, a sequence of zero mean, independent, Gaussian vectors with covariance $\boldsymbol{\Sigma}$. The specification of the associated system matrices is trivial: $\mathbf{T}$ is zero, $\mathbf{Q} = \boldsymbol{\Sigma}$, and the initial condition is nondiffuse ($\mathbf{Q}_1 = \boldsymbol{\Sigma}$ and $\mathbf{A}_1 = 0$).

Multivariate white noise is needed to specify the observation equation noise term for the multivariate models for the time series data. Since the state space formulation for the SSM procedure requires the observation equation noise vector to have the diagonal form, you need to include the noise vector in the state. The noise term for the *i*th response variable is defined by a component that simply picks the *i*th element of this multivariate white noise. For example, the component wn_i defined as follows can be used as a noise term in the MODEL statement of the *i*th response variable:

```
state white(dim) type=wn ...;
component wn_3 = white[3];
```

Multivariate Random Walk Trend

The STATE statement option TYPE=RW specifies a *dim*-dimensional random walk

$$\boldsymbol{\alpha}_{t+1} = \boldsymbol{\alpha}_t + \boldsymbol{\eta}_{t+1}$$

where $\boldsymbol{\eta}_t$ is a sequence of zero mean, independent, Gaussian vectors with covariance $\boldsymbol{\Sigma}$. The specification of the associated system matrices is trivial: $\mathbf{T}$ is a *dim*-dimensional identity matrix, $\mathbf{I}_{dim}$, $\mathbf{Q} = \boldsymbol{\Sigma}$, and the initial condition is fully diffuse ($\mathbf{Q}_1 = 0$ and $\mathbf{A}_1 = \mathbf{I}_{dim}$).

The multivariate random walk is a useful trend model for multivariate time series data. The trend term for the *i*th response variable is defined by a component that simply picks the *i*th ($1 \leq i \leq dim$) element of $\boldsymbol{\alpha}_t$. For example, the component rw_i defined as follows can be used as a trend term in the MODEL statement of the *i*th response variable:

```
state randomWalk(3) type=rw ...;
component rw_2 = randomWalk[2];
```

Multivariate Local Linear Trend

The STATE statement option TYPE=LL specifies a *(2*dim)*-dimensional $\boldsymbol{\alpha}_t$, needed for defining a *dim*-dimensional local linear trend. The first *dim* elements of $\boldsymbol{\alpha}_t$ correspond to the needed multivariate trend, and the subsequent *dim* elements are needed to capture the slope vector of this trend. $\boldsymbol{\alpha}_t$ can be defined as

$$\boldsymbol{\alpha}_{t+1} = \mathbf{T}\boldsymbol{\alpha}_t + \boldsymbol{\eta}_{t+1}$$

where $\boldsymbol{\eta}_t$ is a sequence of zero mean, independent, Gaussian vectors with covariance $\mathrm{Diag}(\boldsymbol{\Sigma},\ \boldsymbol{\Sigma}_{slope})$ and $\mathbf{T}$ is a 2**dim*-dimensional block matrix $\mathbf{T} = (\mathbf{I}_{dim}\ \mathbf{I}_{dim},\ \mathbf{0}\ \mathbf{I}_{dim})$. The initial condition is fully diffuse ($\mathbf{Q}_1 = 0$ and $\mathbf{A}_1 = \mathbf{I}_{2*dim}$). This is a multivariate generalization of the univariate local linear trend.

The multivariate local linear trend is a useful trend model for multivariate time series data. The trend term for the *i*th response variable is defined by a component that simply picks the *i*th element ($1 \leq i \leq dim$) of $\boldsymbol{\alpha}_t$. For example, the component ll_i defined as follows can be used as a trend term in the MODEL statement of the *i*th response variable:

```
state localLin(dim) type=ll(slopecov..) ...;
component ll_3 = localLin[3];
```

Multivariate Cycle

The STATE statement option TYPE=CYCLE specifies a *(2*dim)*-dimensional $\boldsymbol{\alpha}_t$, needed for defining a *dim*-dimensional cycle. As in the LL case, the first *dim* elements of $\boldsymbol{\alpha}_t$ correspond to the needed *dim*-dimensional cycle, and the remaining *dim* elements contain some auxiliary quantities. The cycle model defined in this subsection requires a regular data type—that is, the CT option is not included. Let ρ denote the damping factor, and let $\lambda = 2\pi/\mathit{period}$ be the frequency associated with the cycle. The admissible parameter ranges are $0 < \rho \leq 1$ and $\mathit{period} > 2$, which implies that $0 < \lambda < \pi$. Let $\mathbf{C} = \rho(\cos(\lambda)\ \sin(\lambda),\ -\sin(\lambda)\ \cos(\lambda))$,

a 2×2 matrix, and let $\mathbf{T} = \mathbf{C} \bigotimes \mathbf{I}_{dim}$, a $2 * dim \times 2 * dim$ matrix. With this notation, the transition equation associated with $\boldsymbol{\alpha}_t$ is

$$\boldsymbol{\alpha}_{t+1} = \mathbf{T}\boldsymbol{\alpha}_t + \boldsymbol{\eta}_{t+1}$$

where $\boldsymbol{\eta}_t$ is a sequence of zero mean, independent, $(2 * dim)$-dimensional Gaussian vectors with covariance $\mathrm{Diag}(\boldsymbol{\Sigma}, \boldsymbol{\Sigma})$. If $\rho = 1$, the initial condition is fully diffuse ($\mathbf{Q}_1 = 0$ and $\mathbf{A}_1 = \mathbf{I}_{2*dim}$). Otherwise, it is nondiffuse: $\mathbf{Q}_1 = \frac{1}{(1-\rho^2)}\mathrm{Diag}(\boldsymbol{\Sigma}, \boldsymbol{\Sigma})$ and $\mathbf{A}_1 = 0$.

The multivariate cycle is useful for capturing periodic behavior for multivariate time series data. The cycle term for the *i*th response variable is defined by a component that simply picks the *i*th element of $\boldsymbol{\alpha}_t$. For example, the component cycle_i defined as follows can be used as a cycle term in the MODEL statement of the *i*th response variable:

```
state cycleState(dim) type=cycle  ...;
component cycle_2 = cycleState[2];
```

Multivariate Season

The STATE statement option TYPE=SEASON(LENGTH=*s*) specifies a *((s–1)*dim)*-dimensional $\boldsymbol{\alpha}_t$, needed for defining a *dim*-dimensional trigonometric season component with season length *s*. A (multivariate) trigonometric season component, $\boldsymbol{\zeta}$, is a sum of (multivariate) cycles of different frequencies,

$$\boldsymbol{\zeta} = \sum_{j=1}^{[s/2]} \boldsymbol{\zeta}_j$$

where the constituent cycles $\boldsymbol{\zeta}_j$, called harmonics, have frequencies $\lambda_j = 2\pi j/s$. All the harmonics are assumed to be statistically independent, have the same damping factor $\rho = 1$, and are governed by the disturbances with the same covariance matrix $\boldsymbol{\Sigma}$. The number of harmonics, $[s/2]$, equals $s/2$ if s is even and $(s-1)/2$ if it is odd. This means that specifying TYPE=SEASON(LENGTH=*s*) is equivalent to specifying $[s/2]$ cycle specifications with correct frequencies, damping factor $\rho = 1$, and the COV option restricted to the same covariance $\boldsymbol{\Sigma}$. The resulting $\boldsymbol{\alpha}_t$ is necessarily *((s–1)*dim)*-dimensional. When the season length *s* is even, the last harmonic cycle, $\boldsymbol{\zeta}_{s/2}$, has frequency π and requires special attention. It is of dimension *dim* rather than *2*dim* because its underlying state equation simplifies to a *dim*-variate autoregression with autoregression coefficient $-\mathbf{I}_{dim}$. As a result of this discussion, it is clear that the system matrices $\mathbf{T}$ and $\mathbf{Q}$ associated with the *((s–1)*dim)*-dimensional $\boldsymbol{\alpha}_t$ are block-diagonal with the blocks corresponding to the harmonics. The initial condition is fully diffuse.

For all the models discussed so far, the first *dim* elements of $\boldsymbol{\alpha}_t$ provided the needed (multivariate) component. This is not the case for the (multivariate) season component. Extracting the *i*th seasonal component from $\boldsymbol{\alpha}_t$ requires accumulating the contributions from the $[s/2]$ harmonics that are associated with this *i*th seasonal, which are not organized contiguously in $\boldsymbol{\alpha}_t$. For example, suppose that *dim* is 2 and the season length *s* is 4. In this case $[s/2]$ is 2, and the bivariate seasonal component is a sum of two independent bivariate cycles, $\boldsymbol{\zeta}_1$ and $\boldsymbol{\zeta}_2$. The cycle $\boldsymbol{\zeta}_1$ has frequency $\pi/2$ and its underlying state, say $\boldsymbol{\alpha}_t^a$, has dimension $2 * dim = 4$. The last harmonic, $\boldsymbol{\zeta}_2$, has frequency π, and therefore its underlying state, say $\boldsymbol{\alpha}_t^b$, has dimension 2. The combined state $\boldsymbol{\alpha}_t = (\boldsymbol{\alpha}_t^a, \boldsymbol{\alpha}_t^b)$ has dimension $6 = 4 + 2$. In order to extract the first bivariate seasonal component, you must extract the first components of bivariate cycles $\boldsymbol{\zeta}_1$ and $\boldsymbol{\zeta}_2$, which in turn implies the first elements of $\boldsymbol{\alpha}_t^a$ and $\boldsymbol{\alpha}_t^b$, respectively. Thus, obtaining the first bivariate seasonal component requires extracting the first and the fifth elements of the combined state $\boldsymbol{\alpha}_t$. Similarly, obtaining the second bivariate

seasonal component requires extracting the second and the sixth elements of the combined state $\boldsymbol{\alpha}_t$. All this can be summarized by the dot product expressions

$$\begin{aligned} s_{1t} &= (1\,0\,0\,0\,1\,0)\,\boldsymbol{\alpha}_t \\ s_{2t} &= (0\,1\,0\,0\,0\,1)\,\boldsymbol{\alpha}_t \end{aligned}$$

where s_{1t} and s_{2t} denote the first and second components, respectively, of the bivariate seasonal component. Note that s_{1t} and s_{2t} are univariate seasonal components, each of season length 4, in their own right. They are correlated components; their correlation structure depends on $\boldsymbol{\Sigma}$.

Obtaining the desired components of the multivariate seasonal component is made easy by a special syntax convention of the COMPONENT statement. Continuing with the previous example, the following examples illustrate two equivalent ways of obtaining s_{1t} and s_{2t}. The first set of statements explicitly specify the linear combinations needed for defining s_{1t} and s_{2t}:

```
state seasonState(2) type=season(length=4)  ...;
component s_1 =( 1  0  0  0  1  0 ) * seasonState;
component s_2 =( 0  1  0  0  0  1 ) * seasonState;
```

The following simpler specification achieves the same result:

```
state seasonState(2) type=season(length=4)  ...;
component s_1 = seasonState[1];
component s_2 = seasonState[2];
```

In the latter specification, the meaning of the element operator [] changes if the state in question is defined by using the TYPE= option.

Multivariate ARMA

You can specify a state vector that follows a multivariate autoregressive, moving average (VARMA) model by using the STATE statement option TYPE=VARMA. The autoregressive and moving average orders can be either 0 or 1 ($0 \leq p \leq 1$ and $0 \leq q \leq 1$)—that is, only VAR(1), MA(1), and VARMA(1,1) models can be specified. The notation and the state space form of the VARMA model described here is taken from Reinsel (1997), which is a good reference for VARMA modeling.

A *dim*-dimensional vector process $\boldsymbol{\zeta}_t$ follows a zero-mean, autoregressive order *p*, moving average order *q* (VARMA(*p*, *q*)) model if it satisfies the following matrix difference equation:

$$\boldsymbol{\zeta}_t = \sum_{i=1}^{p} \boldsymbol{\Phi}_i \boldsymbol{\zeta}_{t-i} + \boldsymbol{\epsilon}_t - \sum_{j=1}^{q} \boldsymbol{\Theta}_j \boldsymbol{\epsilon}_{t-i}$$

Here $\boldsymbol{\Phi}_i$ and $\boldsymbol{\Theta}_j$ are *dim*-dimensional square matrices and $\boldsymbol{\epsilon}_t$ is a *dim*-dimensional, Gaussian, white noise sequence with covariance matrix $\boldsymbol{\Sigma}$. If autoregressive order *p* is 0, the term that involves $\boldsymbol{\Phi}_i$ is absent; similarly, if the moving average order *q* is 0, the term that involves $\boldsymbol{\Theta}_j$ is absent. Since AR and MA orders can be at most 1, the subscripts of $\boldsymbol{\Phi}_i$ and $\boldsymbol{\Theta}_j$ can be ignored in this discussion—when applicable, an AR coefficient matrix is denoted by $\boldsymbol{\Phi}$ and an MA coefficient matrix is denoted by $\boldsymbol{\Theta}$. The unknown elements of $\boldsymbol{\Phi}$, $\boldsymbol{\Theta}$, and $\boldsymbol{\Sigma}$ constitute the parameter vector that is associated with a VARMA state. The process $\boldsymbol{\zeta}_t$ defined by the VARMA difference equation is stationary and invertible (Reinsel 1997) if and only if the eigenvalues

of $\mathbf{\Phi}$ and $\mathbf{\Theta}$ are strictly less than 1 in magnitude. By default, the SSM procedure imposes these stationarity and invertibility restrictions on $\mathbf{\Phi}$ and $\mathbf{\Theta}$. However, you can specify $\mathbf{\Phi}$ to be an identity matrix, in which case the resulting process is nonstationary.

A VARMA model can be cast into a state space form. The state space form used by the SSM procedure is described in Reinsel (1997, pp 52–53). The system matrices for the supported VARMA models are as follows:

- The VAR(1) form is the simplest. In this case, the underlying state $\boldsymbol{\alpha}_t$ is the same as the VAR(1) process $\boldsymbol{\zeta}_t$. Therefore, $\mathbf{T} = \mathbf{\Phi}$ and $\mathbf{Q_t} = \mathbf{\Sigma}$.
- Taking $\mathbf{\Phi}$ equal to the zero matrix if $p = 0$, the VARMA(1,1) and MA(1) cases can be treated together. In this case, the underlying state $\boldsymbol{\alpha}_t$ is *2*dim* dimensional and the desired VARMA process $\boldsymbol{\zeta}_t$ corresponds to its first *dim* elements. Let $\mathbf{\Psi} = \mathbf{\Phi} - \mathbf{\Theta}$. Then, in the blocked form,

$$\mathbf{T} = \begin{bmatrix} \mathbf{0} & \mathbf{I}_{dim} \\ \mathbf{0} & \mathbf{\Phi} \end{bmatrix} \quad \text{and} \quad \mathbf{Q}_t = \mathbf{Q} = \begin{bmatrix} \mathbf{\Sigma} & \mathbf{\Sigma\Psi}' \\ \mathbf{\Psi\Sigma} & \mathbf{\Psi\Sigma\Psi}' \end{bmatrix}$$

Unless $\mathbf{\Phi}$ is restricted to be identity, the underlying state $\boldsymbol{\alpha}_t$ is stationary and the covariance of the initial condition is computed by

$$vec(\mathbf{Q}_1) = (\mathbf{I} - \mathbf{T} \bigotimes \mathbf{T})^{-1} vec(\mathbf{Q})$$

where $\bigotimes$ denotes the Kronecker product and the vec operation on a matrix creates a vector formed by vertically stacking the rows of that matrix. If $\mathbf{\Phi}$ is restricted to be identity, the initial condition is fully diffuse.

Continuous-Time Cycle

The STATE statement option TYPE=CYCLE(CT) specifies a two-dimensional $\boldsymbol{\alpha}_t$, needed for defining a univariate continuous time cycle. In this case the nominal dimension, *dim*, must be 1. In particular, $\mathbf{\Sigma}$ becomes one-dimensional, which is denoted by σ^2. This cycle can be used for any data type. As before, the parameters of the cycle are a damping factor ρ, $0 < \rho \leq 1$, and *period* > 0. Unlike in the discrete-time cycle described in the section "Multivariate Cycle" on page 2409, the *period* is not required to be larger than 2. Let $\lambda = 2\pi/$*period*, and let $h_t = (\tau_{t+1} - \tau_t)$ denote the difference between successive time points. In this case, the system matrices $\mathbf{T}$ and $\mathbf{Q}$ that govern $\boldsymbol{\alpha}_t$ depend on h_t. They are:

$$\begin{aligned} \mathbf{T} &= \rho^h \left(\cos(\lambda h)\ \sin(\lambda h),\ -\sin(\lambda h)\ \cos(\lambda h)\right) \\ \mathbf{Q} &= \frac{\sigma^2(1-\rho^{2h})}{-2\ln(\rho)} * \mathbf{I}_2 \qquad \text{if } \rho < 1 \\ \mathbf{Q} &= \sigma^2 h \mathbf{I}_2 \qquad \text{if } \rho = 1 \end{aligned}$$

If $\rho < 1$, the initial condition is nondiffuse: $\mathbf{Q}_1 = \frac{\sigma^2}{-2\ln(\rho)}\mathbf{I}_2$. For $\rho = 1$, the initial condition is fully diffuse.

The first element of $\boldsymbol{\alpha}_t$ corresponds to the needed cycle, and the second element is an auxiliary quantity. You can define a cycle term based on this state as follows:

```
state cycleState(1) type=cycle(CT)  ...;
component cycle = cycleState[1];
```

The CT option must be included in the use of TYPE=CYCLE.

Models with Dependent Lags (Experimental)

Many useful time series models relate the present value of a response variable to its own lagged values and, in the multivariate case, the lagged values of other response variables in the model. In the SSM procedure, you can use the DEPLAG statement to specify the terms in the model that involve lagged response variables. These models apply only to the regular data type. This section describes the state space form of such models; for more information, see Harvey (1989, sec. 7.1.1). As an illustration, consider the following model, where the q-dimensional coefficient matrices $\mathbf{\Phi}_1$ and $\mathbf{\Phi}_2$ are either fully or partially known:

$$\begin{aligned} \mathbf{Y}_t &= \mathbf{\Phi}_1\mathbf{Y}_{t-1} + \mathbf{\Phi}_2\mathbf{Y}_{t-2} + \mathbf{Z}_t\boldsymbol{\alpha}_t + \mathbf{X}_t\boldsymbol{\beta} + \boldsymbol{\epsilon}_t \\ \boldsymbol{\alpha}_{t+1} &= \mathbf{T}_t\boldsymbol{\alpha}_t + \mathbf{W}_{t+1}\boldsymbol{\gamma} + \mathbf{c}_{t+1} + \boldsymbol{\eta}_{t+1} \\ \boldsymbol{\alpha}_1 &= \mathbf{c}_1 + \mathbf{A}_1\boldsymbol{\delta} + \boldsymbol{\eta}_1 \end{aligned}$$

Except for the presence of the terms that involve lagged response vectors ($\mathbf{\Phi}_1\mathbf{Y}_{t-1}$ and $\mathbf{\Phi}_2\mathbf{Y}_{t-2}$) in the observation equation, the form of this model is the same as the standard state space form that is described in the section "State Space Model and Notation" on page 2389. It turns out that this model can be expressed in the standard state space form by suitably enlarging the latent vectors in the state equation and by appropriately reorganizing the system matrices. The enlarged latent vectors and the corresponding system matrices are distinguished by the presence of dagger (†) as a superscript in the following reformulated model,

$$\begin{aligned} \mathbf{Y}_t &= \mathbf{Z}_t^{\dagger}\boldsymbol{\alpha}_t^{\dagger} \\ \boldsymbol{\alpha}_{t+1}^{\dagger} &= \mathbf{T}_t^{\dagger}\boldsymbol{\alpha}_t^{\dagger} + \mathbf{W}_{t+1}^{\dagger}\boldsymbol{\gamma}^{\dagger} + \mathbf{c}_{t+1}^{\dagger} + \boldsymbol{\eta}_{t+1}^{\dagger} \\ \boldsymbol{\alpha}_1^{\dagger} &= \mathbf{c}_1^{\dagger} + \mathbf{A}_1^{\dagger}\boldsymbol{\delta}^{\dagger} + \boldsymbol{\eta}_1^{\dagger} \end{aligned}$$

where the following conditions are true (column vectors are displayed horizontally to save space):

- The enlarged state vector ($\boldsymbol{\alpha}_t^{\dagger}$) is formed by vertically stacking the old state vector ($\boldsymbol{\alpha}_t$), the observation disturbance vector ($\boldsymbol{\epsilon}_t$), and the present and lagged response vectors ($\mathbf{Y}_t$ and $\mathbf{Y}_{t-1}$, respectively). That is, $\boldsymbol{\alpha}_t^{\dagger} = [\boldsymbol{\alpha}_t \;\; \boldsymbol{\epsilon}_t \;\; \mathbf{Y}_t \;\; \mathbf{Y}_{t-1}]$. Because $\boldsymbol{\alpha}_t$ is m-dimensional and $\boldsymbol{\epsilon}_t$, $\mathbf{Y}_t$, and $\mathbf{Y}_{t-1}$ are q-dimensional, the dimension of $\boldsymbol{\alpha}_t^{\dagger}$ is $m^{\dagger} = (m + 3 * q)$.
- The new state regression vector ($\boldsymbol{\gamma}^{\dagger}$) is formed by vertically stacking the old state regression vector ($\boldsymbol{\gamma}$) and the observation equation regression vector ($\boldsymbol{\beta}$). That is, $\boldsymbol{\gamma}^{\dagger} = [\boldsymbol{\gamma} \;\; \boldsymbol{\beta}]$.
- The enlarged disturbance vector ($\boldsymbol{\eta}_t^{\dagger}$) is formed by vertically stacking the old state disturbance vector ($\boldsymbol{\eta}_t$), the observation disturbance vector ($\boldsymbol{\epsilon}_t$), the vector sum ($\mathbf{Z}_t\boldsymbol{\eta}_t + \boldsymbol{\epsilon}_t$), and filling the rest of the vector with zeros. That is, $\boldsymbol{\eta}_t^{\dagger} = [\boldsymbol{\eta}_t \;\; \boldsymbol{\epsilon}_t \;\; (\mathbf{Z}_t\boldsymbol{\eta}_t + \boldsymbol{\epsilon}_t) \;\; \mathbf{0}]$.
- The deterministic vector $\mathbf{c}_{t+1}^{\dagger} = [\mathbf{c}_{t+1} \;\; \mathbf{0} \;\; \mathbf{Z}_{t+1}\mathbf{c}_{t+1} \;\; \mathbf{0}]$.
- The last $2q$ elements of the initial state vector ($\boldsymbol{\alpha}_1^{\dagger}$), which correspond to $\mathbf{Y}_1$, and $\mathbf{Y}_0$, are taken to be diffuse (which means that the diffuse vector $\boldsymbol{\delta}^{\dagger}$ has $2q$ additional elements compared to $\boldsymbol{\delta}$).

The new system matrices can be described in blockwise form in terms of the old system matrices as follows:

- The $q \times (m + 3 * q)$-dimensional $\mathbf{Z}_t^{\dagger} = [\mathbf{0}\ \mathbf{0}\ \mathbf{I}\ \mathbf{0}]$, where $\mathbf{0}$ is either a $q \times m$-dimensional or $q \times q$-dimensional matrix of zeros and $\mathbf{I}$ is a q-dimensional identity matrix.

- The $m^{\dagger} \times m^{\dagger}$ matrices $\mathbf{T}_t^{\dagger}$ (transition matrix) and $\mathbf{Q}_t^{\dagger}$ (covariance of $\boldsymbol{\eta}_{t+1}^{\dagger}$) are

$$\mathbf{T}_t^{\dagger} = \begin{bmatrix} \mathbf{T}_t & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{Z}_{t+1}\mathbf{T}_t & \mathbf{0} & \boldsymbol{\Phi}_1 & \boldsymbol{\Phi}_2 \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \end{bmatrix} \quad \text{and} \quad \mathbf{Q}_t^{\dagger} = \begin{bmatrix} \mathbf{Q}_t & \mathbf{0} & \mathbf{Q}_t\mathbf{Z}_{t+1}^{'} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_{t+1} & \boldsymbol{\Sigma}_{t+1} & \mathbf{0} \\ \mathbf{Z}_{t+1}\mathbf{Q}_t & \boldsymbol{\Sigma}_{t+1} & (\mathbf{Z}_{t+1}\mathbf{Q}_t\mathbf{Z}_{t+1}^{'} + \boldsymbol{\Sigma}_{t+1}) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

  where $\boldsymbol{\Sigma}_t$ denotes the covariance matrix (which is diagonal by design) of the observation error vector $\boldsymbol{\epsilon}_t$. Recall that the system matrices in the transition equation can depend on both t and $t + 1$ even if the subscripts of $\mathbf{T}$ and $\mathbf{Q}$ show dependence on t alone.

- The $m^{\dagger} \times (k + g)$ matrix $\mathbf{W}_t^{\dagger}$ is

$$\mathbf{W}_{t+1}^{\dagger} = \begin{bmatrix} \mathbf{W}_{t+1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{Z}_{t+1}\mathbf{W}_{t+1} & \mathbf{X}_{t+1} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$$

This state space form can be easily extended to account for higher-order lags.

Models that contain dependent lag terms must be used with care. Because the SSM procedure does not impose any special constraints on the lag coefficients (the elements of coefficient matrices $\boldsymbol{\Phi}_1, \boldsymbol{\Phi}_2$, and so on), the resulting models can often be explosive. For an example of a model with lagged response variables, see Example 34.13.

PROC SSM and PROC UCM (see Chapter 41, "The UCM Procedure") handle models that contain dependent lags in essentially the same way. However, there is one difference: if the model parameter vector contains unknown lag parameters, PROC UCM parameters are estimated by optimizing the nondiffuse part of the likelihood whereas PROC SSM continues to use the full diffuse likelihood for parameter estimation.

Covariance Parameterization

The covariance matrices specified by the COV and COV1 options in the STATE statement must be positive semidefinite. When these matrices are of general form and are not user-specified, they are internally parameterized by their Cholesky root. Suppose that Σ, an $m \times m$ positive semidefinite matrix of rank r, is such a covariance matrix. Then, Σ can always be written as

$$\Sigma = RR^{'}$$

where the (generalized) Cholesky root, R, is an $m \times r$ lower triangular matrix with nonnegative diagonal elements (that is, $R[i, j] = 0$ if $j > i$ and $R[i, i] \geq 0,\ 1 \leq i \leq r$). The SSM procedure parameterizes Σ by the elements of its Cholesky root, which adds $r * (r + 1)/2 + r * (m - r)$ elements to the parameter vector $\boldsymbol{\theta}$.

Missing Values

For a variety of reasons the data might contain missing response and predictor values. Before starting the analysis of a particular BY group, SSM procedure makes an internal copy of the data. The actual analysis is done by using this copy. The data in the copy are first examined for missing values in the response, predictor, and the ID variables. No missing values are permitted in the ID variable (if it is specified). If all the missing values are associated with only the response variables, then the internal copy of the data is not altered. However, if any of the predictors in the observation equation—the elements of $\mathbf{X}$ matrix—are found to contain missing values, the internal copy of the data is altered as follows: any missing predictor value is replaced by 0, and the response values that are dependent on that predictor in the corresponding row are set to missing. These missing response values are called the *induced missing values*. The reported analysis is based on the (possibly altered) internal copy of the BY group.

Missing values are not permitted in any of the other system matrices that define the state space model. In particular, missing values are not permitted in $\mathbf{Z}$, $\mathbf{T}$, $\mathbf{W}$, and $\mathbf{Q}$ matrices. In some cases the elements of these matrices depend on the data values. In such cases, care must be taken to ensure that these data values are not missing.

Computational Issues

A Well-Behaved Model

The model defined by the state space model equations (see the section "State Space Model and Notation" on page 2389) is very general. This generality is quite useful because it encompasses a wide variety of data generation processes. On the other hand, it also makes it easy to specify overly complex and numerically unstable models. If a suitable model is not already known and you are in the early phases of modeling, it is important to start with models that are relatively simple and well-behaved from the numerical standpoint. From the numerical and statistical considerations, two aspects of model formulation are particularly important: identifiability and numerical stability. A model is identifiable if the observed data has a distinct probability distribution for each admissible parameter vector. Unless proper care is taken, it is easy to specify an unidentifiable state space model. Similarly, predictions according to some types of state space models can display explosive growth or wild oscillations. This behavior is primarily governed by the transition matrix $\mathbf{T}$ (or $\mathbf{T}_t$ in the time-varying case). Unidentifiable models can run into difficulties during parameter estimation, and explosive growth (and wild oscillation) causes numerical problems associated with finite-precision arithmetic. Unfortunately, no simple identifiability check is available for a general state space model, and it is difficult to decide at the outset whether a specified model might suffer from numerical instability. See Harvey (1989, chap. 4, sec. 4) for a discussion of identifiability issues, and see Harvey (1989, chap. 3, sec. 3) for a discussion of the stability properties of time-invariant state space models. The following guidelines are likely to result in models that are identifiable and numerically stable:

- Build models by composing submodels that are known to be well-behaved. The predefined models provided by the SSM procedure are good submodel candidates (see the sections "Predefined Trend Models" on page 2403 and "Predefined Structural Models" on page 2407).
- Pay careful attention to the way the variety of system matrices are defined. The behavior of their elements, as functions of model parameters and other variables, must be well-understood. If these elements are defined by using DATA steps, you can validate their behavior by running these DATA steps outside of the SSM procedure. In particular, note the following:

- The transition matrix $\mathbf{T}$ (or $\mathbf{T}_t$ in the time-varying case) determines the explosiveness characteristics of the model; it must be well-behaved for all parameters.
- The disturbance covariances $\mathbf{Q}_t$ must be positive semidefinite for all parameters.
- If the system matrices in the state equation, such as the transition matrix $\mathbf{T}_t$ or the disturbance covariance $\mathbf{Q}_t$, are time-varying and the data contain replicate observations (observations with the same ID value), check that the elements of these matrices do not vary during replicate observations. This follows from the fact that the underlying state does not vary during replications (see the state equation in the section "State Space Model and Notation" on page 2389 and the section "Types of Sequence Data" on page 2391).

Convergence Problems

As explained in the section "Likelihood Computation and Model Fitting Phase" on page 2398, the model parameters are estimated by nonlinear optimization of the likelihood. This process is not guaranteed to succeed. For some data sets, the optimization algorithm can fail to converge. Nonconvergence can result from a number of causes, including flat or ridged likelihood surfaces and ill-conditioned data. It is also possible for the algorithm to converge to a point that is not the global optimum of the likelihood.

If you experience convergence problems, consider the following:

- Data that are extremely large or extremely small can adversely affect results because of the internal tolerances used during the filtering steps of the likelihood calculation. Rescaling the data can improve stability.
- Whenever possible, parameterize the disturbance variances in the model on the exponential scale. For illustrations of parameterizing disturbance variances in this manner, see Example 34.12 and Example 34.14.
- Examine your model for redundancies in the included components and regressors. The components or regressors that are nearly collinear to each other can cause the optimization process to become unstable.
- Lack of convergence can indicate model misspecification such as unidentifiable model or a violation of the normality assumption.

Computer Resource Requirements

The computing resources required for the SSM procedure depend on several factors. The memory requirement for the procedure is largely dependent on the number of observations to be processed and the size of the state vector underlying the specified model. If n denotes the sample size and m denotes the size of the state vector, the memory requirement for the smoothing phase of the Kalman filter is of the order of $6 \times 8 \times n \times m^2$ bytes, ignoring the lower-order terms. If the smoothed component estimates are not needed, then the memory requirement is of the order of $6 \times 8 \times (m^2 + n)$ bytes. Besides m and n, the computing time for the parameter estimation depends on the size of the parameter vector $\boldsymbol{\theta}$ and how many likelihood evaluations are needed to reach the optimum.

Displayed Output

The default printed output produced by the SSM procedure contains the following information:

- brief information about the input data set, including the data set name and label
- summary statistics for the response variables in the model, including the names of the variables, the total number of observations and the number of missing observations, the smallest and largest measurements, and the mean and standard deviation
- information about the index variable, including the index value of the first and the last observation, the maximum difference between the successive index values, the number of distinct index values, and the categorization of the data into regular, regular with replication, or irregular types
- estimates of the regression parameters if the model contains any predictors, including their standard errors, t statistics, and p-values
- convergence status of the likelihood optimization process if any parameters are estimated
- estimates of the free parameters at the end of the model-fitting phase, including the parameter estimates and their approximate standard errors
- the likelihood-based goodness-of-fit statistics, including the full likelihood, the sum of squares of residuals normalized by their standard errors, and the information criteria: AIC, AICC, HQIC, BIC, and CAIC
- summary of most significant additive outliers

ODS Table Names

The SSM procedure assigns a name to each table it creates. You can use these names to refer to the table when you use the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in Table 34.9.

Table 34.9 ODS Tables Produced by PROC SSM

ODS Table Name	Description	Statement	Option
Tables That Summarize the Model Information			
ModelSummary	Summary information about the underlying state space model		Default
IdInformation	Summary information about the ID variable		Default
ResponseInfo	Summary information about the response variables		Default
StateSummary	Summary information about the model state vector	PROC SSM	STATEINFO

Table 34.9 *continued*

ODS Table Name	Description	Statement	Option
DiffuseStateSummary	Summary information about the diffuse initial state	PROC SSM	STATEINFO
Tables Related to Model Parameters and the Likelihood			
ConvergenceStatus	Convergence status of the estimation process		Default
RegressionEstimates	Estimates of the regression parameters	MODEL	Default
StateRegressionEstimates	Estimates of the state regression parameters	STATE	W
FixedStateEstimates	Estimates of time-invariant, non-stochastic state subsections		Default
NamedParameterEstimates	Estimates of the parameters specified in the PARMS statement	PARMS	Default
ParameterEstimates	Estimates of the unknown elements in the model system matrices		Default
DisturbanceCovariance	Estimate of the disturbance covariance	STATE	PRINT=COV
InitialCovariance	Estimate of the initial state covariance	STATE	PRINT=COV1
ARCoefficient	Estimate of the autoregressive coefficient matrix	STATE	PRINT=AR
MACoefficient	Estimate of the moving-average coefficient matrix	STATE	PRINT=MA
TransitionMatrix	Estimate of the state transition matrix	STATE	PRINT=T
FitSummary	Summary of the likelihood-based-fit-statistics		Default
InformationCriteria	Likelihood-based information criteria		Default
Tables Related to Series and Component Forecasts			
Forecasts	Series forecasts	MODEL	PRINT=FILTER
SmoothedResponse	Smoothed series values	MODEL	PRINT=SMOOTH
FilteredComponent	Component forecasts	COMPONENT	PRINT=FILTER
SmoothedComponent	Smoothed component	COMPONENT	PRINT=SMOOTH
Tables Related to Outlier Detection and Model Quality			
AOSummary	Summary of additive outliers	Default	Default

Table 34.9 *continued*

ODS Table Name	Description	Statement	Option
ElementTrendBreakSummary	Elementwise trend break summary	TREND	CHECKBREAK
OverallTrendBreakSummary	Overall trend break summary	TREND	CHECKBREAK(OVERALL)
ElementStatedBreakSummary	Elementwise state break summary	STATE	CHECKBREAK
OverallStateBreakSummary	Overall state break summary	STATE	CHECKBREAK(OVERALL)
MaximalShockSummary	Summary of maximal state shocks	OUTPUT	MAXSHOCK
PRESS	Prediction error sum of squares	OUTPUT	PRESS

ODS Graph Names

You can refer to every graph produced through ODS Graphics with a name. The names of the graphs that PROC SSM generates are listed in Table 34.10, along with the required statements and options.

Table 34.10 ODS Graphs Produced by PROC SSM

ODS Graph Name	Description	Statement	Option
Graphs for One-Step-Ahead Residual Analysis			
ResidualNormalityPlot	Normality check	PROC SSM	PLOTS=RESIDUAL(NORMAL)
ResidualHistogram	Residual histogram	PROC SSM	PLOTS(UNPACK)=RESIDUAL
ResidualQQPlot	Residual QQ Plot	PROC SSM	PLOTS(UNPACK)=RESIDUAL
StdResidualPlot	Time series plot of standardized residuals	PROC SSM	Default
Graphs Related to Outlier Detection and Structural Break			
PredErrorNormalityPlot	Normality check	PROC SSM	PLOTS=AO(NORMAL)
PredErrorHistogram	Prediction error histogram	PROC SSM	PLOTS(UNPACK)=AO
PredErrorQQPlot	Prediction error QQ plot	PROC SSM	PLOTS(UNPACK)=AO
StdPredErrorPlot	Time series plot of standardized additive-outlier statistics	PROC SSM	PLOTS=AO(STD)
MaximalShockPlot	Time series plot of maximal state shock chi-square statistics	PROC SSM	PLOTS=MAXSHOCK

OUT= Data Set

You can use the OUT= option in the OUTPUT statement to store the series and component forecasts that are produced by PROC SSM. Which columns are included in the data set depends on the model specification. The model can have one or more response variables, a variety of components that appear in the MODEL statement, and components specified by the EVAL statement. The OUT= data set contains the one-step-ahead and full-sample estimates of the response variables, and all these components.

The following list describes the columns of the data set:

- the BY variables
- the ID variable, if specified by the ID statement
- Obs, a variable that contains the observation number
- the response series (more than one in the multivariate case)
- the following columns associated with the response series (the wildcard * is substituted by the name of one of the response variables):
  - FORECAST_* contains the one-step-ahead predicted values. and the multistep forecasts of the response series.
  - RESIDUAL_* contains the difference between the actual and forecast values.
  - StdErr_* contains the standard error of prediction.
  - Lower_* and Upper_* contain the lower and upper forecast confidence limits.
  - Smoothed_* contains the smoothed values of the response variable.
  - StdErr_Smoothed_* contains standard errors of the smoothed values of the response variable.
  - AO_* contains the additive outlier estimate.
  - StdErr_AO_* contains the standard error of the additive outlier estimate.
- the following columns associated with the components (the wildcard * is substituted by the name of one of the components):
  - FORECAST_* contains the one-step-ahead predicted values and the multistep forecasts of the component.
  - StdErr_* contains the standard error of prediction.
  - Smoothed_* contains the smoothed values of the component.
  - StdErr_Smoothed_* contains standard errors of the smoothed values of the component.
  - Smoothed_Lower_* and Smoothed_Upper_* contain the lower and upper confidence limits of the smoothed component.
- the maximal state shock chi-square statistics at distinct time points (this column is present only if the MAXSHOCK option is used in the OUTPUT statement)

Confidence limits are not produced for the smoothed series values or for the component forecasts; they are produced for the smoothed components.

Examples: SSM Procedure

Example 34.1: Bivariate Basic Structural Model

This example illustrates how you can use the SSM procedure to analyze a bivariate time series. The following data set contains two variables, f_KSI and r_KSI, which are measured quarterly, starting the first quarter of 1969. The variable f_KSI represents the quarterly average of the log of the monthly totals of the front-seat passengers killed or seriously injured during the car accidents, and r_KSI represents a similar number for the rear-seat passengers. The data set has been extended at the end with eight missing values, which represent four quarters, to cause the SSM procedure to produce model forecasts for this span.

```
data seatBelt;
input f_KSI r_KSI @@;
label f_KSI = "Front Seat Passengers Injured--log scale";
label r_KSI = "Rear Seat Passengers Injured--log scale";
date = intnx( 'quarter', '1jan1969'd, _n_-1 );
format date YYQS.;
datalines;
    6.72417 5.64654  6.81728 6.06123  6.92382 6.18190
    6.92375 6.07763  6.84975 5.78544  6.81836 6.04644
    7.00942 6.30167  7.09329 6.14476  6.78554 5.78212
    6.86323 6.09520  6.99369 6.29507  6.98344 6.06194
    6.81499 5.81249  6.92997 6.10534  6.96356 6.21298
    7.02296 6.15261  6.76466 5.77967  6.95563 6.18993
    7.02016 6.40524  6.87849 6.06308  6.55966 5.66084
    6.73627 6.02395  6.91553 6.25736  6.83576 6.03535
    6.52075 5.76028  6.59860 5.91208  6.70597 6.08029
    6.75110 5.98833  6.53117 5.67676  6.52718 5.90572
    6.65963 6.01003  6.76869 5.93226  6.44483 5.55616
    6.62063 5.82533  6.72938 6.04531  6.82182 5.98277
    6.64134 5.76540  6.66762 5.91378  6.83524 6.13387
    6.81594 5.97907  6.60761 5.66838  6.62985 5.88151
    6.76963 6.06895  6.79927 6.01991  6.52728 5.69113
    6.60666 5.92841  6.72242 6.03111  6.76228 5.93898
    6.54290 5.72538  6.62469 5.92028  6.73415 6.11880
    6.74094 5.98009  6.46418 5.63517  6.61537 5.96040
    6.76185 6.15613  6.79546 6.04152  6.21529 5.70139
    6.27565 5.92508  6.40771 6.13903  6.37293 5.96883
    6.16445 5.77021  6.31242 6.05267  6.44414 6.15806
    6.53678 6.13404  . . . . . .
run;
```

These data have been analyzed in Durbin and Koopman (2001, chap. 9, sec. 3). The analysis presented here is similar. To simplify the illustration, the monthly data have been converted to quarterly data and two predictors (the number of kilometers traveled and the real price of petrol) are excluded from the analysis. You can also use PROC SSM to carry out the more elaborate analysis in Durbin and Koopman (2001).

One of the original reasons for studying these data was to assess the effect on f_KSI of the enactment of a seat-belt law in February 1983 that compelled the front seat passengers to wear seat belts. A simple graphical inspection of the data (not shown here) reveals that f_KSI and r_KSI do not show a pronounced upward or downward trend but do show seasonal variation, and that these two series seem to move together. Additional inspection also shows that the seasonal effect is relatively stable throughout the data span. These considerations suggest the following model for $\mathbf{y}$ = (f_KSI, r_KSI):

$$\mathbf{y}_t = \begin{pmatrix} X_t \\ 0 \end{pmatrix} \beta + \boldsymbol{\mu}_t + \boldsymbol{\zeta}_t + \boldsymbol{\xi}_t$$

All the terms on the right-hand side of this equation are assumed to be statistically independent. These terms are as follows:

- The predictor X_t (defined as Q1_83_Shift later in the program) denotes a variable that is 0 before the first quarter of 1983, and 1 thereafter. X_t is supposed to affect only f_KSI (the first element of $\mathbf{y}$); it represents the enactment of the seat-belt law of 1983.
- $\boldsymbol{\mu}_t$ denotes a bivariate random walk. It is supposed to capture the slowly changing level of the vector $\mathbf{y}_t$. To capture the fact that f_KSI and r_KSI move together (that is, they are co-integrated), the covariance of the disturbance term of this random walk is assumed to be of lower than full rank.
- $\boldsymbol{\zeta}_t$ denotes a bivariate trigonometric seasonal term. In this model, it is taken to be fixed (that is, the seasonal effects do not change over time).
- $\boldsymbol{\xi}_t$ denotes a bivariate white noise term, which captures the residual variation that is unexplained by the other terms in the model.

The preceding model is an example of a (bivariate) basic structural model (BSM). The following statements specify and fit this model to f_KSI and r_KSI:

```
proc ssm data=seatBelt stateinfo;
   id date interval=quarter;
   Q1_83_Shift = (date >= '1jan1983'd);
   state error(2) type=WN cov(g) print=cov;
   component wn1 = error[1];
   component wn2 = error[2];
   state level(2) type=RW cov(rank=1) print=cov;
   component rw1 = level[1];
   component rw2 = level[2];
   state season(2) type=season(length=4);
   component s1 = season[1];
   component s2 = season[2];
   model f_KSI = Q1_83_Shift rw1 s1  wn1 / print=(smooth);
   model r_KSI = rw2 s2 wn2;
   eval f_KSI_sa = rw1 + Q1_83_Shift;
   output out=For1;
run;
```

The PROC SSM statement specifies the input data set, seatBelt. The use of the STATEINFO option in the PROC SSM statement produces additional information about the model state vector and its diffuse initial state. The optional ID statement specifies an index variable, date. The INTERVAL=QUARTER option in the ID statement indicates that the measurements were collected on a quarterly basis. Next, a programming

statement defines Q1_83_Shift, the predictor that represents the enactment of the seat-belt law of 1983. It is used later in the MODEL statement for f_KSI. Separate STATE statements specify the terms $\boldsymbol{\mu}_t$, $\boldsymbol{\zeta}_t$, and $\boldsymbol{\xi}_t$ because they are statistically independent. Each model that governs them (white noise for $\boldsymbol{\xi}_t$, random walk for $\boldsymbol{\mu}_t$, and trigonometric seasonal for $\boldsymbol{\zeta}_t$) can be specified by using the TYPE= option of the STATE statement. When you use the TYPE= option, you can use the COV option to specify the information about the disturbance covariance in the state transition equation. The other details, such as the transition matrix specification and the specification of $\mathbf{A}_1$ in the initial condition, are inferred from the TYPE= option. The use of PRINT=COV in the STATE statement causes the estimated disturbance covariance to be printed. For $\boldsymbol{\xi}_t$ (a white noise), $\mathbf{A}_1$ is zero and $\mathbf{Q}_t = \mathbf{Q}$ for all $t \geq 1$, where $\mathbf{Q}$ is specified by the COV option. For $\boldsymbol{\mu}_t$ and $\boldsymbol{\zeta}_t$ the initial condition is fully diffuse—that is, $\mathbf{A}_1$ is an identity matrix of appropriate order and $\mathbf{Q}_1 = 0$. The total diffuse dimension of this model, $(d + k)$, is $9 = 8 + 1$ as a result of one predictor, Q1_83_Shift, and two fully diffuse state subsections, $\boldsymbol{\mu}_t$ and $\boldsymbol{\zeta}_t$. The components in the model are defined by suitable linear combinations of these different state subsections. The program statements define the model as follows:

- **`state error(2) type=WN cov(g);`** defines $\boldsymbol{\xi}_t$ as a two-dimensional white noise, named error, with the covariance of general form. Then two COMPONENT statements define wn1 and wn2 as the first and second elements of error, respectively.

- **`state level(2) type=RW cov(rank=1);`** defines $\boldsymbol{\mu}_t$ as a two-dimensional random walk, named level, with covariance of general form whose rank is restricted to 1. Then two COMPONENT statements define rw1 and rw2 as the first and second elements of level, respectively.

- **`state season(2) type=season(length=4);`** defines $\boldsymbol{\zeta}_t$ as a two-dimensional trigonometric seasonal of season length 4, named season, with zero covariance—signified by the absence of the COV option. Then two COMPONENT statements define s1 and s2 as appropriate linear combinations of season so that s1 represents the seasonal for f_KSI and s2 represents the seasonal for r_KSI. Because TYPE=SEASON in the STATE statement, the COMPONENT statement appropriately interprets **`component s1 = season[1];`** as s1 being a dot product: $(1\ 0\ 0\ 0\ 1\ 0) * \text{season}$. See the section "Multivariate Season" on page 2410 for more information.

- **`model f_KSI = Q1_83_Shift rw1 s1 wn1;`** defines the model for f_KSI, and **`model r_KSI = rw2 s2 wn2;`** defines the model for r_KSI.

The SSM procedure fits the model and reports the parameter estimates, their approximate standard errors, and the likelihood-based goodness-of-fit measures by default. In order to output the one-step-ahead and full-sample estimates of the components in the model, you can either use the PRINT= options in the MODEL statement and the respective COMPONENT statements or you can specify an output data set in the OUTPUT statement. In addition, you can use the EVAL statement to define specific linear combinations of the underlying state that should also be estimated. The statement **`eval f_KSI_sa = rw1 + Q1_83_Shift;`** is an example of one such linear combination. It defines f_KSI_sa, a linear combination that represents the seasonal adjustment of f_KSI. The output data set, For1 (named in the OUTPUT statement) contains estimates of all the model components in addition to the estimate of f_KSI_sa.

The model summary table, shown in Output 34.1.1, provides basic model information, such as the dimension of the underlying state equation ($m = 10$), the diffuse dimension of the model ($(d + k) = 9$), and the number of parameters (5) in the model parameter vector $\boldsymbol{\theta}$.

Output 34.1.1 Bivariate Basic Structural Model

The SSM Procedure

Model Summary	
Model Property	**Value**
Number of Model Equations	2
State Dimension	10
Dimension of the Diffuse Initial Condition	9
Number of Parameters	5

Additional details about the role of different components in forming the model state and its diffuse initial condition are shown in Output 34.1.2 and Output 34.1.3. They show that the 10-dimensional model state vector is made up of subsections that are associated with error and level (each of dimension 2) and season (of dimension 6). Similarly, the nine-dimensional diffuse vector in the initial condition is made up of subsections that correspond to level, season, and the regression variable, Q1_83_Shift. Note that error does not contribute to the diffuse initial vector because it has a fully nondiffuse initial state.

Output 34.1.2 Bivariate Basic Structural Model State Vector Summary

State Vector Composition	
Subsection	**Dimension**
error	2
level	2
season	6

Output 34.1.3 Bivariate Basic Structural Model Initial Diffuse State Vector Summary

Diffuse Initial State Composition (Including Regressors)	
Subsection	**Dimension**
level	2
season	6
Q1_83_Shift	1

The index variable information is shown in Output 34.1.4.

Output 34.1.4 Index Variable Information

ID Variable Information					
Name	**Start**	**End**	**Max Delta**	**NDistinct**	**Type**
date	1969:1	1985:4	1	68	Regular

Output 34.1.5 provides simple summary information about the response variables. It shows that f_KSI and r_KSI have four missing values each and no induced missing values because the predictor in the model, Q1_83_Shift, has no missing values.

Output 34.1.5 Response Variable Summary

Response Variable Information

Name	Number of Observations Total	Missing	Induced Missing	Minimum	Maximum	Mean	Std Deviation
f_KSI	68	4	0	6.16	7.09	6.71	0.206
r_KSI	68	4	0	5.56	6.41	5.97	0.186

The regression coefficient of Q1_83_Shift, shown in Output 34.1.6, is negative and is statistically significant. This is consistent with the expected drop in f_KSI after the enactment of the seat-belt law.

Output 34.1.6 Regression Coefficient of Q1_83_Shift

Regression Parameter Estimates

Response Variable	Regression Variable	Estimate	Standard Error	t Value	Pr > |t|
f_KSI	Q1_83_Shift	-0.408	0.0259	-15.74	<.0001

Output 34.1.7 shows the estimates of the elements of $\boldsymbol{\theta}$. The five parameters in $\boldsymbol{\theta}$ correspond to unknown elements that are associated with the covariance matrices in the specifications of error and level. Whenever a covariance specification is of a general form and is not defined by a user-specified variable list, it is internally parameterized as a product of its Cholesky root: $\mathrm{Cov} = \mathrm{Root}\ \mathrm{Root}'$. This ensures that the resulting covariance is positive semidefinite. The Cholesky root is constrained to be lower triangular, with positive diagonal elements. If rank constraints (such as the rank-one constraint on the covariance in the specification of level) are imposed, the number of free parameters in the Cholesky factor is reduced appropriately. See the section "Covariance Parameterization" on page 2414 for more information. In view of these considerations, the five parameters in $\boldsymbol{\theta}$ are a result of three parameters from the Cholesky root of error and two parameters that are associated with the Cholesky root of level.

Output 34.1.7 Parameter Estimates

Model Parameter Estimates

Component	Type	Parameter	Estimate	Standard Error	t Value
error	Disturbance Covariance	RootCov[1, 1]	0.0361	0.00736	4.91
error	Disturbance Covariance	RootCov[2, 1]	0.0338	0.01131	2.99
error	Disturbance Covariance	RootCov[2, 2]	0.0462	0.00470	9.84
level	Disturbance Covariance	RootCov[1, 1]	0.0375	0.00843	4.45
level	Disturbance Covariance	RootCov[2, 1]	0.0223	0.00569	3.92

Output 34.1.8 shows the resulting covariance estimate of error after multiplying the Cholesky factors.

Output 34.1.8 White Noise Covariance Estimate

Disturbance Covariance for error	Col1	Col2
Row1	0.001307	0.001222
Row2	0.001222	0.003277

Similarly, Output 34.1.9 shows the covariance estimate of level disturbance. Note that because of the rank-one constraint, the determinant of this matrix is 0.

Output 34.1.9 Covariance Estimate of the Random Walk Disturbance

Disturbance Covariance for level		
	Col1	Col2
Row1	0.001408	0.000837
Row2	0.000837	0.000497

Output 34.1.10 shows the likelihood computation summary. This table is produced by using the fitted model to carry out the filtering operation on the data. See the section "Likelihood Computation and Model Fitting Phase" on page 2398 for more information.

Output 34.1.10 Likelihood Computation Summary of the Fitted Model

Likelihood Computation Summary	
Statistic	Value
Nonmissing Response Values Used	128
Estimated Parameters	5
Initialized Diffuse State Elements	9
Normalized Residual Sum of Squares	119.00001
Diffuse Log Likelihood	166.15755
Profile Log Likelihood	199.91165

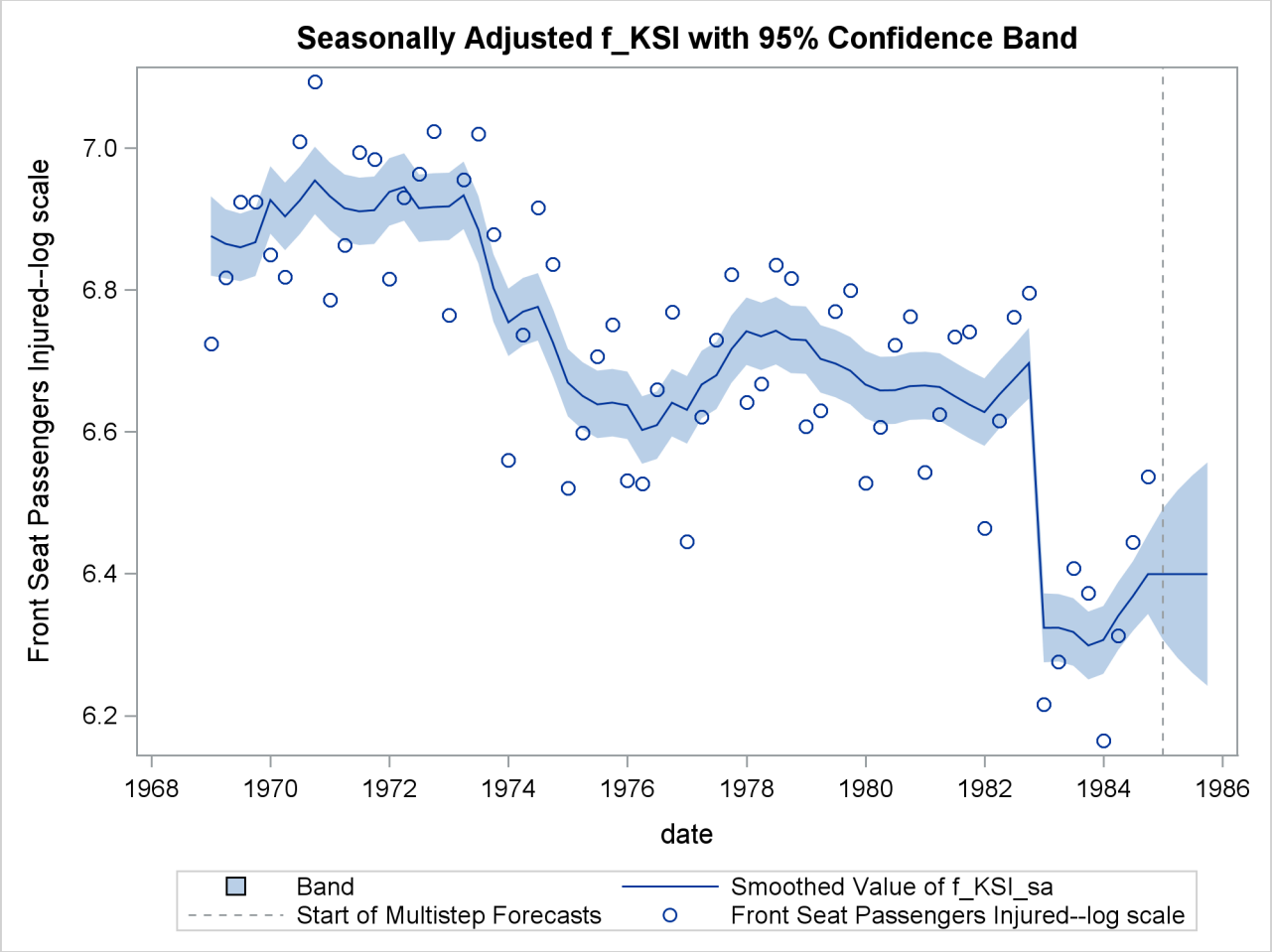
The output data set, For1, specified in the OUTPUT statement contains one-step-ahead and full-sample estimates of all the model components and the user-specified components (defined by the EVAL statement). Their standard errors and the upper and lower confidence limits (by default, 95%) are also produced.

The following statements use the For1 data set to produce a time series plot of the seasonally adjusted f_KSI:

```
proc sgplot data=For1;
   title "Seasonally Adjusted f_KSI with 95% Confidence Band";
   band x=date lower=smoothed_lower_f_KSI_sa
      upper=smoothed_upper_f_KSI_sa ;
   series x=date y=smoothed_f_KSI_sa;
   refline '1jan1985'd / axis=x lineattrs=(pattern=shortdash)
      LEGENDLABEL= "Start of Multistep Forecasts"
      name="Forecast Reference Line";
   scatter x=date y=f_KSI ;
run;
```

The generated plot is shown in Output 34.1.11.

Output 34.1.11 Plot of Seasonally Adjusted f_KSI



Example 34.2: Panel Data: Random-Effects and Autoregressive Models

This example shows how you can use the SSM procedure to specify and fit the two-way random-effects model and the autoregressive model to analyze a panel of time series. The fitting of dynamic panel model for such data is illustrated in Example 34.11. These (and a few other) model types can also be fitted by the PANEL procedure, a SAS/ETS procedure that is specially designed to efficiently handle the cross-sectional time series data. However, because of the differences in their model fitting algorithms, generally the parameter estimates and other fit statistics produced by the SSM and PANEL procedures do not match. The SSM procedure always uses the (restricted) maximum likelihood for parameter estimation. The estimation method used by the PANEL procedure depends on the model type and the particular estimation options.

The cross-sectional data, Cigar, that are used in the section "Getting Started: SSM Procedure" on page 2358 are reused in this example. The output shown here is less extensive than the output shown in that section. The main emphasis of this example is how you can specify the two-way random effects model and the autoregressive model in the SSM procedure.

According to the two-way random effects model, the cigarette sales, lsales, can be described by the following equation:

$$\text{lsales}_{i,t} = \boldsymbol{\mu} + \text{lprice}\,\boldsymbol{\beta}_1 + \text{lndi}\,\boldsymbol{\beta}_2 + \text{lpimin}\,\boldsymbol{\beta}_3 + \boldsymbol{\zeta}_i + \boldsymbol{\eta}_t + \boldsymbol{\epsilon}_{i,t}$$

This model represents lsales in region i and in year t as a sum of an overall intercept $\boldsymbol{\mu}$, the regression effects due to lprice, lndi, and lpimin, a zero-mean, random effect $\boldsymbol{\zeta}_i$ associated with region i, a zero-mean, random effect $\boldsymbol{\eta}_t$ associated with year t, and the observation noise $\boldsymbol{\epsilon}_{i,t}$. The region-specific random effects $\boldsymbol{\zeta}_i$ and the year-specific random effects $\boldsymbol{\eta}_t$ are assumed to be independent, Gaussian sequences with variances σ_ζ^2 and σ_η^2, respectively. In addition, they are assumed to be independent of the observation noise, which is also assumed to be a sequence of independent, zero-mean, Gaussian variables with variance σ_ϵ^2.

You can specify and fit this model by using the following statements:

```
proc ssm data=Cigar;
   id year interval=year;
   parms s2g/ lower=(1.e-6);
   array RegionArray{46} region1-region46;
   do i=1 to 46;
      RegionArray[i] = (region=i);
   end;
   /* region-specific random effects */
   state zeta(46) T(I) cov1(I)=(s2g);
   component regionEffect = zeta * (RegionArray);
   /* year-specific random effect */
   state eta(1) type=wn cov(D);
   component timeEffect = eta[1];
   irregular wn;
   intercept = 1.0;
   model lsales = intercept lprice lndi lpimin
       timeEffect regionEffect wn;
run;
```

The PARMS statement defines s2g, a parameter that is restricted to be positive and is used later as the variance parameter for the region effect. Similarly the 46-dimensional array, RegionArray, of region-specific dummy variables is defined to be used later. The state subsection zeta corresponds to $\boldsymbol{\zeta}$, which is the 46-dimensional vector of region-specific, zero-mean, random effects. The component regionEffect extracts the proper element of $\boldsymbol{\zeta}$ by using the array RegionArray. A constant column, intercept, is defined to be used later as an intercept term. The component timeEffect corresponds to η_t, and wn specifies the observation noise $\boldsymbol{\epsilon}_{it}$. Finally the MODEL statement defines the model. Some of the tables that are produced by running these statements are shown in Output 34.2.1 through Output 34.2.5.

The model summary, shown in Output 34.2.1, shows that the model is defined by one MODEL statement, the dimension of the underlying state vector is 47 (because $\boldsymbol{\zeta}$ is 46-dimensional and η_t is one-dimensional), the diffuse dimension is 4 (because of the four predictors in the model), and there are three parameters to be estimated.

Output 34.2.1 Two-Way Random-Effects Model: Model Summary

The SSM Procedure

Model Summary	
Model Property	**Value**
Number of Model Equations	1
State Dimension	47
Dimension of the Diffuse Initial Condition	4
Number of Parameters	3

Output 34.2.2 provides the likelihood information about the fitted model.

Output 34.2.2 Two-Way Random-Effects Model: Likelihood Summary

Likelihood Computation Summary	
Statistic	**Value**
Nonmissing Response Values Used	1380
Estimated Parameters	3
Initialized Diffuse State Elements	4
Normalized Residual Sum of Squares	1376.0001
Diffuse Log Likelihood	1459.0277
Profile Log Likelihood	1470.8628

Output 34.2.3 shows the regression estimates.

Output 34.2.3 Two-Way Random-Effects Model: Regression Estimates

Regression Parameter Estimates					
Response Variable	**Regression Variable**	**Estimate**	**Standard Error**	**t Value**	**Pr > |t|**
lsales	**intercept**	2.798	0.1136	24.62	<.0001
lsales	**lprice**	-0.903	0.0365	-24.73	<.0001
lsales	**lndi**	0.592	0.0246	24.08	<.0001
lsales	**lpimin**	0.127	0.0398	3.18	0.0015

The ML estimate of s2g, a parameter specified in the PARMS statement, is shown in Output 34.2.4. It corresponds to σ_ξ^2, the variance of the region effect.

Output 34.2.4 Two-Way Random-Effects Model: Estimate of σ_ξ^2

Estimates of Named Parameters			
Parameter	**Estimate**	**Standard Error**	**t Value**
s2g	0.0241	0.00512	4.70

Output 34.2.5 Variance Estimates of η_t and ϵ_{it}

Model Parameter Estimates

Component	Type	Parameter	Estimate	Standard Error	t Value
eta	Disturbance Covariance	Cov[1, 1]	0.000681	0.000264	2.58
wn	Irregular	Variance	0.005698	0.000224	25.40

The estimates of the other unknown parameters in the model are shown in Output 34.2.5. It shows the estimate of the variance of the irregular component wn and the estimate of the variance of the time effect η_t.

The remainder of this example describes how you can specify and fit the following first-order vector autoregessive model to the cigarette data:

$$\begin{aligned} \text{lsales}_{i,t} &= \boldsymbol{\mu} + \text{lprice}\,\boldsymbol{\beta}_1 + \text{lndi}\,\boldsymbol{\beta}_2 + \text{lpimin}\,\boldsymbol{\beta}_3 + \boldsymbol{\zeta}_t[i] \\ \boldsymbol{\zeta}_t &= \boldsymbol{\Phi}\,\boldsymbol{\zeta}_{t-1} + \boldsymbol{\eta}_t \end{aligned}$$

This model represents lsales in region i and in year t as a sum of an overall intercept $\boldsymbol{\mu}$, the regression effects due to lprice, lndi, and lpimin, and the ith element of a vector error term $\boldsymbol{\zeta}_t[i]$. The multidimensional error sequence $\boldsymbol{\zeta}_t$ is assumed to follow a first-order autoregression with a diagonal autoregressive coefficient matrix $\boldsymbol{\Phi}$ and with a multivariate, white noise sequence $\boldsymbol{\eta}_t$ as its disturbance sequence. The covariance matrix of $\boldsymbol{\eta}_t$, $\boldsymbol{\Sigma}$, is assumed to be dense. Note that the dimension of the vectors $\boldsymbol{\zeta}_t$ is the same as the number of cross-sections in the study (the number of regions in this example). Therefore, even for a relatively modest panel study, the total number of parameters to be estimated can get quite large. Therefore, in this example only the first three regions are considered in the analysis. The following statements specify and fit this model to the Cigar data set:

```
proc ssm data=Cigar;
   where region <= 3;
   id year interval=year;
   array RegionArray{3} region1-region3;
   do i=1 to 3;
      RegionArray[i] = (region=i);
   end;
   state zeta(3) type=varma(p(d)=1) cov(g) print=(ar cov);
   component eta = zeta*(RegionArray);
   intercept = 1.0;
   model lsales = intercept lprice lndi lpimin eta;
run;
```

The vectors $\boldsymbol{\zeta}_t$ are specified in the STATE statement. The TYPE= specification signifies that the three-dimensional state subsection, zeta, follows a vector AR(1) model with a diagonal transition matrix and a disturbance covariance of a general form. The PRINT=(AR COV) option causes the SSM procedure to print the estimated AR coefficient matrix, $\boldsymbol{\Phi}$, and the disturbance error covariance $\boldsymbol{\Sigma}$, respectively. The COMPONENT statement defines the appropriate error contribution (named eta), $\boldsymbol{\zeta}_t[i]$. Output 34.2.6 shows the estimated regression coefficients, Output 34.2.7 shows the estimate of $\boldsymbol{\Phi}$, and Output 34.2.8 shows the estimate of $\boldsymbol{\Sigma}$:

Output 34.2.6 Autoregressive Model: Regression Estimates

The SSM Procedure

Regression Parameter Estimates

Response Variable	Regression Variable	Estimate	Standard Error	t Value	Pr > |t|
lsales	intercept	3.576	0.3621	9.88	<.0001
lsales	lprice	-0.230	0.0832	-2.77	0.0056
lsales	lndi	0.218	0.0753	2.89	0.0039
lsales	lpimin	0.068	0.0996	0.68	0.4952

Output 34.2.7 Estimate of the AR Coefficient Φ

AR Coefficient Matrix for zeta

	Col1	Col2	Col3
Row1	0.943078	0	0
Row2	0	0.989301	0
Row3	0	0	0.888549

Output 34.2.8 Estimate of the Disturbance Covariance Σ

Disturbance Covariance for zeta

	Col1	Col2	Col3
Row1	0.000941	0.000445	0.000346
Row2	0.000445	0.00226	0.000242
Row3	0.000346	0.000242	0.000882

Example 34.3: Backcasting, Forecasting, and Interpolation

This example illustrates how you can do model-based extrapolation—backcasting, forecasting, or interpolation—of a response variable. All you need is to appropriately augment the input data set with the relevant ID and predictor information and assign missing values to the response variable in these places. The following DATA step creates one such augmented data set by using a well-known data set that contains recordings of the Nile River water level measured between the years 1871 and 1970. Suppose you want to backcast the Nile water level for two years before 1871, forecast it for two years after 1970, and interpolate its value for the year 1921—for illustration purposes, this value is assumed to be missing in the available data set.

```
data Nile;
   input level @@;
   year = intnx( 'year', '1jan1869'd, _n_-1 );
   format year year4.;
   if year = '1jan1921'd then level=.;
datalines;
. .
1120  1160  963  1210  1160  1160  813  1230   1370  1140
995   935   1110 994   1020  960   1180 799    958   1140
```

```
1100  1210  1150 1250  1260  1220  1030 1100   774   840
874   694   940  833   701   916   692  1020   1050  969
831   726   456  824   702   1120  1100 832    764   821
768   845   864  862   698   845   744  796    1040  759
781   865   845  944   984   897   822  1010   771   676
649   846   812  742   801   1040  860  874    848   890
744   749   838  1050  918   986   797  923    975   815
1020  906   901  1170  912   746   919  718    714   740
. .
;
```

It is also known that for this time span the Nile water level can be reasonably modeled as a sum of a random walk trend, a level shift in the year 1899, and the observation error. The following statements fit this model to the data:

```
proc ssm data=Nile;
    id year interval=year;
    shift1899 = ( year >= '1jan1899'd );
    trend rw(rw);
    irregular wn;
    model level = shift1899 RW wn  / print=smooth;
    output out=nileOut;
quit;
```

The model-based interpolated and extrapolated values of the Nile water level are shown in Output 34.3.1, which is produced by using the PRINT=SMOOTH option in the MODEL statement.

Output 34.3.1 Interpolated and Extrapolated Nile Water Level

The SSM Procedure

Full-Sample Prediction of Missing Values for level

Obs	ID	Estimate	Standard Error	95% Confidence Limits	
1	1869	1098	130	843	1353
2	1870	1098	130	843	1353
53	1921	851	129	599	1104
103	1971	851	129	599	1104
104	1972	851	129	599	1104

Example 34.4: Longitudinal Data: Smoothing of Repeated Measures

This example of a repeated measures study is taken from Diggle, Liang, and Zeger (1994, p. 100). The data consist of body weights of 27 cows, measured at 23 unequally spaced time points over a period of approximately 22 months. Following Diggle, Liang, and Zeger (1994), one animal is removed from the analysis, one observation is removed according to their Figure 5.7, and the time is shifted to start at 0 and is measured in 10-day increments. The design is a 2×2 factorial, and the factors are the infection of an animal with M. paratuberculosis and whether the animal is receiving iron dosing. The data set contains five variables: cow assigns a unique identification number—from 1 to 26—to each cow in the study, tpoint denotes the time of the growth measurement, weight denotes the growth measurement, iron is a dummy variable that indicates

whether the animal is receiving iron or not, and infection is a dummy variable that indicates whether the animal is infected or not. The goal of the study is to assess the effect of iron and infection—and their possible interaction—on weight. The following DATA steps create this data set:

```
data times;
  input time1-time23;
  datalines;
 122  150  166  179  219  247  276  296  324  354  380  445
 478  508  536  569  599  627  655  668  723  751  781
;

data Cows;
  if _n_ = 1 then merge times;
  array t{23} time1   - time23;
  array w{23} weight1 - weight23;
  input cow iron infection weight1-weight23 @@;
  do i=1 to 23;
     weight = w{i};
     tpoint = (t{i}-t{1})/10;
     output;
  end;
  keep cow iron infection tpoint weight;
  datalines;
  1 0 0  4.7    4.905  5.011  5.075  5.136  5.165  5.298  5.323
         5.416  5.438  5.541  5.652  5.687  5.737  5.814  5.799

    ... more lines ...
```

The following DATA step adds ironInf, a grouping variable that is used later during the plotting of the results. In the next step, the data are sorted by the index variable, tpoint.

```
data Cows;
   set Cows;
   ironInf = "No Iron and No Infection";
   if iron=1 and infection=1 then ironInf = "Iron and Infection";
   else if iron=1 and infection=0 then ironInf = "Iron and No Infection";
   else if iron=0 and infection=1 then ironInf = "No Iron and Infection";
   else ironInf = "No Iron and No Infection";
   run;

 proc sort data=Cows;
     by tpoint ;
 run;
```

To assess the effect of iron and infection on weight, the natural growth profile of the animals must also be accounted for. Here two alternate models for this problem are considered. The first model assumes that the observed weight of an animal is the sum of a common growth profile, which is modeled by a polynomial spline trend of order 2, the regression effects of iron and infection, and the observation error—modeled as white noise. An interaction term, for interaction between iron and infection, was found to be insignificant and is not included. In the second model, the common growth profile and the regression variables of the first model are replaced by four environment specific growth profiles.

The following statements fit the first model:

```
proc ssm data=Cows;
   id tpoint;
   trend growth(ps(2));
   irregular wn;
   model weight = iron infection growth wn;
   eval pattern = iron + infection + growth;
   output out=For;
quit;
```

Output 34.4.1 shows that the state dimension of this model is 2 (corresponding to the polynomial trend specification of order 2), the number of diffuse elements in the initial condition is 4 (corresponding to the trend and the two regressors iron and infection), and the number of unknown parameters is 2 (corresponding to the variance parameters of trend and irregular).

Output 34.4.1 Model1: Model Summary Information

The SSM Procedure

Model Summary

Model Property	Value
Number of Model Equations	1
State Dimension	2
Dimension of the Diffuse Initial Condition	4
Number of Parameters	2

Output 34.4.2 shows that the ID variable is irregularly spaced with replication.

Output 34.4.2 ID Variable Information

ID Variable Information

Name	Start	End	Max Delta	NDistinct	Type
tpoint	0	65.9	6.5	23	Irregular with Replication

The estimated regression coefficients of iron and infection, shown in Output 34.4.3, are significant and negative. This implies that both iron and infection adversely affect the response variable, weight.

Output 34.4.3 Model 1: Regression Estimates

Regression Parameter Estimates

Response Variable	Regression Variable	Estimate	Standard Error	t Value	Pr > |t|
weight	iron	-0.0748	0.00761	-9.82	<.0001
weight	infection	-0.1292	0.00859	-15.04	<.0001

The variance estimates of the trend component and the irregular component are shown in Output 34.4.4.

Output 34.4.4 Model 1: Estimates of Unnamed Parameters

Model Parameter Estimates

Component	Type	Parameter	Estimate	Standard Error	t Value
growth	PS(2) Trend	Level Variance	0.0000162	9.01E-06	1.80
wn	Irregular	Variance	0.0085849	5.03E-04	17.06

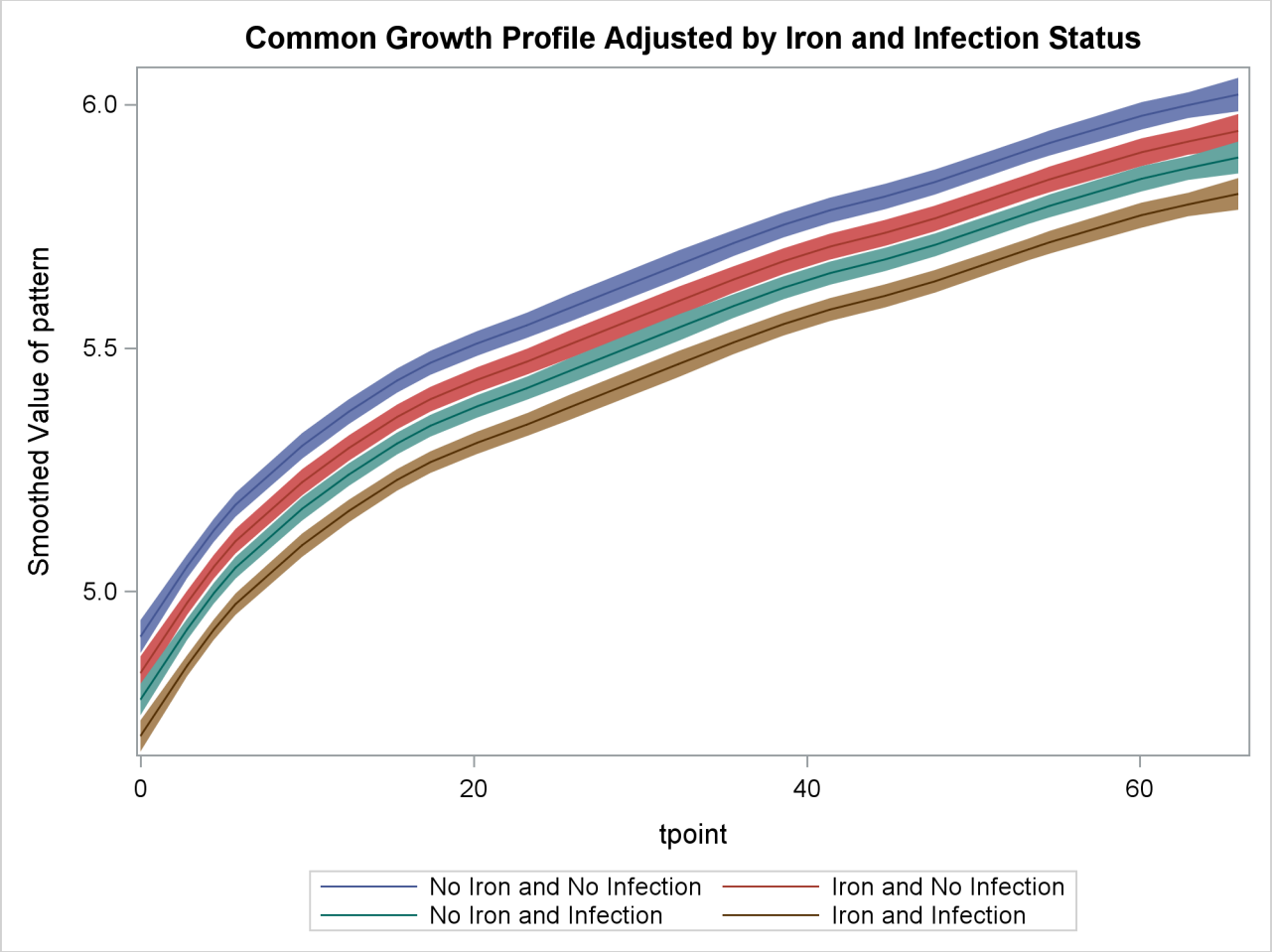
After examining the model fit, it is useful to study how well the patterns implied by the model follow the data. pattern, defined by the EVAL statement, is a sum of the trend component and the regression effects. A graphical examination of the smoothed estimate of pattern is done next. The following DATA step merges the output data set specified in the OUTPUT statement, For, with the input data set, Cows. In particular, this adds ironInf (a grouping variable from Cows) to For.

```
data For;
    merge for Cows;
    by tpoint;
run;
```

The following statements produce the graphs of smoothed_pattern, grouped according to the environment condition (see Output 34.4.5). The plot clearly shows that the control group "No Iron and No Infection" has the best growth profile, while the worst growth profile is for the group "Iron and Infection."

```
proc sgplot data=For noautolegend;
    title 'Common Growth Profile Adjusted by Iron and Infection Status';
    band x=tpoint lower=smoothed_lower_pattern
       upper=smoothed_upper_pattern / group=ironInf name="band";
    series  x=tpoint y=smoothed_pattern / group=ironInf name="series";
    keylegend "series";
run;
```

Output 34.4.5 Model 1: Growth Profile Comparison with 95% Confidence Bands

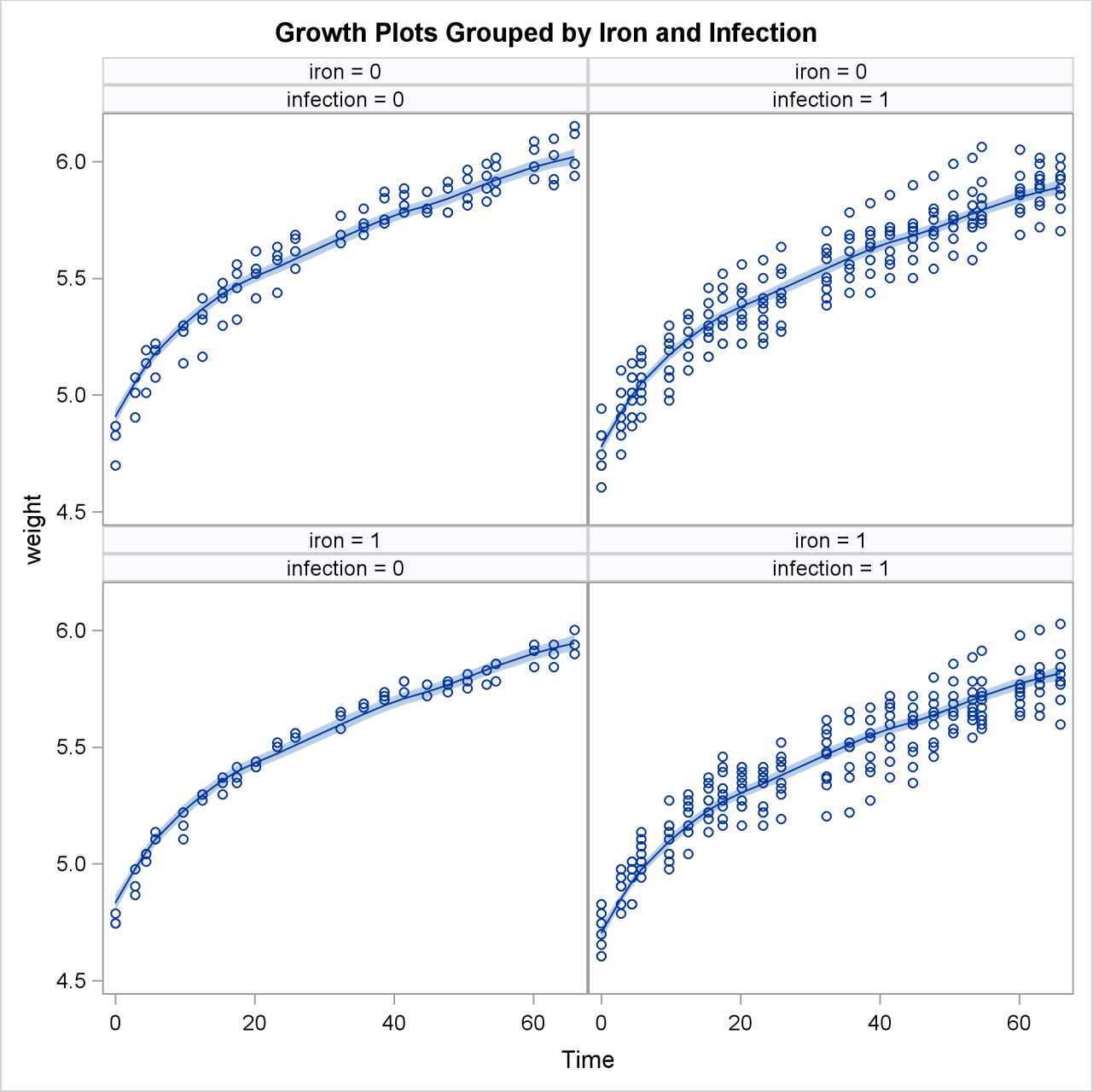


The following statements produce a panel of plots that show how well smoothed_pattern follows the observed data:

```
proc sgpanel data=For noautolegend;
  title 'Growth Plots Grouped by Iron and Infection';
   label tpoint='Time' ;
   panelby iron infection / columns=2;
   band x=tpoint lower=smoothed_lower_pattern
      upper=smoothed_upper_pattern ;
   scatter x=tpoint y=weight;
   series  x=tpoint y=smoothed_pattern ;
run;
```

Output 34.4.6 shows that the model fits the data reasonably well.

Output 34.4.6 Model 1: Smoothed Model Fit Lines



The following statements fit the second model. In this model separate polynomial trends are fit according to different settings of iron and infection by specifying an appropriate list of (dummy) variables in the CROSS= option of the trend specification.

```
proc ssm data=Cows;
    id tpoint;
    a1 = (iron=1 and infection=1);
    a2 = (iron=1 and infection=0);
    a3 = (iron=0 and infection=1);
    a4 = (iron=0 and infection=0);
    trend growth(ps(2)) cross=(a1-a4);
    irregular wn;
    model weight = growth wn;
    /* Define contrasts between a1 and other treatments  */
    comp a1Curve = growth_state_[1];
    comp a2Curve = growth_state_[2];
    comp a3Curve = growth_state_[3];
    comp a4Curve = growth_state_[4];
    eval contrast21 = a2Curve - a1Curve;
    eval contrast31 = a3Curve - a1Curve;
    eval contrast41 = a4Curve - a1Curve;
    output out=for1;
quit;
```

As a result of the CROSS= option, the trend component growth is actually a sum of four separate trends that correspond to the different iron-infection settings. Denoting growth by μ_t and the four independent trends by $\mu_{1,t}$, $\mu_{2,t}$, $\mu_{3,t}$, and $\mu_{4,t}$,

$$\mu_t = a1 * \mu_{1,t} + a2 * \mu_{2,t} + a3 * \mu_{3,t} + a4 * \mu_{4,t}$$

where a1, a2, a3, and a4 are the dummy variables specified in the CROSS= option. This shows that, for any given setting (say, the one for a4) μ_t is simply the corresponding trend $\mu_{4,t}$. In addition, note the form of the COMPONENT statements that define the components a1Curve, a2Curve, a3Curve, and a4Curve. This form of the COMPONENT statement treats the state that is associated with growth, named growth_state_ by convention, as a state of nominal dimension 4—the number of variables in the CROSS= list. This, in turn, implies that a1Curve, which is defined as growth_state_[1], refers to $\mu_{1,t}$. These components are subsequently used in the EVAL statements to define contrasts between the trends—for example, contrast21 corresponds to the difference between the trends $\mu_{2,t}$ and $\mu_{1,t}$. The estimates of these components (a1Curve, a2Curve, . . ., contrast41) are output to the data set For1 named in the OUT= option of the OUTPUT data set.

The model summary, shown in Output 34.4.7, reflects the increased state dimension and the increased number of parameters.

Output 34.4.7 Model2: Model Summary Information

The SSM Procedure

Model Summary	
Model Property	**Value**
Number of Model Equations	1
State Dimension	8
Dimension of the Diffuse Initial Condition	8
Number of Parameters	5

Output 34.4.8 shows the parameter estimates for this model.

Output 34.4.8 Model2: Estimates of Unnamed Parameters (Partial Output)

Component	Parameter	Estimate	StdErr	tValue
growth(Cross = a1)	Level Variance	1.28E-05	6.83E-06	1.87
growth(Cross = a2)	Level Variance	8.72E-06	3.81E-06	2.29
growth(Cross = a3)	Level Variance	9.07E-06	4.23E-06	2.14
growth(Cross = a4)	Level Variance	8.45E-06	3.40E-06	2.49
wn	Variance	8.39E-03	4.98E-04	16.84

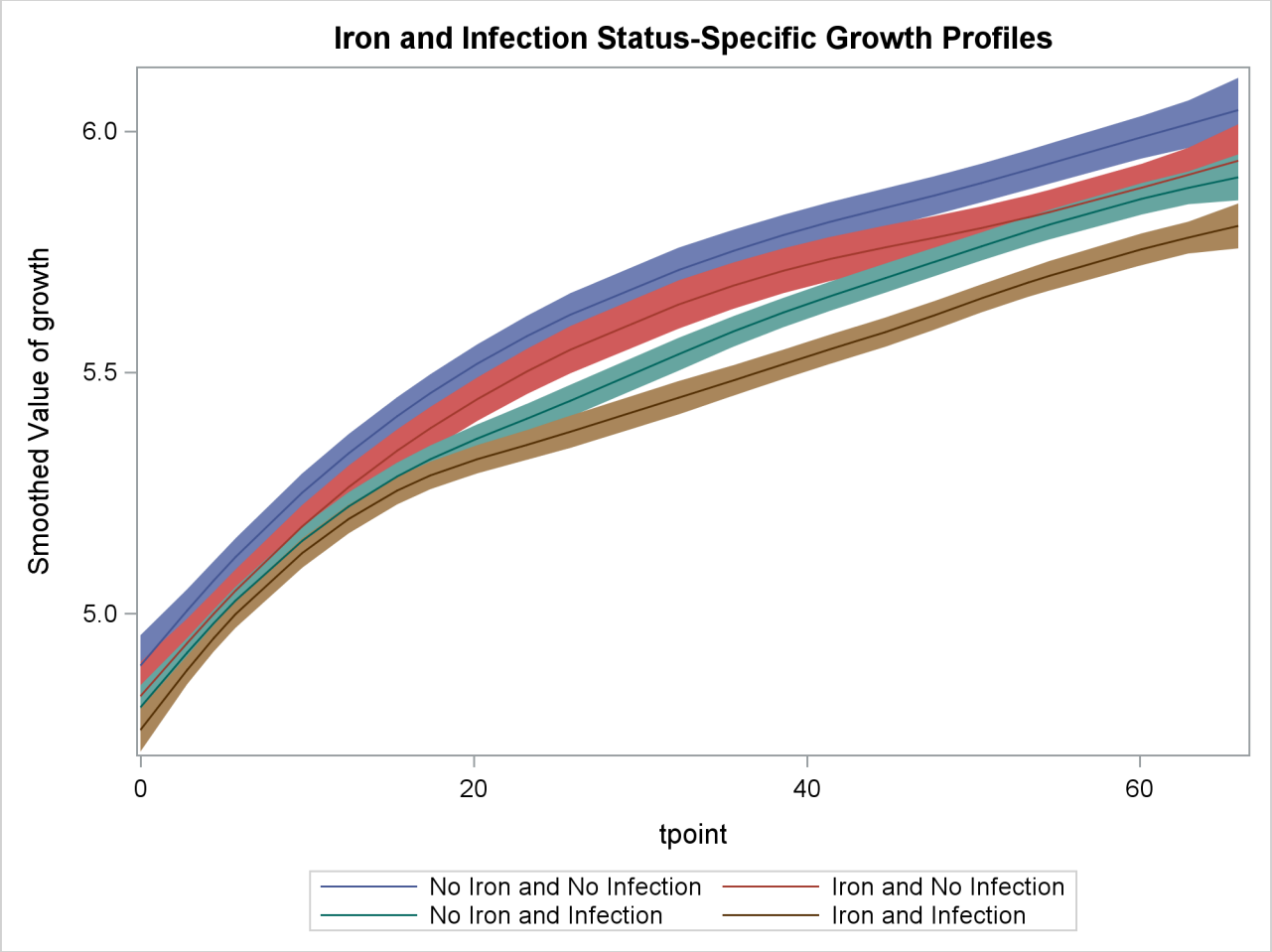
Next, the smoothed estimate of trend (growth) is graphically studied. The following DATA step prepares the data for the grouped plots of smoothed_growth by merging For1 with the input data set Cows. As before, the reason is merely to include ironInf (the grouping variable).

```
data For1;
    merge For1 Cows;
    by tpoint;
run;
```

The following statements produce the graphs of smoothed μ_t for the desired settings (since the grouping variable ironInf exactly corresponds to these settings). Once again, the plot in Output 34.4.9 clearly shows that the control group "No Iron and No Infection" has the best growth profile, while the worst growth profile is for the group "Iron and Infection." However, unlike the first model, the profile curves are not merely shifted versions of a common profile.

```
proc sgplot data=For1 noautolegend;
   title 'Iron and Infection Status-Specific Growth Profiles';
    band x=tpoint lower=smoothed_lower_growth
      upper=smoothed_upper_growth / group=ironInf name="band";
    series  x=tpoint y=smoothed_growth / group=ironInf name="series";
    keylegend "series";
run;
```

Output 34.4.9 Model 2: Growth Profile Comparison with 95% Confidence Bands

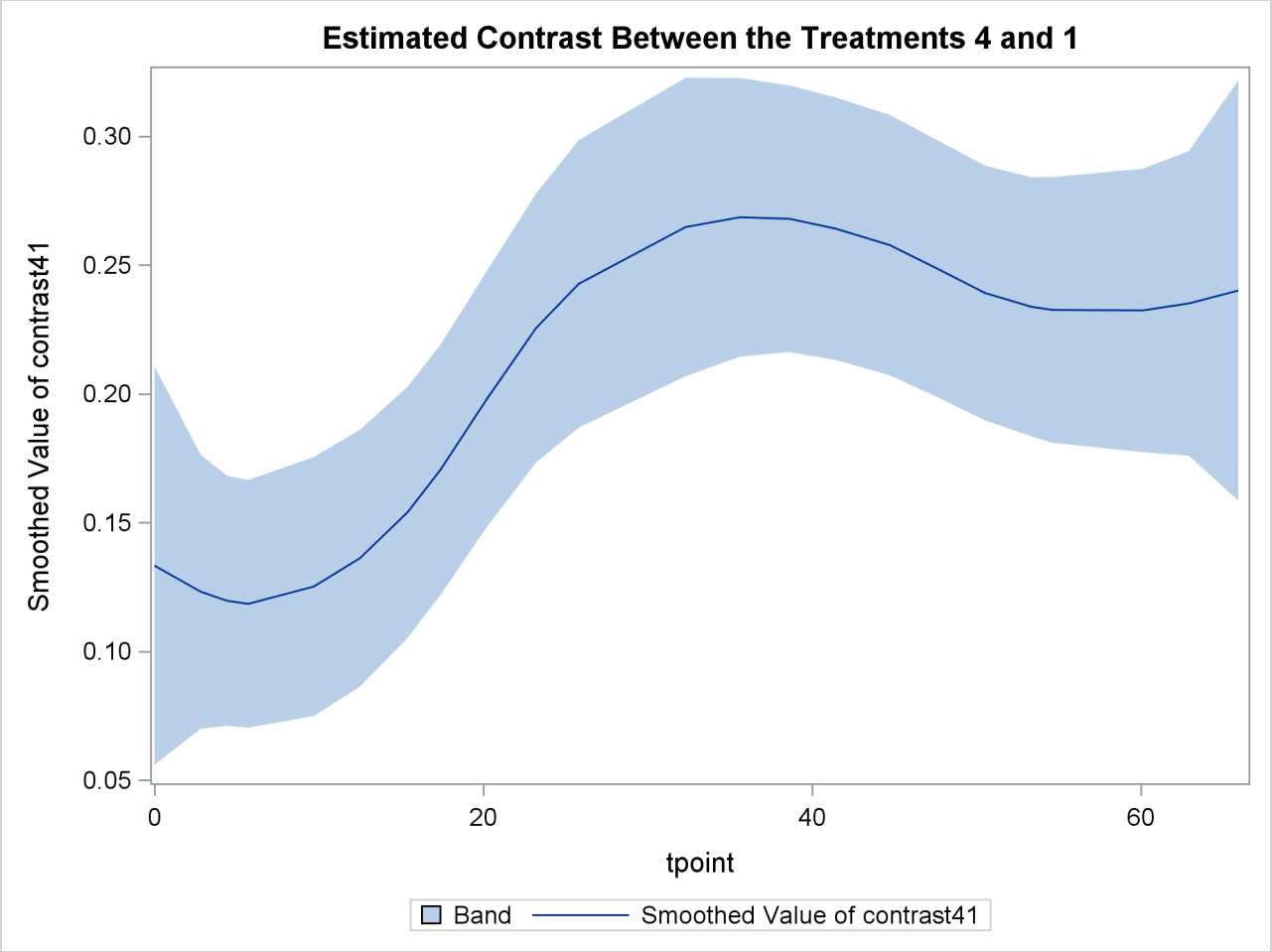


The following statements produce the plot of smoothed $(\mu_{4,t} - \mu_{1,t})$—contrast between the best and the worst growth profiles.

```
proc sgplot data=For1;
   title "Estimated Contrast Between the Treatments 4 and 1 ";
    band x=tpoint lower=smoothed_lower_contrast41
       upper=smoothed_upper_contrast41;
    series x=tpoint y=smoothed_contrast41;
run;
```

Output 34.4.10 shows that the growth pattern of the control group "No Iron and No Infection" consistently remains above the growth pattern of the treatment group "Iron and Infection."

Output 34.4.10 Estimated Contrast Between the Treatments 4 and 1 with 95% Confidence Bands



Example 34.5: A User-Defined Trend Model

This example shows how to specify a continuous-time trend model discussed in Harvey (1989, chap. 9, sec. 9.2.1). This model is not one of the predefined trend models in the SSM procedure. The system matrices that govern the two-dimensional state of this model are

$$\mathbf{T} = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix}$$

$$\mathbf{Q} = \begin{bmatrix} h\sigma_1^2 + \frac{h^3\sigma_2^2}{3} & \frac{h^2\sigma_2^2}{2} \\ \frac{h^2\sigma_2^2}{2} & h\sigma_2^2 \end{bmatrix}$$

where $h = h_t = (\tau_{t+1} - \tau_t)$ denotes the difference between the successive time points, and the parameters σ_1^2 and σ_2^2 are called the level variance and the slope variance, respectively. The initial condition is fully diffuse. The trend component corresponds to the first element of this state vector. The second element of the state vector corresponds to the slope of this trend component. This model reduces to the polynomial spline model of order 2 if the level variance $\sigma_1^2 = 0$. See the section "Polynomial Spline Trend" on page 2405.

The following statements specify a trend-plus-noise model to model the growth of cows in the previous example (Example 34.4). The only cows that are considered are the ones that received iron and are infected.

```
proc ssm data=Cows;
   where iron=1 and infection=1;
   id tpoint;
   parms var1 var2 / lower=(1.e-8 1.e-8);
   array tMat{2,2};
   tMat[1,1] = 1;
   tMat[2,2] = 1;
   tMat[1,2] = _ID_DELTA_;
   array covMat{2,2};
   covMat[1,1] = var1*_ID_DELTA_ + var2*_ID_DELTA_**3/3;
   covMat[1,2] = var2*_ID_DELTA_**2/2;
   covMat[2,1] = covMat[1,2] ;
   covMat[2,2] = var2*_ID_DELTA_;
   state harveyLL(2) T(g)=(tMat) cov(g)=(covMat) a1(2);
   component trend = harveyLL[1];
   component slope = harveyLL[2];
   irregular wn;
   model  weight = trend  wn;
   output out=for;
run;
```

The program is easy to follow. The PARMS statement declares var1 and var2 as positive parameters, which correspond to σ_1^2 and σ_2^2, respectively. The programming statements define arrays tMat and covMat, which later become the matrices $\mathbf{T}$ and $\mathbf{Q}$, respectively. Note that the element tMat[2,1] is left unassigned, since it is a structural zero of $\mathbf{T}$ (see the section "Sparse Transition Matrix Specification" on page 2394 for more information). Recall that the predefined variable _ID_DELTA_ contains the value of h_t, which is needed for defining the elements of $\mathbf{T}$ and $\mathbf{Q}$ (see the section "ID Statement" on page 2376). The STATE statement defines the trend state vector, harveyLL, and the COMPONENT statement defines the trend component, trend, by selecting the first element of harveyLL. An additional COMPONENT statement defines the slope component, slope, as the second element of harveyLL. The slope component (which represents the cow's growth rate) is not part of the observation equation; it is specified so that its estimate is output to For (the OUT= data set specified in the OUTPUT statement). The IRREGULAR statement defines the observation noise, and the MODEL statement defines the trend-plus-noise model.

The estimates of var1 and var2 are shown in Output 34.5.1. It shows that the estimate of the level variance is nearly 0, implying that the fitted trend model is identical to the polynomial spline trend of order 2.

Output 34.5.1 Estimates of the Named Parameters

The SSM Procedure

Estimates of Named Parameters

Parameter	Estimate	Standard Error	t Value
var1	1.00E-08	0.000849	0.00
var2	1.24E-05	.	.

The estimate of the noise variance is shown in Output 34.5.2.

Output 34.5.2 Estimates of the Unnamed Parameters

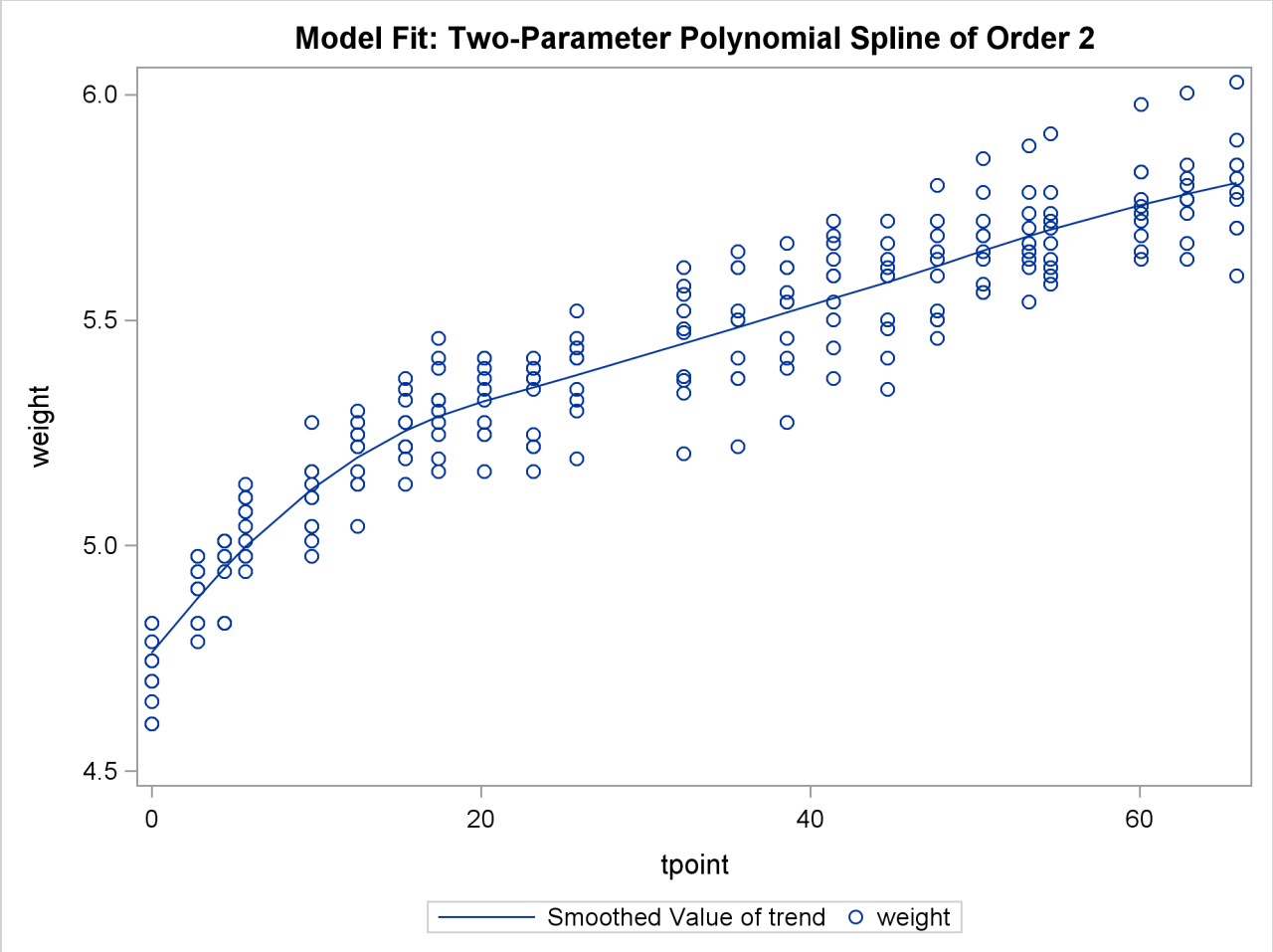
Model Parameter Estimates

Component	Type	Parameter	Estimate	Standard Error	t Value
wn	Irregular	Variance	0.00954	0.000909	10.49

The following statements produce the plot of the fit of this trend model (shown in Output 34.5.3).

```
proc sgplot data=For;
    title "Model Fit: Two-Parameter Polynomial Spline of Order 2";
    series x=tpoint y=smoothed_trend;
    scatter x=tpoint y=weight;
run;
```

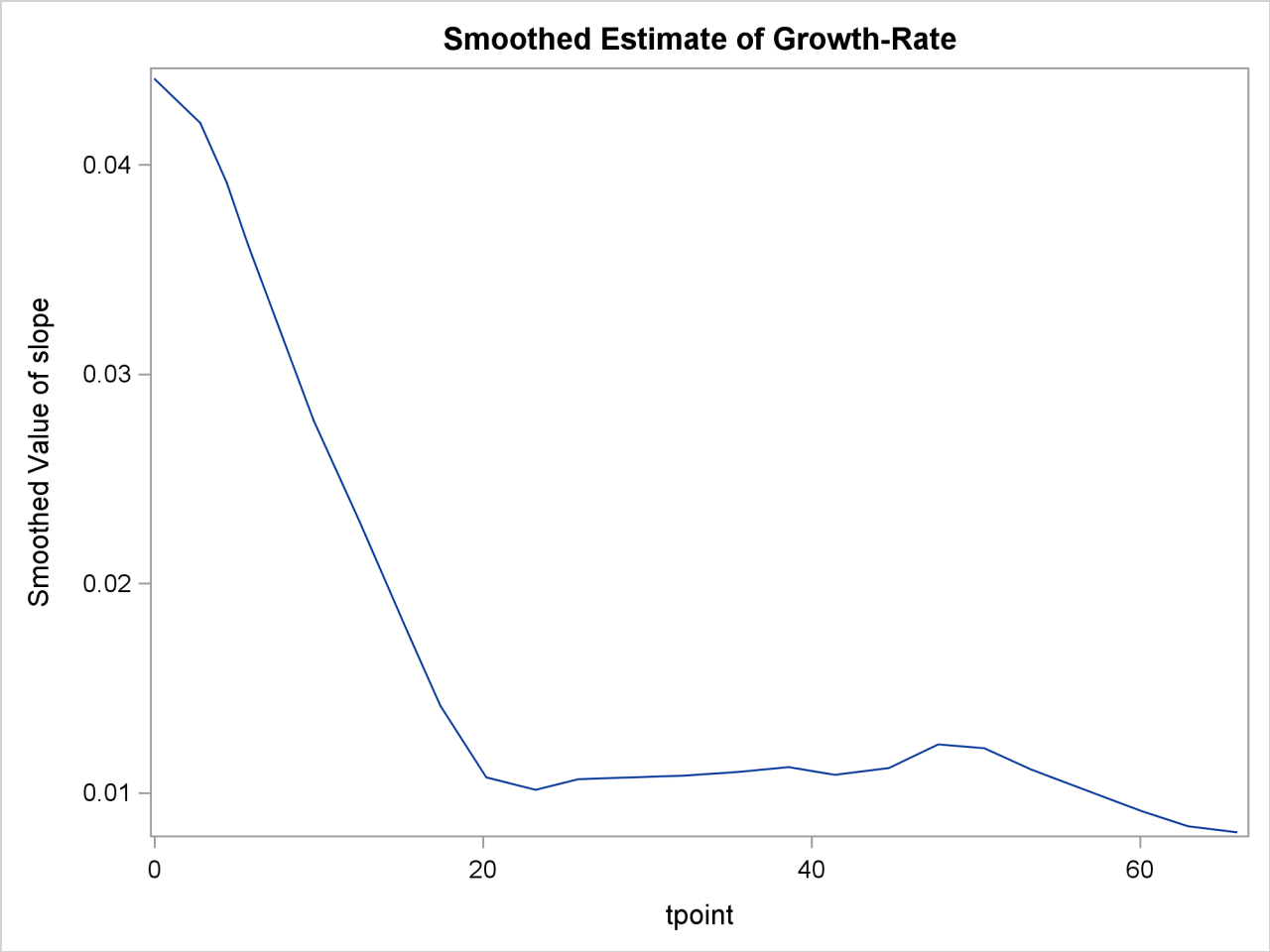
Output 34.5.3 A User-Defined Trend Model



The following statements produce the plot of the estimate of the slope component (shown in Output 34.5.4). This plot complements the preceding plot of trend; it shows the pattern of decline in the growth rate as the animals age.

```
proc sgplot data=For;
    title "Smoothed Estimate of Growth-Rate";
    series x=tpoint y=smoothed_slope;
run;
```

Output 34.5.4 Estimate of the slope Component



Example 34.6: Model with Multiple ARIMA Components

This example shows how you can fit the REGCOMPONENT models in Bell (2011) by using the SSM procedure. The following DATA step generates the data used in the last example of this article (Example 6: "Modeling a time series with a sampling error component"). The variable y in this data set contains monthly values of the VIP series (value of construction put in place), a U.S. Census Bureau publication that measures the value of construction installed or erected at construction sites during a given month. The values of y are known to be contaminated with heterogeneous sampling errors; the variable hwt in the data set is a proxy for this sampling error in the log scale. The variable hwt is treated as a weight variable for the noise component in the model.

```
data Test;
  input y hwt;
  date = intnx('month', '01jan1997'd, _n_-1 );
  format date date.;
  logy = log(y);
  label logy = 'Log value of construction put in place';
  datalines;
115.2    0.042
110.4    0.042
111.5    0.067
127.9    0.122
150.0    0.129
149.5    0.135
139.5    0.152
144.6    0.168
176.0    0.173

   ... more lines ...
```

The article proposes the following model for the log VIP series:

$$\log(y) = \mu_t + hwt * \eta_t$$

where μ_t follows an ARIMA$(0,1,1)\times(0,1,1)_{12}$ model and η_t is a zero-mean, AR(2) error process. In addition, the article fixes the values of some of the model parameters to known values in order to use the known background information. The following statements specify the model in the article:

```
proc ssm data=Test;
   id date interval=month;
   parm var1=0.016565 / lower=1.e-8;
   trend airlineTrend(arma(d=1 sd=1 q=1 sq=1 s=12)) variance=var1;
   trend ar2Noise(arma(p=2)) cross=(hwt) ar=0.600 0.246 variance=0.34488;
   model logy = airlineTrend ar2Noise;
   output outfor=For;
run;
```

Output 34.6.1 Estimates of the MA Parameters in the airlineTrend Model

The SSM Procedure

Model Parameter Estimates

Component	Type	Parameter	Estimate	Standard Error	t Value
airlineTrend	ARMA Trend	MA_1	0.421	0.301	1.40
airlineTrend	ARMA Trend	SMA_1	0.310	0.347	0.89

Output 34.6.2 Estimate of the Error Variance in the airlineTrend Model

Estimates of Named Parameters

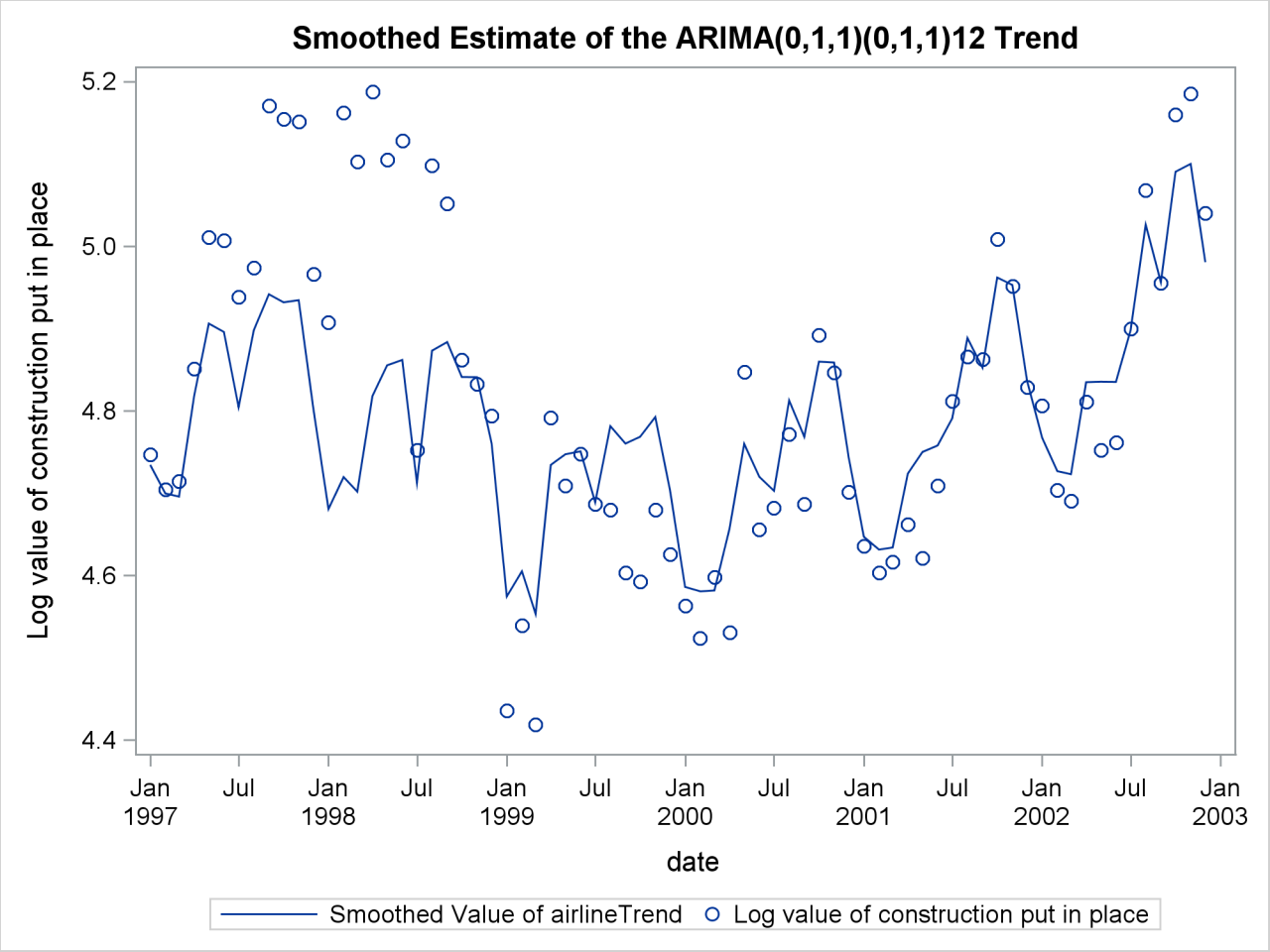
Parameter	Estimate	Standard Error	t Value
var1	0.004	0.00222	1.80

The ARIMA$(0,1,1)\times(0,1,1)_{12}$ trend μ_t is named airlineTrend and the zero-mean, AR(2) error process η_t is named ar2Noise. See the TREND statement for more information about the ARIMA notation. The estimates of model parameters are shown in Output 34.6.1 and Output 34.6.2. These estimates are slightly different from the estimates given in the article; however, the estimated trend and noise series are qualitatively similar.

The following statements produce the plot of the estimate of the airlineTrend component (shown in Output 34.6.3). This plot is very similar to the trend plot shown in the article (the article plots are in the antilog scale).

```
proc sgplot data=For;
    title "Smoothed Estimate of the ARIMA(0,1,1)(0,1,1)12 Trend";
    series x= date y=smoothed_airlineTrend;
    scatter x= date y=logy;
run;
```

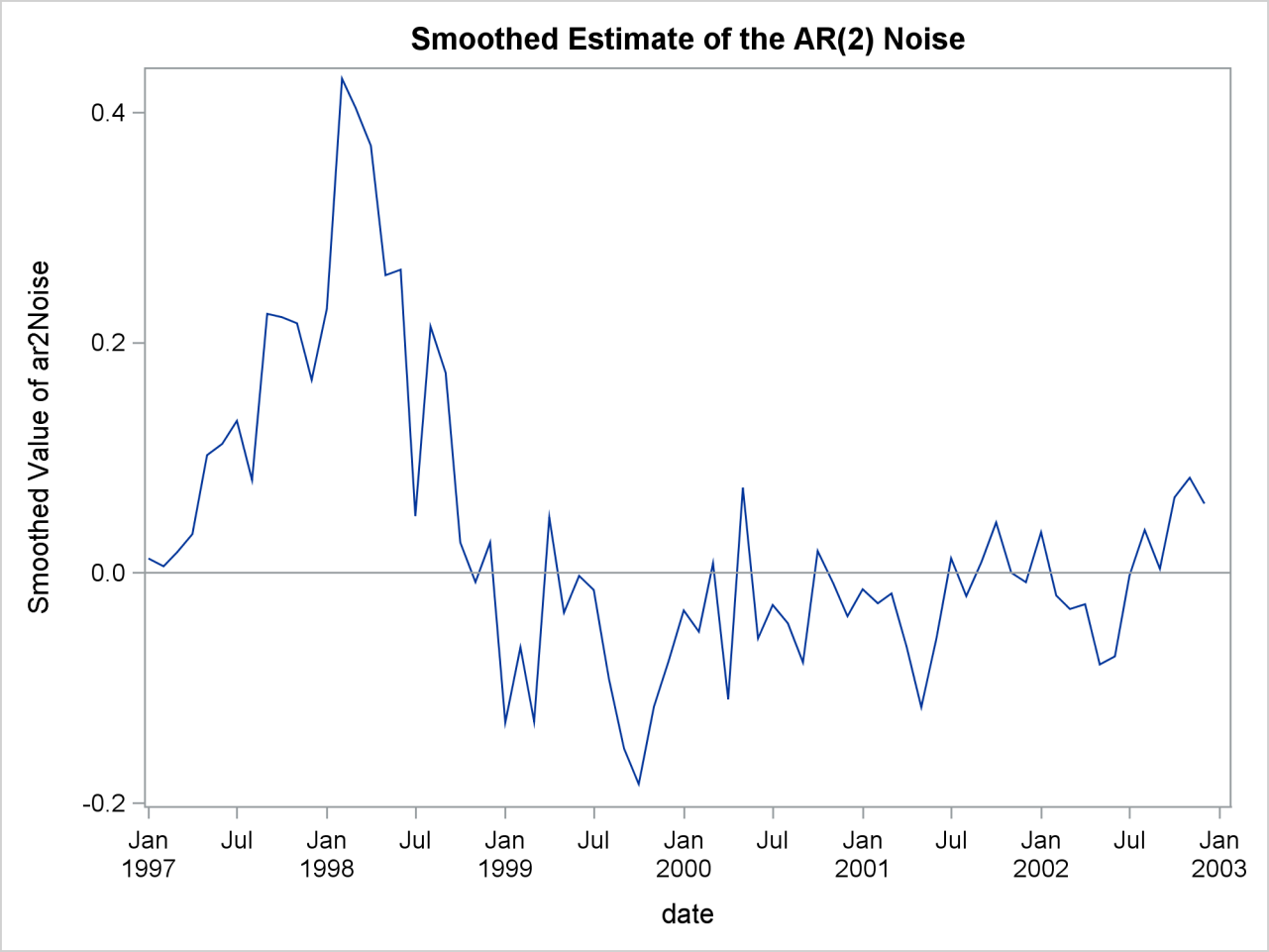
Output 34.6.3 Estimate of the airlineTrend Component



The following statements produce the plot of the estimate of the ar2Noise component (shown in Output 34.6.4). This plot is also very similar to the noise plot shown in the article (once again, the article plots are in the antilog scale).

```
proc sgplot data=For;
    title "Smoothed Estimate of the AR(2) Noise";
    series x= date y=smoothed_ar2Noise;
    refline 0;
run;
```

Output 34.6.4 Estimate of the ar2Noise Component



Example 34.7: Dynamic Factor Modeling

This example shows how you can fit a variant of the dynamic Nelson-Siegel (DNS) factor model discussed in Koopman, Mallee, and van der Wel (2010). The following DATA step creates the yield-curve data set, Dns, that is used in this article. The data are monthly bond yields that were recorded between the start of 1970 to the end of 2000 for 17 bonds of different maturities; the maturities range from three months to 10 years (120 months). The variable date contains the observation date, yield contains the bond yield, maturity contains the associated bond maturity, and mtype contains an index (ranging from 1 to 17) that sequentially labels bonds of increasing maturity. The data have been extended for two more years by adding missing yields for the years 2001 and 2002, which causes the SSM procedure to produce model forecasts for this span.

```
data Dns;
input date : date. yield maturity mtype;
format date date.;
datalines;
1-Jan-70    8.019    3    1
1-Jan-70    8.091    6    2
1-Jan-70    8.108    9    3

   ... more lines ...
```

In addition, suppose you are interested in extrapolating the fitted model to predict the yield of a hypothetical bond that has a maturity of 42 months and is not traded on the general exchange. The following DATA step creates the necessary missing values for this new bond, which is assigned the index of 18—that is, the value of mtype is 18.

```
data tmp1;
   set dns(keep=date);
   by date;
   if first.date then do;
       yield = .;
       maturity = 42;
       mtype = 18;
       output;
   end;
run;

 proc append data=tmp1 base=dns; run;
 proc sort data=dns;
 by date;
 run;
```

Suppose that $\theta_t(\tau)$ denotes the (idealized) yield at time t that is associated with a bond of maturity τ (in months). Even if time is not measured continuously and the bonds of only certain maturities are traded, $\theta_t(\tau)$ is treated as a smooth function of two continuous variables, time t and maturity τ. Koopman, Mallee, and van der Wel (2010) discuss a variety of models for $\theta_t(\tau)$, which is called the yield surface. One of these models depends on a positive, time-varying, scalar parameter λ_t and a time-varying three-dimensional vector parameter $\boldsymbol{\beta}_t$. This model can be described as follows:

$$\begin{aligned}\theta_t(\tau) &= \theta(\tau;\lambda_t,\boldsymbol{\beta}_t)\\ &= \beta_{1t} + \beta_{2t}\left(\frac{1-\exp(-\lambda_t\tau)}{\lambda_t\tau}\right) + \beta_{3t}\left(\frac{1-\exp(-\lambda_t\tau)}{\lambda_t\tau} - \exp(-\lambda_t\tau)\right)\end{aligned}$$

This model is a dynamic version of a static model discussed in Nelson and Siegel (1987), where λ_t and $\boldsymbol{\beta}_t$ are time invariant. For fixed time period t, the three terms in this model have relatively simple interpretation. The first term β_{1t} can be thought of as the overall yield level because it does not depend on τ, the bond maturity. It can also be thought of as the long term yield because as $\tau \uparrow \infty$ the other two terms vanish; the coefficients of both β_{2t} and β_{3t} converge to 0 as $\tau \uparrow \infty$ (recall that λ_t is positive). Next, note that as $\tau \downarrow 0$ the coefficient of β_{2t} in the second term converges to 1 while that of β_{3t} in the third term converges to 0; therefore the second term can be thought of as a correction to the overall yield that is associated with the short term bonds. Finally, note that the coefficient of β_{3t} in the third term is a unimodal function of τ that decays

monotonically to 0 as $\tau \downarrow 0$ and as $\tau \uparrow \infty$; therefore the third term is associated with the medium term bond yields. It is postulated that the observed yield, denoted by $y_t(\tau)$, is a noisy version of this unobserved (true) yield $\theta_t(\tau)$. The observed yield can be modeled as

$$\begin{aligned} y_t(\tau) &= \theta(\tau; \lambda_t, \boldsymbol{\beta}_t) + \epsilon_{t,\tau} \\ &= \beta_{1t} + \beta_{2t}\left(\frac{1-\exp(-\lambda_t\tau)}{\lambda_t\tau}\right) + \beta_{3t}\left(\frac{1-\exp(-\lambda_t\tau)}{\lambda_t\tau} - \exp(-\lambda_t\tau)\right) + \epsilon_{t,\tau} \\ (\boldsymbol{\beta}_t - \boldsymbol{\mu}) &= \boldsymbol{\Phi}(\boldsymbol{\beta}_{t-1} - \boldsymbol{\mu}) + \boldsymbol{\eta}_t \end{aligned}$$

where $\epsilon_{t,\tau}$ are zero-mean, independent, Gaussian variables with variance σ_τ^2, and $\boldsymbol{\eta}_t$ is a three-dimensional, Gaussian white noise. That is, $\boldsymbol{\beta}_t$ is a VAR(1) process with mean vector $\boldsymbol{\mu}$. The remainder of this example explains how to use the SSM procedure to fit this model to the yield data in the Dns data set.

Suppose that variables Z1, Z2, and Z3 are defined as the coefficients of β_{1t}, β_{2t}, and β_{3t}, respectively. That is,

$$\begin{aligned} Z1 &= 1 \\ Z2 &= \frac{1-\exp(-\lambda_t\tau)}{\lambda_t\tau} \\ Z3 &= \frac{1-\exp(-\lambda_t\tau)}{\lambda_t\tau} - \exp(-\lambda_t\tau) \end{aligned}$$

In this case,

$$\theta_t(\tau) = Z1 * \beta_{1t} + Z2 * \beta_{2t} + Z3 * \beta_{3t}$$

Let $\boldsymbol{\zeta}_t = \boldsymbol{\beta}_t - \boldsymbol{\mu}$. Then $\boldsymbol{\zeta}_t$ is a zero-mean VAR(1) process and $\boldsymbol{\beta}_t = \boldsymbol{\zeta}_t + \boldsymbol{\mu}$. In particular,

$$\begin{aligned} \theta_t(\tau) &= Z1 * \beta_{1t} + Z2 * \beta_{2t} + Z3 * \beta_{3t} \\ &= Z1 * \zeta_{1t} + Z2 * \zeta_{2t} + Z3 * \zeta_{3t} + Z1 * \mu_1 + Z2 * \mu_2 + Z3 * \mu_3 \end{aligned}$$

This shows that the model for $y_t(\tau)$ can be cast into a state space form with the following observation equation:

$$y_t(\tau) = \mathbf{Z}\boldsymbol{\zeta}_t + \mathbf{Z}\boldsymbol{\mu} + \epsilon_{t,\tau}$$

The underlying six-dimensional state vector $\boldsymbol{\alpha}_t$ is formed by joining the two independent subvectors, $\boldsymbol{\zeta}_t$ (which is a zero-mean, VAR(1) process) and the constant mean vector $\boldsymbol{\mu}$. That is, $\boldsymbol{\alpha}_t = (\zeta_{1t}\ \zeta_{2t}\ \zeta_{3t}\ \mu_1\ \mu_2\ \mu_3)'$.

Note that the variables Z2 and Z3 depend on the time varying parameter λ_t, which is unknown. λ_t is assumed to be a smooth and positive function of time t. In what follows λ_t is represented as an exponential of a cubic spline—a B-spline— in time with four evenly spaced interior knots between January 1970 and December 2002. A cubic spline with four interior knots can be represented as a sum of seven (number of knots + spline degree + 1) B-spline basis functions, $c_{1t}, c_{2t}, \ldots, c_{7t}$, for example. More specifically, λ_t can be expressed as

$$\lambda_t = \exp(v1 * c_{1t} + \ldots + v7 * c_{7t})$$

for some parameters $v1, v2, \ldots, v7$ and the B-spline basis functions (of time) $c_{1t}, c_{2t}, \ldots, c_{7t}$. Thus, the variables Z2 and Z3 become known functions of time, except for the parameters $v1, v2, \ldots, v7$, which are estimated from the data. The following statements augment the Dns data set with the B-spline basis columns in two steps. First a data set that contains the basis columns, c1–c7, is created by using the BSPLINE function in the IML procedure. This data set is then merged with the Dns data set.

```
proc iml;
   use dns;
   read all var {date} into x;
   bsp = bspline(x, 2, ., 4);
   create spline var{c1 c2 c3 c4 c5 c6 c7};
   append from bsp;
quit;
data dns;
   merge dns spline;
run;
```

The following statements use the SSM procedure to perform the model fitting and forecasting calculations. The variance of the observation equation disturbance for the hypothetical bond (mtype = 18) is taken to be the average of the neighboring bonds (mtype = 10 and 11), whose maturities are 36 and 48 months, respectively.

```
 proc ssm data=Dns optimizer(technique=dbldog maxiter=400);
      id date interval=month;

     /* Time-varying parameter lambda */
     parms v1-v7;
     lambda = exp(v1*c1 + v2*c2 + v3*c3 + v4*c4
          + v5*c5 + v6*c6 + v7*c7);

     /* Observation equation disturbance -- separate variance for each maturity */
     parms sigma1-sigma17 / lower=1.e-4;
     array s_array(17) sigma1-sigma17;
      do i=1 to 17;
         if (mtype=i) then sigma = s_array[i];
      end;
      if (mtype=18) then sigma = (sigma10+sigma11)/2;
      irregular wn variance=sigma;

      /* Variables Z1, Z2, Z3 needed in the observation equation */
      Z1= 1.0;
      tmp = lambda*maturity;
      Z2 = (1-exp(-tmp))/tmp;
      Z3 = ( 1-exp(-tmp)-tmp*exp(-tmp) )/tmp;

       /* Zero-mean VAR(1) factor zeta and the associated component */
      state zeta(3) type=VARMA(p(d)=1) cov(g) print=(cov ar);
      comp zetaComp = (Z1-Z3)*zeta;

      /* Constant mean vector mu and the associated component */
       state mu(3) type=rw;
       comp muComp = (Z1-Z3)*mu;

       /* Observation equation */
       model yield = muComp zetaComp wn;

       /* Various components defined only for output purposes  */
       eval yieldSurface = muComp + zetaComp;

       comp zeta1 = zeta[1];
```

```
    comp zeta2 = zeta[2];
    comp zeta3 = zeta[3];
    comp mu1 = mu[1];
    comp mu2 = mu[2];
    comp mu3 = mu[3];

    comp z2zeta = (Z2)*zeta[2];
    comp z3zeta = (Z3)*zeta[3];
    comp z2Mu =  (Z2)*mu[2];
    comp z3Mu =  (Z3)*mu[3];

    eval beta1 = mu1 + zeta1;
    eval beta2 = mu2 + zeta2;
    eval beta3 = mu3 + zeta3;

    eval shortTem = z2zeta + z2Mu;
    eval medTerm = z3zeta + z3Mu;

    /* output the component estimates and the forecasts */
    output out=dnsFor pdv;
run;
```

The DBLDOG optimization technique is used for parameter estimation since it is computationally more efficient in this example. The transition matrix, $\mathbf{\Phi}$, in the VAR(1) specification of $\boldsymbol{zeta}$ is taken to be diagonal (TYPE=VARMA(P(D)=1)) because the use of more general square matrix did not improve the model fit significantly. The mean vector $\boldsymbol{mu}$ (recall that $\boldsymbol{beta}_t = \boldsymbol{zeta}_t + \boldsymbol{mu}$) is specified as a three-dimensional random walk with zero disturbance covariance (signified by the absence of COV= option). The model specification part of the program ends with the MODEL statement; the subsequent COMP and EVAL statements define some useful linear combinations of the underlying state. Their estimates are computed after the model fit is completed and are output to the output data set dnsFor. The dnsFor data set also contains all the program variables and the parameters defined in the PARMS statement because the OUTPUT statement contains the PDV option.

Output 34.7.1 shows the estimated mean vector ($\boldsymbol{\mu}$). It shows that the mean long-term yield is 7.64. Output 34.7.2 shows the estimates of v1–v7 (used for defining time-varying λ_t) and the maturity specific observation variances. Output 34.7.3 shows the estimate of the VAR(1) transition matrix $\mathbf{\Phi}$, and Output 34.7.4 shows the associated disturbance covariance matrix $\mathbf{\Sigma}$. The model fit summary is shown in Output 34.7.5.

Output 34.7.1 Estimate of the Mean Vector ($\boldsymbol{\mu}$)

The SSM Procedure

Estimates of Fixed State Effects

State	Element Index	Estimate	Standard Error	t Value	Pr > |t|
mu	1	7.639	1.356	5.63	<.0001
mu	2	-1.319	0.777	-1.70	0.0895
mu	3	-0.309	0.268	-1.16	0.2481

Output 34.7.2 Estimates of v1–v7 and Observation Variances

Estimates of Named Parameters			
Parameter	Estimate	Standard Error	t Value
v1	-1.19587	0.304003	-3.93
v2	-2.93679	0.111440	-26.35
v3	-1.88709	0.068972	-27.36
v4	-2.31371	0.079116	-29.24
v5	-3.21875	0.105575	-30.49
v6	-1.66055	0.315692	-5.26
v7	-4.60122	1.548096	-2.97
sigma1	0.05406	0.004708	11.48
sigma2	0.00349	0.000865	4.03
sigma3	0.00869	0.000752	11.56
sigma4	0.01093	0.000901	12.14
sigma5	0.00865	0.000757	11.43
sigma6	0.00603	0.000571	10.56
sigma7	0.00519	0.000491	10.58
sigma8	0.00542	0.000497	10.90
sigma9	0.00562	0.000500	11.25
sigma10	0.00639	0.000559	11.43
sigma11	0.01032	0.000848	12.17
sigma12	0.00742	0.000676	10.98
sigma13	0.01106	0.000947	11.68
sigma14	0.01194	0.001051	11.36
sigma15	0.01244	0.001162	10.70
sigma16	0.02140	0.001842	11.62
sigma17	0.02746	0.002295	11.97

Output 34.7.3 Transition Matrix, $\mathbf{\Phi}$, Associated with $\boldsymbol{\zeta}$

AR Coefficient Matrix for zeta			
	Col1	Col2	Col3
Row1	0.98982	0	0
Row2	0	0.962457	0
Row3	0	0	0.803017

Output 34.7.4 Estimated Disturbance Covariance of ζ

Disturbance Covariance for zeta			
	Col1	**Col2**	**Col3**
Row1	0.108107	-0.02618	0.087107
Row2	-0.02618	0.360656	0.008928
Row3	0.087107	0.008928	1.072137

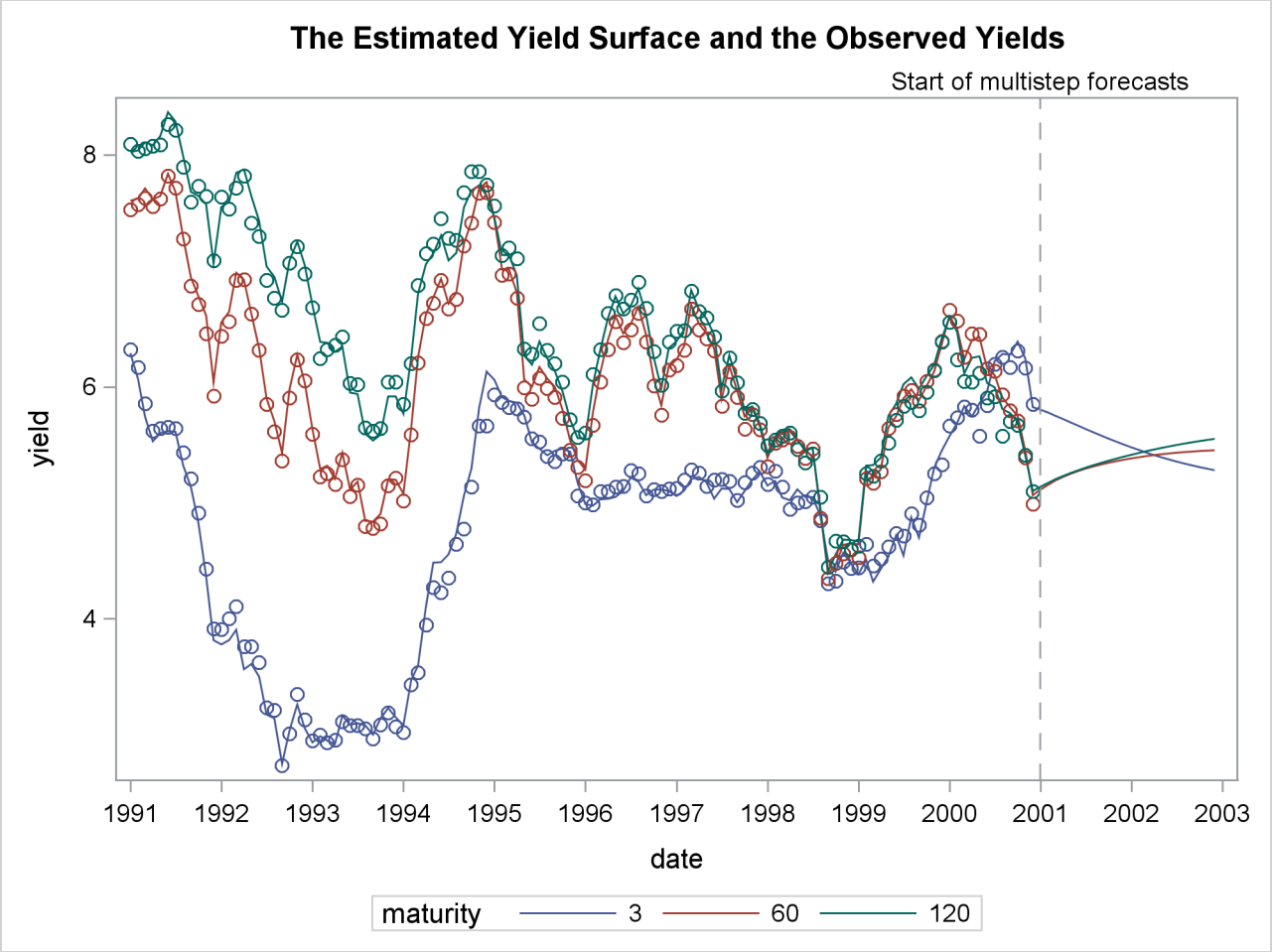
Output 34.7.5 Likelihood Computation Summary for the DNS Factor Model

Likelihood Computation Summary	
Statistic	**Value**
Nonmissing Response Values Used	6324
Estimated Parameters	33
Initialized Diffuse State Elements	3
Normalized Residual Sum of Squares	6320.907
Diffuse Log Likelihood	3548.9546
Profile Log Likelihood	3547.4952

The following statements produce the time series plots of the smoothed estimate of the idealized bond yield ($\theta_t(\tau)$) for bonds with maturities 30, 60, and 120 months (shown in Output 34.7.6). To simplify the display, the plots exclude the time span prior to 1991.

```
proc sgplot data= dnsFor;
    title "The Estimated Yield Surface and the Observed Yields ";
    where maturity in (3 60 120) and date >= '31dec1990'd;
    series x=date y=smoothed_yieldSurface / group=maturity;
    scatter x=date y=yield / group=maturity;
    refline '31dec2000'd / axis=x lineattrs=GraphReference(pattern = Dash)
        name="RefLine"   label="Start of multistep forecasts";
run;
```

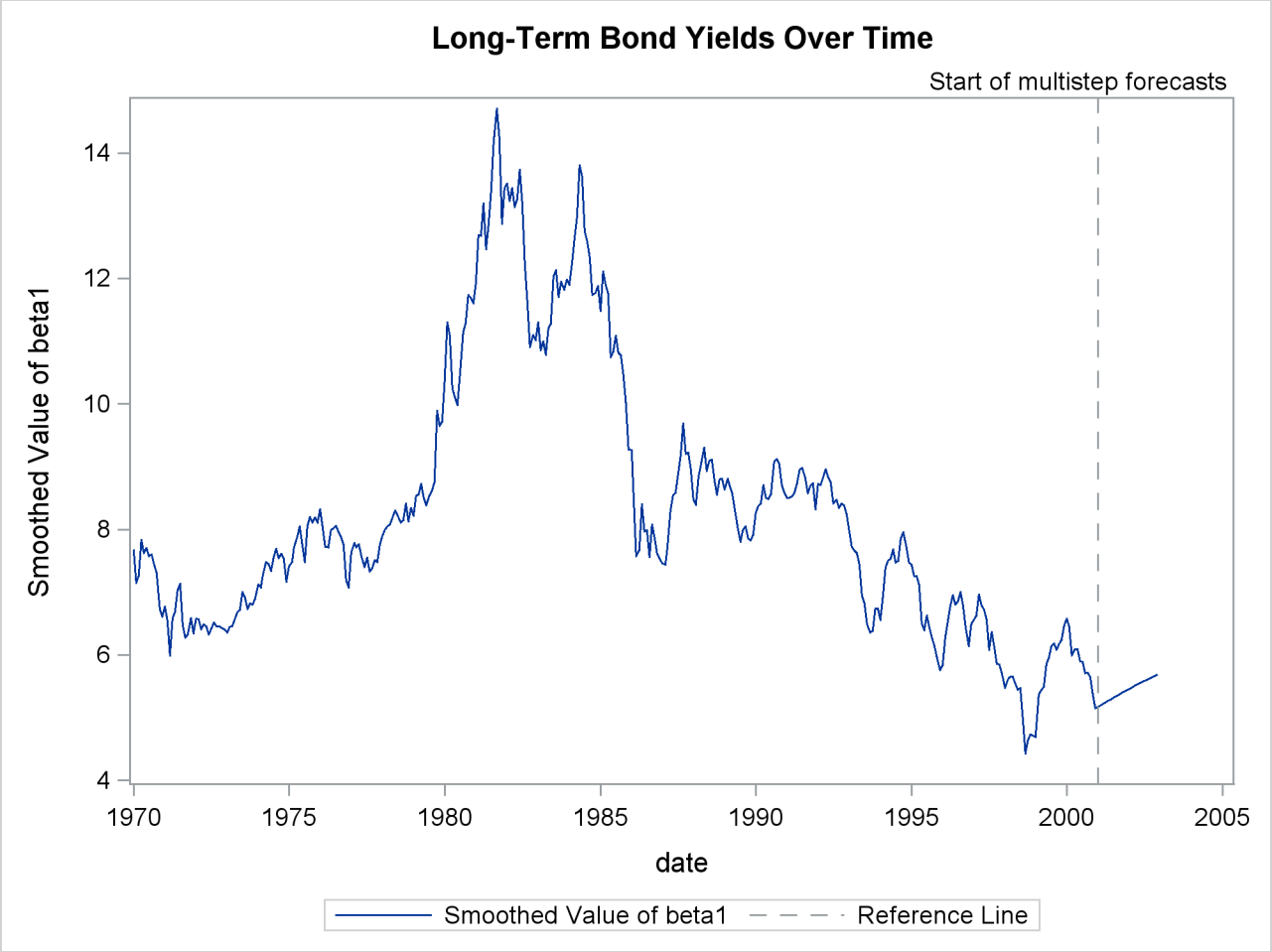
Output 34.7.6 Smoothed Estimate of $\theta_t(\tau)$ for $\tau = 3, 60, 120$



The plots indicate that the DNS model is a reasonable description of the yield data. Similar plots (not shown here) for other maturities also indicate the adequacy of the DNS model. The following statements produce the time series plot of the smoothed estimate of β_{1t}, the long-term bond yield (shown in Output 34.7.7) :

```
proc sgplot data=dnsFor;
   title "Long-Term Bond Yields Over Time ";
   series x=date y=smoothed_beta1 ;
   refline '31dec2000'd / axis=x lineattrs=GraphReference(pattern = Dash)
        name="RefLine"   label="Start of multistep forecasts";
run;
```

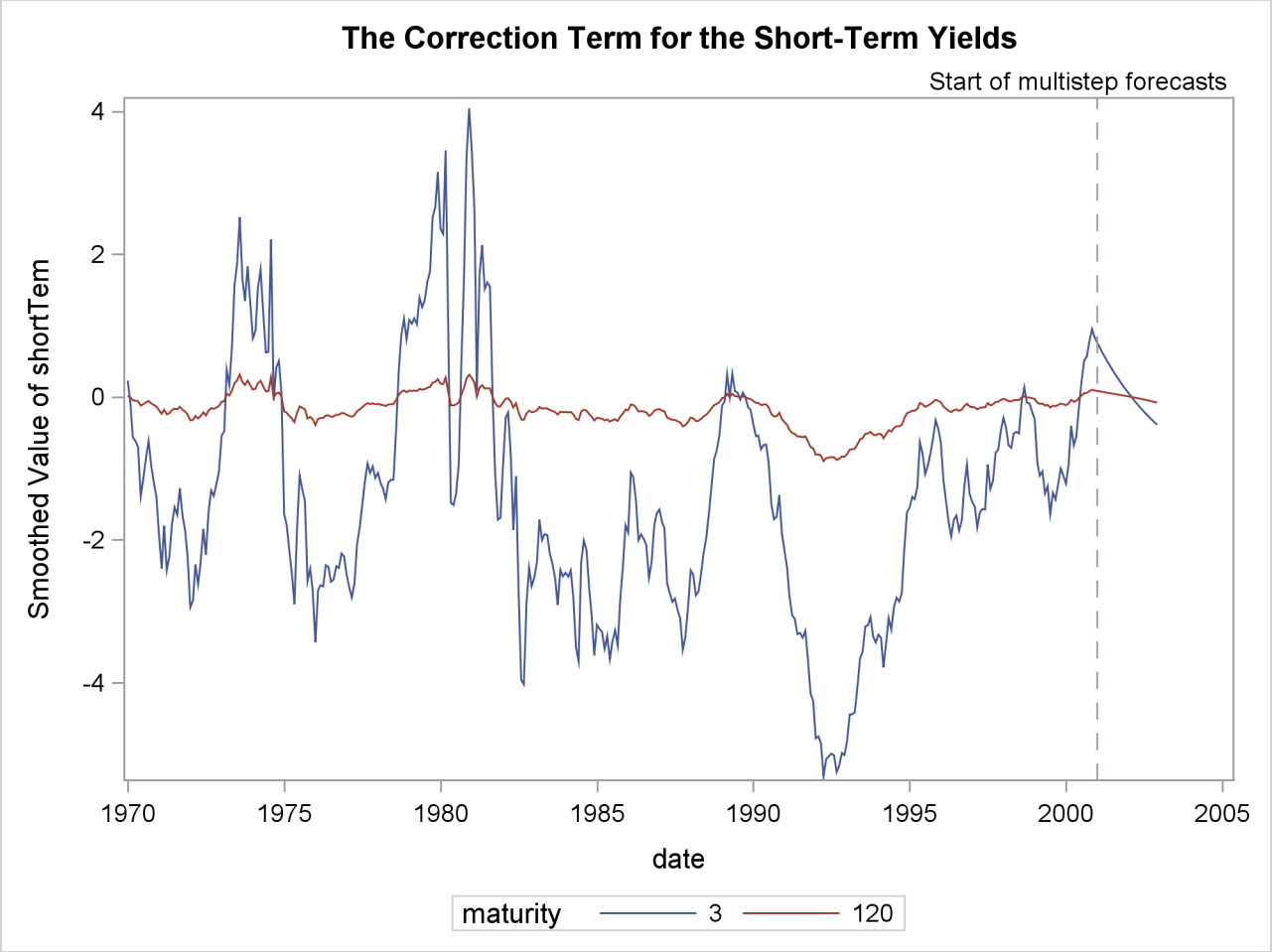
Output 34.7.7 Smoothed Estimate of β_{1t}, the Long-Term Yield



Similarly, Output 34.7.8, which is produced by the following statements, shows the smoothed estimate of the correction to the overall yield that is provided by the second term ($Z2 * \beta_{2t}$) for maturities of 3 months and 120 months. As expected, the correction for the (long-term) maturity of 120 months is negligible compared to the (short-term) maturity of 3 months.

```
proc sgplot data=dnsFor;
  title "The Correction Term for the Short-Term Yields ";
  where maturity in (3 120);
  series x=date y=smoothed_shortTem / group=maturity;
  refline '31dec2000'd / axis=x lineattrs=GraphReference(pattern = Dash)
        name="RefLine"   label="Start of multistep forecasts";
run;
```

Output 34.7.8 Smoothed Estimate of $Z2 * \beta_{2t}$, the Correction Term for the Short-Term Yields

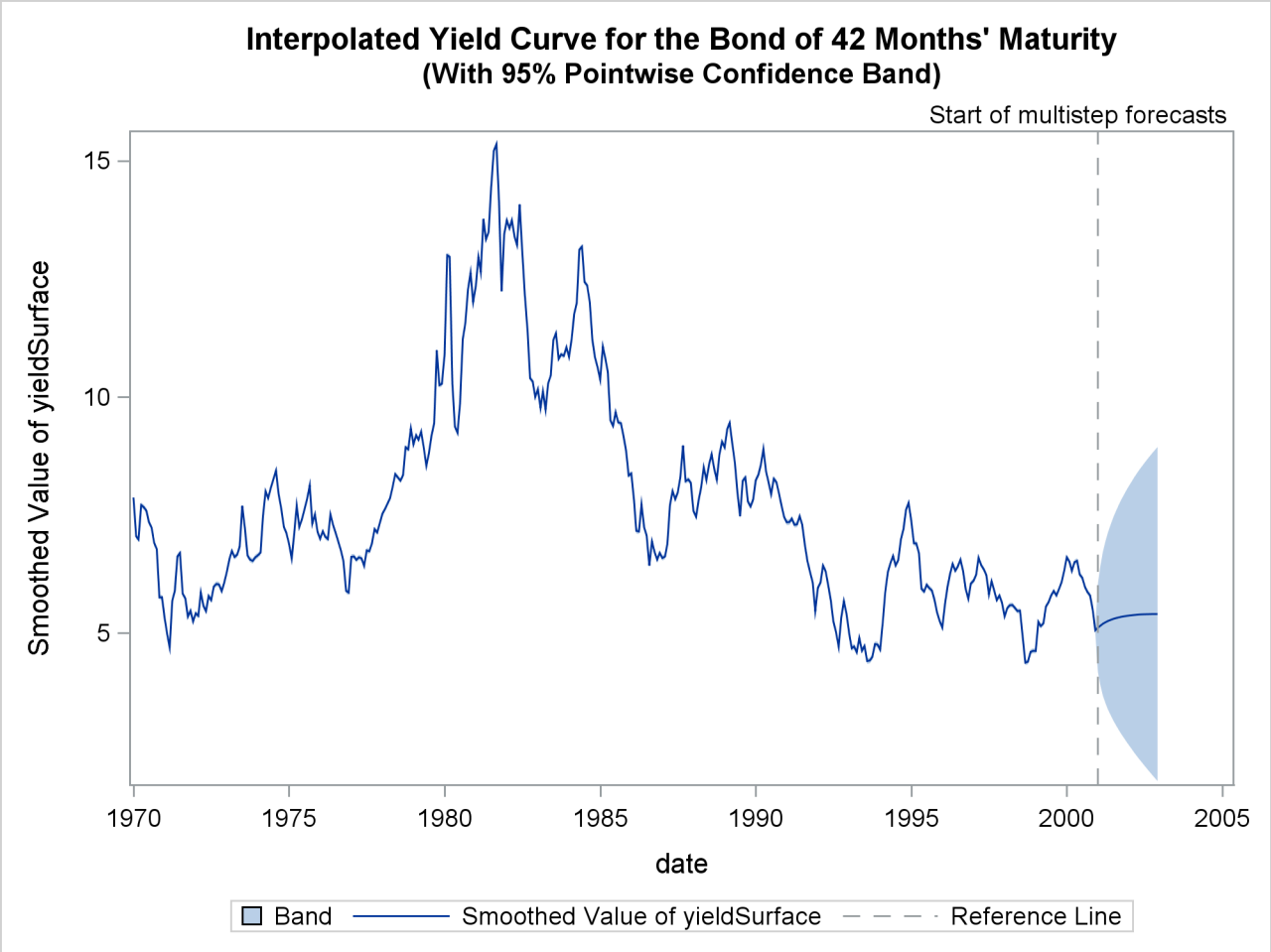


The following statements create plots that show the estimated yield for the hypothetical bond whose maturity is 42 months:

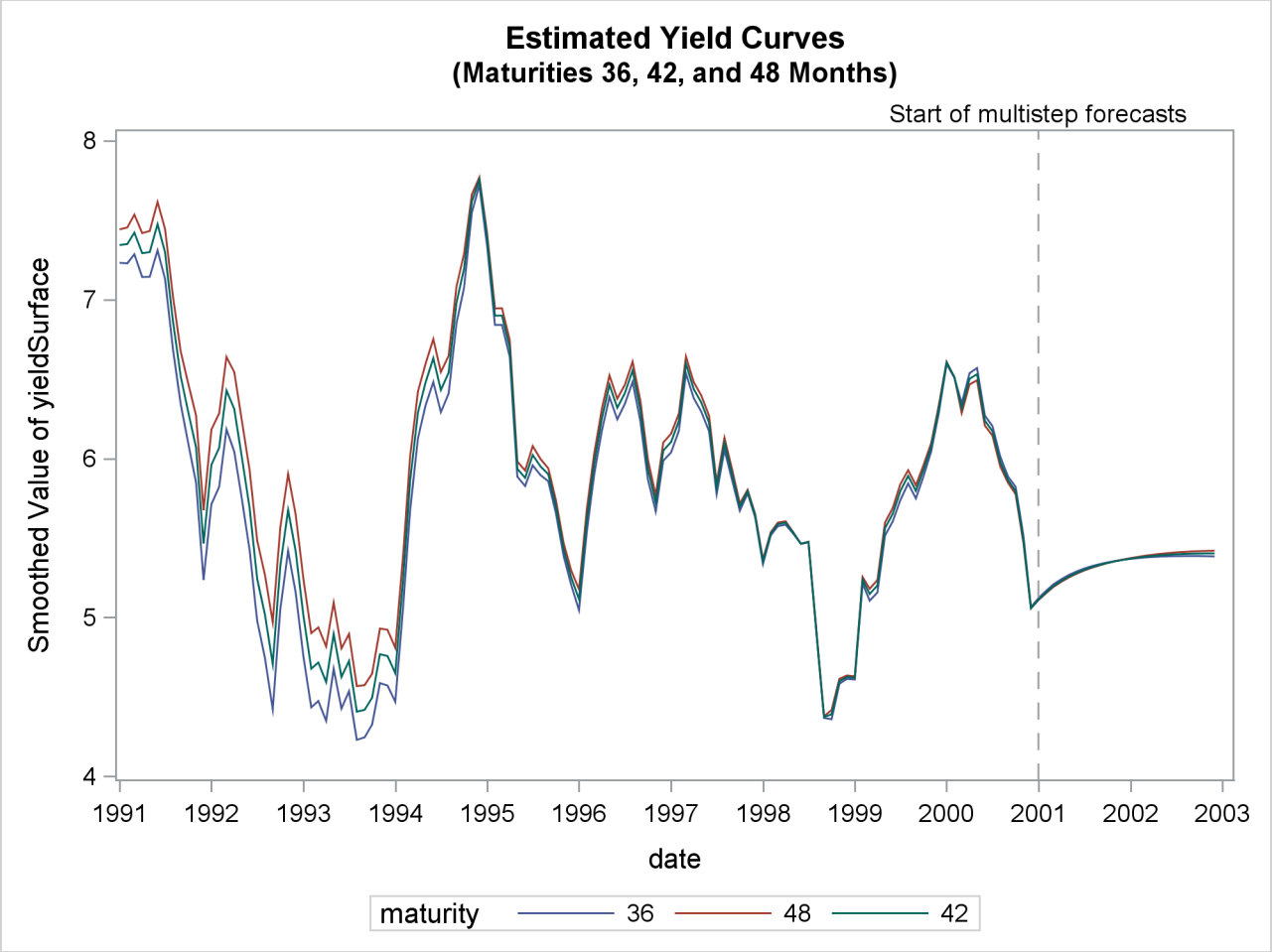
```
proc sgplot data=dnsFor;
   title "Interpolated Yield Curve for the Bond of 42 Months' Maturity";
   title2 "(With 95% Pointwise Confidence Band)";
   where maturity in (42);
   band x=date lower=smoothed_lower_yieldSurface upper=smoothed_upper_yieldSurface;
   series x=date y=smoothed_yieldSurface;
   refline '31dec2000'd / axis=x lineattrs=GraphReference(pattern = Dash)
      name="RefLine"   label="Start of multistep forecasts";
run;

proc sgplot data= dnsFor;
  title "Estimated Yield Curves";
  title2 "(Maturities 36, 42, and 48 Months)";
  where maturity in (36 42 48) and date >= '31dec1990'd;
  series x=date y=smoothed_yieldSurface / group=maturity;
  refline '31dec2000'd / axis=x lineattrs=GraphReference(pattern = Dash)
      name="RefLine"   label="Start of multistep forecasts";
run;
```

Output 34.7.9 shows the interpolated yield curve with a pointwise 95% confidence band. In the historical period, the confidence band appears too tight, mostly because of graphical scaling.

Output 34.7.9 Interpolated Yield Curve for 42 Months' Maturity



Output 34.7.10 Estimated Yield Curves



Output 34.7.10 shows the estimates of $\theta_t(\tau)$ for $\tau = 36$, 42, and 48 months. As expected, the estimated $\theta_t(42)$ lies between the estimates of $\theta_t(36)$ and $\theta_t(48)$.

Example 34.8: Diagnostic Plots and Structural Break Analysis

This example provides information about the diagnostic plots that the SSM procedure produces. In addition, a simple illustration of structural break analysis is also provided. The following plots are available in the SSM procedure:

- a panel of two plots—a histogram and a Q-Q plot—for the normality check of the one-step-ahead residuals $\nu_{t,i}$. A separate panel is produced for each response variable.
- a time series plot of standardized residuals, one per response variable
- a panel of two plots—a histogram and a Q-Q plot—for the normality check of the prediction errors $\mathrm{AO}_{t,i}$. A separate panel is produced for each response variable.
- a time series plot of standardized prediction errors, one per response variable
- a time series plot of maximal state shock chi-square statistics

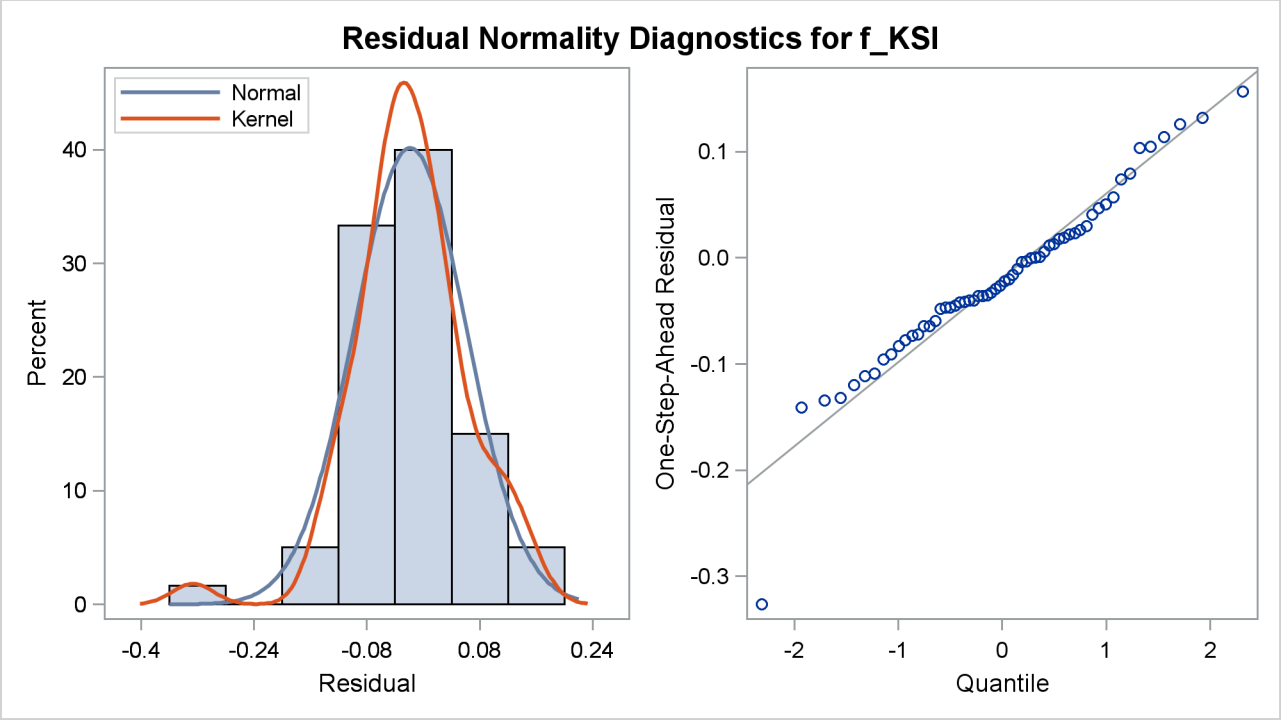
All these plots are used primarily for model diagnostics. In this example, the automobile seat-belt data that are discussed in Example 34.1 are revisited. In Example 34.1, the question under consideration is whether the data show evidence of the effectiveness of the seat-belt law that was introduced in the first quarter of 1983. An intervention variable, Q1_83_Shift, was used in the model to measure the effect of this law on the drivers and front-seat passengers who were killed or seriously injured in car accidents (f_KSI). In the current example, the analysis of these data begins without the knowledge of this seat-belt law. In effect, the same model is fitted without the use of the intervention variable Q1_83_Shift.

The following statements specify the model (without the intervention variable):

```
proc ssm data=seatBelt optimizer(tech=interiorpoint) plots=all;
   id date interval=quarter;
   state error(2) type=WN cov(g);
   component wn1 = error[1];
   component wn2 = error[2];
   state level(2) type=RW cov(rank=1) checkbreak;
   component rw1 = level[1];
   component rw2 = level[2];
   state season(2) type=season(length=4);
   component s1 = season[1];
   component s2 = season[2];
   model f_KSI = rw1 s1  wn1;
   model r_KSI = rw2 s2 wn2;
run;
```

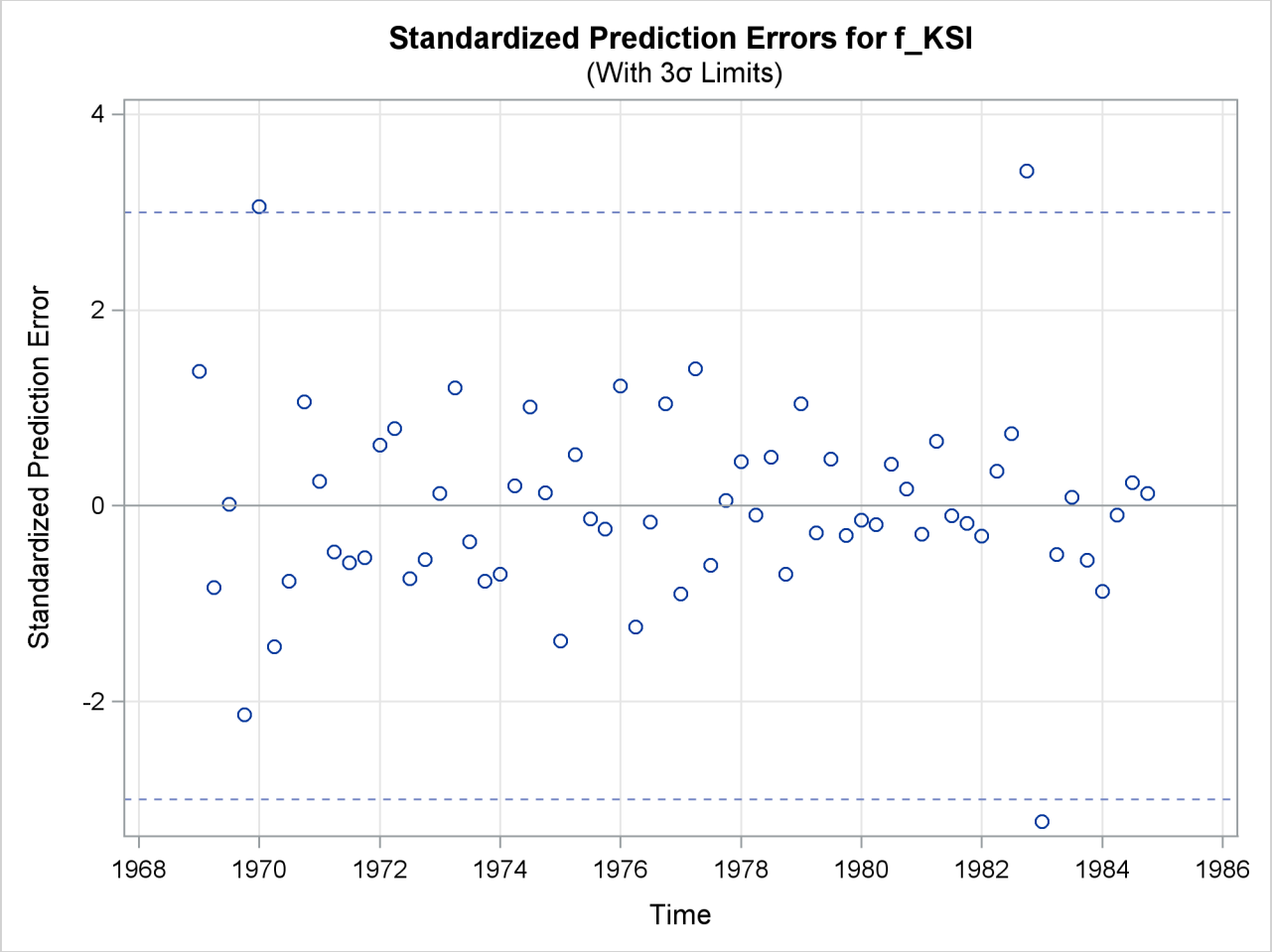
The PLOTS=ALL option in the PROC SSM statement turns on all the plotting options. Because there are two response variables, nine plots in total are produced: a separate set of four plots—two residual and two prediction error—is produced for f_KSI and r_KSI, and one maximal shock plot is produced. Only three of these plots are shown here. Output 34.8.1 shows the normality check for the one-step-ahead residuals for f_KSI. It shows some evidence of lack of normality.

Output 34.8.1 Normality Check of One-Step-Ahead Residuals for f_KSI



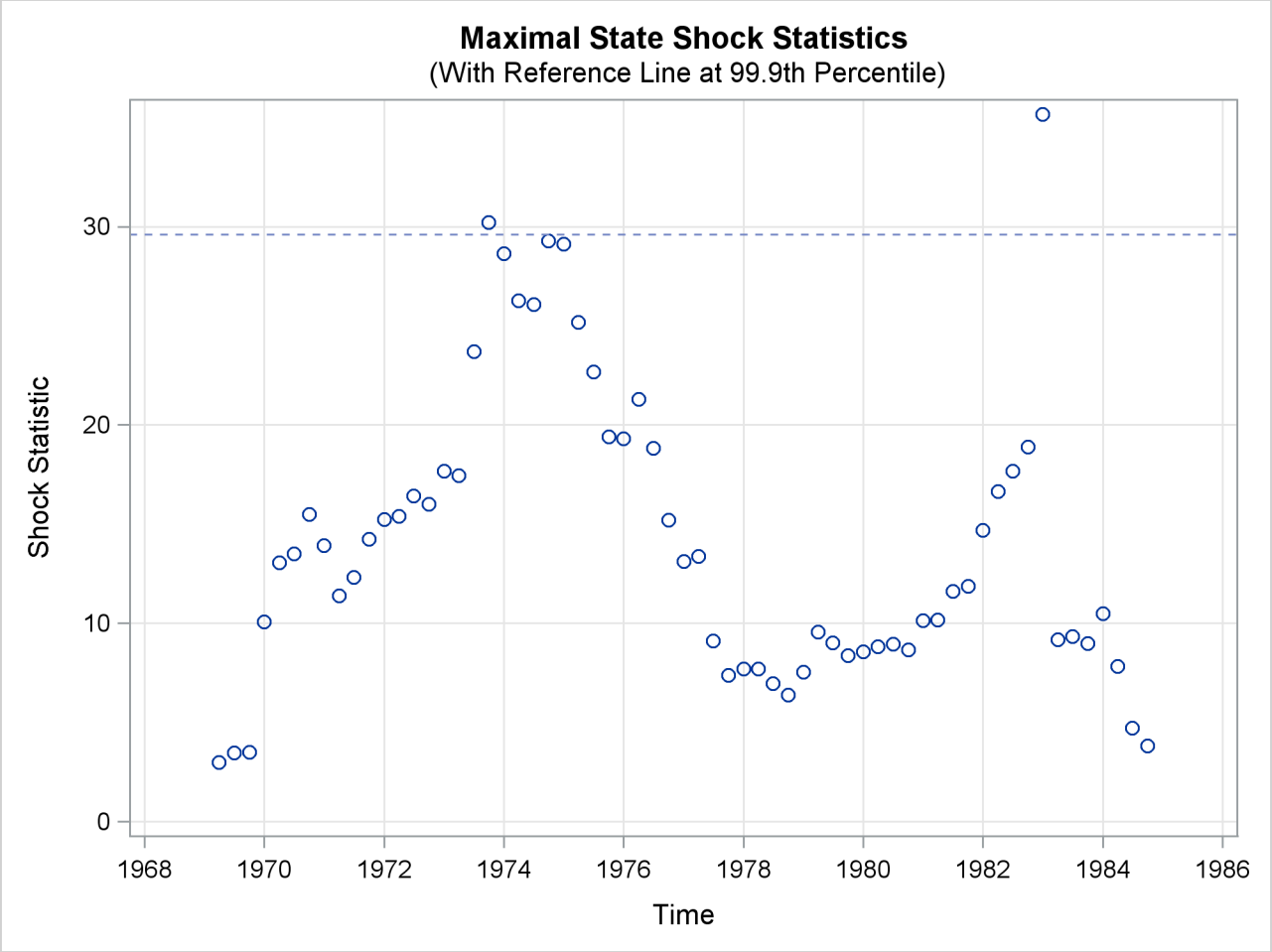
Output 34.8.2 shows the time series plot of standardized prediction errors for f_KSI. It identifies some extreme observations (additive outliers): two near 1983 and one near 1970.

Output 34.8.2 Time Series Plot of Standardized Prediction Errors for f_KSI



Output 34.8.3 shows the time series plot of maximal shock statistics. This plot can be very informative in showing the temporal locations of the structural changes in the overall observation-generation process (treating the fitted model as the reference). It can indicate locations of shifts in the process level or shifts in other characteristics, such as its slope. The precise nature of the shift (whether the shift occurs in the level or in some other aspects) can be determined by using the CHECKBREAK option in the appropriate STATE and TREND statements (as is done in the STATE statement in this example that defines the bivariate state level). In this example, the maximal shock statistics plot indicates two locations—the last quarter of 1973 and the first quarter of 1983—as likely locations for the structural breaks that are associated with the traffic accident process. These are indeed reasonable findings, because the last quarter of 1973 (beginning in October 1973) is associated with the start of the oil crisis that severely curtailed worldwide automobile traffic, and the first quarter of 1983 is associated with the introduction of the seat-belt law that might have improved the safety of drivers and front-seat passengers. In addition, Output 34.8.4 shows the summary of most likely break locations for the bivariate state level. It identifies a break in the first element of level (which corresponds to the drivers and front-seat passengers) in the first quarter of 1983.

Output 34.8.3 Time Series Plot of Maximal Shock Statistics



Output 34.8.4 Elementwise Break Summary for the Bivariate State: level

Elementwise Break Summary for level

Element ID	Index	Z Value	Pr > |z|
1983:1	1	-5.85	<.0001

The following statements fit a revised model that accounts for the break in the first element of level by introducing a dummy variable, Q1_83_Pulse, in the state equation:

```
ods output ElementStateBreakDetails=stateBreak;
proc ssm data=seatBelt optimizer(tech=interiorpoint) plots=all;
   id date interval=quarter;
   Q1_83_Pulse = (date = '1jan1983'd);
   zero = 0;
   state error(2) type=WN cov(g);
   component wn1 = error[1];
   component wn2 = error[2];
   state level(2) type=RW cov(rank=1) W(g)=(Q1_83_Pulse zero)
       checkbreak print=breakdetail;
   component rw1 = level[1];
   component rw2 = level[2];
   state season(2) type=season(length=4);
   component s1 = season[1];
   component s2 = season[2];
   model f_KSI =  rw1 s1  wn1;
   model r_KSI = rw2 s2 wn2;
run;
```

Note that using Q1_83_Pulse in the definition of level is equivalent to using Q1_83_Shift in the MODEL statement for f_KSI in Example 34.1. Output 34.8.5 shows the estimated change in the first element of the state level, which is the same as the estimated level shift shown in Output 34.1.6 (this is not surprising, because these two models are statistically equivalent).

Output 34.8.5 Estimate of the Regression Coefficient of Q1_83_Pulse

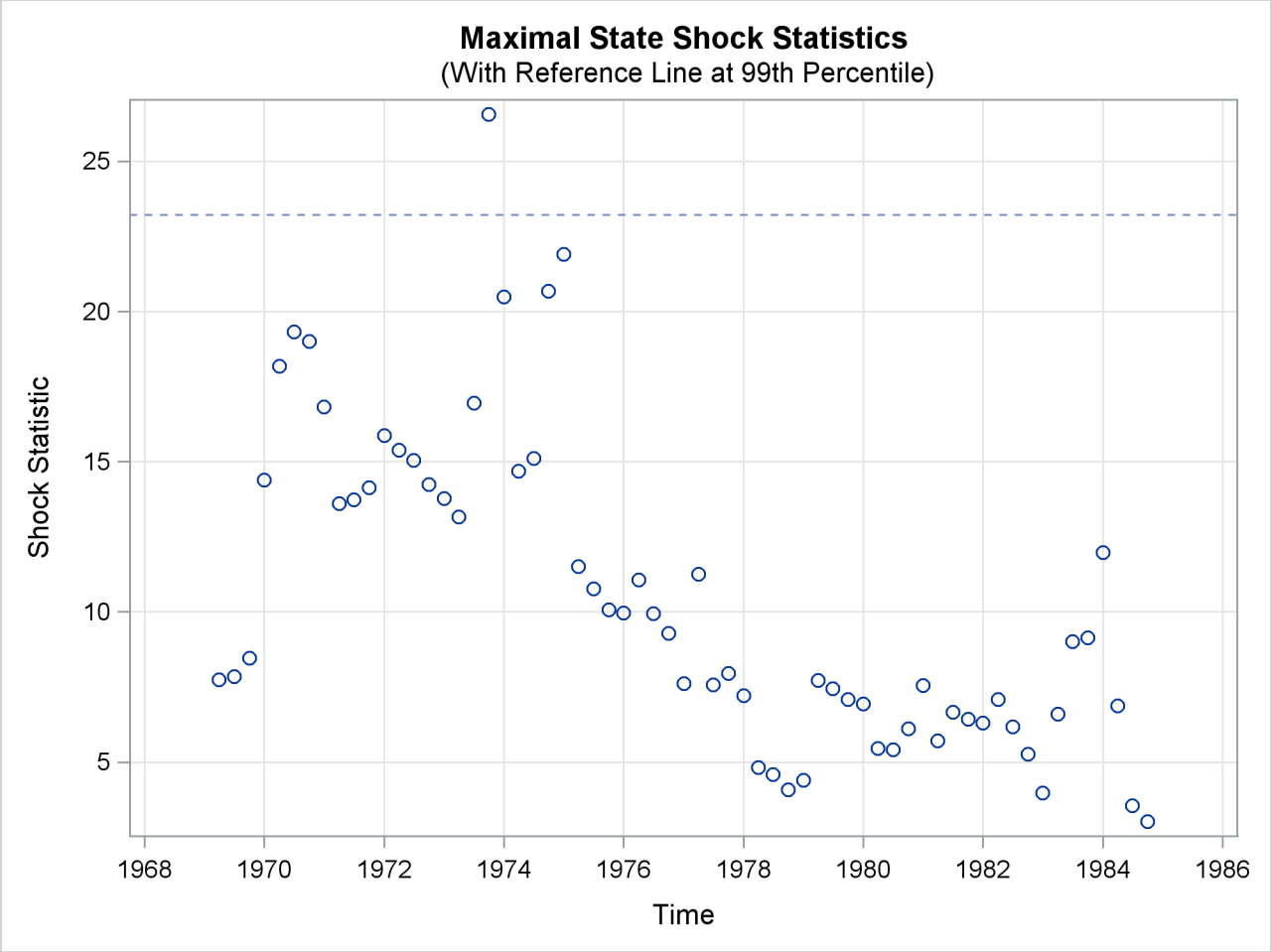
The SSM Procedure

Estimate of the State Equation Regression Vector

State	Element Index	Estimate	Standard Error	t Value	Pr > |t|
level	1	-0.408	0.0259	-15.74	<.0001

In the preceding SSM procedure statements, the CHECKBREAK option is used along with the PRINT=BREAKDETAIL option, which produces a table that contains the break statistics at every distinct time point (this table, in turn, is captured in the output data set stateBreak for later use). Output 34.8.6 shows the time series plot of maximal shock statistics for this revised model. As expected, the plot no longer shows the first quarter of 1983 as a structural break location. It continues to show the last quarter of 1973 as a structural break location, because the fitted model does not try to explicitly account for this shift.

Output 34.8.6 Time Series Plot of Maximal Shock Statistics for the Model with Q1_83_Pulse

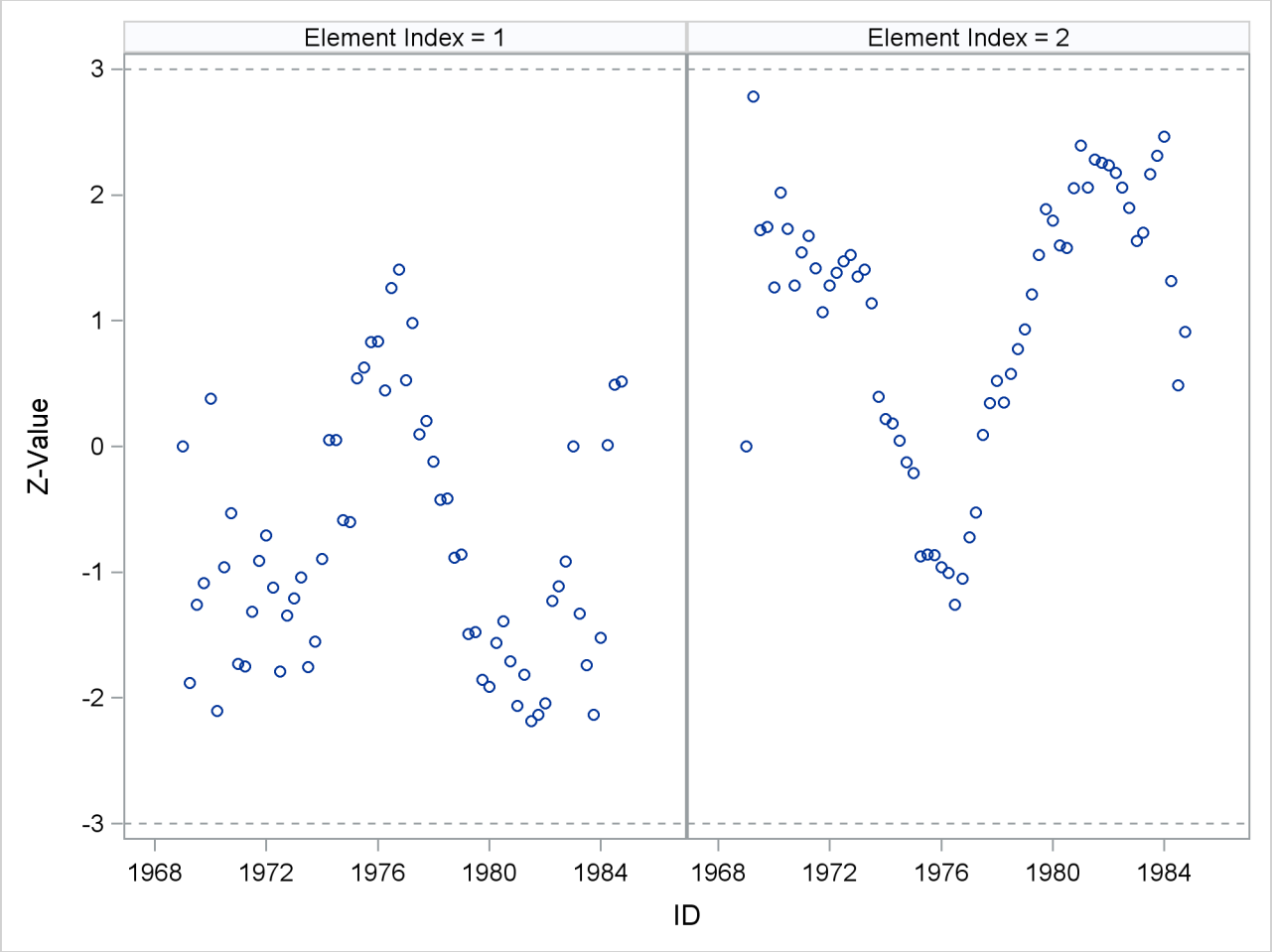


Note that the reference line in Output 34.8.3 is drawn at the 99.9th percentile, whereas the reference line in Output 34.8.6 is drawn at the 99th percentile. The reference line location in the maximal state shock chi-square statistics plot is based on the points in the plot. A reference line is drawn at percentile 80, 90, 99, or 99.9 based on the largest maximal shock statistic that is shown.

The detailed information in the data set stateBreak can be used to further investigate the possibility of significant breaks in the trend in and around 1973. The following statements produce scatter plots for the break statistics for both the drivers and front passengers and the rear passengers (reference lines are also drawn at –3 and 3 to check for extreme Z values):

```
proc sgpanel data=stateBreak;
    panelby elementIndex;
    scatter x=time y=zValue;
    refline 3 / axis=y lineattrs=(pattern=shortdash) noclip;
    refline -3 / axis=y lineattrs=(pattern=shortdash) noclip;
run;
```

Output 34.8.7 Elementwise Structural Break Statistics for level



The resulting graph, shown in Output 34.8.7, shows possible breaks in the second element—rear side passengers—around 1969. In general, however, the evidence of breaks in the elements of level is not very strong. This means that you must look elsewhere to explain the extreme point in Output 34.8.6.

Example 34.9: Longitudinal Data: Variable Bandwidth Smoothing

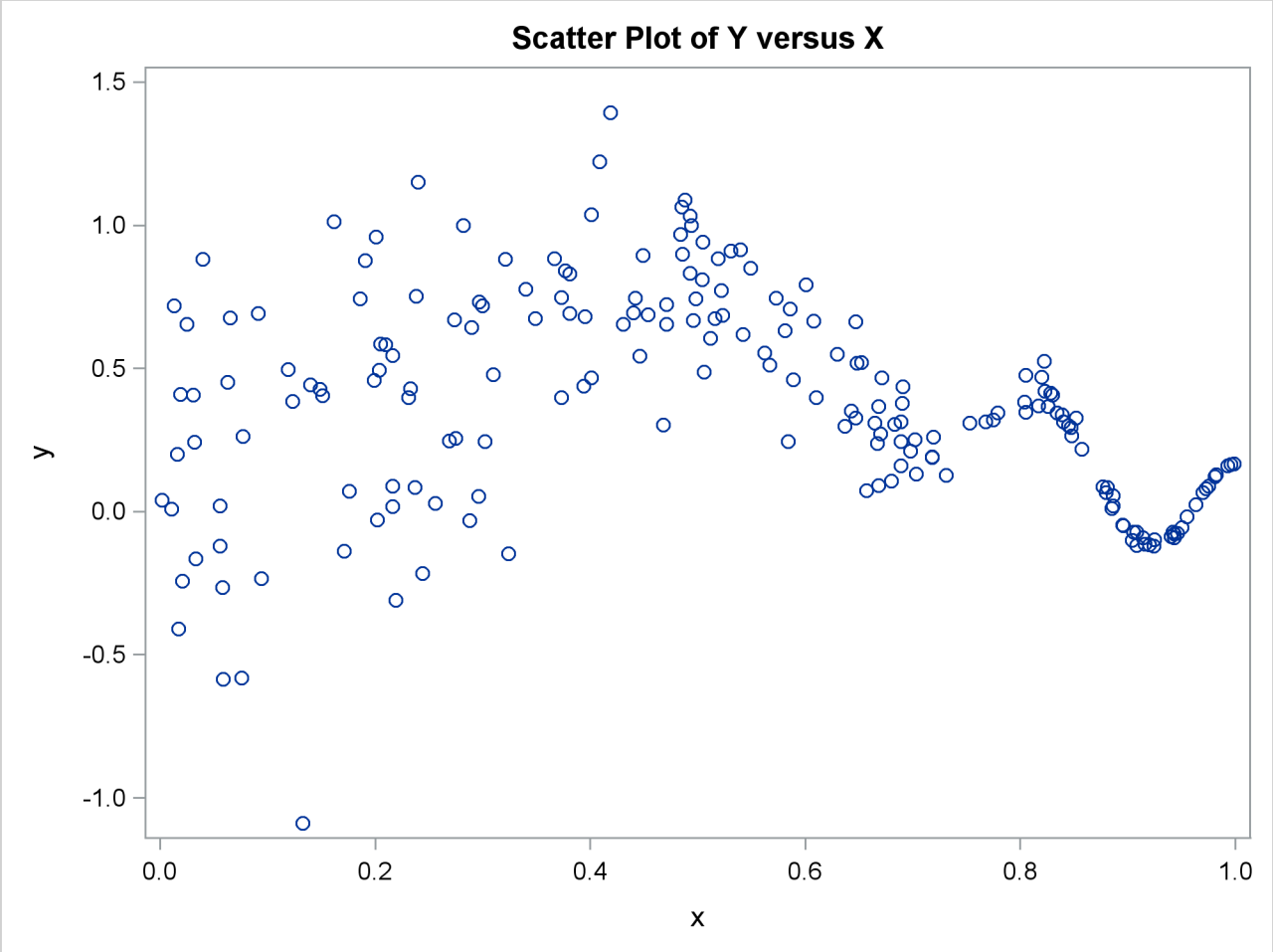
The data for this example, taken from Givens and Hoeting (2005, chap. 11, Example 11.8), contain two variables, x and y. The variable y represents noisy evaluation of an unknown smooth function at x. The data are sorted by x.

```
data Difficult;
input x y;
datalines;
0.002 0.040
0.011 0.009
0.013 0.719
0.016 0.199
0.017 -0.409

  ... more lines ...
```

Output 34.9.1 shows the scatter plot of y against x that is generated by the following statements:

```
proc sgplot data=Difficult;
   title "Scatter Plot of Y versus X";
   scatter x=x y=y ;
run;
```

Output 34.9.1 Scatter Plot of Y versus X



The plot clearly shows that the variance of y values varies considerably over the range of x values—the variance is larger for x values around 0.2 and gets increasingly smaller as the x values get closer to 1. Givens and Hoeting (2005) discuss the difficulties of extracting a smooth pattern from such data. Consider the following model for y:

$$\mathrm{y}(x) = \boldsymbol{\mu}_x + \boldsymbol{\epsilon}_x$$

where $\boldsymbol{\mu}_x$ is a smooth trend component and $\boldsymbol{\epsilon}_x$ is the observation noise with variance, $h(x)$, which changes with x: $\boldsymbol{\epsilon}_x \sim N(0, h(x))$. It is known (Durbin and Koopman 2001, chap. 3, sect. 10 and sect. 11) that modeling the trend $\boldsymbol{\mu}_x$ as a polynomial smoothing spline (for example, the way the growth curves are modeled in Example 34.4) and taking the variance function of the observation noise $\boldsymbol{\epsilon}_x$ a constant results in a trend estimate that can be termed a fixed-bandwidth-smoother. The optimal bandwidth turns out to be a

function of the signal-to-noise ratio: the ratio of the observation noise variance and the disturbance variance of the trend component. On the other hand, allowing the variance function of the observation noise to change with the x values results in a trend estimate that can be termed a variable-bandwidth-smoother. The rest of this example shows how to use the SSM procedure to create a data-dependent variance function $h(x)$ and to extract the associated (variable-bandwidth) smooth trend from such data. Suppose that the (unknown) variance function $h(x)$ can be approximated as

$$h(x) = \exp(\sum_{i=1}^{7} v_i \ \mathrm{SplineBasis}_i(x))$$

where $v_i, i = 1, 2, \ldots, 7$ are unknown parameters and $\mathrm{SplineBasis}_i(x), i = 1, 2, \ldots, 7$, are the full set of cubic spline basis functions (B-splines) with four evenly spaced internal knots between the range of x values—essentially, four equispaced points between 0.0 and 1.0. Note that the number of basis functions in the full set, 7, is the sum of the number of internal knots, 4, and the degree of the polynomial, 3. The following statements create a data set, Combined, that contains the variables x and y, along with the desired spline basis functions (col1–col7) that are created by using the BSPLINE function in PROC IML:

```
proc iml;
   use difficult;
   /* read x and y from difficult into temp */
   read all var _num_ into temp;
   x = temp[,1];
   /* generate B-spline basis for a cubic spline
      with 4 evenly spaced internal knots in the x-range */
   bsp = bspline(x, 2, ., 4);
   Combined = temp || bsp;
   /* create a merged data set with x, y, and
      spline basis columns                    */
   create Combined var {x y col1 col2 col3 col4 col5 col6 col7};
   append from Combined;
quit;
```

The following statements specify and fit the desired model to the data:

```
proc ssm data=Combined opt(tech=dbldog);
   id x;
   /* parameters needed to define h(x) */
   parms v1-v7;
   /* defining h(x) */
   var = exp(v1*col1 + v2*col2 + v3*col3 + v4*col4
      + v5*col5 + v6*col6 + v7*col7);
   /* defining the polynomial spline trend */
   trend trend(ps(2));
   /* defining the observation noise with variance h(x) */
   irregular wn variance=var;
   model y = trend wn;
   output out=For pdv;
run;
```

Output 34.9.2 shows the estimates of $v_i, i = 1, 2, \ldots, 7$, and Output 34.9.3 shows the estimate of the disturbance variance associated with the polynomial spline trend that is specified in the TREND statement.

Output 34.9.2 Estimates of v1–v7

The SSM Procedure

Estimates of Named Parameters

Parameter	Estimate	Standard Error	t Value
v1	-3.302	1.501	-2.20
v2	-0.826	0.619	-1.33
v3	-2.234	0.453	-4.93
v4	-3.130	0.412	-7.60
v5	-4.306	0.415	-10.38
v6	-6.901	0.588	-11.73
v7	-19.514	2.306	-8.46

Output 34.9.3 Estimate of the Disturbance Variance Associated with the Trend

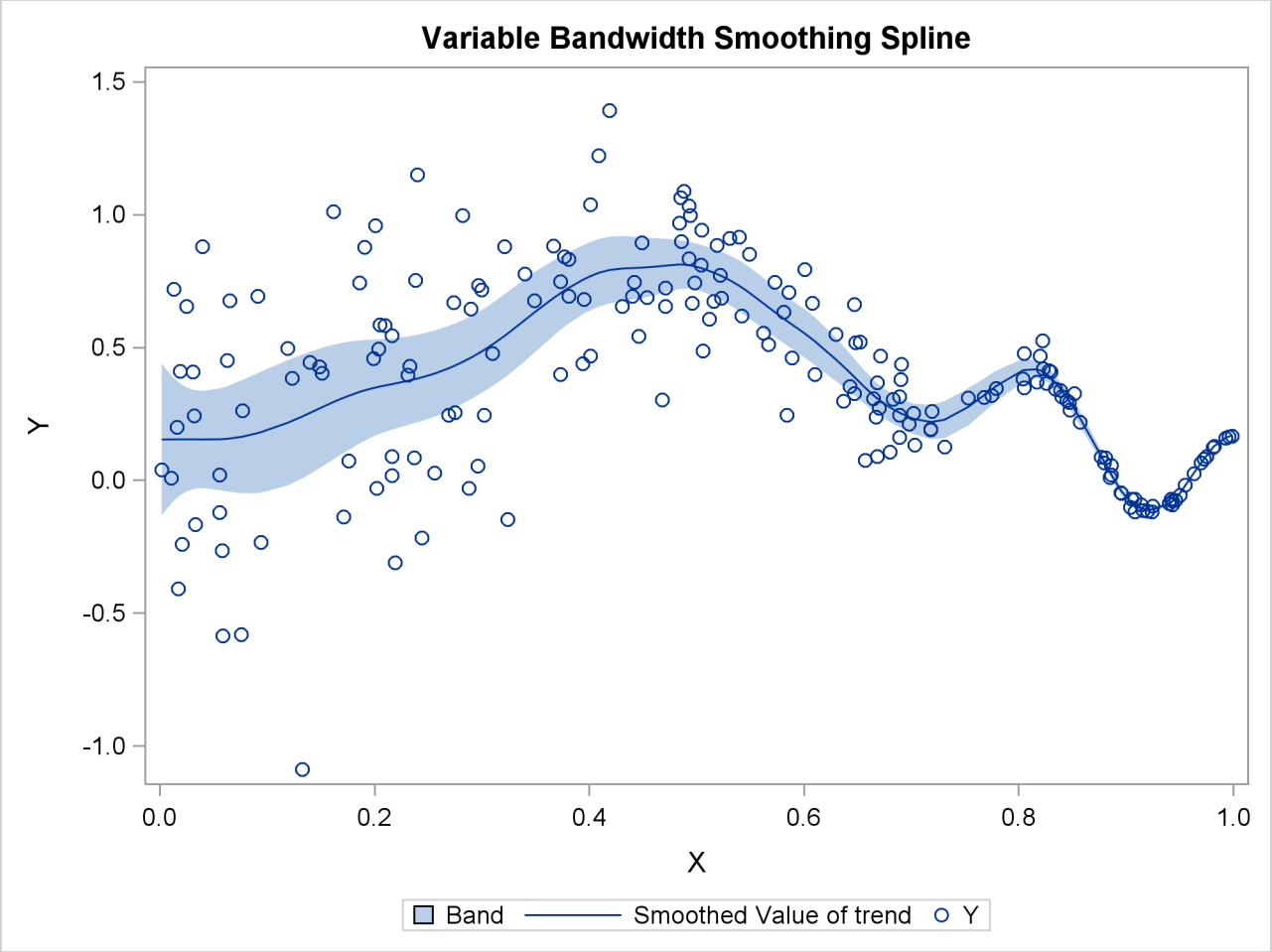
Model Parameter Estimates

Component	Type	Parameter	Estimate	Standard Error	t Value
trend	**PS(2) Trend**	**Level Variance**	339	110	3.07

The following statements produce a plot, shown in Output 34.9.4, of the fitted trend with 95% confidence band:

```
proc sgplot data=For;
   title "Variable Bandwidth Smoothing Spline";
   band x=x lower=smoothed_lower_trend
      upper=smoothed_upper_trend ;
   series x=x y=smoothed_trend;
   scatter x=x y=y;
run;
```

Output 34.9.4 Fitted Trend with 95% Confidence Band

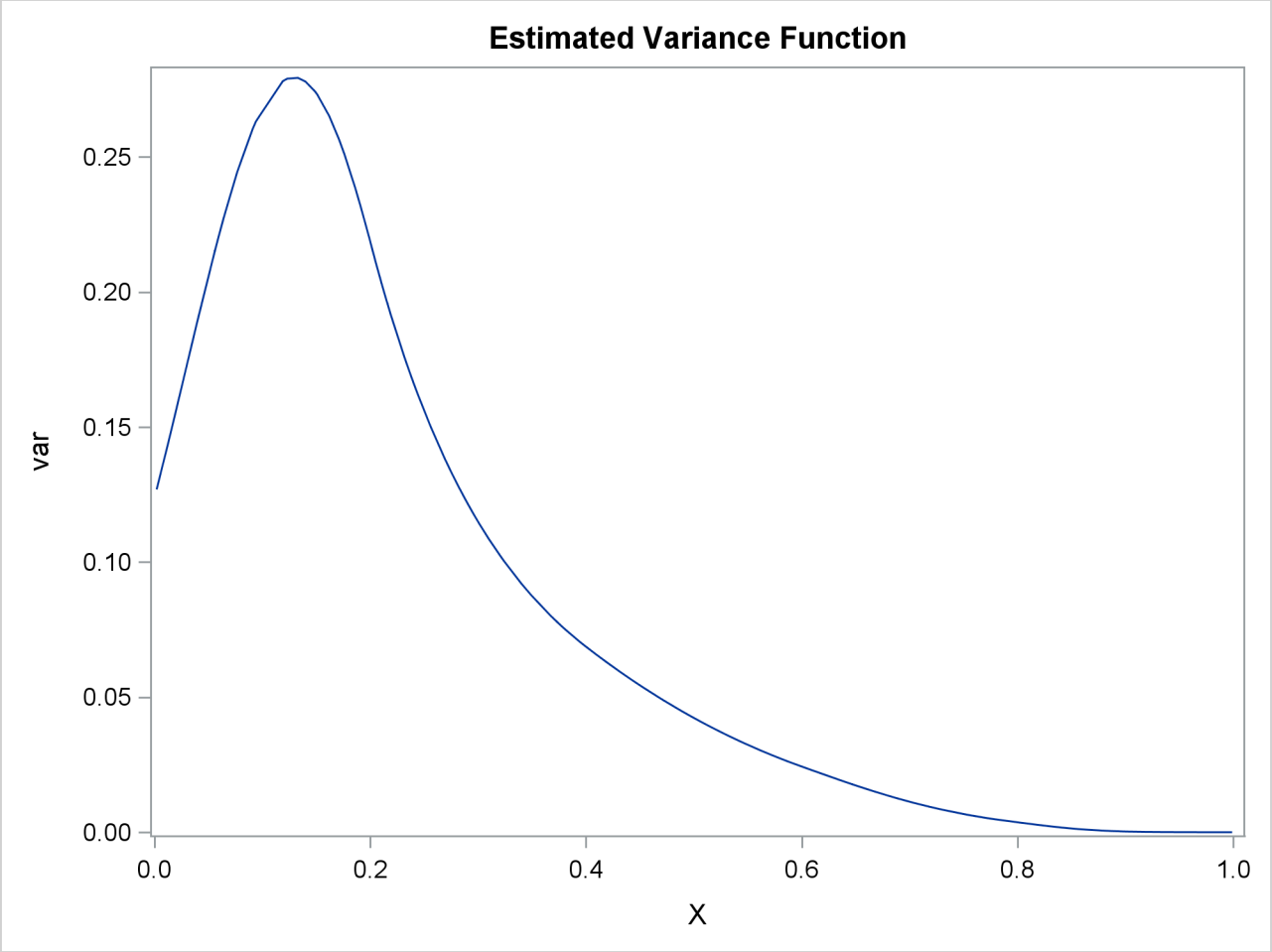


Clearly the fitted curve tracks the data quite well. Lastly, Output 34.9.5 (produced by using the following statements) shows the estimated variance function $h(x)$.

```
proc sgplot data=For;
   title "Estimated Variance Function";
   series x=x y=var;
run;
```

As expected, the curve attains its peak at an x value around 0.18 and decays to nearly 0 as x values reach 1.0.

Output 34.9.5 Estimated Variance Function $h(x) = \exp(\sum_{i=1}^{7} \hat{v}_i \ \mathrm{SplineBasis}_i(x))$



Example 34.10: A Transfer Function Model for the Gas Furnace Data

This example describes how you can include components in your model that follow a transfer function model. Transfer function models, a generalization of distributed lag models, are useful for capturing the contributions from lagged values of the predictor series. Box and Jenkins popularized ARIMA models with transfer function inputs in their famous book (Box and Jenkins 1976). This example shows how you can specify an ARIMA model that is suggested in that book to analyze the data collected in an experiment at a chemical factory. The data set, called Series J by Box and Jenkins, contains sequentially recorded measurements of two variables: x, the input gas rate, and y, the output CO_2. For the output CO_2, Box and Jenkins suggest the model

$$y_t = \mu + f_t + \zeta_t$$

where μ is the intercept, ζ_t is a zero-mean noise term that follows a second-order autoregressive model (that is, $\zeta_t \sim \mathrm{AR}(2)$), and f_t follows a transfer function model

$$f_t = \frac{(\gamma_1 B^3 + \gamma_2 B^4 + \gamma_3 B^5)}{(1 - \delta B)} x_t$$

The model for f_t is specified by using the ratio of two polynomials in the backshift operator B. Alternatively, this model can also be described as follows:

$$f_t = \delta f_{t-1} + \gamma_1 x_{t-3} + \gamma_2 x_{t-4} + \gamma_3 x_{t-5}$$

In this alternate form, it is easy to see that the equation for f_t can also be seen as a state evolution equation for a one-dimensional state with a (1×1) transition matrix δ and state regression variables x_{t-3}, x_{t-4}, and x_{t-5} (lagged values of x). This state equation has no disturbance term.

The following statements define the data set Seriesj. The variables x3, x4, and x5, which denote the appropriately lagged values of x, are also created. These variables are used later in the STATE statement that is used to specify f_t.

```
data Seriesj;
   input x y @@;
   label x = 'Input Gas Rate'
         y = 'Output CO2';
   x3 = lag3(x);
   x4 = lag4(x);
   x5 = lag5(x);
   obsIndex = _n_;
   label obsIndex = 'Observation Index';
datalines;
-0.109  53.8  0.000  53.6  0.178  53.5  0.339  53.5
 0.373  53.4  0.441  53.1  0.461  52.7  0.348  52.4
 0.127  52.2 -0.180  52.0 -0.588  52.0 -1.055  52.4

   ... more lines ...
```

The following SSM procedure statements carry out the modeling of y, the output CO_2, according to the preceding model:

```
proc ssm data=Seriesj(firstobs=6);
    id obsIndex;
    parms delta /lower=-0.9999 upper=0.9999;
    state tfstate(1) T(g)=(delta) W(g)=(x3 x4 x5) a1(1) checkbreak;
    comp tfinput = tfstate[1];
    trend ar2(arma(p=2)) ;
    intercept = 1;
    model y = intercept tfinput ar2 ;
    eval modelCurve = intercept + tfinput;
    forecast out=For;
run;
```

The coefficient of the denominator polynomial, δ, is specified in a PARMS statement. It is constrained to be less than 1 in magnitude, which ensures that the transfer function term does not have explosive growth. The transfer function model for f_t is specified in a STATE statement that defines a one-dimensional state named tfstate. In this statement, the transition matrix (which contains only one element, δ) is specified by using the T(g)= option; the state regression variables x3, x4, and x5 are specified by using the W(g)= option; and the CHECKBREAK option is used to turn on the search for unexpected changes in the behavior of f_t. The COV option is absent from this STATE statement because the disturbance term is absent from the state equation for f_t. Moreover, because nothing can be assumed about the initial condition of this state equation, it is taken to be diffuse (as signified by the A1 option). Note that the first five observations of the input data

set, Seriesj, are excluded from the analysis to ensure that the state regression variables x3, x4, and x5 do not contain any missing values. The component that is associated with tfstate, named tfinput, is specified in a COMPONENT statement that follows the STATE statement. A zero-mean, second-order autoregressive noise term, named ar2, is specified by using a TREND statement. Next, a constant regression variable, intercept, is defined to be used in the MODEL statement to capture the intercept term μ. Finally, the model specification is completed by specifying the response variable, y, and the three right-hand terms in the MODEL statement. Next, an EVAL statement is used to specify a component, modelCurve, which is the sum of the intercept and the transfer function input ($\mu + f_t$). The modelCurve component represents the structural part of the model and is defined only for output purposes: its estimate is output (along with the estimates of other components) to the data set that is specified in the OUT= option of the OUTPUT statement.

Note that the modeling of output CO_2 according to this model is also illustrated in Example 8.3 of the PROC ARIMA documentation (see Chapter 8, "The ARIMA Procedure"). The ARIMA procedure handles the computation of the transfer function f_t slightly differently than the way it is estimated by the SSM procedure. However, despite this algorithmic difference in the modeling procedures, for this example the estimated parameters agree quite closely (barring the sign conventions that are used to specify the model parameters).

Output 34.10.1 shows the estimate of μ, the intercept in the model. Output 34.10.2 shows the estimate of δ, the coefficient in the denominator polynomial of the transfer function. Output 34.10.3 shows the regression estimates of the state regression variables x3, x4, and x5 (which correspond to the coefficients of the numerator polynomial). Output 34.10.4 shows the estimates of the parameters of the AR(2) noise term.

Output 34.10.1 Estimate of μ

The SSM Procedure

Regression Parameter Estimates

Response Variable	Regression Variable	Estimate	Standard Error	t Value	Pr > |t|
y	intercept	53.4	0.145	368.15	<.0001

Output 34.10.2 Estimate of δ, the Coefficient of the Denominator Polynomial in the Transfer Function

Estimates of Named Parameters

Parameter	Estimate	Standard Error	t Value
delta	0.548	0.0396	13.84

Output 34.10.3 Regression Estimates of the State Regression Variables x3, x4, and x5

Estimate of the State Equation Regression Vector

State	Element Index	Estimate	Standard Error	t Value	Pr > |t|
tfstate	1	-0.530	0.0743	-7.13	<.0001
tfstate	2	-0.380	0.1022	-3.72	0.0002
tfstate	3	-0.519	0.0743	-6.99	<.0001

Output 34.10.4 Estimates of the Parameters of the Autoregressive Term

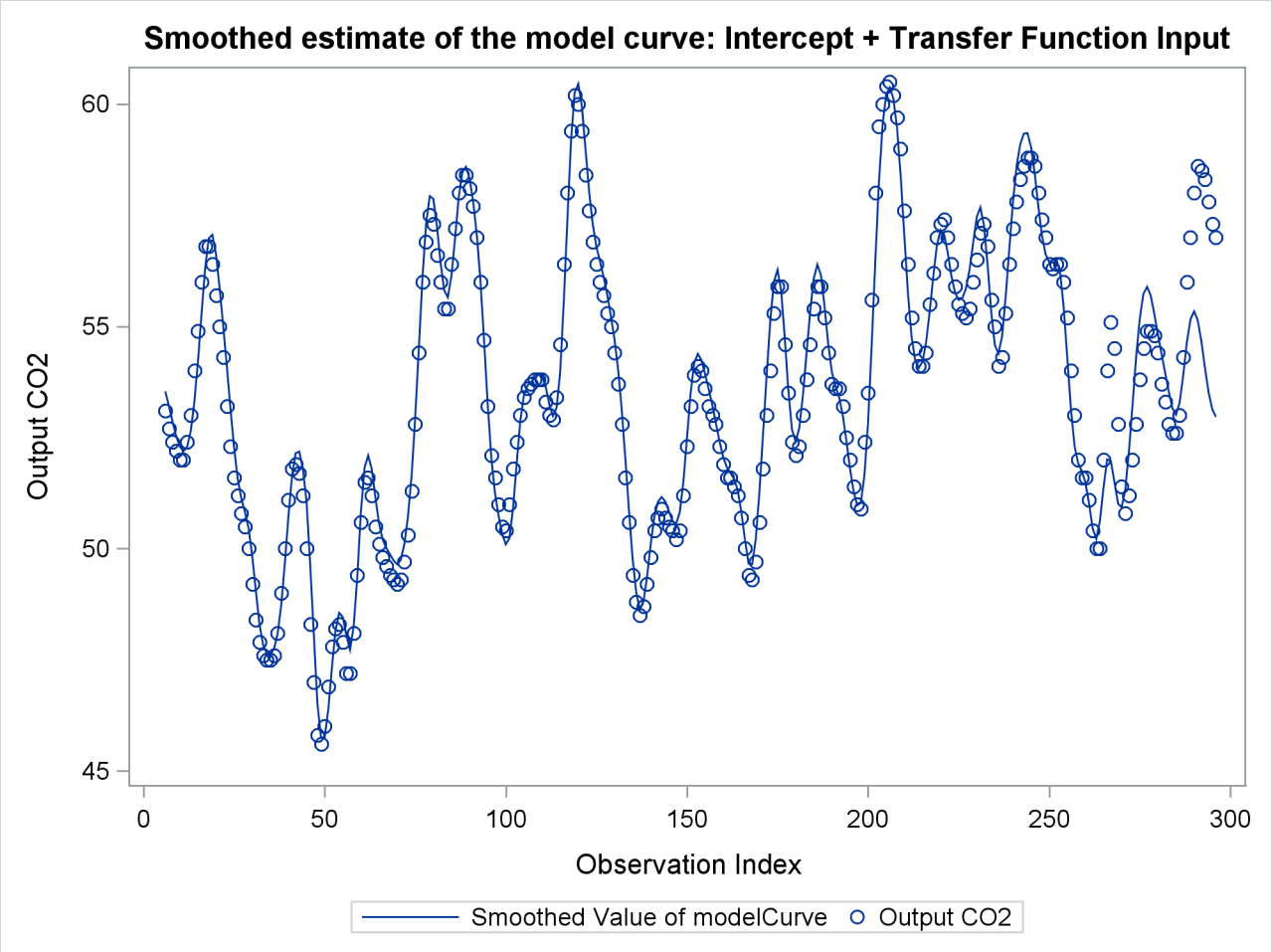
Model Parameter Estimates

Component	Type	Parameter	Estimate	Standard Error	t Value
ar2	ARMA Trend	Error Variance	0.0581	0.00486	11.96
ar2	ARMA Trend	AR_1	1.5320	0.02291	66.86
ar2	ARMA Trend	AR_2	-0.6292	0.01322	-47.58

The following statements produce a time series plot of the estimate of modelCurve—that is, the estimate of the structural part of the model ($\mu + f_t$)—along with the scatter plot of the observed values of the output CO_2. The plot, shown in Output 34.10.5, seems to indicate that the model captures the relationship between the input gas rate and the output CO_2 quite well, at least up to the observation index 250.

```
proc sgplot data=For;
   title "Smoothed estimate of the model curve: Intercept + Transfer Function Input";
   series x=obsIndex y=smoothed_modelCurve;
   scatter x=obsIndex y=y;
run;
```

Output 34.10.5 Smoothed Estimate of $\mu + f_t$

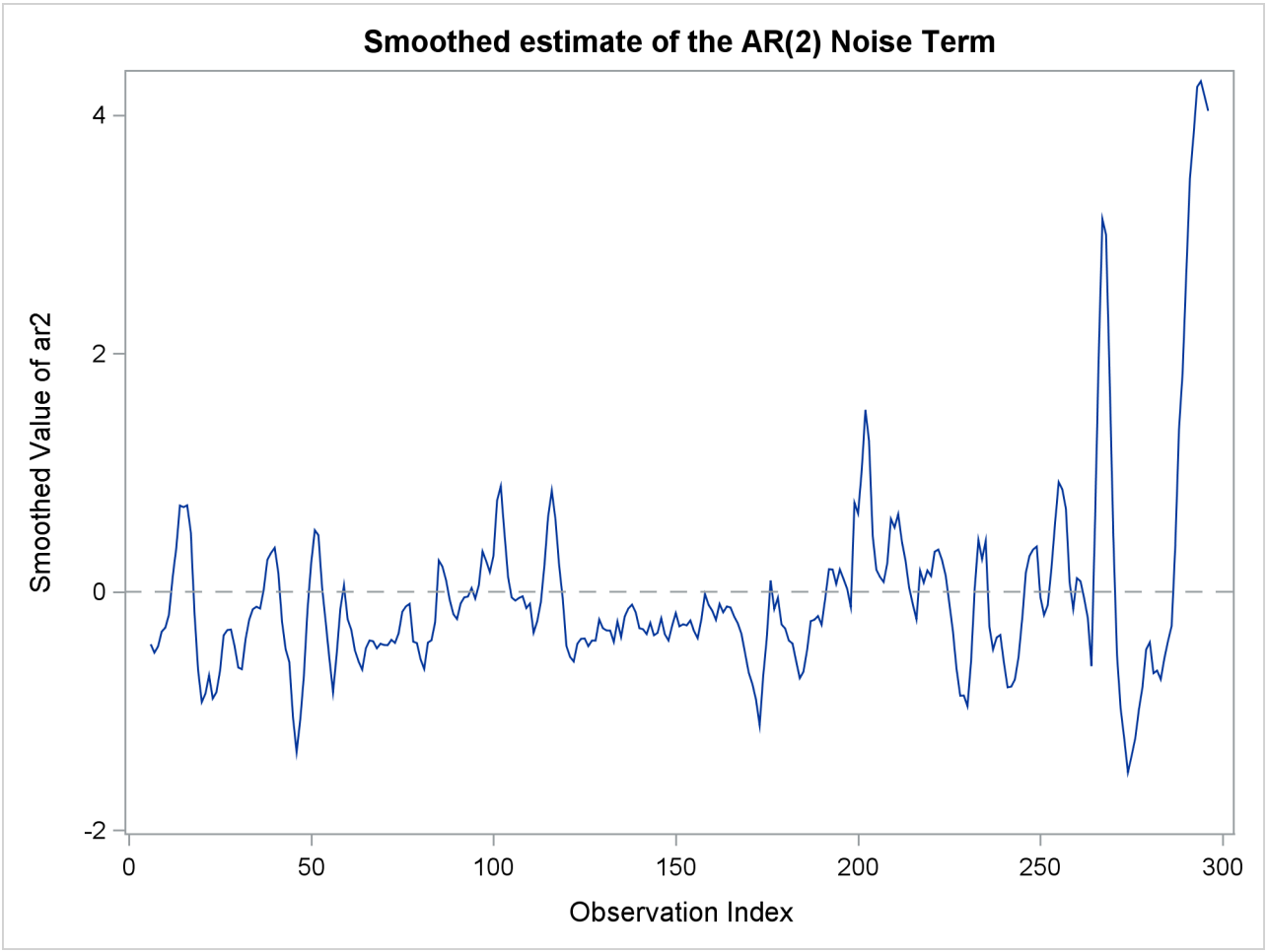


The plot shown in Output 34.10.6 shows the estimate of the noise term, ar2, which is produced by the following statements:

```
proc sgplot data=For;
   title "Smoothed estimate of the AR(2) Noise Term";
   series x=obsIndex y=smoothed_ar2;
   refline 0 / axis=y lineattrs=GraphReference(pattern = Dash);
run;
```

If you compare the scales of these two plots, it appears that the noise term is relatively small and that most of the variation in output CO_2 can be explained by the structural part of the model. It does, however, appear that the model fit deteriorates toward the latter part of the sample. The structural break analysis summary, shown in Output 34.10.7, indicates strong evidence of structural break in the behavior of f_t at or near the observation index 264. Obviously, this type of structural break analysis can be quite useful in industrial quality-control applications.

Output 34.10.6 Smoothed Estimate of the AR(2) Noise Term



Output 34.10.7 Summary of Breaks in f_t

Elementwise Break Summary for tfstate			
ID	Element Index	Z Value	Pr > |z|
264	1	-5.03	<.0001
199	1	4.62	<.0001
198	1	-3.15	0.0016

Example 34.11: Panel Data: Dynamic Panel Model for the Cigar Data

This example shows how you can use the SSM procedure to specify and fit the so-called dynamic panel model, which is commonly used to analyze a panel of time series. Suppose that a panel of time series $y_{t,i}$ follows the model

$$y_{t,i} = \rho y_{(t-1),i} + \mu_i + \beta X_{t,i} + \zeta_t + \epsilon_{t,i}$$

where t denotes the time index (for example, $t = 1, \ldots, T$); i denotes the panel index (for example, $i = 1, \ldots, P$); ρ is the autoregression coefficient; μ_i denote the panel-specific intercepts; $X_{t,i}$ are observations on a regression variable with regression coefficient β (the same for all panels); ζ_t are unobserved, random time effects; and $\epsilon_{t,i}$ are the observation errors. The sequences ζ_t and $\epsilon_{t,i}$ are assumed to be independent, zero-mean Gaussian variables with variances σ_1^2 and σ_0^2, respectively. This is an example of a dynamic panel model that contains one regressor variable. It is easy to formulate this model equation as a state equation with state $\boldsymbol{\alpha}_t$ of size P—the number of panels. Taking $y_{t,i} = \boldsymbol{\alpha}_t[i]$, it is easy to see that the states $\boldsymbol{\alpha}_t$ evolve according to the equation

$$\boldsymbol{\alpha}_{t+1} = \mathbf{T}\boldsymbol{\alpha}_t + \mathbf{W}_{t+1}\boldsymbol{\beta} + \boldsymbol{\eta}_{t+1}$$

where $\mathbf{T} = \rho I_P$ (a P-dimensional, diagonal matrix with all its diagonal elements equal to ρ); $\mathbf{W}_t = (\mathbf{X}_t \;\; I_P)$ is a $P \times (1 + P)$-dimensional matrix (in a block form) of state regression variables, where the first block is a column that includes all the values $X_{t,i}$ that are associated with a given time index (t) and the second block is a P-dimensional identity matrix; $\boldsymbol{\beta} = (\beta \; \mu_1, \ldots, \mu_P)'$ is the $(1 + P)$-dimensional column vector of regression coefficients; and $\boldsymbol{\eta}_t = (\zeta_t + \epsilon_{t,1}, \ldots, \; \zeta_t + \epsilon_{t,P})'$ is a P-dimensional column vector of all the disturbances that are associated with time index t. Because ζ_t and $\epsilon_{t,i}$ are independent, the covariance matrix of $\boldsymbol{\eta}_t$—for example, $\mathbf{Q}_t$—is easy to calculate: $\mathbf{Q}_t[i,i] = \sigma_0^2 + \sigma_1^2$ and, for $i \neq j$, $\mathbf{Q}_t[i,j] = \sigma_1^2$. This formulation can be easily extended to multiple regression variables, such as r variables, by appropriately modifying the term that is associated with the state regression variables—$\mathbf{W}_t\boldsymbol{\beta}$: the new $\mathbf{W}_t$ matrix becomes $P \times (r + P)$-dimensional and the new regression vector $\boldsymbol{\beta}$ becomes $(r + P)$-dimensional.

The cross-sectional data, Cigar, that are used in the section "Getting Started: SSM Procedure" on page 2358 are reused in this example. In order to use the SSM procedure to perform the dynamic panel model–based analysis, the input data set must be reorganized so that it contains the variables that form the $P \times (r + P)$-dimensional matrix $\mathbf{W}_t$. For the Cigar data, the number of panels $P = 46$ (the number of regions considered in the study), and the number of regression variables $r = 3$. Therefore, the input data set needs to be augmented by $46 * (3 + 46) = 2,254$ variables that constitute the matrix $\mathbf{W}_t = (\mathbf{X}_t \quad I_{46})$—the first 46×3-dimensional block $\mathbf{X}_t$ contains the values of the three regression variables, lprice, lndi, and lpimin, at a given time index (a particular year in this case). The following DATA steps accomplish this task in two steps. In the first step, the raw data that form the rows of the Cigar data set are read into a temporary data set, Tmp, such that all 6*46 = 276 values that are associated with a given year (values of six variables—year, region, lsales, lprice, lndi, and lpimin for 46 panels in a given year) are read in a single row that consists of 276 columns. In the second step, the final input data set is formed by rearranging Tmp so that it contains the necessary variables in the proper order—year (the time index), region (the panel index), lsales (the response variable), and the variables that form the 46×49-dimensional $\mathbf{W}$ matrix (w1, . . ., w2254).

```
data Tmp;
    input u1-u276;
datalines;
63 1 4.54223 3.35341 7.3514 3.26194
63 2 4.82831 3.17388 7.5729 3.21487
63 3 4.63860 3.29584 7.3000 3.25037

   ... more lines ...


data cigar(keep=year region lsales w1-w2254);
   array wmat{46, 49} w1-w2254;
   array ivar{46, 6} u1-u276;
   set tmp;
   year = intnx( 'year', '1jan63'd, u1-63 );
   format year year.;
   do i=1 to 46;
      region = ivar[i, 2];
      lsales = ivar[i, 3];
      do j=1 to 46;
          do k=1 to 49;
              wmat[j,k] = 0;
              if k = j+3 then wmat[j,k] = 1;
              if k=1 then wmat[j,k] = ivar[j, 4];
              if k=2 then wmat[j,k] = ivar[j, 5];
              if k=3 then wmat[j,k] = ivar[j, 6];
          end;
      end;
      output;
   end;
run;
```

The following statements specify and fit the dynamic panel model:

```
proc ssm data=Cigar opt(tech=dbldog maxiter=75);
    id year interval=year;
    parms rho / lower=-0.9999 upper=0.9999;
    parms sigma0 sigma1 / lower=1.e-8;
    array RegionArray{46} region1-region46;
    do i=1 to 46;
       RegionArray[i] = (region=i);
    end;
    array cov{46,46};
    do i=1 to 46;
        do j=1 to 46;
           if(i=j) then cov[i,j] = sigma0 + sigma1;
           else cov[i,j] = sigma1;
        end;
    end;
    state panelState(46) T(I)=(rho) W(g)=(w1-w2254)
      cov(g)=(cov) a1(46) checkbreak;
    comp dynPanel = (RegionArray)*panelState;
    model lsales = dynPanel;
    output out=for1 press;
run;
```

The estimates of the regression coefficients and the regional intercepts, which are all statistically significant, are shown in Output 34.11.1. In particular, the estimated coefficients of lprice, lndi, and lpimin, are –0.26, 0.13, and 0.07, respectively.

Output 34.11.1 Estimates of β_1, β_2, β_3 and the Regional Intercepts

The SSM Procedure

Estimate of the State Equation Regression Vector

State	Element Index	Estimate	Standard Error	t Value	Pr > |t|
panelState	1	-0.2627	0.0178	-14.79	<.0001
panelState	2	0.1340	0.0130	10.30	<.0001
panelState	3	0.0748	0.0198	3.78	0.0002
panelState	4	0.4265	0.0581	7.35	<.0001
panelState	5	0.3825	0.0605	6.32	<.0001
panelState	6	0.4425	0.0582	7.61	<.0001
panelState	7	0.3471	0.0631	5.50	<.0001
panelState	8	0.3686	0.0635	5.81	<.0001
panelState	9	0.4357	0.0614	7.10	<.0001
panelState	10	0.3753	0.0655	5.73	<.0001
panelState	11	0.4249	0.0606	7.01	<.0001
panelState	12	0.4185	0.0604	6.92	<.0001
panelState	13	0.3824	0.0602	6.35	<.0001
panelState	14	0.3942	0.0644	6.12	<.0001
panelState	15	0.4154	0.0626	6.64	<.0001
panelState	16	0.3961	0.0610	6.49	<.0001
panelState	17	0.3765	0.0618	6.10	<.0001
panelState	18	0.4528	0.0608	7.44	<.0001
panelState	19	0.4316	0.0586	7.36	<.0001
panelState	20	0.4357	0.0601	7.25	<.0001
panelState	21	0.3771	0.0639	5.90	<.0001
panelState	22	0.3939	0.0629	6.26	<.0001
panelState	23	0.4122	0.0621	6.64	<.0001
panelState	24	0.3949	0.0605	6.52	<.0001
panelState	25	0.4386	0.0565	7.77	<.0001
panelState	26	0.4118	0.0627	6.57	<.0001
panelState	27	0.3898	0.0604	6.45	<.0001
panelState	28	0.3818	0.0613	6.23	<.0001
panelState	29	0.4343	0.0632	6.87	<.0001
panelState	30	0.4619	0.0625	7.39	<.0001
panelState	31	0.3730	0.0636	5.86	<.0001
panelState	32	0.3784	0.0589	6.43	<.0001
panelState	33	0.3825	0.0625	6.12	<.0001
panelState	34	0.3784	0.0598	6.32	<.0001
panelState	35	0.4093	0.0628	6.52	<.0001
panelState	36	0.4155	0.0597	6.96	<.0001
panelState	37	0.3960	0.0615	6.44	<.0001
panelState	38	0.4075	0.0602	6.77	<.0001
panelState	39	0.4045	0.0586	6.91	<.0001
panelState	40	0.3918	0.0599	6.55	<.0001
panelState	41	0.4350	0.0608	7.16	<.0001
panelState	42	0.4007	0.0602	6.65	<.0001
panelState	43	0.3196	0.0597	5.36	<.0001
panelState	44	0.4337	0.0609	7.12	<.0001
panelState	45	0.3790	0.0634	5.98	<.0001

Output 34.11.1 *continued*

The SSM Procedure

Estimate of the State Equation Regression Vector

State	Element Index	Estimate	Standard Error	t Value	Pr > |t|
panelState	46	0.3767	0.0618	6.10	<.0001
panelState	47	0.4392	0.0597	7.36	<.0001
panelState	48	0.3932	0.0603	6.51	<.0001
panelState	49	0.3938	0.0616	6.40	<.0001

Output 34.11.2 shows the estimates of the autoregression coefficient ρ, the observation error variance σ_0^2, and the variance of the time effect (variance of ζ) σ_1^2.

Output 34.11.2 Estimates of ρ, σ_0^2, and σ_1^2

Estimates of Named Parameters

Parameter	Estimate	Standard Error	t Value
rho	0.831679	0.0124338	66.89
sigma0	0.001231	0.0000491	25.08
sigma1	0.000213	0.0000662	3.22

Finally, you can compare the fit of the dynamic panel model with the fit of the model that is discussed in the section "Getting Started: SSM Procedure" on page 2358. Output 34.11.3 shows the likelihood-based information criteria for the dynamic panel model, and Output 34.11.4 shows the same information for the other model.

Output 34.11.3 Likelihood-Based Information Criteria: Dynamic Panel Model

Information Criteria

Statistic	Diffuse Likelihood Based	Profile Likelihood Based
AIC (lower is better)	-4732.722	-4856.398
BIC (lower is better)	-4717.247	-4343.874
AICC (lower is better)	-4732.704	-4841.250
HQIC (lower is better)	-4726.913	-4664.667
CAIC (lower is better)	-4714.247	-4245.874

Output 34.11.4 Likelihood-Based Information Criteria: Getting Started Example

Information Criteria		
Statistic	Diffuse Likelihood Based	Profile Likelihood Based
AIC (lower is better)	-4488.093	-4145.246
BIC (lower is better)	-4477.776	-3637.952
AICC (lower is better)	-4488.084	-4130.417
HQIC (lower is better)	-4484.220	-3955.472
CAIC (lower is better)	-4475.776	-3540.952

Similarly, Output 34.11.5 shows fit criteria based on the delete-one cross validation error for the dynamic panel model, and Output 34.11.6 shows the same information for the other model.

Output 34.11.5 Delete-One Cross Validation Criteria: Dynamic Panel Model

Delete-One Cross Validation Error Criteria			
Variable	N	PRESS	Generalized Cross-Validation
lsales	1380	1.115309	5.62798E-7

Output 34.11.6 Delete-One Cross Validation Criteria: Getting Started Example

Delete-One Cross Validation Error Criteria			
Variable	N	PRESS	Generalized Cross-Validation
lsales	1380	1.290420	6.18144E-7

On the basis of both these considerations, the dynamic panel model appears to provide a better fit for the Cigar data than the model that is fit in the section "Getting Started: SSM Procedure" on page 2358.

Example 34.12: Multivariate Modeling: Long-Term Temperature Trends

In a presentation by Ansley and de Jong (2012), three monthly time series are jointly modeled to obtain long-term—several decades long—temperature predictions for certain regions of the northern hemisphere. This example shows how you can specify and fit the final model that this presentation proposed. The following data set, Temp, contains four variables: date dates the monthly observations; UAH contains monthly satellite global temperature readings, starting in December 1978; CRU contains monthly temperature data, starting in January 1850 (from a different source); and GISS contains monthly temperature data, starting in January 1880 (from yet another source). All these temperature data are scaled suitably so that the numbers represent temperature readings in centigrade.

```
data Temp;
input UAH CRU GISS @@;
date = intnx('month', '01jan1850'd, _n_-1);
format date date.;
datalines;
.    8.243    .
.    9.733    .

   ... more lines ...
```

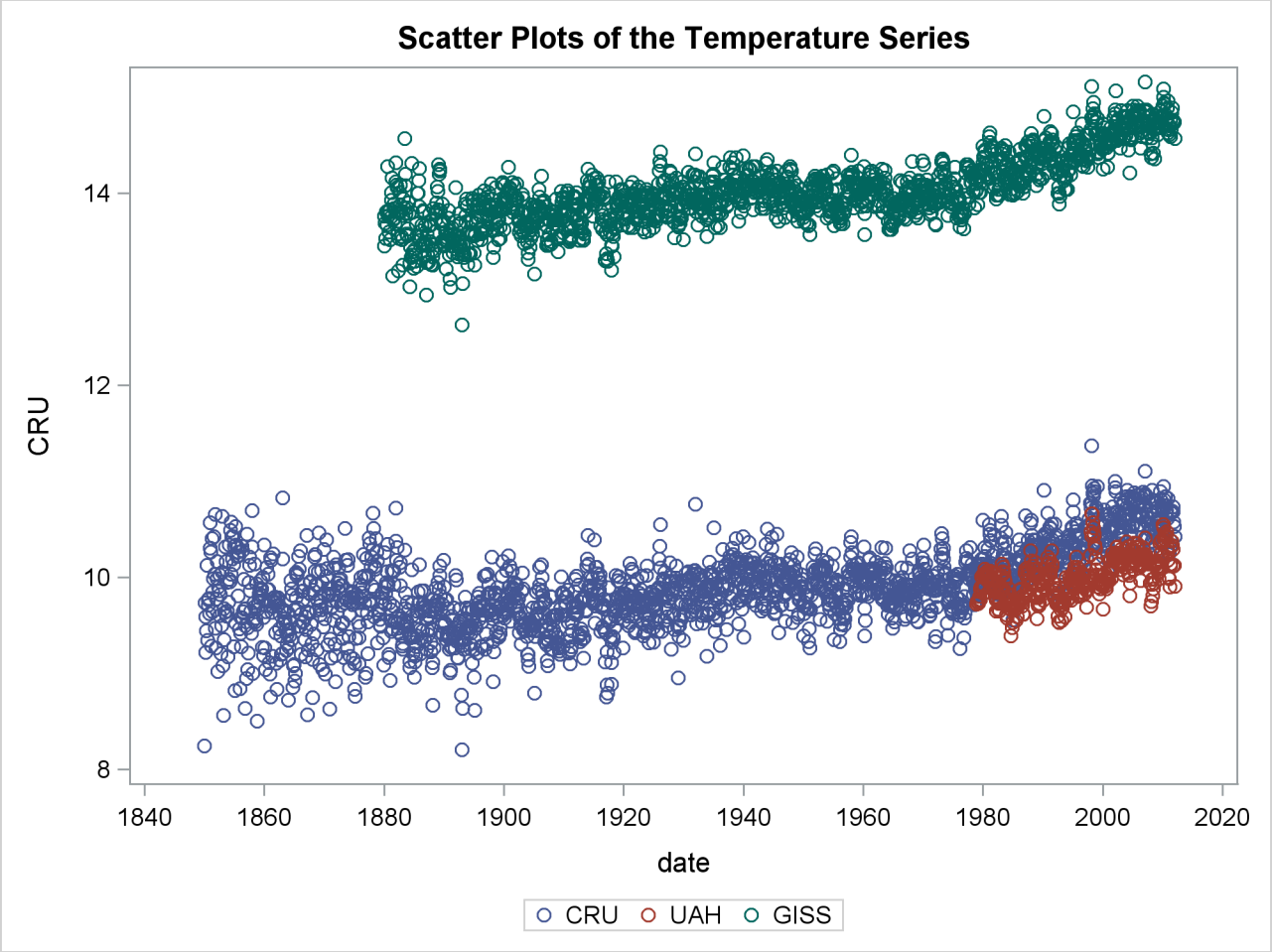
The following statements produce scatter plots of these three series in a single graph:

```
 proc sgplot data=Temp;
       title "Scatter Plots of the Temperature Series";
       scatter x=date y=cru;
       scatter x=date y=uah ;
       scatter x=date y=giss;
  run;
```

Output 34.12.1 shows the resulting graph. As already noted, these three series start at different points in the past. However, they all end at the same time: they all have measurements until January 2012, which is the last month in the data set. The mean levels of these series are different: the GISS measurements are generally larger than CRU and UAH by about 4 degrees. In addition, the variability in the CRU values seems to decrease with time (this is more apparent when the series is plotted by itself). The goal of the analysis is to use these data to make long-term predictions about future temperature levels. The following statements append 1,200 missing measurements to Temp, so that the model fitted by using the SSM procedure can be extrapolated to obtain temperature forecasts 100 years in the future:

```
data append(keep=date cru giss uah);
    do i=1 to 1200;
        cru = .; giss=.; uah=.;
        date = intnx('month','01jan2012'D, i);
    format date monyy7.;
    output;
    end;
run;

 proc append base=Temp data=Append; run;
```

Output 34.12.1 Scatter Plots of CRU, UAH, and GISS



Ansley and de Jong propose a parsimonious model that links these three time series. It can be described as

$$
\begin{aligned}
\mathrm{GISS}_t &= \mu_t + a\,\zeta_t + a\,r_1\,\epsilon_{1t} \\
\mathrm{CRU}_t &= \beta_{cru} + \mu_t + a\,\zeta_t + a\,\epsilon_{2t} \\
\mathrm{UAH}_t &= \beta_{uah} + \mu_t + a\,\zeta_t + a\,r_3\,\epsilon_{3t}
\end{aligned}
$$

where β_{cru} and β_{uah} are intercepts that are associated with CRU and UAH, respectively; μ_t is an integrated random walk trend; ζ_t is a zero-mean, autoregressive noise term (which is scaled by an unknown scaling factor a); and ϵ_{it} $(i = 1, 2, 3)$ are independent observation errors with different variances that are also scaled suitably. Note that the trend μ_t and the autoregressive noise term ζ_t are shared by the models of the three series, and, for identification purposes, the intercept for GISS is taken to be zero. In addition, the model parameters are assumed to be interrelated and are parameterized in a particular way (which leads to fewer parameters to estimate, and their relative scaling helps in parameter estimation). This special parameterization can be expressed as a function of seven basic parameters: *loga1*, *logr1*, *logr3*, *logsigma*, *b*, *c*, and *rhoParm* (this naming convention is different from that used by Ansley and de Jong (2012)).

Let $\sigma^2 = \exp(2\mathrm{logsigma})$, and let the scaling factors $a = \exp(\mathrm{loga1})$, $r_1 = \exp(\mathrm{logr1})$, and $r_3 = \exp(\mathrm{logr3})$. Then the model parameters can be described as follows:

- The parameters that are associated with the autoregression ζ_t: the damping factor $\rho = \frac{\exp(\text{rhoParm})-1}{\exp(\text{rhoParm})+1}$, the variance of the disturbance term = $a^2\sigma^2$, and the variance of the initial state = $a^2\sigma^2/(1-\rho^2)$
- The parameters that are associated with the integrated random walk trend: the variance of the disturbance term in the slope equation = σ^2
- The variance of the observation errors for GISS = $a^2 r_1^2 \sigma^2$
- The variance of the observation errors for UAH = $a^2 r_3^2 \sigma^2$
- The variance of the observation errors for CRU is taken to be time-varying with the following form: for $t \leq nobs$,

  $$\sigma_{2,t}^2 = a^2\sigma^2 \exp(2b + 2c\ t/nobs)$$

  where $nobs$ = 1,945 (the number of observations in the unappended data set). For $t >$ 1,945, it is fixed at its last value: $\sigma_{2,t}^2 = a^2\sigma^2 \exp(2b + 2c)$.

The following DATA step adds an observation index, tindex, to Temp, which is used in the SSM procedure to define the time-varying observation error variance for CRU:

```
data temp;
   set temp;
   tindex = _n_;
run;
```

The following statements fit the preceding model to the Temp data:

```
 ods output RegressionEstimates=regEst;
 proc ssm data=Temp;
     id date interval=month;
     parms loga1 logr1 logr3  logsigma;
     parms b=0 c=0;
     parms rhoParm;
     rho = (exp(rhoParm)-1)/(exp(rhoParm)+1);
     sigmaSq = exp(2*logsigma);
     initSigmaSq = sigmaSq/(1-rho*rho);
     a1 = exp(loga1);
     a1Sq = a1*a1;
     r1sq  = exp(2*logr1);
     r3sq  = exp(2*logr3);
     giss_var = a1Sq*r1sq*sigmaSq;
     nobs=1945;
     if tindex <= nobs then
         cru_var = a1Sq*exp(2*b + 2*c*tindex/nobs)*sigmaSq;
     else cru_var = a1Sq*exp(2*b + 2*c)*sigmaSq;
     uah_var = a1Sq*r3sq*sigmaSq;
     UAH_Intercept=1.0;
     CRU_Intercept=1.0;
     trend level(ll) variance=0 slopevar=sigmaSq;
     state auto(1) T(g)=(rho) cov(g)=(sigmaSq)  cov1(g)=(initSigmaSq);
     comp auto_common = auto*(a1);
     state wn(3) type=wn cov(d)=(giss_var cru_var uah_var);
     comp wn_giss = wn[1];
```

```
      comp wn_cru = wn[2];
      comp wn_uah = wn[3];
      model GISS = level auto_common  wn_giss;
      model CRU = CRU_Intercept level auto_common  wn_cru;
      model UAH = UAH_Intercept level auto_common  wn_uah;
      comp slope = level_state_*(0 1);
      output out=For pdv press;
  run;
```

Output 34.12.2 shows the estimated intercepts $\hat{\beta}_{cru}$ and $\hat{\beta}_{uah}$. As expected, they are quite close (see the scatter plots of CRU and UAH in Output 34.12.1).

Output 34.12.2 Estimated Intercepts for CRU and UAH

The SSM Procedure

Regression Parameter Estimates

Response Variable	Regression Variable	Estimate	Standard Error	t Value	Pr > |t|
CRU	CRU_Intercept	-4.10	0.00360	-1139.2	<.0001
UAH	UAH_Intercept	-4.48	0.00671	-666.48	<.0001

The estimates of the basic parameters that underlie the model parameters are shown in Output 34.12.3.

Output 34.12.3 Estimates of Basic Model Parameters

Estimates of Named Parameters

Parameter	Estimate	Standard Error	t Value
loga1	7.820	0.3819	20.48
logr1	-1.005	0.0884	-11.37
logr3	-0.231	0.0453	-5.09
logsigma	-9.682	0.3806	-25.44
b	0.737	0.0502	14.69
c	-1.403	0.0689	-20.35
rhoParm	1.432	0.0767	18.68

The following DATA steps add two variables (CRU_ADJ = CRU – $\hat{\beta}_{cru}$ and UAH_ADJ = UAH – $\hat{\beta}_{uah}$) to the output data set For. These adjusted versions of CRU and UAH have the same mean level as GISS—estimated μ_t.

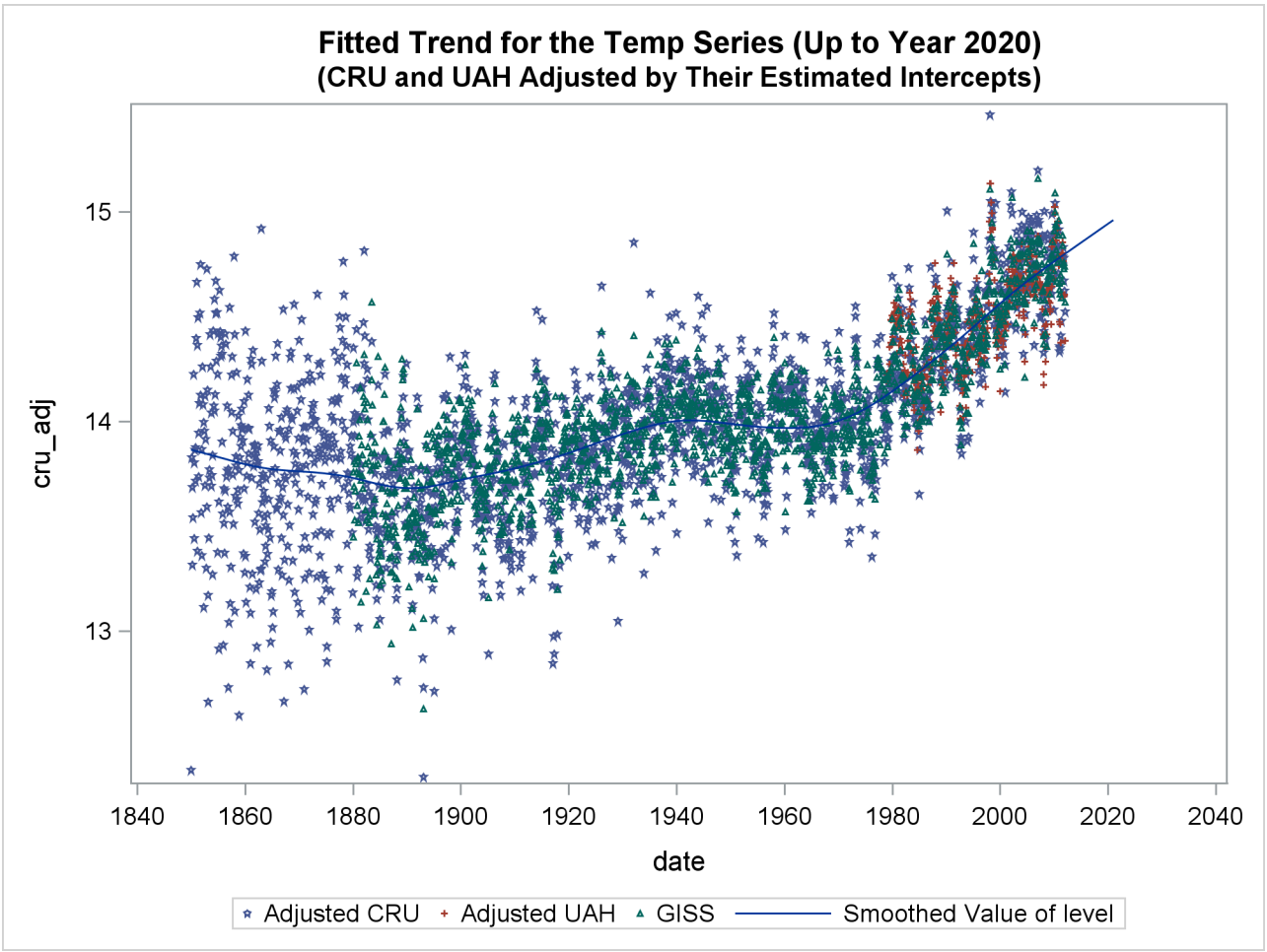
```
data _NULL_;
   set regEst;
   if _n_ = 1 then call symput('intercept1',trim(left(estimate)));
   else call symput('intercept2',trim(left(estimate)));
run;
data for;
    set For;
    cru_adj = cru - &intercept1;
    uah_adj = uah - &intercept2;
run;
```

The following statements produce a graph that contains four plots: scatter plots of GISS, CRU_ADJ, and UAH_ADJ and a series plot of the estimated μ_t.

```
proc sgplot data=For;
   where date < '01feb2021'd;
   title "Fitted Trend for the Temp Series (Up to Year 2020) ";
   title2 "(CRU and UAH Adjusted by Their Estimated Intercepts) ";
   scatter x=date y=cru_adj / LEGENDLABEL="Adjusted CRU"
       MARKERATTRS=GraphData1(symbol=star size=3);
   scatter x=date y=uah_adj / LEGENDLABEL="Adjusted UAH"
       MARKERATTRS=GraphData2(symbol=plus size=3);
   scatter x=date y=giss /
       MARKERATTRS=GraphData3(symbol=triangle size=3);
   series x=date y=smoothed_level / MARKERATTRS=GraphData4;
run;
```

Output 34.12.4 shows the resulting graph. It shows that the estimated mean level μ_t tracks the observed data quite well.

Output 34.12.4 Fitted Trend μ_t

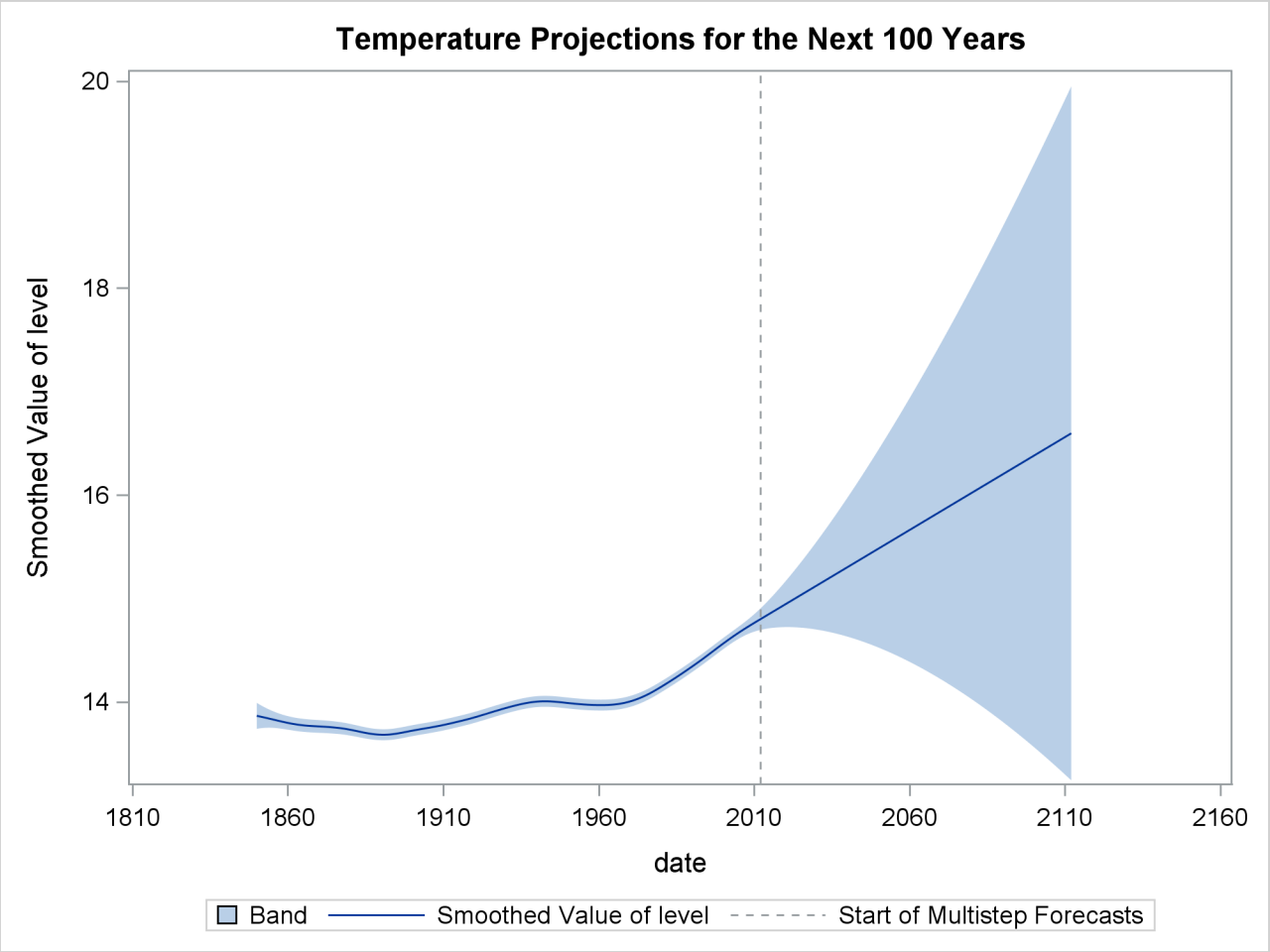


The following statements produce the time series plot of the estimated μ_t along with the 95% confidence band:

```
proc sgplot data=For;
   title "Temperature Projections for the Next 100 Years";
   band x=date lower=smoothed_lower_level upper=smoothed_upper_level;
   series x=date y=smoothed_level;
   refline '01feb2012'd / axis=x lineattrs=(pattern=shortdash)
      LEGENDLABEL= "Start of Multistep Forecasts"
      name="Forecast Reference Line";
run;
```

Output 34.12.5 shows the resulting graph.

Output 34.12.5 Long-Term Forecasts of μ_t



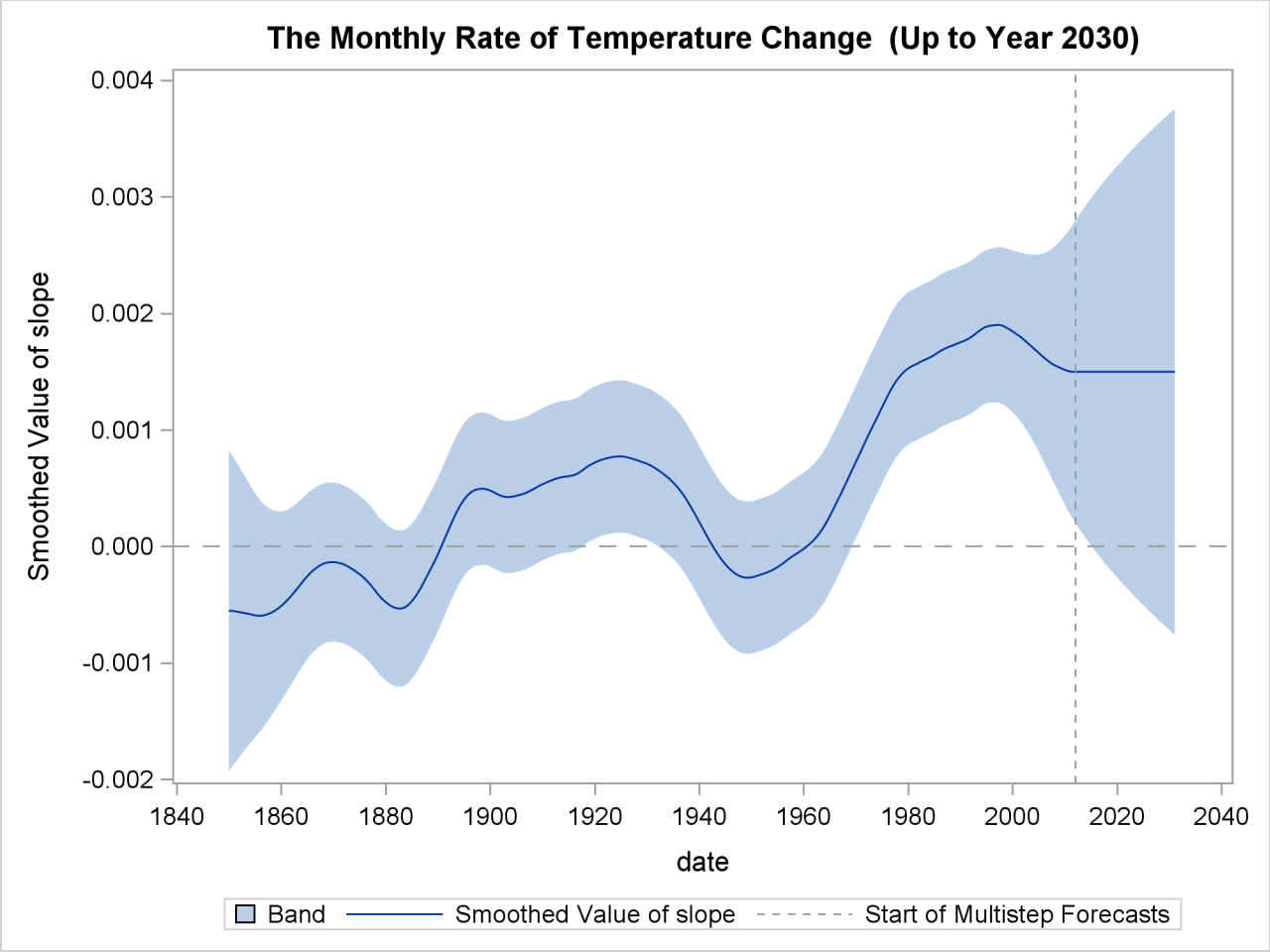
The following statements produce a similar graph for the estimated slope of μ_t:

```
proc sgplot data=For;
   where date <=  '01feb2031'd;
   title "The Monthly Rate of Temperature Change  (Up to Year 2030)";
   band x=date lower=smoothed_lower_slope upper=smoothed_upper_slope;
   series x=date y=smoothed_slope;
   refline '01feb2012'd / axis=x lineattrs=(pattern=shortdash)
      LEGENDLABEL= "Start of Multistep Forecasts"
```

```
      name="Forecast Reference Line";
   refline 0 / axis=y lineattrs=(pattern=dash);
run;
```

Output 34.12.6 shows the resulting plot of the estimated slope of μ_t.

Output 34.12.6 Forecasts of the Slope of μ_t



Based on the preceding analysis (see the plots of μ_t and its slope in Output 34.12.5 and Output 34.12.6), it appears that there has been statistically significant warming over the last 10 years, but the warming does not appear to be accelerating.

Example 34.13: Bivariate Model: Sales of Mink and Muskrat Furs

This example considers a bivariate time series of logarithms of the annual sales of mink and muskrat furs by the Hudson's Bay Company for the years 1850–1911. These data have been analyzed previously by many authors, including Chan and Wallis (1978); Harvey (1989); Reinsel (1997). There is known to be a predator-prey relationship between the mink and muskrat species: minks are principal predators of muskrats. Previous analyses for these data generally conclude the following:

- An increase in the muskrat population is followed by an increase in the mink population a year later, and an increase in the mink population is followed by a decrease in the muskrat population a year later.
- Because muskrats are not the only item in the mink diet and because both mink and muskrat populations are affected by many other factors, the model must include additional terms to explain the year-to-year variation.

The analysis in this example, which loosely follows the discussion in Harvey (1989, chap. 8, sec. 8), also leads to similar conclusions. It begins by taking Harvey's model 8.8.8 (a and b), with autoregressive order one, as the starting model—that is, it assumes that the bivariate (mink, muskrat) process $\mathbf{Y}_t$ satisfies the following relationship:

$$\boldsymbol{\mu}_t = \boldsymbol{\mu}_{t-1} + \boldsymbol{\beta} + \boldsymbol{\nu}_t$$
$$\mathbf{Y}_t = \boldsymbol{\mu}_t + \boldsymbol{\Phi}\mathbf{Y}_{t-1} + \boldsymbol{\xi}_t$$

This model postulates that $\mathbf{Y}_t$ can be expressed as a sum of three terms: $\boldsymbol{\mu}_t$, a bivariate trend that is modeled as a random walk with drift $\boldsymbol{\beta}$; $\boldsymbol{\Phi}\mathbf{Y}_{t-1}$, an AR(1) correction; and $\boldsymbol{\xi}_t$, a bivariate Gaussian white noise disturbance. It is assumed that the AR coefficient matrix $\boldsymbol{\Phi}$ is stable (that is, its eigenvalues are less than 1 in magnitude) and that the bivariate disturbances $\boldsymbol{\nu}_t$ (white noise associated with $\boldsymbol{\mu}_t$) and $\boldsymbol{\xi}_t$ are mutually independent.

The following statements show how you can specify this model in the SSM procedure:

```
proc ssm data=furs plots=residual;

   /* Specify the ID variable */
   id year interval=year;

   /* Define parameters */
   parms rho1 rho2/ lower=-0.9999 upper=0.9999;
   parms msd1 msd2 esd1 esd2 / lower=1.e-6;

   /* Specify the terms with lagged response variables */
   deplag LagsForMink(LogMink) LogMink(lags=1) LogMusk(lags=1);
   deplag LagsForMusk(LogMusk) LogMink(lags=1) LogMusk(lags=1);

   /* Specify the bivariate trend */
   array rwQ{2,2};
   rwQ[1,1] = msd1*msd1; rwQ[1,2] = msd1*msd2*rho1;
   rwQ[2,1] = rwQ[1,2]; rwQ[2,2]=msd2*msd2;
   state alpha(2) type=RW W(I) cov(g)=(rwQ);
   comp minkLevel = alpha[1];
   comp muskLevel = alpha[2];

   /* Specify the bivariate white noise */
   array wnQ{2,2};
   wnQ[1,1] = esd1*esd1; wnQ[1,2] = esd1*esd2*rho2;
   wnQ[2,1] = wnQ[1,2]; wnQ[2,2]=esd2*esd2;
   state error(2) type=WN cov(g)=(wnQ);
   comp minkWn = error[1];
   comp muskWn = error[2];

   /* Specify the observation equation */
```

```
   model LogMink = LagsForMink minkLevel minkWn;
   model LogMusk = LagsForMusk muskLevel muskWn;

   /* Specify an output data set to store component estimates */
   output out=salesFor press;
run;
```

The different parts of the program are explained as follows:

- The PARMS statements define parameters that are used to form the elements of Σ_1 (the covariance of $\boldsymbol{\nu}_t$, the disturbance term in the bivariate level equation) and Σ_2 (the covariance of $\boldsymbol{\xi}_t$, which is the bivariate white noise). Σ_1 is parameterized as (`msd1*msd1 msd1*msd2*rho1; msd1*msd2*rho1 msd2*msd2`). Σ_2 is similarly parameterized by using esd1, esd2, and rho2. In addition to ensuring that Σ_1 and Σ_2 are positive semidefinite, it turns out that this parameterization leads to an interpretable model at the end.
- The DEPLAG statements help define the terms that are associated with $\boldsymbol{\Phi}\mathbf{Y}_{t-1}$.
- The remaining statements are self-explanatory.

Output 34.13.1 shows the estimate of the drift vector $\boldsymbol{\beta}$ in the equation of $\boldsymbol{\mu}_t$ ($\boldsymbol{\mu}_t = \boldsymbol{\mu}_{t-1} + \boldsymbol{\beta} + \boldsymbol{\nu}_t$).

Output 34.13.1 Estimate of the Drift Vector $\boldsymbol{\beta}$

Estimate of the State Equation Regression Vector

State	Element Index	Estimate	Standard Error	t Value	Pr > |t|
alpha	1	-0.000817	0.0323	-0.03	0.9798
alpha	2	0.005953	0.0258	0.23	0.8175

Clearly, both elements of $\boldsymbol{\beta}$ are statistically insignificant, and the $\boldsymbol{\mu}_t$ equation can be simplified as $\boldsymbol{\mu}_t = \boldsymbol{\mu}_{t-1} + \boldsymbol{\nu}_t$. Next, Output 34.13.2 shows the estimates of the elements of Σ_1, and Σ_2, and Output 34.13.3 shows the estimates of the lag coefficients.

Output 34.13.2 Estimates of Σ_1, and Σ_2

Estimates of Named Parameters

Parameter	Estimate	Standard Error	t Value
rho1	0.8310	0.1377	6.03
rho2	-0.9999	1.6555	-0.60
msd1	0.2500	0.0354	7.06
msd2	0.1991	0.0592	3.36
esd1	0.0662	0.0597	1.11
esd2	0.1344	0.0527	2.55

Output 34.13.3 Estimates of Lag Coefficients (Elements of Φ)

Model Parameter Estimates

Component	Type	Parameter	Estimate	Standard Error	t Value
LagsForMink	Lag Coefficient Of LogMink	Lag[1]	-0.0011	0.173	-0.01
LagsForMink	Lag Coefficient Of LogMusk	Lag[1]	0.3349	0.137	2.45
LagsForMusk	Lag Coefficient Of LogMink	Lag[1]	-0.9905	0.142	-6.98
LagsForMusk	Lag Coefficient Of LogMusk	Lag[1]	0.6570	0.121	5.44

The main points of the output can be summarized as follows:

- phi11, the first element of Φ, which relates the current value of LogMink with its lagged value, is statistically insignificant. That is, lagged LogMink term could be dropped from the model equation for LogMink.
- rho2, the correlation coefficient between the elements of $\boldsymbol{\xi}_t$—the bivariate noise vector in the equation $\mathbf{Y}_t = \boldsymbol{\mu}_t + \boldsymbol{\Phi}\mathbf{Y}_{t-1} + \boldsymbol{\xi}_t$—is very near its lower boundary of –1 (in such cases the standard error of the parameter estimate is not reliable). This implies that the two elements of $\boldsymbol{\xi}_t$ are perfectly negatively correlated.

Taken together, these observations suggest the reduced model

$$\boldsymbol{\mu}_t = \boldsymbol{\mu}_{t-1} + \boldsymbol{\nu}_t$$
$$\mathbf{Y}_t = \boldsymbol{\mu}_t + \boldsymbol{\Phi}\mathbf{Y}_{t-1} + \boldsymbol{\xi}_t$$

where $\boldsymbol{\Phi} = (0 \quad \phi_{12};\ \phi_{21} \quad \phi_{22})$ and $\mathrm{Cov}(\boldsymbol{\xi}_t) = \Sigma_2$ is parameterized as (`esd1*esd1 -esd1*esd2; -esd1*esd2 esd2*esd2`). The program that produces the reduced model is a simple modification of the preceding program and is not shown.

Output 34.13.4 Estimates of Σ_1, and Σ_2 (Reduced Model)

Estimates of Named Parameters

Parameter	Estimate	Standard Error	t Value
rho1	0.8526	0.0978	8.71
msd1	0.2472	0.0282	8.78
msd2	0.1955	0.0365	5.36
esd1	0.0679	0.0385	1.76
esd2	0.1372	0.0328	4.18

Output 34.13.5 Estimates of Lag Coefficients (elements of Φ) (Reduced Model)

Model Parameter Estimates

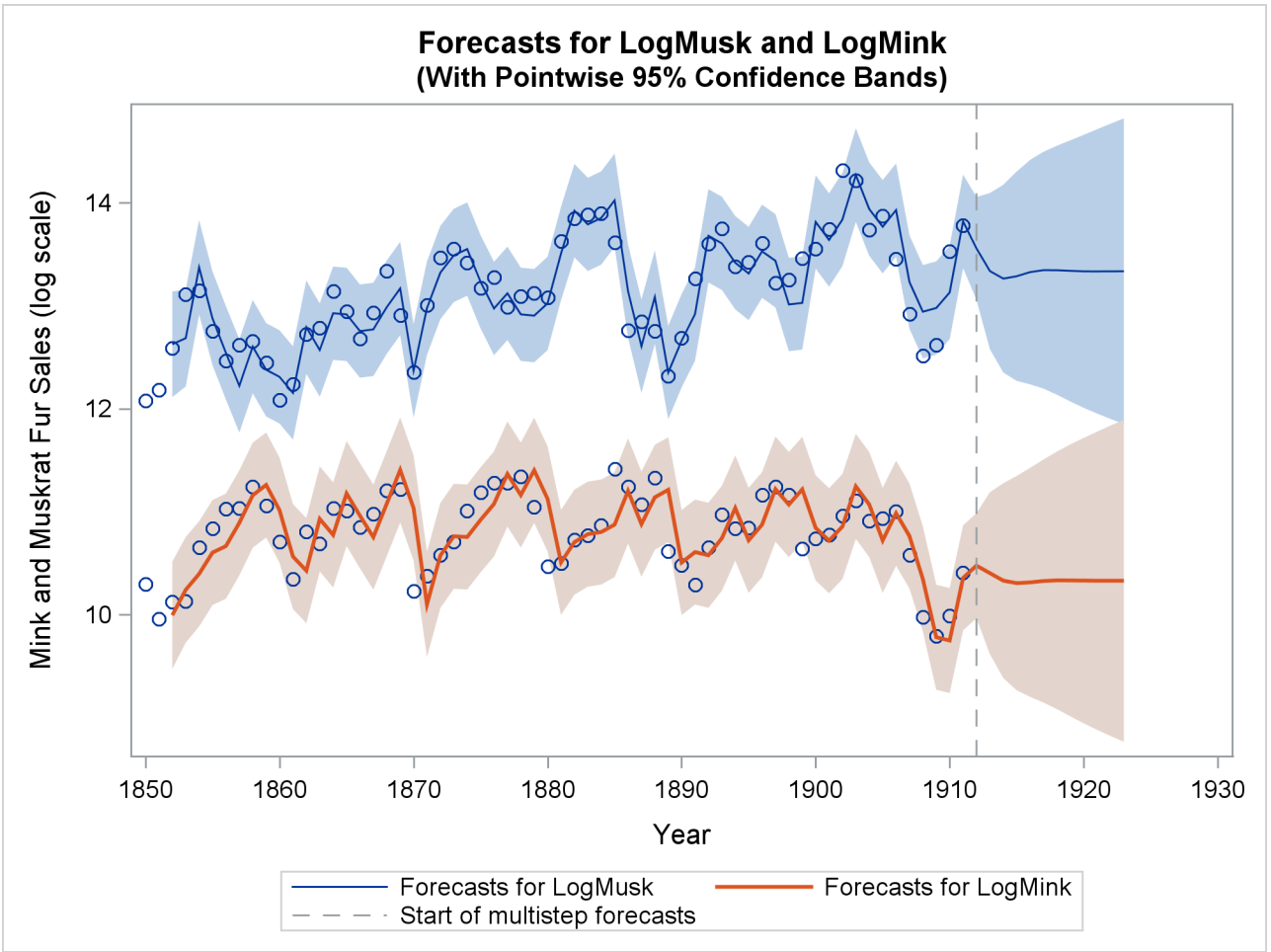
Component	Type	Parameter	Estimate	Standard Error	t Value
LagsForMink	Lag Coefficient Of LogMusk	Lag[1]	0.330	0.0986	3.35
LagsForMusk	Lag Coefficient Of LogMink	Lag[1]	-0.997	0.1168	-8.54
LagsForMusk	Lag Coefficient Of LogMusk	Lag[1]	0.668	0.1003	6.66

The tables in Output 34.13.4 and Output 34.13.5 show the new parameter estimates. By examining the parameter estimates, you can easily see that this model supports the general conclusions mentioned at the start of this example. In particular, note the following:

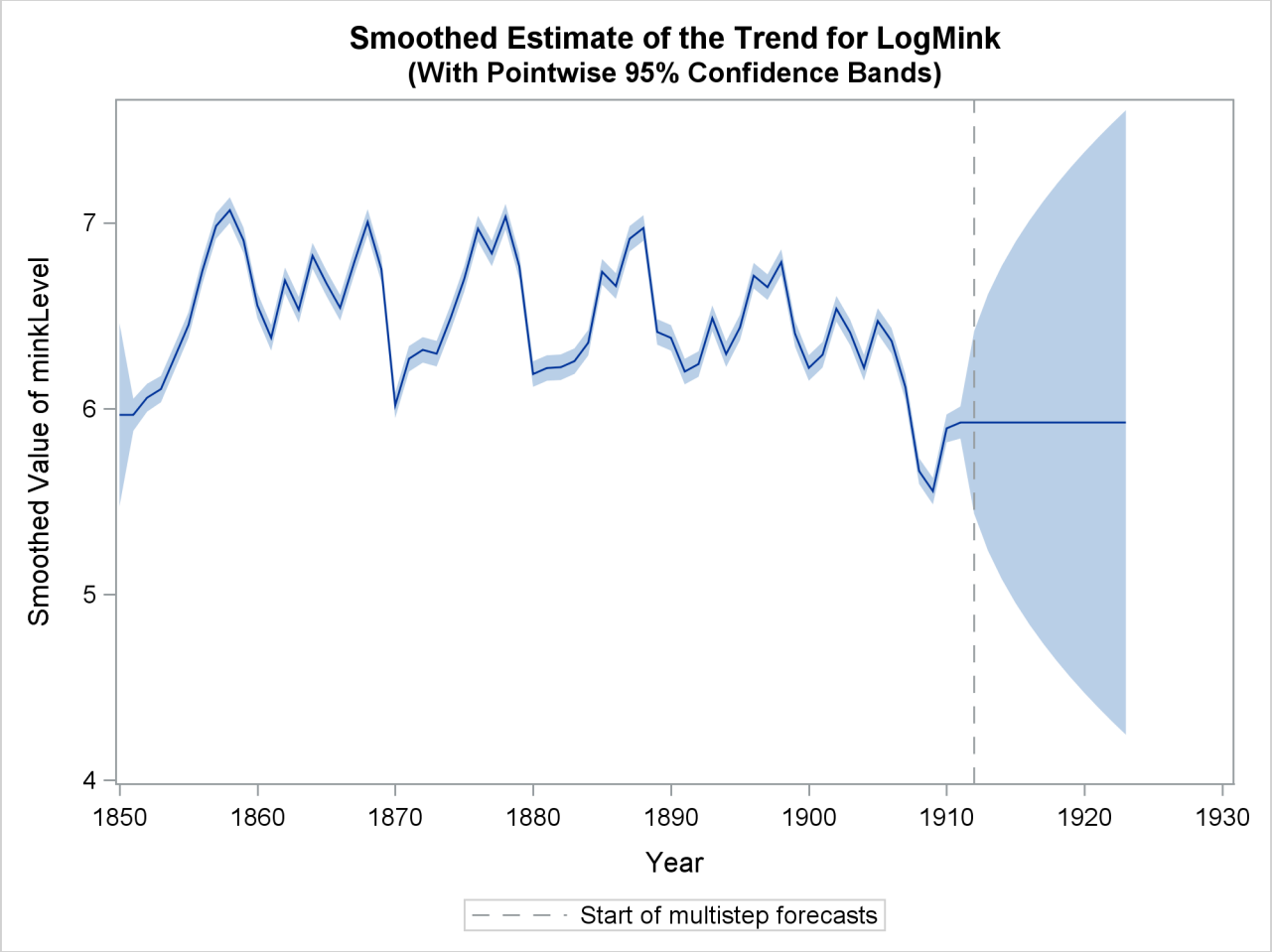
- $\hat{\phi}_{12} = 0.33$ implies that this year's mink abundance is positively correlated with last year's muskrat abundance.
- $\hat{\phi}_{21} = -0.99$ and $\hat{\phi}_{22} = 0.66$ imply that this year's muskrat abundance is negatively correlated with last year's mink abundance and positively correlated with last year's muskrat abundance.
- Even though the parameters were not restricted to ensure stability, the estimated $\mathbf{\Phi}$ turns out to be stable with a pair of complex eigenvalues, $0.317 \pm i\ 0.473$, and a modulus of 0.570 (these calculations are done separately by using the IML procedure).
- The fact that elements of $\boldsymbol{\xi}_t$ are perfectly negatively correlated further supports the predator-prey relationship.

Finally, Output 34.13.6 shows the plots of one-step-ahead and post-sample forecasts for LogMink and LogMusk, and Output 34.13.7 shows the plot of the smoothed (full-sample) estimate of the first element of $\boldsymbol{\mu}_t$: LogMink Trend.

Output 34.13.6 Forecasts for Mink and Muskrat Fur Sales in Logarithms



Output 34.13.7 Smoothed Estimate of LogMink Trend



Example 34.14: Factor Model: Now-Casting the US Economy

A well-known business conditions index, the Aruoba-Diebold-Scotti (ADS) business conditions index, is designed to track real business conditions at high frequency (for more information about this index, see http://www.philadelphiafed.org/research-and-data/real-time-center/business-conditions-index/). Its underlying (seasonally adjusted) economic indicators (weekly initial jobless claims, monthly payroll employment, industrial production, personal income less transfer payments, manufacturing and trade sales, and quarterly real GDP) blend high- and low-frequency information with stock and flow data. The ADS index is based on a rather elaborate state space model that takes into account the stock and flow nature of the underlying economic indicators. To simplify the illustration, this example uses the same economic indicators to develop a similar index by using a simpler factor model. You can also use PROC SSM to carry out the more elaborate modeling that underlies the ADS index. All these economic indicators are freely available from the Federal Reserve Economic Data (FRED). You can access these data by using the SASEFRED interface engine; see Chapter 48, "The SASEFRED Interface Engine." The names of analysis variables and the relevant information that is needed for using the SASEFRED engine to obtain these data are shown in Table 34.11. All variables are transformed versions of the original series: all, except l_icsa, are both logged and differenced; l_icsa is only logged. The input data set for the analysis,

named econ, is formed by merging these six series of different frequencies. This merging is done by treating them as daily series that have missing values on the days when the series values are not available—for example, ld_pinc is nonmissing only for the first day of the month. The econ data set contains one more variable, recession, which is an indicator variable for the recessionary periods. This time series is an interpretation of the US Business Cycle Expansions and Contractions data that are provided by the National Bureau of Economic Research (NBER) at http://www.nber.org/cycles/cyclesmain.html (also see http://research.stlouisfed.org/fred2/series/USRECDM). The variable recession is not used in modeling. It is used later to demonstrate the adequacy of the activity index that is created in this example.

Table 34.11 Analysis Variables and Their FRED IDs

Name	FRED ID	Frequency	Description
ld_payemp	PAYEMS	Monthly	Payroll employment
ld_pinc	W875RX1	Monthly	Real personal income excluding current transfer receipts
ld_mnfctr	CMRMTSPL	Monthly	Real manufacturing and trade industries sales
ld_indpro	INDPRO	Monthly	Industrial production index
ld_gdp	GDPC1	Quarterly	Real GDP
l_icsa	ICSA	Weekly	Initial jobless claims

For easier model description, the variables ld_payemp to ld_gdp are also denoted as y_{1t} to y_{5t}, and the variable l_icsa is denoted as y_{6t}. Using this notation, the following model is postulated for the six daily time series:

$$\begin{aligned} \mathrm{y}_{it} &= \mathrm{intercept}_i + \beta_i * \mathrm{irw}_t + \epsilon_{it} \qquad 1 \le i \le 5 \\ \mathrm{y}_{6t} &= \mathrm{intercept}_6 + \beta_6 * \mathrm{irw}_t + \mu_t + \epsilon_{6t} \end{aligned}$$

A justification for this model is based on the following observations:

- The five time series y_{1t} to y_{5t} are logged and differenced versions of the underlying economic variables. Their plots (not shown here) show them to be hovering around a constant level, with some periods of deviation from this level. The plot of the sixth series, y_{6t}, which is logged but not differenced, shows a pronounced nonstationary pattern.

- All these series can be considered as proxies, possibly noisy, for the national economic activity. It is therefore reasonable to assume that a model for each of them will contain a common component, appropriately weighted, that is associated with the economic activity. In the current model this common component, named irw_t, is modeled as an integrated random walk. For y_{1t} to y_{5t}, the only other terms in the model are the respective intercepts, $\mathrm{intercept}_i$, and the random disturbances, ϵ_{it}. Because y_{6t} shows a pronounced nonstationary pattern, its model includes an additional term, μ_t, which is also modeled as an integrated random walk. For identifiability purposes, the initial condition for irw_t is taken to be 0. For the same reason, β_1, the coefficient of irw_t in the model for y_{1t} is taken to be 1.

- The underlying economic variables of the five time series y_{1t} to y_{5t} are positively correlated with the economic activity—for example, payroll employment is expected to increase with increased economic activity. On the other hand, y_{6t}, which is associated with the initial jobless claims, is negatively correlated with the economic activity. This means that, with β_1 taken to be 1, the estimates of $\beta_2, \ldots, \beta_5$ are expected to be positive and the estimate of β_6 is expected to be negative. In the factor modeling terminology, irw_t is called a factor and $\beta_1, \ldots, \beta_6$ are called the associated factor loadings.

The following statements show you how to specify this model in the SSM procedure:

```
ods output NamedParameterEstimates = named;
proc ssm data=econ opt(tech=activeset);
  id date interval=day;
  parms beta2-beta6;
  parms lv1-lv8;
  avar = exp(lv7);
  wnv1 = exp(lv1);  wnv2 = exp(lv2);
  wnv3 = exp(lv3);  wnv4 = exp(lv4);
  wnv5 = exp(lv5);  wnv6 = exp(lv6);
  tvar = exp(lv8);
  zero = 0;

   /* --- start of model spec ----*/
  state latent(2) t(g)=(1 1 0 1) cov(d)=(zero avar);
  comp c1 = latent[1];
  comp c2 = (beta2)*latent[1];
  comp c3 = (beta3)*latent[1];
  comp c4 = (beta4)*latent[1];
  comp c5 = (beta5)*latent[1];
  comp c6 = (beta6)*latent[1];

  irregular w1 variance=wnv1;
  int1 = 1;
  model ld_payemp = int1 c1 w1;

  irregular w2 variance=wnv2;
  int2 = 1;
  model ld_pinc = int2 c2 w2;

  irregular w3 variance=wnv3;
  int3 = 1;
  model ld_mnfctr = int3 c3 w3;

  irregular w4 variance=wnv4;
  int4 = 1;
  model ld_indpro = int4 c4 w4;

  irregular w5 variance=wnv5;
  int5 = 1;
  model ld_gdp = int5 c5 w5;

  irregular w6 variance=wnv6 ;
  trend t_icsa(ll) levelvar=0 slopevar=tvar;
  model l_icsa = c6 t_icsa w6;
  /* ---model spec done----*/

  eval icsaPattern = c6 + t_icsa;
  /*--index is a scaled version of the common factor--*/
  eval Index = 1000*c1;
  comp slope = latent[2];
  eval IndexSlope = 1000*slope;
  /*--just so recession is output to the output data set--*/
```

```
 rec = recession;
 output out=forecast1 press pdv;
run;
```

A few comments about the program:

- If the model and data are in reasonable accord, the default likelihood optimization settings work in most situations. However, in some cases the likelihood optimization process needs additional customization. Some experimentation with alternative optimization techniques and different parameterization of the model parameters can help. This example turns out to be one such case. The optimization technique ACTIVESET (`opt(tech=activeset)`) works better for WINDOWS and a few other platforms, whereas the default optimization technique works better for the AIX platform. In addition, the variances of all the disturbance terms in the model are parameterized in the exponential scale.

- The two-dimensional state that is associated with irw_t is named latent, and irw_t (the first element of latent) itself is named c1. Note that the second element of latent corresponds to the slope of irw_t. The components c2 to c6 correspond to $\beta_i * \text{irw}_t$ for $2 \leq i \leq 6$.

- The desired business index, named index, is a scaled version of irw_t (`eval Index = 1000*c1;`). This scaling is done purely for ease of display—the scaled values turn out to be in the range of –6.0 to 5.0. Another component, named IndexSlope, contains the slope of index, which is also a quantity of interest.

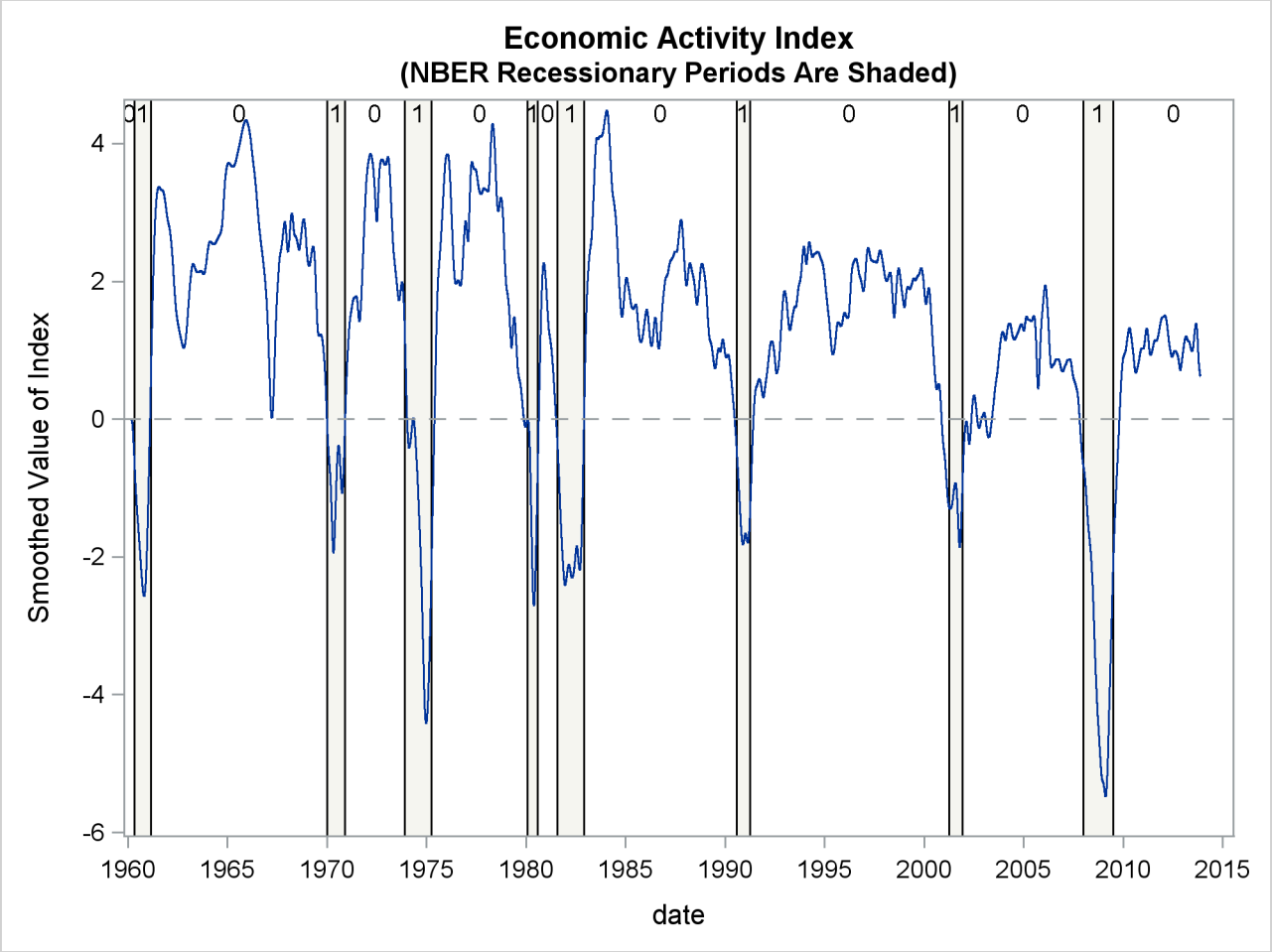
Output 34.14.1 Estimated Factor Loadings

The Estimated Loadings

Parameter	Estimate	StdErr	tValue
beta2	1.15	0.1276	8.98
beta3	1.96	0.2390	8.20
beta4	2.48	0.1646	15.08
beta5	3.27	0.2653	12.33
beta6	-96.45	9.6008	-10.05

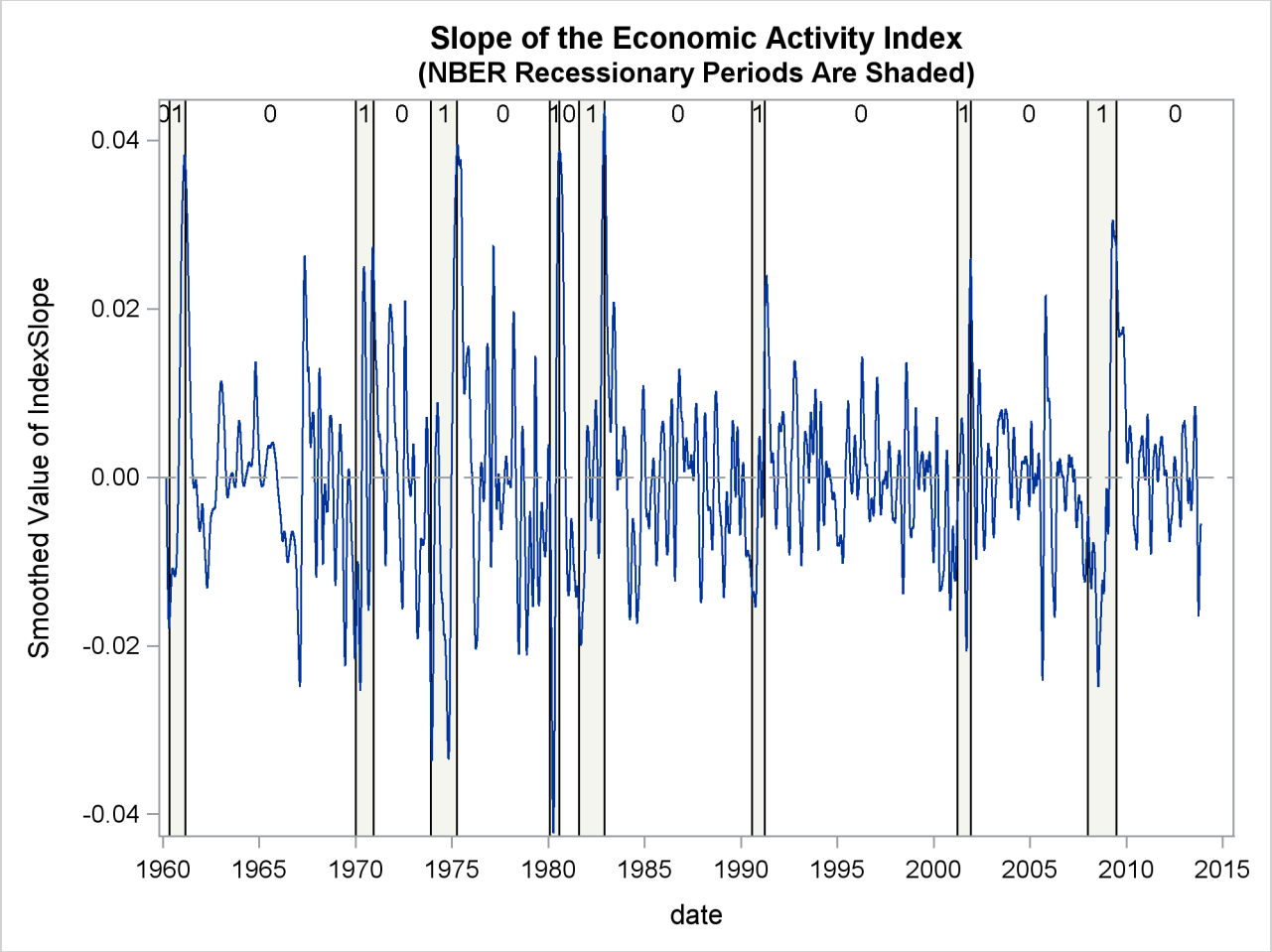
Output 34.14.1 shows the estimated factor loadings. They are statistically significant and their signs are consistent.

Output 34.14.2 The Estimate of Economic Activity Index



Output 34.14.2 shows the plot of the smoothed index. Note that it coheres quite well with the NBER recessionary periods. In Aruoba, Diebold, and Scotti (2009, sec. 4.4) the features of an earlier version of the ADS index are discussed in detail. Similar comments apply to this indicator also.

Output 34.14.3 Estimated Slope of the Economic Activity Index



Finally, Output 34.14.3 shows the plot of the slope of the index, which gives an idea of the direction of the economic activity.

Example 34.15: Longitudinal Data: Lung Function Analysis

The data for this example, which consist of 209 measurements of the lung function of an asthma patient, are analyzed in Wang (2013). The time series is measured mostly at two-hour time intervals but with irregular gaps. Wang (2013) fit a fourth-order continuous-time autoregressive model, CAR(4), to these data. The analysis results in a decomposition of the observed time series in three components:

- a slowly varying trend pattern, which appears to have a slight downward drift
- a diurnal component—a periodic pattern with a period of 24 hours
- a residual component

As shown in Wang (2013), the continuous-time autoregressive models can be formulated as state space models. However, in general, the form of such SSMs is quite complex. Consequently, specifying such a model by using the current SSM procedure syntax is impractical. On the other hand, you can analyze these types of longitudinal data by using continuous-time structural models, which are easy to specify in the SSM procedure. In this example, the lung function measurements, `y`, are modeled as

$$y_t = \text{intercept} + \beta * t + \zeta_t + \epsilon_t$$

where (intercept $+ \beta * t$) is a simple linear time trend, ζ_t is a continuous-time stochastic cycle, and ϵ_t is a Gaussian white noise sequence. Replacing the linear time trend with a more general time trend, such as a spline smoother, does not seem to change the fit, because the estimated smoothing spline turns out to be almost perfectly linear.

The following statements show you how to specify this model in the SSM procedure:

```
proc ssm data=asth;
   id time;
   state s1(1) type=cycle(CT) cov(g);
   comp c1 = s1[1];
   intercept = 1;
   irregular wn;
   model y= intercept time c1 wn;
   output out=for1 press;
   eval pattern=intercept+time+c1;
run;
```

The continuous-time stochastic cycle, named `c1`, is defined by a pair of STATE and COMPONENT statements. The STATE statement defines `s1` as a state subsection that is associated with a univariate, continuous-time cycle (signified by the use of **`type=cycle(CT)`**). The COMPONENT statement defines `c1` as its first element.

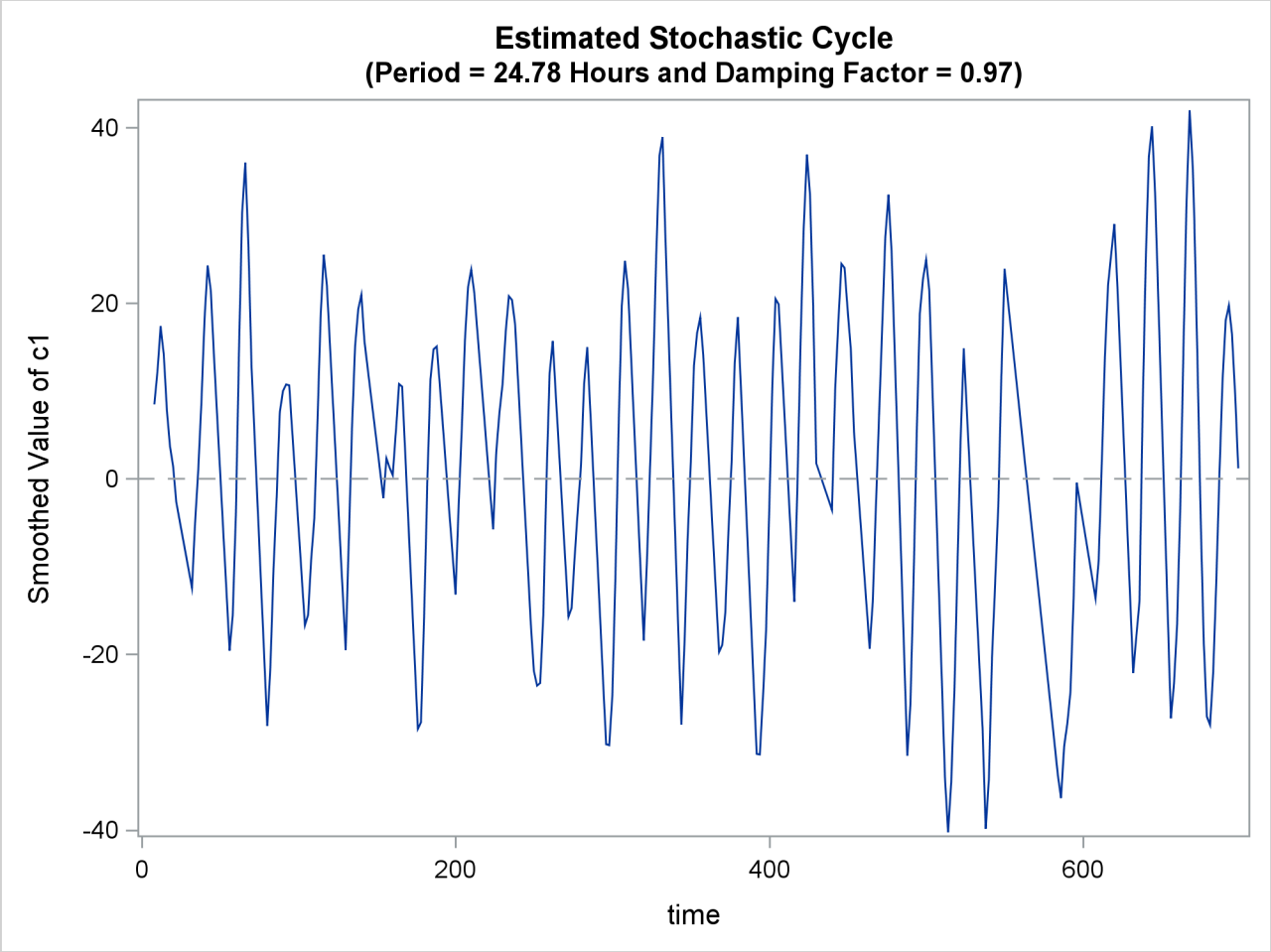
Output 34.15.1 Linear Time Trend: Estimates of Intercept and Slope

Regression Parameter Estimates

Response Variable	Regression Variable	Estimate	Standard Error	t Value	Pr > |t|
y	intercept	502.1637	3.50470	143.28	<.0001
y	time	-0.0201	0.00918	-2.19	0.0286

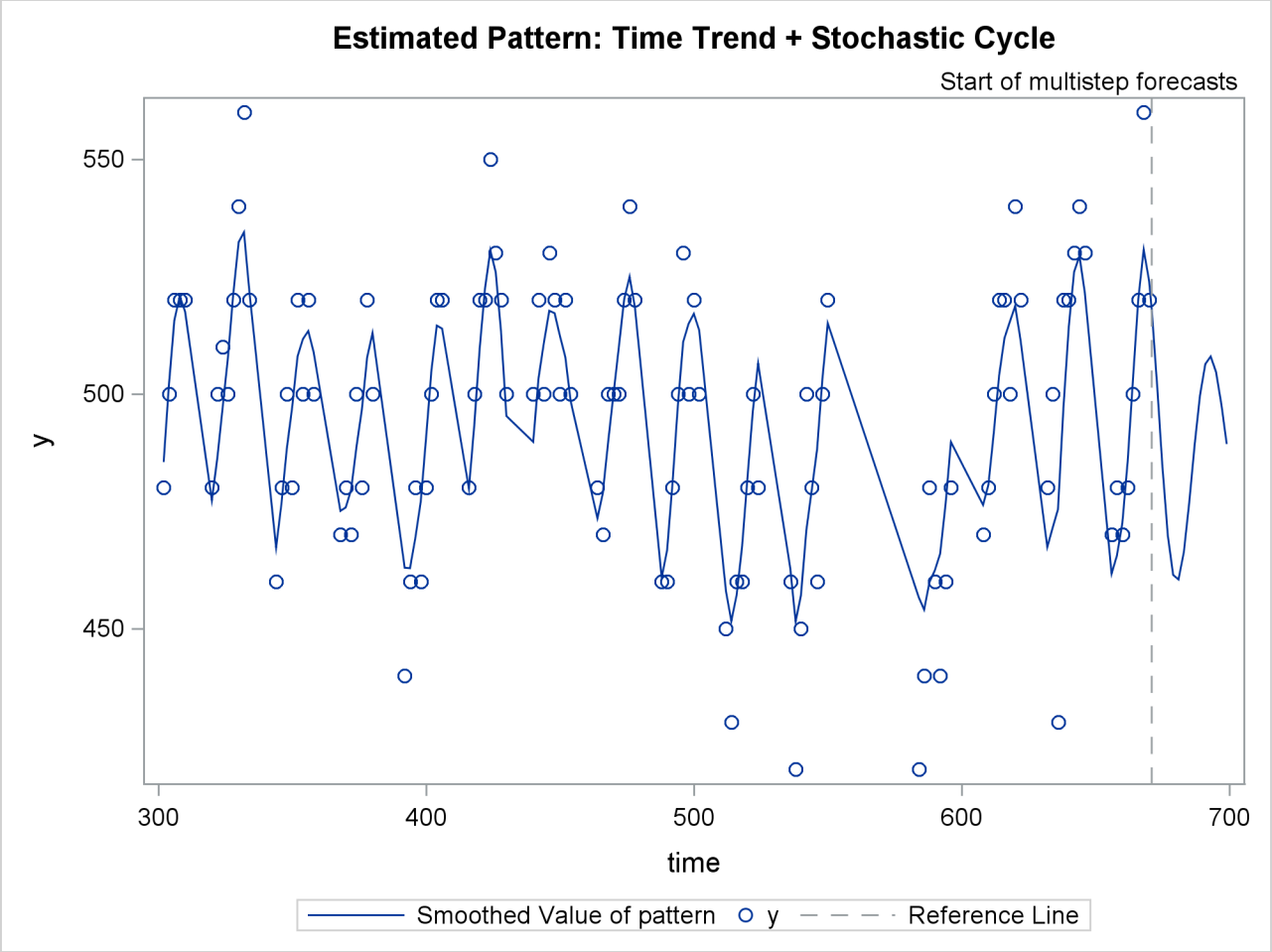
Output 34.15.1 shows the estimated intercept and slope of the time trend. The estimated slope (only marginally significant) is negative, which is consistent with the overall downward drift.

Output 34.15.2 Estimated Stochastic Cycle: ζ_t



Output 34.15.2 shows the plot of the estimated cycle component, which has a period of 24.78 hours and a damping factor of 0.97. That is, it is a nearly persistent diurnal cycle.

Output 34.15.3 Estimated Pattern: Intercept $+ \beta * t + \zeta_t$



Output 34.15.3 shows the fit of the de-noised y values (intercept $+ \beta * t + \zeta_t$). To reduce the clutter, only the second half of the data are plotted. The fit appears to be quite reasonable.

References

Akaike, H. (1974). "A New Look at the Statistical Model Identification." *IEEE Transactions on Automatic Control* AC-19:716–723.

Anderson, B. D., and Moore, J. B. (1979). *Optimal Filtering*. Englewood Cliffs, NJ: Prentice-Hall.

Ansley, C., and de Jong, P. (2012). "Inferring and Predicting Global Temperature Trends." Slide presentation at Time Series Econometrics: Conference in Honour of Andrew Harvey. http://www.oxford-man.ox.ac.uk/sites/default/files/events/Piet%20de%20Jong.pdf.

Aruoba, B. S., Diebold, F. X., and Scotti, C. (2009). "Real-Time Measurement of Business Conditions." *Journal of Business and Economic Statistics* 27:417–427.

Baltagi, B. H. (1995). *Econometric Analysis of Panel Data*. New York: John Wiley & Sons.

Baltagi, B. H., and Levin, D. (1992). "Cigarette Taxation: Raising Revenues and Reducing Consumption." *Structural Change and Economic Dynamics* 3:321–335.

Bell, W. R. (2011). "REGCMPNT—a Fortran Program for Regression Models with ARIMA Component Errors." *Journal of Statistical Software* 41:1–23. http://www.jstatsoft.org/v41/i07/.

Box, G. E. P., and Jenkins, G. M. (1976). *Time Series Analysis: Forecasting and Control*. Rev. ed. San Francisco: Holden-Day.

Bozdogan, H. (1987). "Model Selection and Akaike's Information Criterion (AIC): The General Theory and Its Analytical Extensions." *Psychometrika* 52:345–370.

Burnham, K. P., and Anderson, D. R. (1998). *Model Selection and Inference: A Practical Information-Theoretic Approach*. New York: Springer-Verlag.

Chan, W.-Y. T., and Wallis, K. F. (1978). "Multiple Time Series Modelling: Another Look at the Mink-Muskrat Interaction." *Journal of the Royal Statistical Society, Series C* 27:168–175.

De Jong, P. (1989). "Smoothing and Interpolation with the State-Space Model." *Journal of the American Statistical Association* 84:1085–1088.

De Jong, P. (1991). "The Diffuse Kalman Filter." *Annals of Statistics* 19:1073–1083.

De Jong, P., and Chu-Chun-Lin, S. (2003). "Smoothing with an Unknown Initial Condition." *Journal of Time Series Analysis* 24:141–148.

De Jong, P., and Mazzi, S. (2001). "Modeling and Smoothing Unequally Spaced Sequence Data." *Statistical Inference for Stochastic Processes* 4:53–71.

De Jong, P., and Penzer, J. (1998). "Diagnosing Shocks in Time Series." *Journal of the American Statistical Association* 93:796–806.

Diggle, P. J., Liang, K.-Y., and Zeger, S. L. (1994). *Analysis of Longitudinal Data*. Oxford: Clarendon Press.

Durbin, J., and Koopman, S. J. (2001). *Time Series Analysis by State Space Methods*. Oxford: Oxford University Press.

Eubank, R. L., Huang, C., and Wang, S. (2003). "Adaptive Order Selection for Spline Smoothing." *Journal of Computational and Graphical Statistics* 12:382–397.

Francke, M. K., Koopman, S. J., and de Vos, A. F. (2010). "Likelihood Functions for State Space Models with Diffuse Initial Conditions." *Journal of Time Series Analysis* 31:407–414.

Givens, G. H., and Hoeting, J. A. (2005). *Computational Statistics*. Hoboken, NJ: John Wiley & Sons.

Hannan, E. J., and Quinn, B. G. (1979). "The Determination of the Order of an Autoregression." *Journal of the Royal Statistical Society, Series B* 41:190–195.

Harvey, A. C. (1989). *Forecasting, Structural Time Series Models, and the Kalman Filter*. Cambridge: Cambridge University Press.

Hurvich, C. M., and Tsai, C.-L. (1989). "Regression and Time Series Model Selection in Small Samples." *Biometrika* 76:297–307.

Jones, R. H. (1980). "Maximum Likelihood Fitting of ARMA Models to Time Series with Missing Observations." *Technometrics* 22:389–396.

Jones, R. H. (1993). *Longitudinal Data with Serial Correlation: A State Space Approach*. London: Chapman & Hall.

Kohn, R., Ansley, C., and Tharm, D. (1991). "The Performance of Cross-validation and Maximum Likelihood Estimators of Spline Smoothing Parameters." *Journal of the American Statistical Association* 86:1042–1050.

Kohn, R., and Ansley, C. F. (1991). "A Signal Extraction Approach to the Estimation of Treatment and Control Curves." *Journal of the American Statistical Association* 86:1034–1041.

Koopman, S. J., Mallee, M. I. P., and van der Wel, M. (2010). "Analyzing the Term Structure of Interest Rates Using the Dynamic Nelson-Siegel Model with Time-Varying Parameters." *Journal of Business and Economic Statistics* 28:329–343.

Laird, N. (2004). *Analysis of Longitudinal and Cluster-Correlated Data.* Vol. 8 of NSF-CBMS Regional Conference Series in Probability and Statistics. Beachwood, OH: Institute of Mathematical Statistics. http://www.jstor.org/stable/4153193.

Nelson, C. R., and Siegel, A. F. (1987). "Parsimonious Modeling of Yield Curves." *Journal of Business* 60:473–489.

Reinsel, G. C. (1997). *Elements of Multivariate Time Series Analysis*. 2nd ed. New York: Springer-Verlag.

Schwarz, G. (1978). "Estimating the Dimension of a Model." *Annals of Statistics* 6:461–464.

Selukar, R. S. (2010). "Estimability of the Linear Effects in State Space Models with an Unknown Initial Condition." *Journal of Time Series Analysis* 31:167–168.

Selukar, R. S. (2015). "Functional Modeling of Longitudinal Data with the SSM Procedure." In *Proceedings of the SAS Global Forum 2015 Conference*. Cary, NC: SAS Institute Inc. http://support.sas.com/resources/papers/proceedings15/SAS1580-2015.pdf.

Wang, Z. (2013). "cts: An R Package for Continuous Time Autoregressive Models via Kalman Filter." *Journal of Statistical Software* 53:1–19.

Wecker, W. E., and Ansley, C. F. (1983). "The Signal Extraction Approach to Nonlinear Regression and Spline Smoothing." *Journal of the American Statistical Association* 78:81–89.

Chapter 35
The STATESPACE Procedure

Contents

Overview: STATESPACE Procedure **2504**
The State Space Model 2504
How PROC STATESPACE Works 2506
Getting Started: STATESPACE Procedure **2506**
Automatic State Space Model Selection 2507
Specifying the State Space Model 2514
Syntax: STATESPACE Procedure **2516**
Functional Summary 2516
PROC STATESPACE Statement 2517
BY Statement 2520
FORM Statement 2521
ID Statement 2521
INITIAL Statement 2521
RESTRICT Statement 2522
VAR Statement 2522
Details: STATESPACE Procedure **2523**
Missing Values 2523
Stationarity and Differencing 2523
Preliminary Autoregressive Models 2524
Canonical Correlation Analysis 2527
Parameter Estimation 2530
Forecasting 2531
Relation of ARMA and State Space Forms 2533
OUT= Data Set 2534
OUTAR= Data Set 2535
OUTMODEL= Data Set 2536
Printed Output 2537
ODS Table Names 2538
Examples: STATESPACE Procedure **2539**
Example 35.1: Series J from Box and Jenkins 2539
References **2545**

Overview: STATESPACE Procedure

The STATESPACE procedure has largely been superseded by the newer SSM procedure. PROC SSM fits and forecasts very general linear state space models. It supports irregularly spaced time series and replicated longitudinal data, in addition to supporting regular fixed-period time series. The SSM procedure also provides a powerful expressive language for specifying state space models, and allows programming statements to define model elements through user-written functions of unlimited complexity. The SSM procedure also provides more modern estimation, filtering, and forecasting algorithms than the older STATESPACE procedure. See Chapter 34, "The SSM Procedure," for information about PROC SSM.

Although the SSM procedure should be preferred to the STATESPACE procedure for most state space modeling applications, the STATESPACE procedure should be considered if you wish to perform automated multivariate forecasting using a state space model selected through the modeling strategy proposed by Akaike (1976). This strategy employs an initial sequence of unrestricted vector autoregressive (VAR) models, selection of an initial VAR model using Akaike's information criterion (AIC), followed by a canonical correlation analysis for the automatic identification of the state space model to use to forecast the vector of time series.

The operation of the STATESPACE procedure and the form of state space model it supports are described in the following.

The STATESPACE procedure uses the state space model to analyze and forecast multivariate time series. The STATESPACE procedure is appropriate for jointly forecasting several related time series that have dynamic interactions. By taking into account the autocorrelations among all the variables in a set, it is possible that the STATESPACE procedure may give better forecasts than methods that model each series separately.

By default, the STATESPACE procedure automatically selects a state space model appropriate for the time series, making the procedure a good tool for automatic forecasting of multivariate time series. Alternatively, you can specify the state space model by giving the form of the state vector and the state transition and innovation matrices.

The methods used by the STATESPACE procedure assume that the time series are jointly stationary. Nonstationary series must be made stationary by some preliminary transformation, usually by differencing. The STATESPACE procedure enables you to specify differencing of the input data. When differencing is specified, the STATESPACE procedure automatically integrates forecasts of the differenced series to produce forecasts of the original series.

The State Space Model

The *state space model* represents a multivariate time series through auxiliary variables, some of which might not be directly observable. These auxiliary variables are called the *state vector*. The state vector summarizes all the information from the present and past values of the time series that is relevant to the prediction of future values of the series. The observed time series are expressed as linear combinations of the state variables. The state space model is also called a Markovian representation, or a canonical representation, of a multivariate time series process. The state space approach to modeling a multivariate stationary time series is summarized in Akaike (1976).

The state space form encompasses a very rich class of models. Any Gaussian multivariate stationary time series can be written in a state space form, provided that the dimension of the predictor space is finite. In particular, any autoregressive moving average (ARMA) process has a state space representation and, conversely, any state space process can be expressed in an ARMA form (Akaike 1974). More details on the relation of the state space and ARMA forms are given in the section "Relation of ARMA and State Space Forms" on page 2533.

Let $\mathbf{x}_t$ be the $r \times 1$ vector of observed variables, after differencing (if differencing is specified) and subtracting the sample mean. Let $\mathbf{z}_t$ be the state vector of dimension
Mathtexts, $s \geq r$, where the first r components of $\mathbf{z}_t$ consist of $\mathbf{x}_t$. Let the notation $\mathbf{x}_{t+k|t}$ represent the conditional expectation (or prediction) of $\mathbf{x}_{t+k}$ based on the information available at time t. Then the last $s - r$ elements of $\mathbf{z}_t$ consist of elements of $\mathbf{x}_{t+k|t}$, where $k > 0$ is specified or determined automatically by the procedure.

There are various forms of the state space model in use. The form of the state space model used by the STATESPACE procedure is based on Akaike (1976). The model is defined by the following *state transition equation* :

$$\mathbf{z}_{t+1} = \mathbf{F}\mathbf{z}_t + \mathbf{G}\mathbf{e}_{t+1}$$

In the state transition equation, the $s \times s$ coefficient matrix $\mathbf{F}$ is called the *transition matrix*; it determines the dynamic properties of the model.

The $s \times r$ coefficient matrix $\mathbf{G}$ is called the *input matrix*; it determines the variance structure of the transition equation. For model identification, the first r rows and columns of $\mathbf{G}$ are set to an $r \times r$ identity matrix.

The input vector $\mathbf{e}_t$ is a sequence of independent normally distributed random vectors of dimension r with mean $\mathbf{0}$ and covariance matrix $\Sigma_{\mathbf{ee}}$. The random error $\mathbf{e}_t$ is sometimes called the innovation vector or shock vector.

In addition to the state transition equation, state space models usually include a *measurement equation* or *observation equation* that gives the observed values $\mathbf{x}_t$ as a function of the state vector $\mathbf{z}_t$. However, since PROC STATESPACE always includes the observed values $\mathbf{x}_t$ in the state vector $\mathbf{z}_t$, the measurement equation in this case merely represents the extraction of the first r components of the state vector.

The measurement equation used by the STATESPACE procedure is

$$\mathbf{x}_t = [\mathbf{I}_r \mathbf{0}]\mathbf{z}_t$$

where $\mathbf{I}_r$ is an $r \times r$ identity matrix. In practice, PROC STATESPACE performs the extraction of $\mathbf{x}_t$ from $\mathbf{z}_t$ without reference to an explicit measurement equation.

In summary:

$\mathbf{x}_t$	is an observation vector of dimension r.
$\mathbf{z}_t$	is a state vector of dimension s, whose first r elements are $\mathbf{x}_t$ and whose last $s - r$ elements are conditional prediction of future $\mathbf{x}_t$.
$\mathbf{F}$	is an $s \times s$ transition matrix.
$\mathbf{G}$	is an $s \times r$ input matrix, with the identity matrix $\mathbf{I}_r$ forming the first r rows and columns.
$\mathbf{e}_t$	is a sequence of independent normally distributed random vectors of dimension r with mean $\mathbf{0}$ and covariance matrix $\Sigma_{\mathbf{ee}}$.

How PROC STATESPACE Works

The design of the STATESPACE procedure closely follows the modeling strategy proposed by Akaike (1976). This strategy employs canonical correlation analysis for the automatic identification of the state space model.

Following Akaike (1976), the procedure first fits a sequence of unrestricted vector autoregressive (VAR) models and computes Akaike's information criterion (AIC) for each model. The vector autoregressive models are estimated using the sample autocovariance matrices and the Yule-Walker equations. The order of the VAR model that produces the smallest Akaike information criterion is chosen as the order (number of lags into the past) to use in the canonical correlation analysis.

The elements of the state vector are then determined via a sequence of canonical correlation analyses of the sample autocovariance matrices through the selected order. This analysis computes the sample canonical correlations of the past with an increasing number of steps into the future. Variables that yield significant correlations are added to the state vector; those that yield insignificant correlations are excluded from further consideration. The importance of the correlation is judged on the basis of another information criterion proposed by Akaike. See the section "Canonical Correlation Analysis Options" on page 2518 for details. If you specify the state vector explicitly, these model identification steps are omitted.

After the state vector is determined, the state space model is fit to the data. The free parameters in the $\mathbf{F}$, $\mathbf{G}$, and $\mathbf{\Sigma}_{\mathbf{ee}}$ matrices are estimated by approximate maximum likelihood. By default, the $\mathbf{F}$ and $\mathbf{G}$ matrices are unrestricted, except for identifiability requirements. Optionally, conditional least squares estimates can be computed. You can impose restrictions on elements of the $\mathbf{F}$ and $\mathbf{G}$ matrices.

After the parameters are estimated, the Kalman filtering technique is used to produce forecasts from the fitted state space model. If differencing was specified, the forecasts are integrated to produce forecasts of the original input variables.

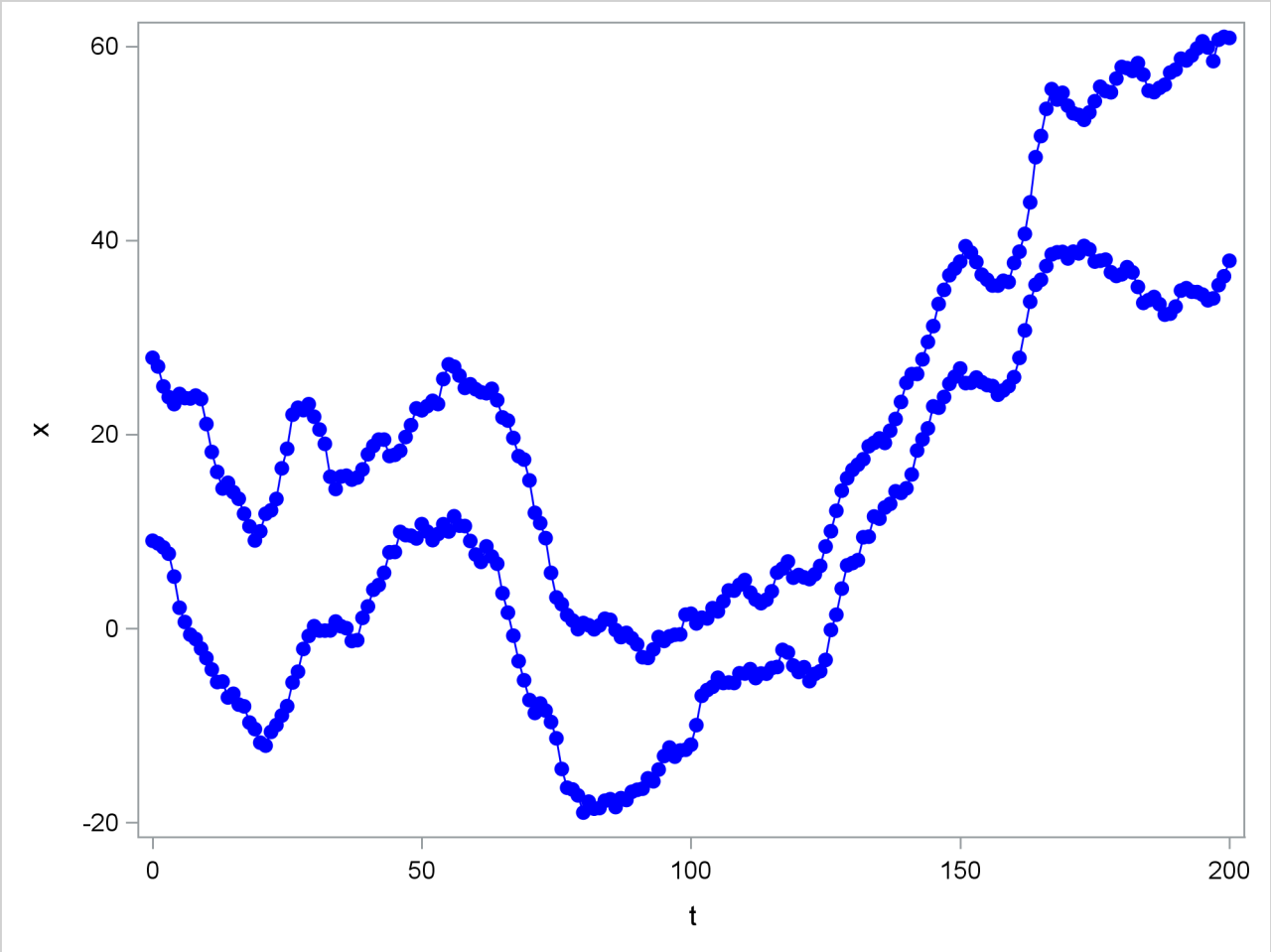
Getting Started: STATESPACE Procedure

The following introductory example uses simulated data for two variables X and Y. The following statements generate the X and Y series.

```
data in;
   x=10;  y=40;
   x1=0; y1=0;
   a1=0; b1=0;
   iseed=123;
   do t=-100 to 200;
      a=rannor(iseed);
      b=rannor(iseed);
      dx = 0.5*x1 + 0.3*y1 + a - 0.2*a1 - 0.1*b1;
      dy = 0.3*x1 + 0.5*y1 + b;
      x = x + dx + .25;
      y = y + dy + .25;
      if t >= 0 then output;
      x1 = dx; y1 = dy;
      a1 = a; b1 = b;
   end;
   keep t x y;
run;
```

The simulated series X and Y are shown in Figure 35.1.

Figure 35.1 Example Series



Automatic State Space Model Selection

The STATESPACE procedure is designed to automatically select the best state space model for forecasting the series. You can specify your own model if you want, and you can use the output from PROC STATESPACE to help you identify a state space model. However, the easiest way to use PROC STATESPACE is to let it choose the model.

Stationarity and Differencing

Although PROC STATESPACE selects the state space model automatically, it does assume that the input series are stationary. If the series are nonstationary, then the process might fail. Therefore the first step is to examine your data and test to see if differencing is required. (See the section "Stationarity and Differencing" on page 2523 for further discussion of this issue.)

The series shown in Figure 35.1 are nonstationary. In order to forecast X and Y with a state space model, you must difference them (or use some other detrending method). If you fail to difference when needed and try to use PROC STATESPACE with nonstationary data, an inappropriate state space model might be selected, and the model estimation might fail to converge.

The following statements identify and fit a state space model for the first differences of X and Y, and forecast X and Y 10 periods ahead:

```
proc statespace data=in out=out lead=10;
   var x(1) y(1);
   id t;
run;
```

The DATA= option specifies the input data set and the OUT= option specifies the output data set for the forecasts. The LEAD= option specifies forecasting 10 observations past the end of the input data. The VAR statement specifies the variables to forecast and specifies differencing. The notation X(1) Y(1) specifies that the state space model analyzes the first differences of X and Y.

Descriptive Statistics and Preliminary Autoregressions

The first page of the printed output produced by the preceding statements is shown in Figure 35.2.

Figure 35.2 Descriptive Statistics and VAR Order Selection

The STATESPACE Procedure

Number of Observations	200

Variable	Mean	Standard Error	
x	0.144316	1.233457	Has been differenced. With period(s) = 1.
y	0.164871	1.304358	Has been differenced. With period(s) = 1.

The STATESPACE Procedure

Information Criterion for Autoregressive Models										
Lag=0	Lag=1	Lag=2	Lag=3	Lag=4	Lag=5	Lag=6	Lag=7	Lag=8	Lag=9	Lag=10
149.697	8.387786	5.517099	12.05986	15.36952	21.79538	24.00638	29.88874	33.55708	41.17606	47.70222

Schematic Representation of Correlations											
Name/Lag	0	1	2	3	4	5	6	7	8	9	10
x	++	++	++	++	++	++	+.	..	+.	+.	..
y	++	++	++	++	++	+.	+.	+.	+.	..	..

+ is > 2*std error, - is < -2*std error, . is between

Descriptive statistics are printed first, giving the number of nonmissing observations after differencing and the sample means and standard deviations of the differenced series. The sample means are subtracted before the series are modeled (unless the NOCENTER option is specified), and the sample means are added back when the forecasts are produced.

Let X_t and Y_t be the observed values of X and Y, and let x_t and y_t be the values of X and Y after differencing and subtracting the mean difference. The series $\mathbf{x}_t$ modeled by the STATEPSPACE procedure is

$$\mathbf{x}_t = \begin{bmatrix} x_t \\ y_t \end{bmatrix} = \begin{bmatrix} (1-B)X_t - 0.144316 \\ (1-B)Y_t - 0.164871 \end{bmatrix}$$

where B represents the backshift operator.

After the descriptive statistics, PROC STATESPACE prints the Akaike information criterion (AIC) values for the autoregressive models fit to the series. The smallest AIC value, in this case 5.517 at lag 2, determines the number of autocovariance matrices analyzed in the canonical correlation phase.

A schematic representation of the autocorrelations is printed next. This indicates which elements of the autocorrelation matrices at different lags are significantly greater than or less than 0.

The second page of the STATESPACE printed output is shown in Figure 35.3.

Figure 35.3 Partial Autocorrelations and VAR Model

Schematic Representation of Partial Autocorrelations

Name/Lag	1	2	3	4	5	6	7	8	9	10
x	++	+.	..	..	..	..	..	..	..	..
y	++	..	..	..	..	..	..	..	..	..

+ is > 2*std error, - is < -2*std error, . is between

Yule-Walker Estimates for Minimum AIC

	Lag=1		Lag=2	
	x	y	x	y
x	0.257438	0.202237	0.170812	0.133554
y	0.292177	0.469297	-0.00537	-0.00048

Figure 35.3 shows a schematic representation of the partial autocorrelations, similar to the autocorrelations shown in Figure 35.2. The selection of a second order autoregressive model by the AIC statistic looks reasonable in this case because the partial autocorrelations for lags greater than 2 are not significant.

Next, the Yule-Walker estimates for the selected autoregressive model are printed. This output shows the coefficient matrices of the vector autoregressive model at each lag.

Selected State Space Model Form and Preliminary Estimates

After the autoregressive order selection process has determined the number of lags to consider, the canonical correlation analysis phase selects the state vector. By default, output for this process is not printed. You can use the CANCORR option to print details of the canonical correlation analysis. See the section "Canonical Correlation Analysis Options" on page 2518 for an explanation of this process.

After the state vector is selected, the state space model is estimated by approximate maximum likelihood. Information from the canonical correlation analysis and from the preliminary autoregression is used to form preliminary estimates of the state space model parameters. These preliminary estimates are used as starting values for the iterative estimation process.

The form of the state vector and the preliminary estimates are printed next, as shown in Figure 35.4.

Figure 35.4 Preliminary Estimates of State Space Model

The STATESPACE Procedure
Selected Statespace Form and Preliminary Estimates

State Vector		
x(T;T)	y(T;T)	x(T+1;T)

Estimate of Transition Matrix		
0	0	1
0.291536	0.468762	-0.00411
0.24869	0.24484	0.204257

Input Matrix for Innovation	
1	0
0	1
0.257438	0.202237

Variance Matrix for Innovation	
0.945196	0.100786
0.100786	1.014703

Figure 35.4 first prints the state vector as X[T;T] Y[T;T] X[T+1;T]. This notation indicates that the state vector is

$$\mathbf{z}_t = \begin{bmatrix} x_{t|t} \\ y_{t|t} \\ x_{t+1|t} \end{bmatrix}$$

The notation $x_{t+1|t}$ indicates the conditional expectation or prediction of x_{t+1} based on the information available at time t, and $x_{t|t}$ and $y_{t|t}$ are x_t and y_t, respectively.

The remainder of Figure 35.4 shows the preliminary estimates of the transition matrix $\mathbf{F}$, the input matrix $\mathbf{G}$, and the covariance matrix $\mathbf{\Sigma}_{ee}$.

Estimated State Space Model

The next page of the STATESPACE output prints the final estimates of the fitted model, as shown in Figure 35.5. This output has the same form as in Figure 35.4, but it shows the maximum likelihood estimates instead of the preliminary estimates.

Figure 35.5 Fitted State Space Model

The STATESPACE Procedure
Selected Statespace Form and Fitted Model

State Vector		
x(T;T)	y(T;T)	x(T+1;T)

Figure 35.5 *continued*

Estimate of Transition Matrix		
0	0	1
0.297273	0.47376	-0.01998
0.2301	0.228425	0.256031

Input Matrix for Innovation	
1	0
0	1
0.257284	0.202273

Variance Matrix for Innovation	
0.945188	0.100752
0.100752	1.014712

The estimated state space model shown in Figure 35.5 is

$$\begin{bmatrix} x_{t+1|t+1} \\ y_{t+1|t+1} \\ x_{t+2|t+1} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0.297 & 0.474 & -0.020 \\ 0.230 & 0.228 & 0.256 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \\ x_{t+1|t} \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0.257 & 0.202 \end{bmatrix} \begin{bmatrix} e_{t+1} \\ n_{t+1} \end{bmatrix}$$

$$var \begin{bmatrix} e_{t+1} \\ n_{t+1} \end{bmatrix} = \begin{bmatrix} 0.945 & 0.101 \\ 0.101 & 1.015 \end{bmatrix}$$

The next page of the STATESPACE output lists the estimates of the free parameters in the **F** and **G** matrices with standard errors and t statistics, as shown in Figure 35.6.

Figure 35.6 Final Parameter Estimates

Parameter Estimates			
Parameter	Estimate	Standard Error	t Value
F(2,1)	0.297273	0.129995	2.29
F(2,2)	0.473760	0.115688	4.10
F(2,3)	-0.01998	0.313025	-0.06
F(3,1)	0.230100	0.126226	1.82
F(3,2)	0.228425	0.112978	2.02
F(3,3)	0.256031	0.305256	0.84
G(3,1)	0.257284	0.071060	3.62
G(3,2)	0.202273	0.068593	2.95

Convergence Failures

The maximum likelihood estimates are computed by an iterative nonlinear maximization algorithm, which might not converge. If the estimates fail to converge, warning messages are printed in the output.

If you encounter convergence problems, you should recheck the stationarity of the data and ensure that the specified differencing orders are correct. Attempting to fit state space models to nonstationary data is a common cause of convergence failure. You can also use the MAXIT= option to increase the number of iterations allowed, or experiment with the convergence tolerance options DETTOL= and PARMTOL=.

Forecast Data Set

The following statements print the output data set. The WHERE statement excludes the first 190 observations from the output, so that only the forecasts and the last 10 actual observations are printed.

```
proc print data=out;
   id t;
   where t > 190;
run;
```

The PROC PRINT output is shown in Figure 35.7.

Figure 35.7 OUT= Data Set Produced by PROC STATESPACE

t	x	FOR1	RES1	STD1	y	FOR2	RES2	STD2
191	34.8159	33.6299	1.18600	0.97221	58.7189	57.9916	0.72728	1.00733
192	35.0656	35.6598	-0.59419	0.97221	58.5440	59.7718	-1.22780	1.00733
193	34.7034	35.5530	-0.84962	0.97221	59.0476	58.5723	0.47522	1.00733
194	34.6626	34.7597	-0.09707	0.97221	59.7774	59.2241	0.55330	1.00733
195	34.4055	34.8322	-0.42664	0.97221	60.5118	60.1544	0.35738	1.00733
196	33.8210	34.6053	-0.78434	0.97221	59.8750	60.8260	-0.95102	1.00733
197	34.0164	33.6230	0.39333	0.97221	58.4698	59.4502	-0.98046	1.00733
198	35.3819	33.6251	1.75684	0.97221	60.6782	57.9167	2.76150	1.00733
199	36.2954	36.0528	0.24256	0.97221	60.9692	62.1637	-1.19450	1.00733
200	37.8945	37.1431	0.75142	0.97221	60.8586	61.4085	-0.54984	1.00733
201	.	38.5068	.	0.97221	.	61.3161	.	1.00733
202	.	39.0428	.	1.59125	.	61.7509	.	1.83678
203	.	39.4619	.	2.28028	.	62.1546	.	2.62366
204	.	39.8284	.	2.97824	.	62.5099	.	3.38839
205	.	40.1474	.	3.67689	.	62.8275	.	4.12805
206	.	40.4310	.	4.36299	.	63.1139	.	4.84149
207	.	40.6861	.	5.03040	.	63.3755	.	5.52744
208	.	40.9185	.	5.67548	.	63.6174	.	6.18564
209	.	41.1330	.	6.29673	.	63.8435	.	6.81655
210	.	41.3332	.	6.89383	.	64.0572	.	7.42114

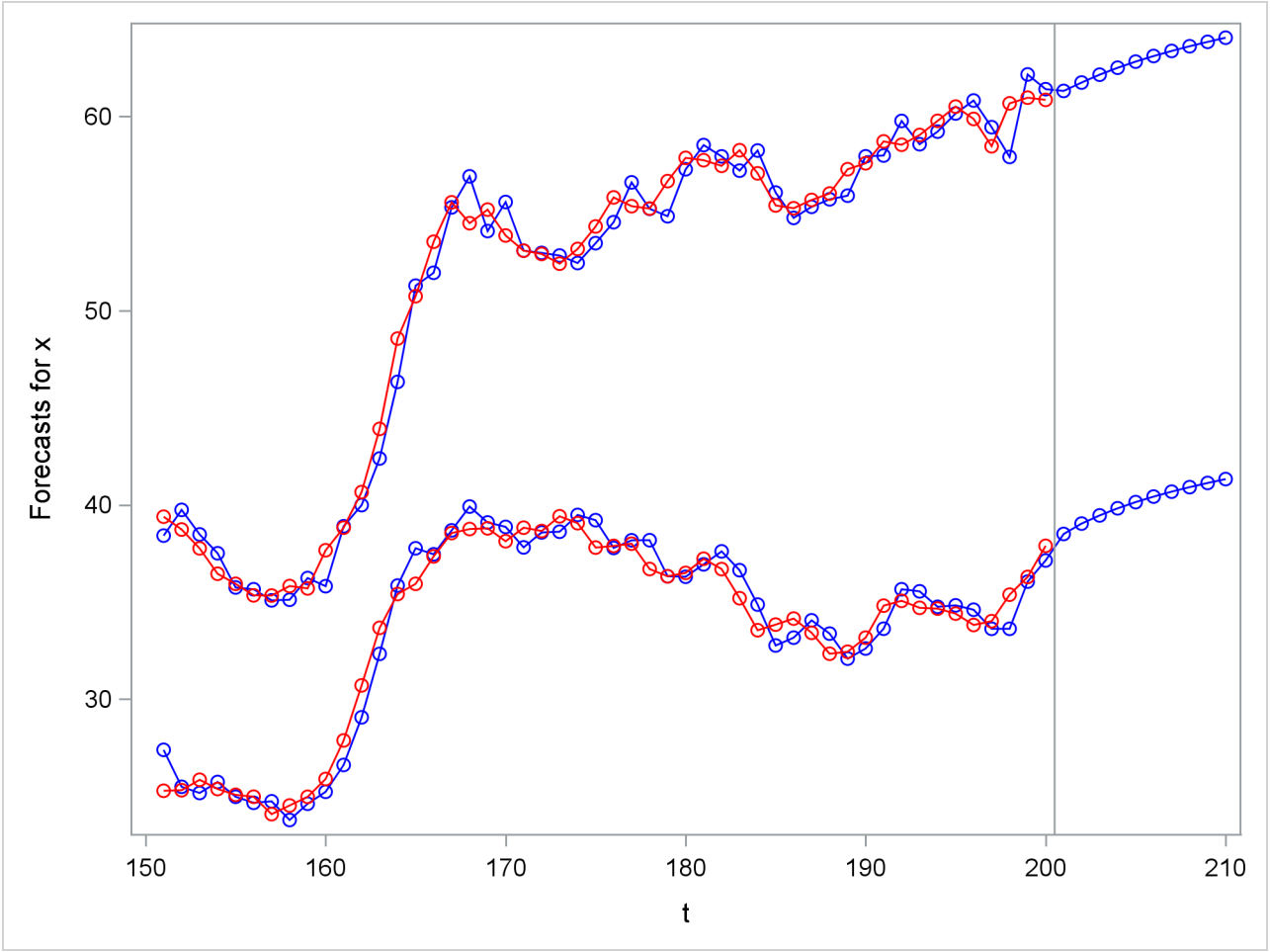
The OUT= data set produced by PROC STATESPACE contains the VAR and ID statement variables. In addition, for each VAR statement variable, the OUT= data set contains the variables FOR*i*, RES*i*, and STD*i*. These variables contain the predicted values, residuals, and forecast standard errors for the *i*th variable in the VAR statement list. In this case, X is listed first in the VAR statement, so FOR1 contains the forecasts of X, while FOR2 contains the forecasts of Y.

The following statements plot the forecasts and actuals for the series.

```
proc sgplot data=out noautolegend;
   where t > 150;
   series x=t y=for1 / markers
    markerattrs=(symbol=circle color=blue)
    lineattrs=(pattern=solid color=blue);
  series x=t y=for2 / markers
   markerattrs=(symbol=circle color=blue)
   lineattrs=(pattern=solid color=blue);
  series x=t y=x / markers
   markerattrs=(symbol=circle color=red)
   lineattrs=(pattern=solid color=red);
  series x=t y=y / markers
   markerattrs=(symbol=circle color=red)
   lineattrs=(pattern=solid color=red);
  refline 200.5 / axis=x;
run;
```

The forecast plot is shown in Figure 35.8. The last 50 observations are also plotted to provide context, and a reference line is drawn between the historical and forecast periods.

Figure 35.8 Plot of Forecasts



Controlling Printed Output

By default, the STATESPACE procedure produces a large amount of printed output. The NOPRINT option suppresses all printed output. You can suppress the printed output for the autoregressive model selection process with the PRINTOUT=NONE option. The descriptive statistics and state space model estimation output are still printed when PRINTOUT=NONE is specified. You can produce more detailed output with the PRINTOUT=LONG option and by specifying the printing control options CANCORR, COVB, and PRINT.

Specifying the State Space Model

Instead of allowing the STATESPACE procedure to select the model automatically, you can use FORM and RESTRICT statements to specify a state space model.

Specifying the State Vector

Use the FORM statement to control the form of the state vector. You can use this feature to force PROC STATESPACE to estimate and forecast a model different from the model it would select automatically. You can also use this feature to reestimate the automatically selected model (possibly with restrictions) without repeating the canonical correlation analysis.

The FORM statement specifies the number of lags of each variable to include in the state vector. For example, the statement FORM X 3; forces the state vector to include $x_{t|t}$, $x_{t+1|t}$, and $x_{t+2|t}$. The following statement specifies the state vector $(x_{t|t}, y_{t|t}, x_{t+1|t})$, which is the same state vector selected in the preceding example:

```
form x 2 y 1;
```

You can specify the form for only some of the variables and allow PROC STATESPACE to select the form for the other variables. If only some of the variables are specified in the FORM statement, canonical correlation analysis is used to determine the number of lags included in the state vector for the remaining variables not specified by the FORM statement. If the FORM statement includes specifications for all the variables listed in the VAR statement, the state vector is completely defined and the canonical correlation analysis is not performed.

Restricting the F and G matrices

After you know the form of the state vector, you can use the RESTRICT statement to fix some parameters in the **F** and **G** matrices to specified values. One use of this feature is to remove insignificant parameters by restricting them to 0.

In the introductory example shown in the preceding section, the F[2,3] parameter is not significant. (The parameters estimation output shown in Figure 35.6 gives the t statistic for F[2,3] as –0.06. F[3,3] and F[3,1] also have low significance with $t < 2$.)

The following statements reestimate this model with F[2,3] restricted to 0. The FORM statement is used to specify the state vector and thus bypass the canonical correlation analysis.

```
proc statespace data=in out=out lead=10;
   var x(1) y(1);
   id t;
   form x 2 y 1;
   restrict f(2,3)=0;
run;
```

The final estimates produced by these statements are shown in Figure 35.10.

Figure 35.9 Results Using RESTRICT Statement

The STATESPACE Procedure
Selected Statespace Form and Fitted Model

State Vector		
x(T;T)	y(T;T)	x(T+1;T)

Estimate of Transition Matrix		
0	0	1
0.290051	0.467468	0
0.227051	0.226139	0.26436

Input Matrix for Innovation	
1	0
0	1
0.256826	0.202022

Variance Matrix for Innovation	
0.945175	0.100696
0.100696	1.014733

Figure 35.10 Restricted Parameter Estiamtes

Parameter Estimates

Parameter	Estimate	Standard Error	t Value
F(2,1)	0.290051	0.063904	4.54
F(2,2)	0.467468	0.060430	7.74
F(3,1)	0.227051	0.125221	1.81
F(3,2)	0.226139	0.111711	2.02
F(3,3)	0.264360	0.299537	0.88
G(3,1)	0.256826	0.070994	3.62
G(3,2)	0.202022	0.068507	2.95

Syntax: STATESPACE Procedure

The STATESPACE procedure uses the following statements:

PROC STATESPACE *options* ;
  BY *variable . . .* ;
  FORM *variable value . . .* ;
  ID *variable* ;
  INITIAL *F (row,column)=value . . . G(row,column)=value . . .* ;
  RESTRICT *F (row,column)=value . . . G (row,column)=value . . .* ;
  VAR *variable (difference, difference, . . .) . . .* ;

Functional Summary

Table 35.1 summarizes the statements and options used by PROC STATESPACE.

Table 35.1 STATESPACE Functional Summary

Description	Statement	Option
Input Data Set Options		
specify the input data set	PROC STATESPACE	DATA=
prevent subtraction of sample mean	PROC STATESPACE	NOCENTER
specify the ID variable	ID	
specify the observed series and differencing	VAR	
Options for Autoregressive Estimates		
specify the maximum order	PROC STATESPACE	ARMAX=
specify maximum lag for autocovariances	PROC STATESPACE	LAGMAX=
output only minimum AIC model	PROC STATESPACE	MINIC
specify the amount of detail printed	PROC STATESPACE	PRINTOUT=
write preliminary AR models to a data set	PROC STATESPACE	OUTAR=
Options for Canonical Correlation Analysis		
print the sequence of canonical correlations	PROC STATESPACE	CANCORR
specify upper limit of dimension of state vector	PROC STATESPACE	DIMMAX=
specify the minimum number of lags	PROC STATESPACE	PASTMIN=
specify the multiplier of the degrees of freedom	PROC STATESPACE	SIGCORR=
Options for State Space Model Estimation		
specify starting values	INITIAL	
print covariance matrix of parameter estimates	PROC STATESPACE	COVB
specify the convergence criterion	PROC STATESPACE	DETTOL=
specify the convergence criterion	PROC STATESPACE	PARMTOL=
print the details of the iterations	PROC STATESPACE	ITPRINT
specify an upper limit of the number of lags	PROC STATESPACE	KLAG=

Description	Statement	Option
specify maximum number of iterations allowed	PROC STATESPACE	MAXIT=
suppress the final estimation	PROC STATESPACE	NOEST
write the state space model parameter estimates to an output data set	PROC STATESPACE	OUTMODEL=
use conditional least squares for final estimates	PROC STATESPACE	RESIDEST
specify criterion for testing for singularity	PROC STATESPACE	SINGULAR=
Options for Forecasting		
start forecasting before end of the input data	PROC STATESPACE	BACK=
specify the time interval between observations	PROC STATESPACE	INTERVAL=
specify multiple periods in the time series	PROC STATESPACE	INTPER=
specify how many periods to forecast	PROC STATESPACE	LEAD=
specify the output data set for forecasts	PROC STATESPACE	OUT=
print forecasts	PROC STATESPACE	PRINT
Options to Specify the State Space Model		
specify the state vector	FORM	
specify the parameter values	RESTRICT	
BY Groups		
specify BY-group processing	BY	
Printing		
suppresses all printed output	NOPRINT	

PROC STATESPACE Statement

PROC STATESPACE *options* **;**

The following options can be specified in the PROC STATESPACE statement.

Printing Options

NOPRINT
suppresses all printed output.

Input Data Options

DATA=*SAS-data-set*
specifies the name of the SAS data set to be used by the procedure. If the DATA= option is omitted, the most recently created SAS data set is used.

LAGMAX=*k*

specifies the number of lags for which the sample autocovariance matrix is computed. The LAGMAX= option controls the number of lags printed in the schematic representation of the autocorrelations.

The sample autocovariance matrix of lag i, denoted as $\mathbf{C}_i$, is computed as

$$\mathbf{C}_i = \frac{1}{N-1} \sum_{t=1+i}^{N} \mathbf{x}_t \mathbf{x}'_{t-i}$$

where $\mathbf{x}_t$ is the differenced and centered data and N is the number of observations. (If the NOCENTER option is specified, 1 is not subtracted from N.) LAGMAX= k specifies that $\mathbf{C}_0$ through $\mathbf{C}_k$ are computed. The default is LAGMAX=10.

NOCENTER

prevents subtraction of the sample mean from the input series (after any specified differencing) before the analysis.

Options for Preliminary Autoregressive Models

ARMAX=*n*

specifies the maximum order of the preliminary autoregressive models. The ARMAX= option controls the autoregressive orders for which information criteria are printed, and controls the number of lags printed in the schematic representation of partial autocorrelations. The default is ARMAX=10. See the section "Preliminary Autoregressive Models" on page 2524 for details.

MINIC

writes to the OUTAR= data set only the preliminary Yule-Walker estimates for the VAR model that produces the minimum AIC. See the section "OUTAR= Data Set" on page 2535 for details.

OUTAR=*SAS-data-set*

writes the Yule-Walker estimates of the preliminary autoregressive models to a SAS data set. See the section "OUTAR= Data Set" on page 2535 for details.

PRINTOUT=SHORT | LONG | NONE

determines the amount of detail printed. PRINTOUT=LONG prints the lagged covariance matrices, the partial autoregressive matrices, and estimates of the residual covariance matrices from the sequence of autoregressive models. PRINTOUT=NONE suppresses the output for the preliminary autoregressive models. The descriptive statistics and state space model estimation output are still printed when PRINTOUT=NONE is specified. PRINTOUT=SHORT is the default.

Canonical Correlation Analysis Options

CANCORR

prints the canonical correlations and information criterion for each candidate state vector considered. See the section "Canonical Correlation Analysis Options" on page 2518 for details.

DIMMAX=*n*

specifies the upper limit to the dimension of the state vector. The DIMMAX= option can be used to limit the size of the model selected. The default is DIMMAX=10.

PASTMIN=*n*

specifies the minimum number of lags to include in the canonical correlation analysis. The default is PASTMIN=0. See the section "Canonical Correlation Analysis Options" on page 2518 for details.

SIGCORR=*value*

specifies the multiplier of the degrees of freedom for the penalty term in the information criterion used to select the state space form. The default is SIGCORR=2. The larger the value of the SIGCORR= option, the smaller the state vector tends to be. Hence, a large value causes a simpler model to be fit. See the section "Canonical Correlation Analysis Options" on page 2518 for details.

State Space Model Estimation Options

COVB

prints the inverse of the observed information matrix for the parameter estimates. This matrix is an estimate of the covariance matrix for the parameter estimates.

DETTOL=*value*

specifies the convergence criterion. The DETTOL= and PARMTOL= option values are used together to test for convergence of the estimation process. If, during an iteration, the relative change of the parameter estimates is less than the PARMTOL= value and the relative change of the determinant of the innovation variance matrix is less than the DETTOL= value, then iteration ceases and the current estimates are accepted. The default is DETTOL=1E–5.

ITPRINT

prints the iterations during the estimation process.

KLAG=*n*

sets an upper limit for the number of lags of the sample autocovariance matrix used in computing the approximate likelihood function. If the data have a strong moving average character, a larger KLAG= value might be necessary to obtain good estimates. The default is KLAG=15. See the section "Parameter Estimation" on page 2530 for details.

MAXIT=*n*

sets an upper limit to the number of iterations in the maximum likelihood or conditional least squares estimation. The default is MAXIT=50.

NOEST

suppresses the final maximum likelihood estimation of the selected model.

OUTMODEL=*SAS-data-set*

writes the parameter estimates and their standard errors to a SAS data set. See the section "OUTMODEL= Data Set" on page 2536 for details.

PARMTOL=*value*

specifies the convergence criterion. The DETTOL= and PARMTOL= option values are used together to test for convergence of the estimation process. If, during an iteration, the relative change of the parameter estimates is less than the PARMTOL= value and the relative change of the determinant of the innovation variance matrix is less than the DETTOL= value, then iteration ceases and the current estimates are accepted. The default is PARMTOL=0.001.

RESIDEST
computes the final estimates by using conditional least squares on the raw data. This type of estimation might be more stable than the default maximum likelihood method but is usually more computationally expensive. See the section "Parameter Estimation" on page 2530 for details about the conditional least squares method.

SINGULAR=*value*
specifies the criterion for testing for singularity of a matrix. A matrix is declared singular if a scaled pivot is less than the SINGULAR= value when sweeping the matrix. The default is SINGULAR=1E–7.

Forecasting Options

BACK=*n*
starts forecasting *n* periods before the end of the input data. The BACK= option value must not be greater than the number of observations. The default is BACK=0.

INTERVAL=*interval*
specifies the time interval between observations. The INTERVAL= value is used in conjunction with the ID variable to check that the input data are in order and have no missing periods. The INTERVAL= option is also used to extrapolate the ID values past the end of the input data. See Chapter 5, "Date Intervals, Formats, and Functions," for details about the INTERVAL= values allowed.

INTPER=*n*
specifies that each input observation corresponds to *n* time periods. For example, the options INTERVAL=MONTH and INTPER=2 specify bimonthly data and are equivalent to specifying INTERVAL=MONTH2. If the INTERVAL= option is not specified, the INTPER= option controls the increment used to generate ID values for the forecast observations. The default is INTPER=1.

LEAD=*n*
specifies how many forecast observations are produced. The forecasts start at the point set by the BACK= option. The default is LEAD=0, which produces no forecasts.

OUT=*SAS-data-set*
writes the residuals, actual values, forecasts, and forecast standard errors to a SAS data set. See the section "OUT= Data Set" on page 2534 for details.

PRINT
prints the forecasts.

BY Statement

BY *variable . . .* **;**

A BY statement can be used with the STATESPACE procedure to obtain separate analyses on observations in groups defined by the BY variables.

FORM Statement

FORM *variable value . . .* **;**

The FORM statement specifies the number of times a variable is included in the state vector. Values can be specified for any variable listed in the VAR statement. If a value is specified for each variable in the VAR statement, the state vector for the state space model is entirely specified, and automatic selection of the state space model is not performed.

The FORM statement forces the state vector, $\mathbf{z}_t$, to contain a specific variable a given number of times. For example, if Y is one of the variables in $\mathbf{x}_t$, then the statement

```
form y 3;
```

forces the state vector to contain Y_t, $Y_{t+1|t}$, and $Y_{t+2|t}$, possibly along with other variables.

The following statements illustrate the use of the FORM statement:

```
proc statespace data=in;
   var x y;
   form x 3 y 2;
run;
```

These statements fit a state space model with the following state vector:

$$\mathbf{z_t} = \begin{bmatrix} x_{t|t} \\ y_{t|t} \\ x_{t+1|t} \\ y_{t+1|t} \\ x_{t+2|t} \end{bmatrix}$$

ID Statement

ID *variable* **;**

The ID statement specifies a variable that identifies observations in the input data set. The variable specified in the ID statement is included in the OUT= data set. The values of the ID variable are extrapolated for the forecast observations based on the values of the INTERVAL= and INTPER= options.

INITIAL Statement

INITIAL *F (row,column)= value . . . G(row, column)= value . . .* **;**

The INITIAL statement gives initial values to the specified elements of the **F** and **G** matrices. These initial values are used as starting values for the iterative estimation.

Parts of the **F** and **G** matrices represent fixed structural identities. If an element specified is a fixed structural element instead of a free parameter, the corresponding initialization is ignored.

The following is an example of an INITIAL statement:

```
initial f(3,2)=0 g(4,1)=0 g(5,1)=0;
```

RESTRICT Statement

RESTRICT *F(row,column)= value . . . G(row,column)= value . . .* **;**

The RESTRICT statement restricts the specified elements of the **F** and **G** matrices to the specified values.

To use the restrict statement, you need to know the form of the model. Either specify the form of the model with the FORM statement, or do a preliminary run (perhaps with the NOEST option) to find the form of the model that PROC STATESPACE selects for the data.

The following is an example of a RESTRICT statement:

```
restrict f(3,2)=0 g(4,1)=0 g(5,1)=0 ;
```

Parts of the **F** and **G** matrices represent fixed structural identities. If a restriction is specified for an element that is a fixed structural element instead of a free parameter, the restriction is ignored.

VAR Statement

VAR *variable (difference, difference, . . .) . . .* **;**

The VAR statement specifies the variables in the input data set to model and forecast. The VAR statement also specifies differencing of the input variables. The VAR statement is required.

Differencing is specified by following the variable name with a list of difference periods separated by commas. See the section "Stationarity and Differencing" on page 2523 for more information about differencing of input variables.

The order in which variables are listed in the VAR statement controls the order in which variables are included in the state vector. Usually, potential inputs should be listed before potential outputs.

For example, assuming the input data are monthly, the following VAR statement specifies modeling and forecasting of the one period and seasonal second difference of X and Y:

```
var x(1,12) y(1,12);
```

In this example, the vector time series analyzed is

$$\mathbf{x}_t = \begin{bmatrix} (1-B)(1-B^{12})X_t - \overline{x} \\ (1-B)(1-B^{12})Y_t - \overline{y} \end{bmatrix}$$

where B represents the back shift operator and $\overline{x}$ and $\overline{y}$ represent the means of the differenced series. If the NOCENTER option is specified, the mean differences are not subtracted.

Details: STATESPACE Procedure

Missing Values

The STATESPACE procedure does not support missing values. The procedure uses the first contiguous group of observations with no missing values for any of the VAR statement variables. Observations at the beginning of the data set with missing values for any VAR statement variable are not used or included in the output data set.

Stationarity and Differencing

The state space model used by the STATESPACE procedure assumes that the time series are stationary. Hence, the data should be checked for stationarity. One way to check for stationarity is to plot the series. A graph of series over time can show a time trend or variability changes.

You can also check stationarity by using the sample autocorrelation functions displayed by the ARIMA procedure. The autocorrelation functions of nonstationary series tend to decay slowly. See Chapter 8, "The ARIMA Procedure," for more information.

Another alternative is to use the STATIONARITY= option in the IDENTIFY statement in PROC ARIMA to apply Dickey-Fuller tests for unit roots in the time series. See Chapter 8, "The ARIMA Procedure," for more information about Dickey-Fuller unit root tests.

The most popular way to transform a nonstationary series to stationarity is by differencing. Differencing of the time series is specified in the VAR statement. For example, to take a simple first difference of the series X, use this statement:

```
var x(1);
```

In this example, the change in X from one period to the next is analyzed. When the series has a seasonal pattern, differencing at a period equal to the length of the seasonal cycle can be desirable. For example, suppose the variable X is measured quarterly and shows a seasonal cycle over the year. You can use the following statement to analyze the series of changes from the same quarter in the previous year:

```
var x(4);
```

To difference twice, add another differencing period to the list. For example, the following statement analyzes the series of second differences $(X_t - X_{t-1}) - (X_{t-1} - X_{t-2}) = X_t - 2X_{t-1} + X_{t-2}$:

```
var x(1,1);
```

The following statement analyzes the seasonal second difference series:

```
var x(1,4);
```

The series that is being modeled is the 1-period difference of the 4-period difference:
$(X_t - X_{t-4}) - (X_{t-1} - X_{t-5}) = X_t - X_{t-1} - X_{t-4} + X_{t-5}$.

Another way to obtain stationary series is to use a regression on time to detrend the data. If the time series has a deterministic linear trend, regressing the series on time produces residuals that should be stationary. The following statements write residuals of X and Y to the variable RX and RY in the output data set DETREND.

```
data a;
   set a;
   t=_n_;
run;

proc reg data=a;
   model x y = t;
   output out=detrend r=rx ry;
run;
```

You then use PROC STATESPACE to forecast the detrended series RX and RY. A disadvantage of this method is that you need to add the trend back to the forecast series in an additional step. A more serious disadvantage of the detrending method is that it assumes a deterministic trend. In practice, most time series appear to have a stochastic rather than a deterministic trend. Differencing is a more flexible and often more appropriate method.

There are several other methods to handle nonstationary time series. For more information and examples, see Brockwell and Davis (1991).

Preliminary Autoregressive Models

After computing the sample autocovariance matrices, PROC STATESPACE fits a sequence of vector autoregressive models. These preliminary autoregressive models are used to estimate the autoregressive order of the process and limit the order of the autocovariances considered in the state vector selection process.

Yule-Walker Equations for Forward and Backward Models

Unlike a univariate autoregressive model, a multivariate autoregressive model has different forms, depending on whether the present observation is being predicted from the past observations or from the future observations.

Let $\mathbf{x}_t$ be the r-component stationary time series given by the VAR statement after differencing and subtracting the vector of sample means. (If the NOCENTER option is specified, the mean is not subtracted.) Let n be the number of observations of $\mathbf{x}_t$ from the input data set.

Let $\mathbf{e}_t$ be a vector white noise sequence with mean vector $\mathbf{0}$ and variance matrix $\mathbf{\Sigma}_p$, and let $\mathbf{n}_t$ be a vector white noise sequence with mean vector $\mathbf{0}$ and variance matrix $\mathbf{\Omega}_p$. Let p be the order of the vector autoregressive model for $\mathbf{x}_t$.

The forward autoregressive form based on the past observations is written as follows:

$$\mathbf{x}_t = \sum_{i=1}^{p} \mathbf{\Phi}_i^p \mathbf{x}_{t-i} + \mathbf{e}_t$$

The backward autoregressive form based on the future observations is written as follows:

$$\mathbf{x}_t = \sum_{i=1}^{p} \mathbf{\Psi}_i^p \mathbf{x}_{t+i} + \mathbf{n}_t$$

Letting E denote the expected value operator, the autocovariance sequence for the $\mathbf{x}_t$ series, $\mathbf{\Gamma}_i$, is

$$\mathbf{\Gamma}_i = E\mathbf{x}_t\mathbf{x}_{t-i}'$$

The Yule-Walker equations for the autoregressive model that matches the first p elements of the autocovariance sequence are

$$\begin{bmatrix} \mathbf{\Gamma}_0 & \mathbf{\Gamma}_1 & \cdots & \mathbf{\Gamma}_{p-1} \\ \mathbf{\Gamma}_1' & \mathbf{\Gamma}_0 & \cdots & \mathbf{\Gamma}_{p-2} \\ \vdots & \vdots & & \vdots \\ \mathbf{\Gamma}_{p-1}' & \mathbf{\Gamma}_{p-2}' & \cdots & \mathbf{\Gamma}_0 \end{bmatrix} \begin{bmatrix} \mathbf{\Phi}_1^p \\ \mathbf{\Phi}_2^p \\ \vdots \\ \mathbf{\Phi}_p^p \end{bmatrix} = \begin{bmatrix} \mathbf{\Gamma}_1 \\ \mathbf{\Gamma}_2 \\ \vdots \\ \mathbf{\Gamma}_p \end{bmatrix}$$

and

$$\begin{bmatrix} \mathbf{\Gamma}_0 & \mathbf{\Gamma}_1' & \cdots & \mathbf{\Gamma}_{p-1}' \\ \mathbf{\Gamma}_1 & \mathbf{\Gamma}_0 & \cdots & \mathbf{\Gamma}_{p-2}' \\ \vdots & \vdots & & \vdots \\ \mathbf{\Gamma}_{p-1} & \mathbf{\Gamma}_{p-2} & \cdots & \mathbf{\Gamma}_0 \end{bmatrix} \begin{bmatrix} \mathbf{\Psi}_1^p \\ \mathbf{\Psi}_2^p \\ \vdots \\ \mathbf{\Psi}_p^p \end{bmatrix} = \begin{bmatrix} \mathbf{\Gamma}_1' \\ \mathbf{\Gamma}_2' \\ \vdots \\ \mathbf{\Gamma}_p' \end{bmatrix}$$

Here $\mathbf{\Phi}_i^p$ are the coefficient matrices for the past observation form of the vector autoregressive model, and $\mathbf{\Psi}_i^p$ are the coefficient matrices for the future observation form. More information about the Yule-Walker equations in the multivariate setting can be found in Whittle (1963); Ansley and Newbold (1979).

The innovation variance matrices for the two forms can be written as follows:

$$\mathbf{\Sigma}_p = \mathbf{\Gamma}_0 - \sum_{i=1}^{p} \mathbf{\Phi}_i^p \mathbf{\Gamma}_i'$$

$$\mathbf{\Omega}_p = \mathbf{\Gamma}_0 - \sum_{i=1}^{p} \mathbf{\Psi}_i^p \mathbf{\Gamma}_i$$

The autoregressive models are fit to the data by using the preceding Yule-Walker equations with $\mathbf{\Gamma}_i$ replaced by the sample covariance sequence $\mathbf{C_i}$. The covariance matrices are calculated as

$$\mathbf{C}_i = \frac{1}{N-1} \sum_{t=i+1}^{N} \mathbf{x}_t \mathbf{x}_{t-i}'$$

Let $\widehat{\mathbf{\Phi}}_p$, $\widehat{\mathbf{\Psi}}_p$, $\widehat{\mathbf{\Sigma}}_p$, and $\widehat{\mathbf{\Omega}}_p$ represent the Yule-Walker estimates of $\mathbf{\Phi}_p$, $\mathbf{\Psi}_p$, $\mathbf{\Sigma}_p$, and $\mathbf{\Omega}_p$, respectively. These matrices are written to an output data set when the OUTAR= option is specified.

When the PRINTOUT=LONG option is specified, the sequence of matrices $\widehat{\mathbf{\Sigma}}_p$ and the corresponding correlation matrices are printed. The sequence of matrices $\widehat{\mathbf{\Sigma}}_p$ is used to compute Akaike information criteria for selection of the autoregressive order of the process.

Akaike Information Criterion

The Akaike information criterion (AIC) is defined as –2(*maximum of log likelihood*)+2(*number of parameters*). Since the vector autoregressive models are estimates from the Yule-Walker equations, not by maximum likelihood, the exact likelihood values are not available for computing the AIC. However, for the vector autoregressive model the maximum of the log likelihood can be approximated as

$$\ln(L) \approx -\frac{n}{2}\ln(|\widehat{\mathbf{\Sigma}}_p|)$$

Thus, the AIC for the order p model is computed as

$$AIC_p = n\ln(|\widehat{\mathbf{\Sigma}}_p|) + 2pr^2$$

You can use the printed AIC array to compute a likelihood ratio test of the autoregressive order. The log-likelihood ratio test statistic for testing the order p model against the order $p-1$ model is

$$-n\ln(|\widehat{\mathbf{\Sigma}}_p|) + n\ln(|\widehat{\mathbf{\Sigma}}_{p-1}|)$$

This quantity is asymptotically distributed as a χ^2 with r^2 degrees of freedom if the series is autoregressive of order $p-1$. It can be computed from the AIC array as

$$AIC_{p-1} - AIC_p + 2r^2$$

You can evaluate the significance of these test statistics with the PROBCHI function in a SAS DATA step or with a χ^2 table.

Determining the Autoregressive Order

Although the autoregressive models can be used for prediction, their primary value is to aid in the selection of a suitable portion of the sample covariance matrix for use in computing canonical correlations. If the multivariate time series $\mathbf{x}_t$ is of autoregressive order p, then the vector of past values to lag p is considered to contain essentially all the information relevant for prediction of future values of the time series.

By default, PROC STATESPACE selects the order p that produces the autoregressive model with the smallest AIC_p. If the value p for the minimum AIC_p is less than the value of the PASTMIN= option, then p is set to the PASTMIN= value. Alternatively, you can use the ARMAX= and PASTMIN= options to force PROC STATESPACE to use an order you select.

Significance Limits for Partial Autocorrelations

The STATESPACE procedure prints a schematic representation of the partial autocorrelation matrices that indicates which partial autocorrelations are significantly greater than or significantly less than 0. Figure 35.11 shows an example of this table.

Figure 35.11 Significant Partial Autocorrelations

Schematic Representation of Partial Autocorrelations										
Name/Lag	1	2	3	4	5	6	7	8	9	10
x	++	+.	..	..	..	..	..	..	..	..
y	++	..	..	..	..	..	..	..	..	..
+ is > 2*std error, - is < -2*std error, . is between										

The partial autocorrelations are from the sample partial autoregressive matrices $\widehat{\mathbf{\Phi}}_p^p$. The standard errors used for the significance limits of the partial autocorrelations are computed from the sequence of matrices $\mathbf{\Sigma}_p$ and $\mathbf{\Omega}_p$.

Under the assumption that the observed series arises from an autoregressive process of order $p-1$, the pth sample partial autoregressive matrix $\widehat{\mathbf{\Phi}}_p^p$ has an asymptotic variance matrix $\frac{1}{n}\mathbf{\Omega}_p^{-1}\otimes\mathbf{\Sigma}_p$.

The significance limits for $\widehat{\mathbf{\Phi}}_p^p$ used in the schematic plot of the sample partial autoregressive sequence are derived by replacing $\mathbf{\Omega}_p$ and $\mathbf{\Sigma}_p$ with their sample estimators to produce the variance estimate, as follows:

$$\widehat{Var}\left(\widehat{\mathbf{\Phi}}_p^p\right) = \left(\frac{1}{n-rp}\right)\widehat{\mathbf{\Omega}}_p^{-1}\otimes\widehat{\mathbf{\Sigma}}_p$$

Canonical Correlation Analysis

Given the order p, let $\mathbf{p}_t$ be the vector of current and past values relevant to prediction of $\mathbf{x}_{t+1}$:

$$\mathbf{p}_t = (\mathbf{x}_t', \mathbf{x}_{t-1}', \cdots, \mathbf{x}_{t-p}')'$$

Let $\mathbf{f}_t$ be the vector of current and future values:

$$\mathbf{f}_t = (\mathbf{x}_t', \mathbf{x}_{t+1}', \cdots, \mathbf{x}_{t+p}')'$$

In the canonical correlation analysis, consider submatrices of the sample covariance matrix of $\mathbf{p}_t$ and $\mathbf{f}_t$. This covariance matrix, $\mathbf{V}$, has a block Hankel form:

$$\mathbf{V} = \begin{bmatrix} \mathbf{C}_0 & \mathbf{C}_1' & \mathbf{C}_2' & \cdots & \mathbf{C}_p' \\ \mathbf{C}_1' & \mathbf{C}_2' & \mathbf{C}_3' & \cdots & \mathbf{C}_{p+1}' \\ \vdots & \vdots & \vdots & & \vdots \\ \mathbf{C}_p' & \mathbf{C}_{p+1}' & \mathbf{C}_{p+2}' & \cdots & \mathbf{C}_{2p}' \end{bmatrix}$$

State Vector Selection Process

The canonical correlation analysis forms a sequence of potential state vectors $\mathbf{z}_t^j$. Examine a sequence $\mathbf{f}_t^j$ of subvectors of $\mathbf{f}_t$, form the submatrix $\mathbf{V}^j$ that consists of the rows and columns of $\mathbf{V}$ that correspond to the components of $\mathbf{f}_t^j$, and compute its canonical correlations.

The smallest canonical correlation of $\mathbf{V}^j$ is then used in the selection of the components of the state vector. The selection process is described in the following discussion. For more details about this process, see Akaike (1976).

In the following discussion, the notation $\mathbf{x}_{t+k|t}$ denotes the wide sense conditional expectation (best linear predictor) of $\mathbf{x}_{t+k}$, given all $\mathbf{x}_s$ with s less than or equal to t. In the notation $x_{i,t+1}$, the first subscript denotes the ith component of $\mathbf{x}_{t+1}$.

The initial state vector $\mathbf{z}_t^1$ is set to $\mathbf{x}_t$. The sequence $\mathbf{f}_t^j$ is initialized by setting

$$\mathbf{f}_t^1 = (\mathbf{z}_t^{1'}, x_{1,t+1|t})' = (\mathbf{x}_t', x_{1,t+1|t})'$$

That is, start by considering whether to add $x_{1,t+1|t}$ to the initial state vector $\mathbf{z}_t^1$.

The procedure forms the submatrix $\mathbf{V}^1$ that corresponds to $\mathbf{f}_t^1$ and computes its canonical correlations. Denote the smallest canonical correlation of $\mathbf{V}^1$ as ρ_{min}. If ρ_{min} is significantly greater than 0, $x_{1,t+1|t}$ is added to the state vector.

If the smallest canonical correlation of $\mathbf{V}^1$ is not significantly greater than 0, then a linear combination of $\mathbf{f}_t^1$ is uncorrelated with the past, $\mathbf{p}_t$. Assuming that the determinant of $\mathbf{C}_0$ is not 0, (that is, no input series is a constant), you can take the coefficient of $x_{1,t+1|t}$ in this linear combination to be 1. Denote the coefficients of $\mathbf{z}_t^1$ in this linear combination as ℓ. This gives the relationship:

$$x_{1,t+1|t} = \ell' \mathbf{x}_t$$

Therefore, the current state vector already contains all the past information useful for predicting $x_{1,t+1}$ and any greater leads of $x_{1,t}$. The variable $x_{1,t+1|t}$ is not added to the state vector, nor are any terms $x_{1,t+k|t}$ considered as possible components of the state vector. The variable x_1 is no longer active for state vector selection.

The process described for $x_{1,t+1|t}$ is repeated for the remaining elements of $\mathbf{f}_t$. The next candidate for inclusion in the state vector is the next component of $\mathbf{f}_t$ that corresponds to an active variable. Components of $\mathbf{f}_t$ that correspond to inactive variables that produced a zero ρ_{min} in a previous step are skipped.

Denote the next candidate as $x_{l,t+k|t}$. The vector $\mathbf{f}_t^j$ is formed from the current state vector and $x_{l,t+k|t}$ as follows:

$$\mathbf{f}_t^j = (\mathbf{z}_t^{j'}, x_{l,t+k|t})'$$

The matrix $\mathbf{V}^j$ is formed from $\mathbf{f}_t^j$ and its canonical correlations are computed. The smallest canonical correlation of $\mathbf{V}^j$ is judged to be either greater than or equal to 0. If it is judged to be greater than 0, $x_{l,t+k|t}$ is added to the state vector. If it is judged to be 0, then a linear combination of $\mathbf{f}_t^j$ is uncorrelated with the $\mathbf{p}_t$, and the variable x_l is now inactive.

The state vector selection process continues until no active variables remain.

Testing Significance of Canonical Correlations

For each step in the canonical correlation sequence, the significance of the smallest canonical correlation ρ_{min} is judged by an information criterion from Akaike (1976). This information criterion is

$$-n\ln(1-\rho_{min}^2) - \lambda(r(p+1) - q + 1)$$

where q is the dimension of $\mathbf{f}_t^j$ at the current step, r is the order of the state vector, p is the order of the vector autoregressive process, and λ is the value of the SIGCORR= option. The default is SIGCORR=2. If this information criterion is less than or equal to 0, ρ_{min} is taken to be 0; otherwise, it is taken to be significantly greater than 0. (Do not confuse this information criterion with the AIC.)

Variables in $\mathbf{x}_{t+p|t}$ are not added in the model, even with positive information criterion, because of the singularity of $\mathbf{V}$. You can force the consideration of more candidate state variables by increasing the size of the $\mathbf{V}$ matrix by specifying a PASTMIN= option value larger than p.

Printing the Canonical Correlations

To print the details of the canonical correlation analysis process, specify the CANCORR option in the PROC STATESPACE statement. The CANCORR option prints the candidate state vectors, the canonical correlations, and the information criteria for testing the significance of the smallest canonical correlation.

Bartlett's χ^2 and its degrees of freedom are also printed when the CANCORR option is specified. The formula used for Bartlett's χ^2 is

$$\chi^2 = -(n - .5(r(p+1) - q + 1))\ln(1 - \rho_{min}^2)$$

with $r(p+1) - q + 1$ degrees of freedom.

Figure 35.12 shows the output of the CANCORR option for the introductory example shown in the "Getting Started: STATESPACE Procedure" on page 2506.

```
proc statespace data=in out=out lead=10 cancorr;
   var x(1) y(1);
   id t;
run;
```

Figure 35.12 Canonical Correlations Analysis

The STATESPACE Procedure
Canonical Correlations Analysis

x(T;T)	y(T;T)	x(T+1;T)	Information Criterion	Chi-Square	DF
1	1	0.237045	3.566167	11.4505	4

New variables are added to the state vector if the information criteria are positive. In this example, $\mathbf{y}_{t+1|t}$ and $\mathbf{x}_{t+2|t}$ are not added to the state space vector because the information criteria for these models are negative.

If the information criterion is nearly 0, then you might want to investigate models that arise if the opposite decision is made regarding ρ_{min}. This investigation can be accomplished by using a FORM statement to specify part or all of the state vector.

Preliminary Estimates of F

When a candidate variable $x_{l,t+k|t}$ yields a zero ρ_{min} and is not added to the state vector, a linear combination of $\mathbf{f}_t^j$ is uncorrelated with the $\mathbf{p}_t$. Because of the method used to construct the $\mathbf{f}_t^j$ sequence, the coefficient of $x_{l,t+k|t}$ in $\mathbf{l}$ can be taken as 1. Denote the coefficients of $\mathbf{z}_t^j$ in this linear combination as $\mathbf{l}$.

This gives the relationship:

$$x_{l,t+k|t} = \mathrm{l}'\mathbf{z}_t^j$$

The vector l is used as a preliminary estimate of the first r columns of the row of the transition matrix $\mathbf{F}$ corresponding to $x_{l,t+k-1|t}$.

Parameter Estimation

The model is $\mathbf{z}_{t+1} = \mathbf{F}\mathbf{z}_t + \mathbf{G}\mathbf{e}_{t+1}$, where $\mathbf{e}_t$ is a sequence of independent multivariate normal innovations with mean vector $\mathbf{0}$ and variance $\Sigma_{\mathbf{ee}}$. The observed sequence $\mathbf{x}_t$ composes the first r components of $\mathbf{z}_t$, and thus $\mathbf{x}_t = \mathbf{H}\mathbf{z}_t$, where $\mathbf{H}$ is the $r \times s$ matrix $[\mathbf{I}_r\ \mathbf{0}]$.

Let $\mathbf{E}$ be the $r \times n$ matrix of innovations:

$$\mathbf{E} = \begin{bmatrix} \mathbf{e}_1 & \cdots & \mathbf{e}_n \end{bmatrix}$$

If the number of observations n is reasonably large, the log likelihood L can be approximated up to an additive constant as follows:

$$L = -\frac{n}{2}\ln(|\mathbf{\Sigma}_{\mathbf{ee}}|) - \frac{1}{2}trace(\mathbf{\Sigma}_{\mathbf{ee}}^{-1}\mathbf{E}\mathbf{E}')$$

The elements of $\mathbf{\Sigma}_{\mathbf{ee}}$ are taken as free parameters and are estimated as follows:

$$\mathbf{S}_0 = \frac{1}{n}\mathbf{E}\mathbf{E}'$$

Replacing $\mathbf{\Sigma}_{\mathbf{ee}}$ by $\mathbf{S}_0$ in the likelihood equation, the log likelihood, up to an additive constant, is

$$\mathbf{L} = -\frac{n}{2}\ln(|\mathbf{S}_0|)$$

Letting B be the backshift operator, the formal relation between $\mathbf{x}_t$ and $\mathbf{e}_t$ is

$$\mathbf{x}_t = \mathbf{H}(\mathbf{I} - B\mathbf{F})^{-1}\mathbf{G}\mathbf{e}_t$$

$$\mathbf{e}_t = (\mathbf{H}(\mathbf{I} - B\mathbf{F})^{-1}\mathbf{G})^{-1}\mathbf{x}_t = \sum_{i=0}^{\infty} \mathbf{\Xi}_i \mathbf{x}_{t-i}$$

Letting $\mathbf{C}_i$ be the ith lagged sample covariance of $\mathbf{x}_t$ and neglecting end effects, the matrix $\mathbf{S}_0$ is

$$\mathbf{S}_0 = \sum_{i,j=0}^{\infty} \mathbf{\Xi}_i \mathbf{C}_{-i+j} \mathbf{\Xi}'_j$$

For the computation of $\mathbf{S}_0$, the infinite sum is truncated at the value of the KLAG= option. The value of the KLAG= option should be large enough that the sequence $\mathbf{\Xi}_i$ is approximately 0 beyond that point.

Let $\boldsymbol{\theta}$ be the vector of free parameters in the $\mathbf{F}$ and $\mathbf{G}$ matrices. The derivative of the log likelihood with respect to the parameter $\boldsymbol{\theta}$ is

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = -\frac{n}{2} \operatorname{trace}\left(\mathbf{S}_0^{-1} \frac{\partial \mathbf{S}_0}{\partial \boldsymbol{\theta}}\right)$$

The second derivative is

$$\frac{\partial^2 \mathbf{L}}{\partial \boldsymbol{\theta}\, \partial \boldsymbol{\theta}'} = \frac{n}{2}\left(\operatorname{trace}\left(\mathbf{S}_0^{-1} \frac{\partial \mathbf{S}_0}{\partial \boldsymbol{\theta}'} \mathbf{S}_0^{-1} \frac{\partial \mathbf{S}_0}{\partial \boldsymbol{\theta}}\right) - \operatorname{trace}\left(\mathbf{S}_0^{-1} \frac{\partial^2 \mathbf{S}_0}{\partial \boldsymbol{\theta}\, \partial \boldsymbol{\theta}'}\right)\right)$$

Near the maximum, the first term is unimportant and the second term can be approximated to give the following second derivative approximation:

$$\frac{\partial^2 L}{\partial \boldsymbol{\theta}\, \partial \boldsymbol{\theta}'} \cong -n \operatorname{trace}\left(\mathbf{S}_0^{-1} \frac{\partial \mathbf{E}}{\partial \boldsymbol{\theta}} \frac{\partial \mathbf{E}'}{\partial \boldsymbol{\theta}'}\right)$$

The first derivative matrix and this second derivative matrix approximation are computed from the sample covariance matrix $\mathbf{C}_0$ and the truncated sequence $\boldsymbol{\Xi}_i$. The approximate likelihood function is maximized by a modified Newton-Raphson algorithm that employs these derivative matrices.

The matrix $\mathbf{S}_0$ is used as the estimate of the innovation covariance matrix, $\boldsymbol{\Sigma}_{\mathbf{ee}}$. The negative of the inverse of the second derivative matrix at the maximum is used as an approximate covariance matrix for the parameter estimates. The standard errors of the parameter estimates printed in the parameter estimates tables are taken from the diagonal of this covariance matrix. The parameter covariance matrix is printed when the COVB option is specified.

If the data are nearly nonstationary, a better estimate of $\boldsymbol{\Sigma}_{\mathbf{ee}}$ and the other parameters can sometimes be obtained by specifying the RESIDEST option. The RESIDEST option estimates the parameters by using conditional least squares instead of maximum likelihood.

The residuals are computed using the state space equation and the sample mean values of the variables in the model as start-up values. The estimate of $\mathbf{S}_0$ is then computed using the residuals from the ith observation on, where i is the maximum number of times any variable occurs in the state vector. A multivariate Gauss-Marquardt algorithm is used to minimize $|\mathbf{S}_0|$. See Harvey (1981a) for a further description of this method.

Forecasting

Given estimates of $\mathbf{F}$, $\mathbf{G}$, and $\boldsymbol{\Sigma}_{\mathbf{ee}}$, forecasts of $\mathbf{x}_t$ are computed from the conditional expectation of $\mathbf{z}_t$.

In forecasting, the parameters $\mathbf{F}$, $\mathbf{G}$, and $\boldsymbol{\Sigma}_{\mathbf{ee}}$ are replaced with the estimates or by values specified in the RESTRICT statement. One-step-ahead forecasting is performed for the observation $\mathbf{x}_t$, where $t \leq n - b$. Here n is the number of observations and b is the value of the BACK= option. For the observation $\mathbf{x}_t$, where $t > n - b$, m-step-ahead forecasting is performed for $m = t - n + b$. The forecasts are generated recursively with the initial condition $\mathbf{z}_0 = 0$.

The m-step-ahead forecast of $\mathbf{z}_{t+m}$ is $\mathbf{z}_{t+m|t}$, where $\mathbf{z}_{t+m|t}$ denotes the conditional expectation of $\mathbf{z}_{t+m}$ given the information available at time t. The m-step-ahead forecast of $\mathbf{x}_{t+m}$ is $\mathbf{x}_{t+m|t} = \mathbf{H}\mathbf{z}_{t+m|t}$, where the matrix $\mathbf{H} = [\mathbf{I}_r \mathbf{0}]$.

Let $\Psi_i = \mathbf{F}^i \mathbf{G}$. Note that the last $s - r$ elements of $\mathbf{z}_t$ consist of the elements of $\mathbf{x}_{u|t}$ for $u > t$.

The state vector $\mathbf{z}_{t+m}$ can be represented as

$$\mathbf{z}_{t+m} = \mathbf{F}^m \mathbf{z}_t + \sum_{i=0}^{m-1} \mathbf{\Psi}_i \mathbf{e}_{t+m-i}$$

Since $\mathbf{e}_{t+i|t} = \mathbf{0}$ for $i > 0$, the m-step-ahead forecast $\mathbf{z}_{t+m|t}$ is

$$\mathbf{z}_{t+m|t} = \mathbf{F}^m \mathbf{z}_t = \mathbf{F}\mathbf{z}_{t+m-1|t}$$

Therefore, the m-step-ahead forecast of $\mathbf{x}_{t+m}$ is

$$\mathbf{x}_{t+m|t} = \mathbf{H}\mathbf{z}_{t+m|t}$$

The m-step-ahead forecast error is

$$\mathbf{z}_{t+m} - \mathbf{z}_{t+m|t} = \sum_{i=0}^{m-1} \mathbf{\Psi}_i \mathbf{e}_{t+m-i}$$

The variance of the m-step-ahead forecast error is

$$\mathbf{V}_{z,m} = \sum_{i=0}^{m-1} \mathbf{\Psi}_i \mathbf{\Sigma}_{\mathbf{ee}} \Psi_i'$$

Letting $\mathbf{V}_{z,0} = \mathbf{0}$, the variance of the m-step-ahead forecast error of $\mathbf{z}_{t+m}$, $\mathbf{V}_{z,m}$, can be computed recursively as follows:

$$\mathbf{V}_{z,m} = \mathbf{V}_{z,m-1} + \mathbf{\Psi}_{m-1} \mathbf{\Sigma}_{\mathbf{ee}} \mathbf{\Psi}_{m-1}^{'}$$

The variance of the m-step-ahead forecast error of $\mathbf{x}_{t+m}$ is the $r \times r$ left upper submatrix of $\mathbf{V}_{z,m}$; that is,

$$\mathbf{V}_{x,m} = \mathbf{H}\mathbf{V}_{z,m}\mathbf{H}'$$

Unless the NOCENTER option is specified, the sample mean vector is added to the forecast. When differencing is specified, the forecasts $\mathbf{x}_{t+m|t}$ plus the sample mean vector are integrated back to produce forecasts for the original series.

Let $\mathbf{y}_t$ be the original series specified by the VAR statement, with some 0 values appended that correspond to the unobserved past observations. Let B be the backshift operator, and let $\mathbf{\Delta}(B)$ be the $s \times s$ matrix polynomial in the backshift operator that corresponds to the differencing specified by the VAR statement. The off-diagonal elements of $\mathbf{\Delta}_i$ are 0. Note that $\mathbf{\Delta}_0 = \mathbf{I}_s$, where $\mathbf{I}_s$ is the $s \times s$ identity matrix. Then $\mathbf{z}_t = \mathbf{\Delta}(B)\mathbf{y}_t$.

This gives the relationship

$$\mathbf{y}_t = \boldsymbol{\Delta}^{-1}(B)\mathbf{z}_t = \sum_{i=0}^{\infty} \boldsymbol{\Lambda}_i \mathbf{z}_{t-i}$$

where $\boldsymbol{\Delta}^{-1}(B) = \sum_{i=0}^{\infty} \boldsymbol{\Lambda}_i B^i$ and $\boldsymbol{\Lambda}_0 = \mathbf{I}_s$.

The m-step-ahead forecast of $\mathbf{y}_{t+m}$ is

$$\mathbf{y}_{t+m|t} = \sum_{i=0}^{m-1} \boldsymbol{\Lambda}_i \mathbf{z}_{t+m-i|t} + \sum_{i=m}^{\infty} \boldsymbol{\Lambda}_i \mathbf{z}_{t+m-i}$$

The m-step-ahead forecast error of $\mathbf{y}_{t+m}$ is

$$\sum_{i=0}^{m-1} \boldsymbol{\Lambda}_i \left(\mathbf{z}_{t+m-i} - \mathbf{z}_{t+m-i|t}\right) = \sum_{i=0}^{m-1} \left(\sum_{u=0}^{i} \boldsymbol{\Lambda}_u \boldsymbol{\Psi}_{i-u} \right) \mathbf{e}_{t+m-i}$$

Letting $\mathbf{V}_{y,0} = \mathbf{0}$, the variance of the m-step-ahead forecast error of $\mathbf{y}_{t+m}$, $\mathbf{V}_{y,m}$, is

$$\begin{aligned} \mathbf{V}_{y,m} &= \sum_{i=0}^{m-1} \left(\sum_{u=0}^{i} \boldsymbol{\Lambda}_u \boldsymbol{\Psi}_{i-u} \right) \boldsymbol{\Sigma}_{\mathbf{ee}} \left(\sum_{u=0}^{i} \boldsymbol{\Lambda}_u \boldsymbol{\Psi}_{i-u} \right)' \\ &= \mathbf{V}_{y,m-1} + \left(\sum_{u=0}^{m-1} \boldsymbol{\Lambda}_u \boldsymbol{\Psi}_{m-1-u} \right) \boldsymbol{\Sigma}_{\mathbf{ee}} \left(\sum_{u=0}^{m-1} \boldsymbol{\Lambda}_u \boldsymbol{\Psi}_{m-1-u} \right)' \end{aligned}$$

Relation of ARMA and State Space Forms

Every state space model has an ARMA representation, and conversely every ARMA model has a state space representation. This section discusses this equivalence. The following material is adapted from Akaike (1974), where there is a more complete discussion. Pham (1978) also contains a discussion of this material.

Suppose you are given the following ARMA model:

$$\boldsymbol{\Phi}(B)\mathbf{x}_t = \boldsymbol{\Theta}(B)\mathbf{e}_t$$

or, in more detail,

$$\mathbf{x}_t - \boldsymbol{\Phi}_1 \mathbf{x}_{t-1} - \cdots - \boldsymbol{\Phi}_p \mathbf{x}_{t-p} = \mathbf{e}_t + \boldsymbol{\Theta}_1 \mathbf{e}_{t-1} + \cdots + \boldsymbol{\Theta}_q \mathbf{e}_{t-q} \qquad (1)$$

where $\mathbf{e}_t$ is a sequence of independent multivariate normal random vectors with mean $\mathbf{0}$ and variance matrix $\boldsymbol{\Sigma}_{\mathbf{ee}}$, B is the backshift operator ($B\mathbf{x}_t = \mathbf{x}_{t-1}$), $\boldsymbol{\Phi}(B)$ and $\boldsymbol{\Theta}(B)$ are matrix polynomials in B, and $\mathbf{x}_t$ is the observed process.

If the roots of the determinantial equation $|\boldsymbol{\Phi}(B)| = 0$ are outside the unit circle in the complex plane, the model can also be written as

$$\mathbf{x}_t = \boldsymbol{\Phi}^{-1}(B)\boldsymbol{\Theta}(B)\mathbf{e}_t = \sum_{i=0}^{\infty} \boldsymbol{\Psi}_i \mathbf{e}_{t-i}$$

The $\mathbf{\Psi}_i$ matrices are known as the impulse response matrices and can be computed as $\mathbf{\Phi}^{-1}(B)\mathbf{\Theta}(B)$.

You can assume $p > q$ since, if this is not initially true, you can add more terms $\mathbf{\Phi}_i$ that are identically 0 without changing the model.

To write this set of equations in a state space form, proceed as follows. Let $\mathbf{x}_{t+i|t}$ be the conditional expectation of $\mathbf{x}_{t+i}$ given $\mathbf{x}_w$ for $w \leq t$. The following relations hold:

$$\mathbf{x}_{t+i|t} = \sum_{j=i}^{\infty} \mathbf{\Psi}_j \mathbf{e}_{t+i-j}$$

$$\mathbf{x}_{t+i|t+1} = \mathbf{x}_{t+i|t} + \mathbf{\Psi}_{i-1}\mathbf{e}_{t+1}$$

However, from equation (1) you can derive the following relationship:

$$\mathbf{x}_{t+p|t} = \mathbf{\Phi}_1 \mathbf{x}_{t+p-1|t} + \cdots + \mathbf{\Phi}_p \mathbf{x}_t \qquad (2)$$

Hence, when $i = p$, you can substitute for $\mathbf{x}_{t+p|t}$ in the right-hand side of equation (2) and close the system of equations.

This substitution results in the following model in the state space form $\mathbf{z}_{t+1} = \mathbf{F}\mathbf{z}_t + \mathbf{G}\mathbf{e}_{t+1}$:

$$\begin{bmatrix} \mathbf{x}_{t+1} \\ \mathbf{x}_{t+2|t+1} \\ \vdots \\ \mathbf{x}_{t+p|t+1} \end{bmatrix} = \begin{bmatrix} 0 & \mathbf{I} & 0 & \cdots & 0 \\ 0 & 0 & \mathbf{I} & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ \mathbf{\Phi}_p & \mathbf{\Phi}_{p-1} & & \cdots & \mathbf{\Phi}_1 \end{bmatrix} \begin{bmatrix} \mathbf{x}_t \\ \mathbf{x}_{t+1|t} \\ \vdots \\ \mathbf{x}_{t+p-1|t} \end{bmatrix} + \begin{bmatrix} \mathbf{I} \\ \mathbf{\Psi}_1 \\ \vdots \\ \mathbf{\Psi}_{p-1} \end{bmatrix} \mathbf{e}_{t+1}$$

Note that the state vector $\mathbf{z}_t$ is composed of conditional expectations of $\mathbf{x}_t$ and the first r components of $\mathbf{z}_t$ are equal to $\mathbf{x}_t$.

The state space form can be cast into an ARMA form by solving the system of difference equations for the first r components.

When converting from an ARMA form to a state space form, you can generate a state vector larger than needed; that is, the state space model might not be a minimal representation. When going from a state space form to an ARMA form, you can have nontrivial common factors in the autoregressive and moving average operators that yield an ARMA model larger than necessary.

If the state space form used is not a minimal representation, some but not all components of $\mathbf{x}_{t+i|t}$ might be linearly dependent. This situation corresponds to $[\mathbf{\Phi}_p \mathbf{\Theta}_{p-1}]$ being of less than full rank when $\mathbf{\Phi}(B)$ and $\mathbf{\Theta}(B)$ have no common nontrivial left factors. In this case, $\mathbf{z}_t$ consists of a subset of the possible components of $[\mathbf{x}_{t+i|t}] \quad i = 1, 2, \cdots, p-1$. However, once a component of $\mathbf{x}_{t+i|t}$ (for example, the jth one) is linearly dependent on the previous conditional expectations, then all subsequent jth components of $\mathbf{x}_{t+k|t}$ for $k > i$ must also be linearly dependent. Note that in this case, equivalent but seemingly different structures can arise if the order of the components within $\mathbf{x}_t$ is changed.

OUT= Data Set

The forecasts are contained in the output data set specified by the OUT= option in the PROC STATESPACE statement. The OUT= data set contains the following variables:

- the BY variables
- the ID variable
- the VAR statement variables. These variables contain the actual values from the input data set.
- FOR*i*, numeric variables that contain the forecasts. The variable FOR*i* contains the forecasts for the *i*th variable in the VAR statement list. Forecasts are one-step-ahead predictions until the end of the data or until the observation specified by the BACK= option.
- RES*i*, numeric variables that contain the residual for the forecast of the *i*th variable in the VAR statement list. For forecast observations, the actual values are missing and the RES*i* variables contain missing values.
- STD*i*, numeric variables that contain the standard deviation for the forecast of the *i*th variable in the VAR statement list. The values of the STD*i* variables can be used to construct univariate confidence limits for the corresponding forecasts. However, such confidence limits do not take into account the covariance of the forecasts.

OUTAR= Data Set

The OUTAR= data set contains the estimates of the preliminary autoregressive models. The OUTAR= data set contains the following variables:

- ORDER, a numeric variable that contains the order p of the autoregressive model that the observation represents
- AIC, a numeric variable that contains the value of the information criterion AIC_p
- SIGF*l*, numeric variables that contain the estimate of the innovation covariance matrices for the forward autoregressive models. The variable SIGF*l* contains the *l*th column of $\widehat{\mathbf{\Sigma}}_p$ in the observations with ORDER=*p*.
- SIGB*l*, numeric variables that contain the estimate of the innovation covariance matrices for the backward autoregressive models. The variable SIGB*l* contains the *l*th column of $\widehat{\mathbf{\Omega}}_p$ in the observations with ORDER=*p*.
- FOR*k* _*l*, numeric variables that contain the estimates of the autoregressive parameter matrices for the forward models. The variable FOR*k* _*l* contains the *l*th column of the lag *k* autoregressive parameter matrix $\widehat{\mathbf{\Phi}}_k^p$ in the observations with ORDER=*p*.
- BAC*k* _*l*, numeric variables that contain the estimates of the autoregressive parameter matrices for the backward models. The variable BAC*k* _*l* contains the *l*th column of the lag *k* autoregressive parameter matrix $\widehat{\mathbf{\Psi}}_k^p$ in the observations with ORDER=*p*.

The estimates for the order p autoregressive model can be selected as those observations with ORDER=*p*. Within these observations, the *k*,*l*th element of $\mathbf{\Phi}_i^p$ is given by the value of the FOR*i* _*l* variable in the *k*th observation. The *k*,*l*th element of $\mathbf{\Psi}_i^p$ is given by the value of BAC*i* _*l* variable in the *k*th observation. The

k,lth element of $\mathbf{\Sigma}_p$ is given by SIGFl in the kth observation. The k,lth element of $\mathbf{\Omega}_p$ is given by SIGBl in the kth observation.

Table 35.2 shows an example of the OUTAR= data set, with ARMAX=3 and $\mathbf{x}_t$ of dimension 2. In Table 35.2, (i, j) indicate $(i, (n)$ element of the matrix.

Table 35.2 Values in the OUTAR= Data Set

Obs	ORDER	AIC	SIGF1	SIGF2	SIGB1	SIGB2	FOR1_1	FOR1_2	FOR2_1	FOR2_2	FOR3_1
1	0	AIC_0	$\mathbf{\Sigma}_{0(1,1)}$	$\mathbf{\Sigma}_{0(1,2)}$	$\mathbf{\Omega}_{0(1,1)}$	$\mathbf{\Omega}_{0(1,2)}$	.	.	.	.	.
2	0	AIC_0	$\mathbf{\Sigma}_{0(2,1)}$	$\mathbf{\Sigma}_{0(2,2)}$	$\mathbf{\Omega}_{0(2,1)}$	$\mathbf{\Omega}_{0(2,2)}$	.	.	.	.	.
3	1	AIC_1	$\mathbf{\Sigma}_{1(1,1)}$	$\mathbf{\Sigma}_{1(1,2)}$	$\mathbf{\Omega}_{1(1,1)}$	$\mathbf{\Omega}_{1(1,2)}$	$\mathbf{\Phi}^1_{1(1,1)}$	$\mathbf{\Phi}^1_{1(1,2)}$	.	.	.
4	1	AIC_1	$\mathbf{\Sigma}_{1(2,1)}$	$\mathbf{\Sigma}_{1(1,2)}$	$\mathbf{\Omega}_{1(2,1)}$	$\mathbf{\Omega}_{1(1,2)}$	$\mathbf{\Phi}^1_{1(2,1)}$	$\mathbf{\Phi}^1_{1(2,2)}$	.	.	.
5	2	AIC_2	$\mathbf{\Sigma}_{2(1,1)}$	$\mathbf{\Sigma}_{2(1,2)}$	$\mathbf{\Omega}_{2(1,1)}$	$\mathbf{\Omega}_{2(1,2)}$	$\mathbf{\Phi}^2_{1(1,1)}$	$\mathbf{\Phi}^2_{1(1,2)}$	$\mathbf{\Phi}^2_{2(1,1)}$	$\mathbf{\Phi}^2_{2(1,2)}$	.
6	2	AIC_2	$\mathbf{\Sigma}_{2(2,1)}$	$\mathbf{\Sigma}_{2(1,2)}$	$\mathbf{\Omega}_{2(2,1)}$	$\mathbf{\Omega}_{2(1,2)}$	$\mathbf{\Phi}^2_{1(2,1)}$	$\mathbf{\Phi}^2_{1(2,2)}$	$\mathbf{\Phi}^2_{2(2,1)}$	$\mathbf{\Phi}^2_{2(2,2)}$	.
7	3	AIC_3	$\mathbf{\Sigma}_{3(1,1)}$	$\mathbf{\Sigma}_{3(1,2)}$	$\mathbf{\Omega}_{3(1,1)}$	$\mathbf{\Omega}_{3(1,2)}$	$\mathbf{\Phi}^3_{1(1,1)}$	$\mathbf{\Phi}^3_{1(1,2)}$	$\mathbf{\Phi}^3_{2(1,1)}$	$\mathbf{\Phi}^3_{2(1,2)}$	$\mathbf{\Phi}^3_{3(1,1)}$
8	3	AIC_3	$\mathbf{\Sigma}_{3(2,1)}$	$\mathbf{\Sigma}_{3(1,2)}$	$\mathbf{\Omega}_{3(2,1)}$	$\mathbf{\Omega}_{3(1,2)}$	$\mathbf{\Phi}^3_{1(2,1)}$	$\mathbf{\Phi}^3_{1(2,2)}$	$\mathbf{\Phi}^3_{2(2,1)}$	$\mathbf{\Phi}^3_{2(2,2)}$	$\mathbf{\Phi}^3_{3(2,1)}$

Obs	FOR3_2	BACK1_1	BACK1_2	BACK2_1	BACK2_2	BACK3_1	BACK3_2
1	.	.	.	.	.	.	.
2	.	.	.	.	.	.	.
3	.	$\mathbf{\Psi}^1_{1(1,1)}$	$\mathbf{\Psi}^1_{1(1,2)}$	.	.	.	.
4	.	$\mathbf{\Psi}^1_{1(2,1)}$	$\mathbf{\Psi}^1_{1(2,2)}$	.	.	.	.
5	.	$\mathbf{\Psi}^2_{1(1,1)}$	$\mathbf{\Psi}^2_{1(1,2)}$	$\mathbf{\Psi}^2_{2(1,1)}$	$\mathbf{\Psi}^2_{2(1,2)}$	.	.
6	.	$\mathbf{\Psi}^2_{1(2,1)}$	$\mathbf{\Psi}^2_{1(2,2)}$	$\mathbf{\Psi}^2_{2(2,1)}$	$\mathbf{\Psi}^2_{2(2,2)}$	.	.
7	$\mathbf{\Phi}^3_{3(1,2)}$	$\mathbf{\Psi}^3_{1(1,1)}$	$\mathbf{\Psi}^3_{1(1,2)}$	$\mathbf{\Psi}^3_{2(1,1)}$	$\mathbf{\Psi}^3_{2(1,2)}$	$\mathbf{\Psi}^3_{3(1,1)}$	$\mathbf{\Psi}^3_{3(1,2)}$
8	$\mathbf{\Phi}^3_{3(2,2)}$	$\mathbf{\Psi}^3_{1(2,1)}$	$\mathbf{\Psi}^3_{1(2,2)}$	$\mathbf{\Psi}^3_{2(2,1)}$	$\mathbf{\Psi}^3_{2(2,2)}$	$\mathbf{\Psi}^3_{3(2,1)}$	$\mathbf{\Psi}^3_{3(2,2)}$

The estimated autoregressive parameters can be used in the IML procedure to obtain autoregressive estimates of the spectral density function or forecasts based on the autoregressive models.

OUTMODEL= Data Set

The OUTMODEL= data set contains the estimates of the $\mathbf{F}$ and $\mathbf{G}$ matrices and their standard errors, the names of the components of the state vector, and the estimates of the innovation covariance matrix. The variables contained in the OUTMODEL= data set are as follows:

- the BY variables
- STATEVEC, a character variable that contains the name of the component of the state vector corresponding to the observation. The STATEVEC variable has the value STD for standard deviations observations, which contain the standard errors for the estimates given in the preceding observation.
- F_j, numeric variables that contain the columns of the $\mathbf{F}$ matrix. The variable F_j contains the jth column of $\mathbf{F}$. The number of F_j variables is equal to the value of the DIMMAX= option. If the model is of smaller dimension, the extraneous variables are set to missing.
- G_j, numeric variables that contain the columns of the $\mathbf{G}$ matrix. The variable G_j contains the jth column of $\mathbf{G}$. The number of G_j variables is equal to r, the dimension of $\mathbf{x}_t$ given by the number of variables in the VAR statement.
- SIG_j, numeric variables that contain the columns of the innovation covariance matrix. The variable SIG_j contains the jth column of $\mathbf{\Sigma}_{ee}$. There are r variables SIG_j.

Table 35.3 shows an example of the OUTMODEL= data set, with $\mathbf{x}_t = (x_t, y_t)'$, $\mathbf{z}_t = (x_t, y_t, x_{t+1|t})'$, and DIMMAX=4. In Table 35.3, $\mathbf{F}_{i,j}$ and $\mathbf{G}_{i,j}$ are the (i, j) elements of $\mathbf{F}$ and $\mathbf{G}$ respectively. Note that all elements for F_4 are missing because $\mathbf{F}$ is a 3×3 matrix.

Table 35.3 Value in the OUTMODEL= Data Set

Obs	STATEVEC	F_1	F_2	F_3	F_4	G_1	G_2	SIG_1	SIG_2
1	X(T;T)	0	0	1	.	1	0	$\mathbf{\Sigma}_{1,1}$	$\mathbf{\Sigma}_{1,2}$
2	STD	.	.	.	.	.	.	.	.
3	Y(T;T)	$\mathbf{F}_{2,1}$	$\mathbf{F}_{2,2}$	$\mathbf{F}_{2,3}$	.	0	1	$\mathbf{\Sigma}_{2,1}$	$\mathbf{\Sigma}_{2,2}$
4	STD	std $\mathbf{F}_{2,1}$	std $\mathbf{F}_{2,2}$	std $\mathbf{F}_{2,3}$	.	.	.	.	.
5	X(T+1;T)	$\mathbf{F}_{3,1}$	$\mathbf{F}_{3,2}$	$\mathbf{F}_{3,3}$	.	$\mathbf{G}_{3,1}$	$\mathbf{G}_{3,2}$	.	.
6	STD	std $\mathbf{F}_{3,1}$	std $\mathbf{F}_{3,2}$	std $\mathbf{F}_{3,3}$	.	std $\mathbf{G}_{3,1}$	std $\mathbf{G}_{3,2}$	.	.

Printed Output

The printed output produced by the STATESPACE procedure includes the following:

1. descriptive statistics, which include the number of observations used, the names of the variables, their means and standard deviations (Std), and the differencing operations used
2. the Akaike information criteria for the sequence of preliminary autoregressive models
3. if the PRINTOUT=LONG option is specified, the sample autocovariance matrices of the input series at various lags
4. if the PRINTOUT=LONG option is specified, the sample autocorrelation matrices of the input series
5. a schematic representation of the autocorrelation matrices, showing the significant autocorrelations
6. if the PRINTOUT=LONG option is specified, the partial autoregressive matrices. (These are $\mathbf{\Phi}_p^p$ as described in the section "Preliminary Autoregressive Models" on page 2524.)
7. a schematic representation of the partial autocorrelation matrices, showing the significant partial autocorrelations
8. the Yule-Walker estimates of the autoregressive parameters for the autoregressive model with the minimum AIC
9. if the PRINTOUT=LONG option is specified, the autocovariance matrices of the residuals of the minimum AIC model. This is the sequence of estimated innovation variance matrices for the solutions of the Yule-Walker equations.
10. if the PRINTOUT=LONG option is specified, the autocorrelation matrices of the residuals of the minimum AIC model
11. If the CANCORR option is specified, the canonical correlations analysis for each potential state vector considered in the state vector selection process. This includes the potential state vector, the canonical correlations, the information criterion for the smallest canonical correlation, Bartlett's χ^2 statistic ("Chi Square") for the smallest canonical correlation, and the degrees of freedom of Bartlett's χ^2.

12. the components of the chosen state vector
13. the preliminary estimate of the transition matrix, **F**, the input matrix, **G**, and the variance matrix for the innovations, $\mathbf{\Sigma_{ee}}$
14. if the ITPRINT option is specified, the iteration history of the likelihood maximization. For each iteration, this shows the iteration number, the number of step halvings, the determinant of the innovation variance matrix, the damping factor Lambda, and the values of the parameters.
15. the state vector, printed again to aid interpretation of the following listing of **F** and **G**
16. the final estimate of the transition matrix **F**
17. the final estimate of the input matrix **G**
18. the final estimate of the variance matrix for the innovations $\mathbf{\Sigma_{ee}}$
19. a table that lists the estimates of the free parameters in **F** and **G** and their standard errors and *t* statistics
20. if the COVB option is specified, the covariance matrix of the parameter estimates
21. if the COVB option is specified, the correlation matrix of the parameter estimates
22. if the PRINT option is specified, the forecasts and their standard errors

ODS Table Names

PROC STATESPACE assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in the following table.

Table 35.4 ODS Tables Produced in PROC STATESPACE

ODS Table Name	Description	Option
NObs	number of observations	default
Summary	simple summary statistics table	default
InfoCriterion	information criterion table	default
CovLags	covariance matrices of input series	PRINTOUT=LONG
CorrLags	correlation matrices of input series	PRINTOUT=LONG
PartialAR	partial autoregressive matrices	PRINTOUT=LONG
YWEstimates	Yule-Walker estimates for minimum AIC	default
CovResiduals	covariance of residuals	PRINTOUT=LONG
CorrResiduals	residual correlations from AR models	PRINTOUT=LONG
StateVector	state vector table	default
CorrGraph	schematic representation of correlations	default
TransitionMatrix	transition matrix	default
InputMatrix	input matrix	default
VarInnov	variance matrix for the innovation	default

Table 35.4 *continued*

ODS Table Name	Description	Option
CovB	covariance of parameter estimates	COVB
CorrB	correlation of parameter estimates	COVB
CanCorr	canonical correlation analysis	CANCORR
IterHistory	iterative fitting table	ITPRINT
ParameterEstimates	parameter estimates table	default
Forecasts	forecasts table	PRINT
ConvergenceStatus	convergence status table	default

Examples: STATESPACE Procedure

Example 35.1: Series J from Box and Jenkins

This example analyzes the gas furnace data (series J) from Box and Jenkins. (The data are not shown; see Box and Jenkins (1976) for the data.)

First, a model is selected and fit automatically using the following statements.

```
title1 'Gas Furnace Data';
title2 'Box & Jenkins Series J';
title3 'Automatically Selected Model';

proc statespace data=seriesj cancorr;
   var x y;
run;
```

The results for the automatically selected model are shown in Output 35.1.1.

Output 35.1.1 Results for Automatically Selected Model

Gas Furnace Data
Box & Jenkins Series J
Automatically Selected Model

The STATESPACE Procedure

Number of Observations	296

Variable	Mean	Standard Error
x	-0.05683	1.072766
y	53.50912	3.202121

Output 35.1.1 *continued*

Gas Furnace Data
Box & Jenkins Series J
Automatically Selected Model

The STATESPACE Procedure

Information Criterion for Autoregressive Models										
Lag=0	Lag=1	Lag=2	Lag=3	Lag=4	Lag=5	Lag=6	Lag=7	Lag=8	Lag=9	Lag=10
651.3862	-1033.57	-1632.96	-1645.12	-1651.52	-1648.91	-1649.34	-1643.15	-1638.56	-1634.8	-1633.59

Schematic Representation of Correlations											
Name/Lag	0	1	2	3	4	5	6	7	8	9	10
x	+−	+−	+−	+−	+−	+−	+−	+−	+−	+−	+−
y	−+	−+	−+	−+	−+	−+	−+	−+	−+	−+	−+
+ is > 2*std error, - is < -2*std error, . is between											

Output 35.1.2 Results for Automatically Selected Model

Schematic Representation of Partial Autocorrelations										
Name/Lag	1	2	3	4	5	6	7	8	9	10
x	+.	−.	+.	..	..	−.	..	..	..	..
y	−+	−−	−.	.+	..	..	..	..	..	.+
+ is > 2*std error, - is < -2*std error, . is between										

Yule-Walker Estimates for Minimum AIC								
	Lag=1		Lag=2		Lag=3		Lag=4	
	x	y	x	y	x	y	x	y
x	1.925887	-0.00124	-1.20166	0.004224	0.116918	-0.00867	0.104236	0.003268
y	0.050496	1.299793	-0.02046	-0.3277	-0.71182	-0.25701	0.195411	0.133417

Output 35.1.3 Results for Automatically Selected Model

Gas Furnace Data
Box & Jenkins Series J
Automatically Selected Model

The STATESPACE Procedure
Canonical Correlations Analysis

x(T;T)	y(T;T)	x(T+1;T)	Information Criterion	Chi-Square	DF
1	1	0.804883	292.9228	304.7481	8

Output 35.1.4 Results for Automatically Selected Model

Gas Furnace Data
Box & Jenkins Series J
Automatically Selected Model

The STATESPACE Procedure
Selected Statespace Form and Preliminary Estimates

State Vector				
x(T;T)	y(T;T)	x(T+1;T)	y(T+1;T)	y(T+2;T)

Estimate of Transition Matrix				
0	0	1	0	0
0	0	0	1	0
-0.84718	0.026794	1.711715	-0.05019	0
0	0	0	0	1
-0.19785	0.334274	-0.18174	-1.23557	1.787475

Input Matrix for Innovation	
1	0
0	1
1.925887	-0.00124
0.050496	1.299793
0.142421	1.361696

Output 35.1.5 Results for Automatically Selected Model

Variance Matrix for Innovation	
0.035274	-0.00734
-0.00734	0.097569

Output 35.1.6 Results for Automatically Selected Model

Gas Furnace Data
Box & Jenkins Series J
Automatically Selected Model

The STATESPACE Procedure
Selected Statespace Form and Fitted Model

State Vector				
x(T;T)	y(T;T)	x(T+1;T)	y(T+1;T)	y(T+2;T)

Output 35.1.6 *continued*

Estimate of Transition Matrix				
0	0	1	0	0
0	0	0	1	0
-0.86192	0.030609	1.724235	-0.05483	0
0	0	0	0	1
-0.34839	0.292124	-0.09435	-1.09823	1.671418

Input Matrix for Innovation	
1	0
0	1
1.92442	-0.00416
0.015621	1.258495
0.08058	1.353204

Output 35.1.7 Results for Automatically Selected Model

Variance Matrix for Innovation	
0.035579	-0.00728
-0.00728	0.095577

Parameter Estimates			
Parameter	Estimate	Standard Error	t Value
F(3,1)	-0.86192	0.072961	-11.81
F(3,2)	0.030609	0.026167	1.17
F(3,3)	1.724235	0.061599	27.99
F(3,4)	-0.05483	0.030169	-1.82
F(5,1)	-0.34839	0.135253	-2.58
F(5,2)	0.292124	0.046299	6.31
F(5,3)	-0.09435	0.096527	-0.98
F(5,4)	-1.09823	0.109525	-10.03
F(5,5)	1.671418	0.083737	19.96
G(3,1)	1.924420	0.058162	33.09
G(3,2)	-0.00416	0.035255	-0.12
G(4,1)	0.015621	0.095771	0.16
G(4,2)	1.258495	0.055742	22.58
G(5,1)	0.080580	0.151622	0.53
G(5,2)	1.353204	0.091388	14.81

The two series are believed to have a transfer function relation with the gas rate (variable X) as the input and the CO_2 concentration (variable Y) as the output. Since the parameter estimates shown in Output 35.1.1 support this kind of model, the model is reestimated with the feedback parameters restricted to 0. The following statements fit the transfer function (no feedback) model.

```
title3 'Transfer Function Model';
proc statespace data=seriesj printout=none;
   var x y;
   restrict f(3,2)=0 f(3,4)=0
            g(3,2)=0 g(4,1)=0 g(5,1)=0;
run;
```

The last two pages of the output are shown in Output 35.1.8.

Output 35.1.8 STATESPACE Output for Transfer Function Model

Gas Furnace Data
Box & Jenkins Series J
Transfer Function Model

The STATESPACE Procedure
Selected Statespace Form and Fitted Model

State Vector				
x(T;T)	y(T;T)	x(T+1;T)	y(T+1;T)	y(T+2;T)

Estimate of Transition Matrix				
0	0	1	0	0
0	0	0	1	0
-0.68882	0	1.598717	0	0
0	0	0	0	1
-0.35944	0.284179	-0.0963	-1.07313	1.650047

Input Matrix for Innovation	
1	0
0	1
1.923446	0
0	1.260856
0	1.346332

Output 35.1.9 STATESPACE Output for Transfer Function Model

Variance Matrix for Innovation	
0.036995	-0.0072
-0.0072	0.095712

Parameter Estimates

Parameter	Estimate	Standard Error	t Value
F(3,1)	-0.68882	0.050549	-13.63
F(3,3)	1.598717	0.050924	31.39
F(5,1)	-0.35944	0.229044	-1.57
F(5,2)	0.284179	0.096944	2.93
F(5,3)	-0.09630	0.140876	-0.68
F(5,4)	-1.07313	0.250385	-4.29
F(5,5)	1.650047	0.188533	8.75
G(3,1)	1.923446	0.056328	34.15
G(4,2)	1.260856	0.056464	22.33
G(5,2)	1.346332	0.091086	14.78

References

Akaike, H. (1974). "Markovian Representation of Stochastic Processes and Its Application to the Analysis of Autoregressive Moving Average Processes." *Annals of the Institute of Statistical Mathematics* 26:363–387.

Akaike, H. (1976). "Canonical Correlations Analysis of Time Series and the Use of an Information Criterion." In *System Identification: Advances and Case Studies*, edited by R. Mehra, and D. G. Lainiotis, 27–96. New York: Academic Press.

Anderson, T. W. (1971). *The Statistical Analysis of Time Series*. New York: John Wiley & Sons.

Ansley, C. F., and Newbold, P. (1979). "Multivariate Partial Autocorrelations." In *Proceedings of the Business and Economic Statistics Section*, 349–353. Washington, DC: American Statistical Association.

Box, G. E. P., and Jenkins, G. M. (1976). *Time Series Analysis: Forecasting and Control*. Rev. ed. San Francisco: Holden-Day.

Brockwell, P. J., and Davis, R. A. (1991). *Time Series: Theory and Methods*. 2nd ed. New York: Springer-Verlag.

Hannan, E. J. (1970). *Multiple Time Series*. New York: John Wiley & Sons.

Hannan, E. J. (1976). "The Identification and Parameterization of ARMAX and State Space Forms." *Econometrica* 44:713–722.

Harvey, A. C. (1981a). *The Econometric Analysis of Time Series*. New York: John Wiley & Sons.

Harvey, A. C. (1981b). *Time Series Models*. New York: John Wiley & Sons.

Jones, R. H. (1974). "Identification and Autoregressive Spectrum Estimation." *IEEE Transactions on Automatic Control* 19:894–898.

Pham, D.-T. (1978). "On the Fitting of Multivariate Processes of the Autoregressive Moving Average Type." *Biometrika* 65:99–107.

Priestley, M. B. (1980). "System Identification, Kalman Filtering, and Stochastic Control." In *Directions in Time Series*, edited by D. R. Brillinger, and G. C. Tiao, 228–254. Bethesda, MD: Institute of Mathematical Statistics.

Whittle, P. (1963). "On the Fitting of Multivariate Autoregressions and the Approximate Canonical Factorization of a Spectral Density Matrix." *Biometrika* 50:129–134.

Chapter 36
The SYSLIN Procedure

Contents

Overview: SYSLIN Procedure **2548**
Getting Started: SYSLIN Procedure **2549**
An Example Model 2549
Variables in a System of Equations 2550
Using PROC SYSLIN 2550
OLS Estimation 2551
Two-Stage Least Squares Estimation 2553
LIML, K-Class, and MELO Estimation 2554
SUR, 3SLS, and FIML Estimation 2555
Computing Reduced Form Estimates 2558
Restricting Parameter Estimates 2559
Testing Parameters 2561
Saving Residuals and Predicted Values 2563
Plotting Residuals 2563
Syntax: SYSLIN Procedure **2565**
Functional Summary 2565
PROC SYSLIN Statement 2567
BY Statement 2569
ENDOGENOUS Statement 2570
IDENTITY Statement 2570
INSTRUMENTS Statement 2570
MODEL Statement 2570
OUTPUT Statement 2572
RESTRICT Statement 2573
SRESTRICT Statement 2574
STEST Statement 2575
TEST Statement 2576
VAR Statement 2577
WEIGHT Statement 2578
Details: SYSLIN Procedure **2578**
Input Data Set 2578
Estimation Methods 2578
ANOVA Table for Instrumental Variables Methods 2581
The R-Square Statistics 2581
Computational Details 2582
Missing Values 2585

OUT= Data Set 2585
OUTEST= Data Set 2585
OUTSSCP= Data Set 2586
Printed Output 2587
ODS Table Names 2589
ODS Graphics 2590
Examples: SYSLIN Procedure **2590**
Example 36.1: Klein's Model I Estimated with LIML and 3SLS 2590
Example 36.2: Grunfeld's Model Estimated with SUR 2596
Example 36.3: Illustration of ODS Graphics 2599
References **2603**

Overview: SYSLIN Procedure

The SYSLIN procedure estimates parameters in an interdependent system of linear regression equations.

Ordinary least squares (OLS) estimates are biased and inconsistent when current period endogenous variables appear as regressors in other equations in the system. The errors of a set of related regression equations are often correlated, and the efficiency of the estimates can be improved by taking these correlations into account. The SYSLIN procedure provides several techniques that produce consistent and asymptotically efficient estimates for systems of regression equations.

The SYSLIN procedure provides the following estimation methods:

- ordinary least squares (OLS)
- two-stage least squares (2SLS)
- limited information maximum likelihood (LIML)
- K-class
- seemingly unrelated regressions (SUR)
- iterated seemingly unrelated regressions (ITSUR)
- three-stage least squares (3SLS)
- iterated three-stage least squares (IT3SLS)
- full information maximum likelihood (FIML)
- minimum expected loss (MELO)

Other features of the SYSLIN procedure enable you to:

- impose linear restrictions on the parameter estimates

- test linear hypotheses about the parameters
- write predicted and residual values to an output SAS data set
- write parameter estimates to an output SAS data set
- write the crossproducts matrix (SSCP) to an output SAS data set
- use raw data, correlations, covariances, or cross products as input

Getting Started: SYSLIN Procedure

This section introduces the use of the SYSLIN procedure. The problem of dependent regressors is introduced using a supply and demand example. This section explains the terminology used for variables in a system of regression equations and introduces the SYSLIN procedure statements for declaring the roles the variables play. The syntax used for the different estimation methods and the output produced is shown.

An Example Model

In simultaneous systems of equations, endogenous variables are determined jointly rather than sequentially. Consider the following supply and demand functions for some product:

$$Q_D = a_1 + b_1 P + c_1 Y + d_1 S + \epsilon_1 (\text{demand})$$

$$Q_S = a_2 + b_2 P + c_2 U + \epsilon_2 (\text{supply})$$

$$Q = Q_D = Q_S (\text{market equilibrium})$$

The variables in this system are as follows:

Q_D	quantity demanded
Q_S	quantity supplied
Q	the observed quantity sold, which equates quantity supplied and quantity demanded in equilibrium
P	price per unit
Y	income
S	price of substitutes
U	unit cost
ϵ_1	the random error term for the demand equation
ϵ_2	the random error term for the supply equation

In this system, quantity demanded depends on price, income, and the price of substitutes. Consumers normally purchase more of a product when prices are lower and when income and the price of substitute goods are higher. Quantity supplied depends on price and the unit cost of production. Producers supply more when price is high and when unit cost is low. The actual price and quantity sold are determined jointly by the values that equate demand and supply.

Since price and quantity are jointly endogenous variables, both structural equations are necessary to adequately describe the observed values. A critical assumption of OLS is that the regressors are uncorrelated with the residual. When current endogenous variables appear as regressors in other equations (endogenous variables depend on each other), this assumption is violated and the OLS parameter estimates are biased and inconsistent. The bias caused by the violated assumptions is called *simultaneous equation bias.* Neither the demand nor supply equation can be estimated consistently by OLS.

Variables in a System of Equations

Before explaining how to use the SYSLIN procedure, it is useful to define some terms. The variables in a system of equations can be classified as follows:

- *Endogenous variables,* which are also called *jointly dependent* or *response variables,* are the variables determined by the system. Endogenous variables can also appear on the right-hand side of equations.
- *Exogenous variables* are independent variables that do not depend on any of the endogenous variables in the system.
- *Predetermined variables* include both the exogenous variables and *lagged endogenous variables,* which are past values of endogenous variables determined at previous time periods. PROC SYSLIN does not compute lagged values; any lagged endogenous variables must be computed in a preceding DATA step.
- *Instrumental variables* are predetermined variables used in obtaining predicted values for the current period endogenous variables by a first-stage regression. The use of instrumental variables characterizes estimation methods such as two-stage least squares and three-stage least squares. Instrumental variables estimation methods substitute these first-stage predicted values for endogenous variables when they appear as regressors in model equations.

Using PROC SYSLIN

First specify the input data set and estimation method in the PROC SYSLIN statement. If any model uses dependent regressors, and you are using an instrumental variables regression method, declare the dependent regressors with an ENDOGENOUS statement and declare the instruments with an INSTRUMENTS statement. Next, use MODEL statements to specify the structural equations of the system.

The use of different estimation methods is shown by the following examples. These examples use the simulated dataset WORK.IN given below.

```
data in;
   label q = "Quantity"
         p = "Price"
         s = "Price of Substitutes"
         y = "Income"
         u = "Unit Cost";
   drop i e1 e2;
   p = 0; q = 0;
   do i = 1 to 60;
      y = 1 + .05*i  + .15*rannor(123);
      u = 2          + .05*rannor(123) + .05*rannor(123);
      s = 4 - .001*(i-10)*(i-110) + .5*rannor(123);
      e1 = .15 * rannor(123);
      e2 = .15 * rannor(123);
      demandx = 1 + .3 * y + .35 * s + e1;
      supplyx = -1 - 1 * u + e2 - .4*e1;
      q = 1.4/2.15 * demandx + .75/2.15 * supplyx;
      p = ( - q + supplyx ) / -1.4;
      output;
   end;
run;
```

OLS Estimation

PROC SYSLIN performs OLS regression if you do not specify a method of estimation in the PROC SYSLIN statement. OLS does not use instruments, so the ENDOGENOUS and INSTRUMENTS statements can be omitted.

The following statements estimate the supply and demand model shown previously:

```
proc syslin data=in;
   demand: model q = p y s;
   supply: model q = p u;
run;
```

The PROC SYSLIN output for the demand equation is shown in Figure 36.1, and the output for the supply equation is shown in Figure 36.2.

Figure 36.1 OLS Results for Demand Equation

The SYSLIN Procedure
Ordinary Least Squares Estimation

Model	DEMAND
Dependent Variable	q
Label	Quantity

Figure 36.1 *continued*

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	3	9.587901	3.195967	398.31	<.0001
Error	56	0.449338	0.008024		
Corrected Total	59	10.03724			

Root MSE	0.08958	R-Square	0.95523
Dependent Mean	1.30095	Adj R-Sq	0.95283
Coeff Var	6.88542		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	Variable Label
Intercept	1	-0.47677	0.210239	-2.27	0.0272	Intercept
p	1	0.123326	0.105177	1.17	0.2459	Price
y	1	0.201282	0.032403	6.21	<.0001	Income
s	1	0.167258	0.024091	6.94	<.0001	Price of Substitutes

Figure 36.2 OLS Results for Supply Equation

The SYSLIN Procedure
Ordinary Least Squares Estimation

Model	SUPPLY
Dependent Variable	q
Label	Quantity

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	2	9.033902	4.516951	256.61	<.0001
Error	57	1.003337	0.017602		
Corrected Total	59	10.03724			

Root MSE	0.13267	R-Square	0.90004
Dependent Mean	1.30095	Adj R-Sq	0.89653
Coeff Var	10.19821		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	Variable Label
Intercept	1	-0.30389	0.471397	-0.64	0.5217	Intercept
p	1	1.218743	0.053914	22.61	<.0001	Price
u	1	-1.07757	0.234150	-4.60	<.0001	Unit Cost

For each MODEL statement, the output first shows the model label and dependent variable name and label. This is followed by an analysis-of-variance table for the model, which shows the model, error, and total mean squares, and an F test for the no-regression hypothesis. Next, the procedure prints the root mean squared error, dependent variable mean and coefficient of variation, and the R^2 and adjusted R^2 statistics.

Finally, the table of parameter estimates shows the estimated regression coefficients, standard errors, and *t* tests. You would expect the price coefficient in a demand equation to be negative. However, note that the OLS estimate of the price coefficient P in the demand equation (0.1233) has a positive sign. This could be caused by simultaneous equation bias.

Two-Stage Least Squares Estimation

In the supply and demand model, P is an endogenous variable, and consequently the OLS estimates are biased. The following example estimates this model using two-stage least squares.

```
proc syslin data=in 2sls;
   endogenous  p;
   instruments y u s;
   demand: model q = p y s;
   supply: model q = p u;
run;
```

The 2SLS option in the PROC SYSLIN statement specifies the two-stage least squares method. The ENDOGENOUS statement specifies that P is an endogenous regressor for which first-stage predicted values are substituted. You need to declare an endogenous variable in the ENDOGENOUS statement only if it is used as a regressor; thus although Q is endogenous in this model, it is not necessary to list it in the ENDOGENOUS statement.

Usually, all predetermined variables that appear in the system are used as instruments. The INSTRUMENTS statement specifies that the exogenous variables Y, U, and S are used as instruments for the first-stage regression to predict P.

The 2SLS results are shown in Figure 36.3 and Figure 36.4. The first-stage regressions are not shown. To see the first-stage regression results, use the FIRST option in the PROC SYSLIN statement.

Figure 36.3 2SLS Results for Demand Equation

The SYSLIN Procedure
Two-Stage Least Squares Estimation

Model	DEMAND
Dependent Variable	q
Label	Quantity

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	3	9.670892	3.223631	115.58	<.0001
Error	56	1.561956	0.027892		
Corrected Total	59	10.03724			

Root MSE	0.16701	**R-Square**	0.86095
Dependent Mean	1.30095	**Adj R-Sq**	0.85350
Coeff Var	12.83744		

Figure 36.3 *continued*

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	Variable Label
Intercept	1	1.901048	1.171231	1.62	0.1102	Intercept
p	1	-1.11519	0.607395	-1.84	0.0717	Price
y	1	0.419546	0.117955	3.56	0.0008	Income
s	1	0.331475	0.088472	3.75	0.0004	Price of Substitutes

Figure 36.4 2SLS Results for Supply Equation

The SYSLIN Procedure
Two-Stage Least Squares Estimation

Model	SUPPLY
Dependent Variable	q
Label	Quantity

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	2	9.646109	4.823054	253.96	<.0001
Error	57	1.082503	0.018991		
Corrected Total	59	10.03724			

Root MSE	0.13781	**R-Square**	0.89910
Dependent Mean	1.30095	**Adj R-Sq**	0.89556
Coeff Var	10.59291		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	Variable Label
Intercept	1	-0.51878	0.490999	-1.06	0.2952	Intercept
p	1	1.333080	0.059271	22.49	<.0001	Price
u	1	-1.14623	0.243491	-4.71	<.0001	Unit Cost

The 2SLS output is similar in form to the OLS output. However, the 2SLS results are based on predicted values for the endogenous regressors from the first stage instrumental regressions. This makes the analysis-of-variance table and the R^2 statistics difficult to interpret. See the sections "ANOVA Table for Instrumental Variables Methods" on page 2581 and "The R-Square Statistics" on page 2581 for details.

Note that, unlike the OLS results, the 2SLS estimate for the P coefficient in the demand equation (–1.115) is negative.

LIML, K-Class, and MELO Estimation

To obtain limited information maximum likelihood, general K-class, or minimum expected loss estimates, use the ENDOGENOUS, INSTRUMENTS, and MODEL statements as in the 2SLS case but specify the

LIML, K=, or MELO option instead of 2SLS in the PROC SYSLIN statement. The following statements show this for K-class estimation.

```
proc syslin data=in k=.5;
   endogenous  p;
   instruments y u s;
   demand: model q = p y s;
   supply: model q = p u;
run;
```

For more information about these estimation methods, see the section "Estimation Methods" on page 2578 and consult econometrics textbooks.

SUR, 3SLS, and FIML Estimation

In a multivariate regression model, the errors in different equations might be correlated. In this case, the efficiency of the estimation might be improved by taking these cross-equation correlations into account.

Seemingly Unrelated Regression

Seemingly unrelated regression (SUR), also called joint generalized least squares (JGLS) or Zellner estimation, is a generalization of OLS for multi-equation systems. Like OLS, the SUR method assumes that all the regressors are independent variables, but SUR uses the correlations among the errors in different equations to improve the regression estimates. The SUR method requires an initial OLS regression to compute residuals. The OLS residuals are used to estimate the cross-equation covariance matrix.

The SUR option in the PROC SYSLIN statement specifies seemingly unrelated regression, as shown in the following statements:

```
proc syslin data=in sur;
   demand: model q = p y s;
   supply: model q = p u;
run;
```

INSTRUMENTS and ENDOGENOUS statements are not needed for SUR, because the SUR method assumes there are no endogenous regressors. For SUR to be effective, the models must use different regressors. SUR produces the same results as OLS unless the model contains at least one regressor not used in the other equations.

Three-Stage Least Squares

The three-stage least squares method generalizes the two-stage least squares method to take into account the correlations between equations in the same way that SUR generalizes OLS. Three-stage least squares requires three steps: first-stage regressions to get predicted values for the endogenous regressors; a two-stage least squares step to get residuals to estimate the cross-equation correlation matrix; and the final 3SLS estimation step.

The 3SLS option in the PROC SYSLIN statement specifies the three-stage least squares method, as shown in the following statements.

```
proc syslin data=in 3sls;
   endogenous  p;
   instruments y u s;
   demand: model q = p y s;
   supply: model q = p u;
run;
```

The 3SLS output begins with a two-stage least squares regression to estimate the cross-model correlation matrix. This output is the same as the 2SLS results shown in Figure 36.3 and Figure 36.4, and is not repeated here. The next part of the 3SLS output prints the cross-model correlation matrix computed from the 2SLS residuals. This output is shown in Figure 36.5 and includes the cross-model covariances, correlations, the inverse of the correlation matrix, and the inverse covariance matrix.

Figure 36.5 Estimated Cross-Model Covariances Used for 3SLS Estimates

The SYSLIN Procedure
Three-Stage Least Squares Estimation

Cross Model Covariance

	DEMAND	SUPPLY
DEMAND	0.027892	-.011283
SUPPLY	-.011283	0.018991

Cross Model Correlation

	DEMAND	SUPPLY
DEMAND	1.00000	-0.49022
SUPPLY	-0.49022	1.00000

Cross Model Inverse Correlation

	DEMAND	SUPPLY
DEMAND	1.31634	0.64530
SUPPLY	0.64530	1.31634

Cross Model Inverse Covariance

	DEMAND	SUPPLY
DEMAND	47.1941	28.0379
SUPPLY	28.0379	69.3130

The final 3SLS estimates are shown in Figure 36.6.

Figure 36.6 Three-Stage Least Squares Results

System Weighted MSE	0.5711
Degrees of freedom	113
System Weighted R-Square	0.9627

Model	DEMAND
Dependent Variable	q
Label	Quantity

Figure 36.6 *continued*

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	Variable Label
Intercept	1	1.980269	1.169176	1.69	0.0959	Intercept
p	1	-1.17654	0.605015	-1.94	0.0568	Price
y	1	0.404117	0.117179	3.45	0.0011	Income
s	1	0.359204	0.085077	4.22	<.0001	Price of Substitutes

Model	SUPPLY
Dependent Variable	q
Label	Quantity

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	Variable Label
Intercept	1	-0.51878	0.490999	-1.06	0.2952	Intercept
p	1	1.333080	0.059271	22.49	<.0001	Price
u	1	-1.14623	0.243491	-4.71	<.0001	Unit Cost

This output first prints the system weighted mean squared error and system weighted R^2 statistics. The system weighted MSE and system weighted R^2 measure the fit of the joint model obtained by stacking all the models together and performing a single regression with the stacked observations weighted by the inverse of the model error variances. See the section "The R-Square Statistics" on page 2581 for details.

Next, the table of 3SLS parameter estimates for each model is printed. This output has the same form as for the other estimation methods.

Note that, in some cases, the 3SLS and 2SLS results can be the same. Such a case could arise because of the same principle that causes OLS and SUR results to be identical, unless an equation includes a regressor not used in the other equations of the system. However, the application of this principle is more complex when instrumental variables are used. When all the exogenous variables are used as instruments, linear combinations of all the exogenous variables appear in the third-stage regressions through substitution of first-stage predicted values.

In this example, 3SLS produces different (and, it is hoped, more efficient) estimates for the demand equation. However, the 3SLS and 2SLS results for the supply equation are the same. This is because the supply equation has one endogenous regressor and one exogenous regressor not used in other equations. In contrast, the demand equation has fewer endogenous regressors than exogenous regressors not used in other equations in the system.

Full Information Maximum Likelihood

The FIML option in the PROC SYSLIN statement specifies the full information maximum likelihood method, as shown in the following statements.

```
proc syslin data=in fiml;
   endogenous  p q;
   instruments y u s;
   demand: model q = p y s;
   supply: model q = p u;
run;
```

The FIML results are shown in Figure 36.7.

Figure 36.7 FIML Results

The SYSLIN Procedure
Full-Information Maximum Likelihood Estimation

NOTE: Convergence criterion met at iteration 3.

Model	DEMAND
Dependent Variable	q
Label	Quantity

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	Variable Label
Intercept	1	1.988538	1.233632	1.61	0.1126	Intercept
p	1	-1.18148	0.652278	-1.81	0.0755	Price
y	1	0.402312	0.107270	3.75	0.0004	Income
s	1	0.361345	0.103817	3.48	0.0010	Price of Substitutes

Model	SUPPLY
Dependent Variable	q
Label	Quantity

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	Variable Label
Intercept	1	-0.52443	0.479522	-1.09	0.2787	Intercept
p	1	1.336083	0.057939	23.06	<.0001	Price
u	1	-1.14804	0.237793	-4.83	<.0001	Unit Cost

Computing Reduced Form Estimates

A system of structural equations with endogenous regressors can be represented as functions of only the predetermined variables. For this to be possible, there must be as many equations as endogenous variables. If there are more endogenous variables than regression models, you can use IDENTITY statements to complete the system. See the section "Reduced Form Estimates" on page 2583 for details.

The REDUCED option in the PROC SYSLIN statement prints reduced form estimates. The following statements show this by using the 3SLS estimates of the structural parameters.

```
proc syslin data=in 3sls reduced;
   endogenous  p;
   instruments y u s;
   demand: model q = p y s;
   supply: model q = p u;
run;
```

The first four pages of this output were as shown previously and are not repeated here. (See Figure 36.3, Figure 36.4, Figure 36.5, and Figure 36.6.) The final page of the output from this example contains the reduced form coefficients from the 3SLS structural estimates, as shown in Figure 36.8.

Figure 36.8 Reduced Form 3SLS Results

The SYSLIN Procedure
Three-Stage Least Squares Estimation

Endogenous Variables

	p	q
DEMAND	1.176543	1
SUPPLY	-1.33308	1

Exogenous Variables

	Intercept	y	s	u
DEMAND	1.980269	0.404117	0.359204	0
SUPPLY	-0.51878	0	0	-1.14623

Inverse Endogenous Variables

	DEMAND	SUPPLY
p	0.398466	-0.39847
q	0.531187	0.468813

Reduced Form

	Intercept	y	s	u
p	0.995788	0.161027	0.143131	0.456735
q	0.808682	0.214662	0.190804	-0.53737

Restricting Parameter Estimates

You can impose restrictions on the parameter estimates with RESTRICT and SRESTRICT statements. The RESTRICT statement imposes linear restrictions on parameters in the equation specified by the preceding MODEL statement. The SRESTRICT statement imposes linear restrictions that relate parameters in different models.

To impose restrictions involving parameters in different equations, use the SRESTRICT statement. Specify the parameters in the linear hypothesis as *model-label.regressor-name*. (If the MODEL statement does not have a label, you can use the dependent variable name as the label for the model, provided the dependent variable uniquely labels the model.)

Tests for the significance of the restrictions are printed when RESTRICT or SRESTRICT statements are used. You can label RESTRICT and SRESTRICT statements to identify the restrictions in the output.

The RESTRICT statement in the following example restricts the price coefficient in the demand equation to equal 0.015. The SRESTRICT statement restricts the estimate of the income coefficient in the demand equation to be 0.01 times the estimate of the unit cost coefficient in the supply equation.

```
proc syslin data=in 3sls;
   endogenous  p;
   instruments y u s;
   demand: model q = p y s;
   peq015: restrict p = .015;
   supply: model q = p u;
   yeq01u: srestrict demand.y = .01 * supply.u;
run;
```

The restricted estimation results are shown in Figure 36.9.

Figure 36.9 Restricted Estimates

The SYSLIN Procedure
Three-Stage Least Squares Estimation

Model	DEMAND
Dependent Variable	q
Label	Quantity

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	Variable Label
Intercept	1	-0.46584	0.053307	-8.74	<.0001	Intercept
p	1	0.015000	0	.	.	Price
y	1	-0.00679	0.002357	-2.88	0.0056	Income
s	1	0.325589	0.009872	32.98	<.0001	Price of Substitutes
RESTRICT	-1	50.59353	7.464988	6.78	<.0001	PEQ015

Model	SUPPLY
Dependent Variable	q
Label	Quantity

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	Variable Label
Intercept	1	-1.31894	0.477633	-2.76	0.0077	Intercept
p	1	1.291718	0.059101	21.86	<.0001	Price
u	1	-0.67887	0.235679	-2.88	0.0056	Unit Cost

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	Variable Label
RESTRICT	-1	342.3605	38.12094	8.98	<.0001	YEQ01U

The standard error for P in the demand equation is 0, since the value of the P coefficient was specified by the RESTRICT statement and not estimated from the data. The "Parameter Estimates" table for the demand equation contains an additional row for the restriction specified by the RESTRICT statement. The parameter estimate for the restriction is the value of the Lagrange multiplier used to impose the restriction. The restriction is highly significant ($t = 6.777$), which means that the data are not consistent with the restriction, and the model does not fit as well with the restriction imposed. See the section "RESTRICT Statement" on page 2573 for details.

Following the "Parameter Estimates" table for the supply equation, the results for the cross model restrictions are printed. This shows that the restriction specified by the SRESTRICT statement is not consistent with the data ($t = 8.98$). See the section "SRESTRICT Statement" on page 2574 for details.

Testing Parameters

You can test linear hypotheses about the model parameters with TEST and STEST statements. The TEST statement tests hypotheses about parameters in the equation specified by the preceding MODEL statement. The STEST statement tests hypotheses that relate parameters in different models.

For example, the following statements test the hypothesis that the price coefficient in the demand equation is equal to 0.015.

```
proc syslin data=in 3sls;
   endogenous  p;
   instruments y u s;
   demand: model q = p y s;
   test_1: test p = .015;
   supply: model q = p u;
run;
```

The TEST statement results are shown in Figure 36.10. This reports an F test for the hypothesis specified by the TEST statement. In this case, the F statistic is 6.79 (3.879/.571) with 1 and 113 degrees of freedom. The p value for this F statistic is 0.0104, which indicates that the hypothesis tested is almost but not quite rejected at the 0.01 level. See the section "TEST Statement" on page 2576 for details.

Figure 36.10 TEST Statement Results

The SYSLIN Procedure
Three-Stage Least Squares Estimation

System Weighted MSE	0.5711
Degrees of freedom	113
System Weighted R-Square	0.9627

Model	DEMAND
Dependent Variable	q
Label	Quantity

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	Variable Label
Intercept	1	1.980269	1.169176	1.69	0.0959	Intercept
p	1	-1.17654	0.605015	-1.94	0.0568	Price
y	1	0.404117	0.117179	3.45	0.0011	Income
s	1	0.359204	0.085077	4.22	<.0001	Price of Substitutes

Test Results

Num DF	Den DF	F Value	Pr > F	Label
1	113	6.79	0.0104	TEST_1

To test hypotheses that involve parameters in different equations, use the STEST statement. Specify the parameters in the linear hypothesis as *model-label.regressor-name.* (If the MODEL statement does not have a label, you can use the dependent variable name as the label for the model, provided the dependent variable uniquely labels the model.)

For example, the following statements test the hypothesis that the income coefficient in the demand equation is 0.01 times the unit cost coefficient in the supply equation:

```
proc syslin data=in 3sls;
   endogenous  p;
   instruments y u s;
   demand: model q = p y s;
   supply: model q = p u;
   stest1: stest demand.y = .01 * supply.u;
run;
```

The STEST statement results are shown in Figure 36.11. The form and interpretation of the STEST statement results are like the TEST statement results. In this case, the F test produces a p value less than 0.0001, and strongly rejects the hypothesis tested. See the section "STEST Statement" on page 2575 for details.

Figure 36.11 STEST Statement Results

The SYSLIN Procedure
Three-Stage Least Squares Estimation

System Weighted MSE	0.5711
Degrees of freedom	113
System Weighted R-Square	0.9627

Model	DEMAND
Dependent Variable	q
Label	Quantity

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	Variable Label
Intercept	1	1.980269	1.169176	1.69	0.0959	Intercept
p	1	-1.17654	0.605015	-1.94	0.0568	Price
y	1	0.404117	0.117179	3.45	0.0011	Income
s	1	0.359204	0.085077	4.22	<.0001	Price of Substitutes

Model	SUPPLY
Dependent Variable	q
Label	Quantity

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	Variable Label
Intercept	1	-0.51878	0.490999	-1.06	0.2952	Intercept
p	1	1.333080	0.059271	22.49	<.0001	Price
u	1	-1.14623	0.243491	-4.71	<.0001	Unit Cost

Figure 36.11 *continued*

Test Results				
Num DF	Den DF	F Value	Pr > F	Label
1	113	22.46	0.0001	STEST1

You can combine TEST and STEST statements with RESTRICT and SRESTRICT statements to perform hypothesis tests for restricted models. Of course, the validity of the TEST and STEST statement results depends on the correctness of any restrictions you impose on the estimates.

Saving Residuals and Predicted Values

You can store predicted values and residuals from the estimated models in a SAS data set. Specify the OUT= option in the PROC SYSLIN statement and use the OUTPUT statement to specify names for new variables to contain the predicted and residual values.

For example, the following statements store the predicted quantity from the supply and demand equations in a data set PRED:

```
proc syslin data=in out=pred 3sls;
   endogenous  p;
   instruments y u s;
   demand: model q = p y s;
   output predicted=q_demand;
   supply: model q = p u;
   output predicted=q_supply;
run;
```

Plotting Residuals

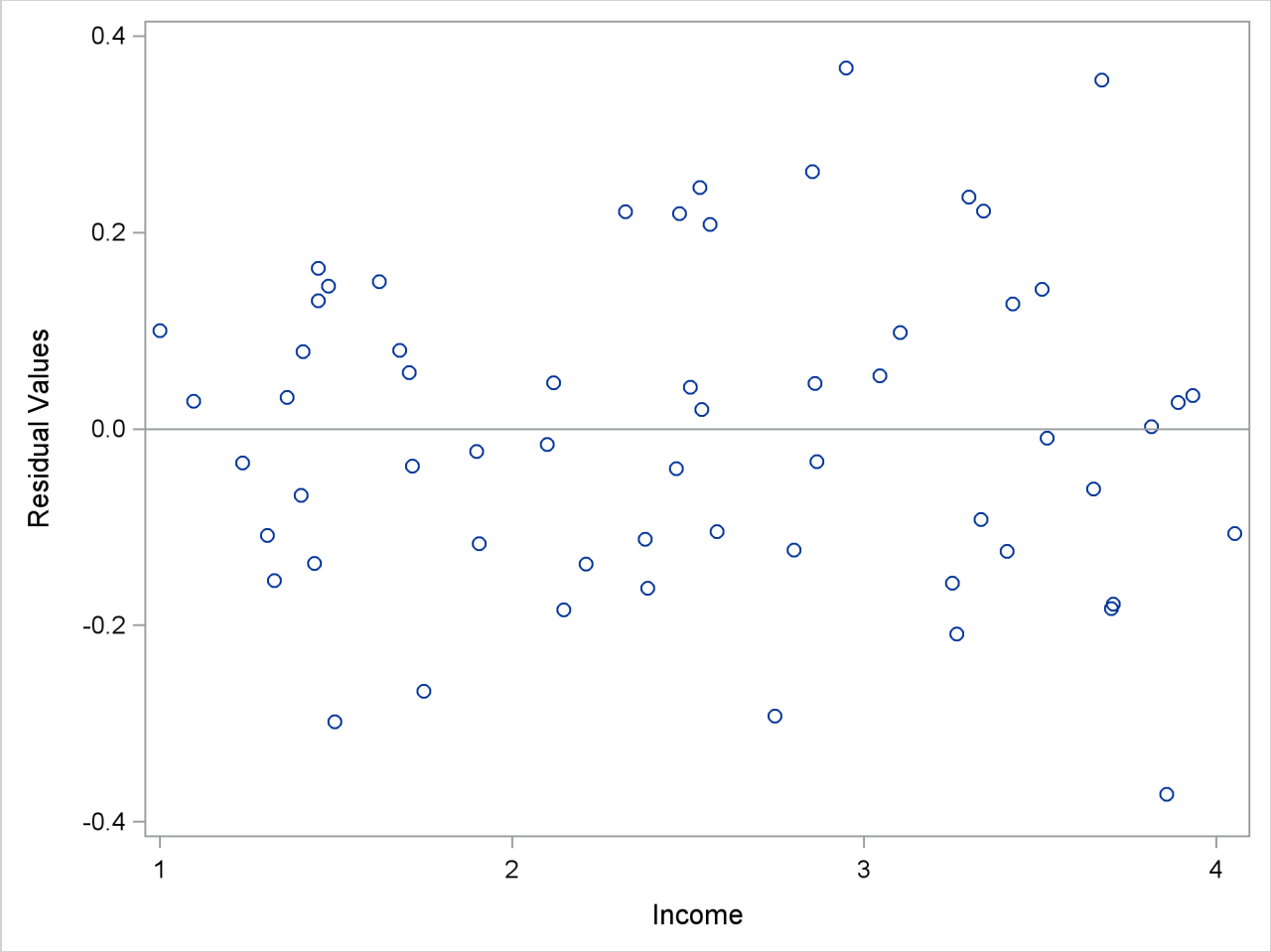
You can plot the residuals against the regressors by using the PROC SGPLOT. For example, the following statements plot the 2SLS residuals for the demand model against price, income, and price of substitutes.

```
proc syslin data=in 2sls out=out;
   endogenous  p;
   instruments y u s;
   demand: model q = p y s;
   output residual=residual_q;
run;

proc sgplot data=out;
   scatter x=p y=residual_q;
   refline 0 / axis=y;
run;

proc sgplot data=out;
   scatter x=y y=residual_q;
   refline 0 / axis=y;
run;
```

```
proc sgplot data=out;
   scatter x=s y=residual_q;
   refline 0 / axis=y;
run;
```

The plot for income is shown in Figure 36.12. The other plots are not shown.

Figure 36.12 Plot of Residuals against Income



Syntax: SYSLIN Procedure

The SYSLIN procedure uses the following statements:

```
PROC SYSLIN options ;
  BY variables ;
  ENDOGENOUS variables ;
  IDENTITY identities ;
  INSTRUMENTS variables ;
  MODEL response = regressors / options ;
  OUTPUT PREDICTED= variable RESIDUAL= variable ;
  RESTRICT restrictions ;
  SRESTRICT restrictions ;
  STEST equations ;
  TEST equations ;
  VAR variables ;
  WEIGHT variable ;
```

Functional Summary

The SYSLIN procedure statements and options are summarized in the following table.

Description	Statement	Option
Data Set Options		
specify the input data set	PROC SYSLIN	DATA=
specify the output data set	PROC SYSLIN	OUT=
write parameter estimates to an output data set	PROC SYSLIN	OUTEST=
write covariances to the OUTEST= data set	PROC SYSLIN	OUTCOV OUTCOV3
write the SSCP matrix to an output data set	PROC SYSLIN	OUTSSCP=
Estimation Method Options		
specify full information maximum likelihood estimation	PROC SYSLIN	FIML
specify iterative SUR estimation	PROC SYSLIN	ITSUR
specify iterative 3SLS estimation	PROC SYSLIN	IT3SLS
specify K-class estimation	PROC SYSLIN	K=
specify limited information maximum likelihood estimation	PROC SYSLIN	LIML
specify minimum expected loss estimation	PROC SYSLIN	MELO
specify ordinary least squares estimation	PROC SYSLIN	OLS
specify seemingly unrelated estimation	PROC SYSLIN	SUR
specify two-stage least squares estimation	PROC SYSLIN	2SLS
specify three-stage least squares estimation	PROC SYSLIN	3SLS

Description	Statement	Option
specify Fuller's modification to LIML	PROC SYSLIN	ALPHA=
specify convergence criterion	PROC SYSLIN	CONVERGE=
specify maximum number of iterations	PROC SYSLIN	MAXIT=
use diagonal of **S** instead of **S**	PROC SYSLIN	SDIAG
exclude RESTRICT statements in final stage	PROC SYSLIN	NOINCLUDE
specify criterion for testing for singularity	PROC SYSLIN	SINGULAR=
specify denominator for variance estimates	PROC SYSLIN	VARDEF=
Printing Control Options		
print all results	PROC SYSLIN	ALL
print first-stage regression statistics	PROC SYSLIN	FIRST
print estimates and SSE at each iteration	PROC SYSLIN	ITPRINT
print the reduced form estimates	PROC SYSLIN	REDUCED
print descriptive statistics	PROC SYSLIN	SIMPLE
print uncorrected SSCP matrix	PROC SYSLIN	USSCP
print correlations of the parameter estimates	MODEL	CORRB
print covariances of the parameter estimates	MODEL	COVB
print Durbin-Watson statistics	MODEL	DW
print Basmann's test	MODEL	OVERID
plot residual values against regressors	MODEL	PLOT
print standardized parameter estimates	MODEL	STB
print unrestricted parameter estimates	MODEL	UNREST
print the model crossproducts matrix	MODEL	XPX
print the inverse of the crossproducts matrix	MODEL	I
suppress printed output	MODEL	NOPRINT
suppress all printed output	PROC SYSLIN	NOPRINT
Model Specification		
specify structural equations	MODEL	
suppress the intercept parameter	MODEL	NOINT
specify linear relationship among variables	IDENTITY	
perform weighted regression	WEIGHT	
Tests and Restrictions on Parameters		
place restrictions on parameter estimates	RESTRICT	
place restrictions on parameter estimates	SRESTRICT	
test linear hypothesis	STEST	
test linear hypothesis	TEST	
Other Statements		
specify BY-group processing	BY	

Description	Statement	Option
specify the endogenous variables	ENDOGENOUS	
specify instrumental variables	INSTRUMENTS	
write predicted and residual values to a data set	OUTPUT	
name variable for predicted values	OUTPUT	PREDICTED=
name variable for residual values	OUTPUT	RESIDUAL=
include additional variables in $X'X$ matrix	VAR	

PROC SYSLIN Statement

PROC SYSLIN *options* **;**

The following options can be used with the PROC SYSLIN statement.

Data Set Options

DATA=*SAS-data-set*
specifies the input data set. If the DATA= option is omitted, the most recently created SAS data set is used. In addition to ordinary SAS data sets, PROC SYSLIN can analyze data sets of TYPE=CORR, TYPE=COV, TYPE=UCORR, TYPE=UCOV, and TYPE=SSCP. See the section "Special TYPE= Input Data Sets" on page 2578 for details.

OUT=*SAS-data-set*
specifies an output SAS data set for residuals and predicted values. The OUT= option is used in conjunction with the OUTPUT statement. See the section "OUT= Data Set" on page 2585 for details.

OUTEST=*SAS-data-set*
writes the parameter estimates to an output data set. See the section "OUTEST= Data Set" on page 2585 for details.

OUTCOV
COVOUT
writes the covariance matrix of the parameter estimates to the OUTEST= data set in addition to the parameter estimates.

OUTCOV3
COV3OUT
writes covariance matrices for each model in a system to the OUTEST= data set when the 3SLS, SUR, or FIML option is used.

OUTSSCP=*SAS-data-set*
writes the sum-of-squares-and-crossproducts matrix to an output data set. See the section "OUTSSCP= Data Set" on page 2586 for details.

Estimation Method Options

2SLS
specifies the two-stage least squares estimation method.

3SLS
specifies the three-stage least squares estimation method.

ALPHA=*value*
specifies Fuller's modification to the LIML estimation method. See the section "Fuller's Modification to LIML" on page 2584 for details.

CONVERGE=*value*
specifies the convergence criterion for the iterative estimation methods IT3SLS, ITSUR, and FIML. The default is CONVERGE=0.0001.

FIML
specifies the full information maximum likelihood estimation method.

ITSUR
specifies the iterative seemingly unrelated estimation method.

IT3SLS
specifies the iterative three-stage least squares estimation method.

K=*value*
specifies the K-class estimation method.

LIML
specifies the limited information maximum likelihood estimation method.

MAXITER=*n*
specifies the maximum number of iterations allowed for the IT3SLS, ITSUR, and FIML estimation methods. The MAXITER= option can be abbreviated as MAXIT=. The default is MAXITER=30.

MELO
specifies the minimum expected loss estimation method.

NOINCLUDE
excludes the RESTRICT statements from the final stage for the 3SLS, IT3SLS, SUR, and ITSUR estimation methods.

OLS
specifies the ordinary least squares estimation method. This is the default.

SDIAG
uses the diagonal of **S** instead of **S** to do the estimation, where **S** is the covariance matrix of equation errors. See the section "Uncorrelated Errors across Equations" on page 2584 for details.

SINGULAR=*value*
specifies a criterion for testing singularity of the crossproducts matrix. This is a tuning parameter used to make PROC SYSLIN more or less sensitive to singularities. The value must be between 0 and 1. The default is SINGULAR=1E–8.

SUR
specifies the seemingly unrelated estimation method.

Printing Control Options

ALL
specifies the CORRB, COVB, DW, I, OVERID, PLOT, STB, and XPX options for every MODEL statement.

FIRST
prints first-stage regression statistics for the endogenous variables regressed on the instruments. This output includes sums of squares, estimates, variances, and standard deviations.

ITPRINT
prints parameter estimates, system-weighted residual sum of squares, and R^2 at each iteration for the IT3SLS and ITSUR estimation methods. For the FIML method, the ITPRINT option prints parameter estimates, negative of log-likelihood function, and norm of gradient vector at each iteration.

NOPRINT
suppresses all printed output. Specifying NOPRINT in the PROC SYSLIN statement is equivalent to specifying NOPRINT in every MODEL statement.

REDUCED
prints the reduced form estimates. If the REDUCED option is specified, you should specify any IDENTITY statements needed to make the system square. See the section "Reduced Form Estimates" on page 2583 for details.

SIMPLE
prints descriptive statistics for the dependent variables. The statistics printed include the sum, mean, uncorrected sum of squares, variance, and standard deviation.

USSCP
prints the uncorrected sum-of-squares-and-crossproducts matrix.

USSCP2
prints the uncorrected sum-of-squares-and-crossproducts matrix for all variables used in the analysis, including predicted values of variables generated by the procedure.

VARDEF=DF | N | WEIGHT | WGT
specifies the denominator to use in calculating cross-equation error covariances and parameter standard errors and covariances. The default is VARDEF=DF, which corrects for model degrees of freedom. VARDEF=N specifies no degrees-of-freedom correction. VARDEF=WEIGHT specifies the sum of the observation weights. VARDEF=WGT specifies the sum of the observation weights minus the model degrees of freedom. See the section "Computation of Standard Errors" on page 2583 for details.

BY Statement

BY *variables* **;**

A BY statement can be used with PROC SYSLIN to obtain separate analyses on observations in groups defined by the BY variables.

ENDOGENOUS Statement

ENDOGENOUS *variables* **;**

The ENDOGENOUS statement declares the jointly dependent variables that are projected in the first-stage regression through the instrument variables. The ENDOGENOUS statement is not needed for the SUR, ITSUR, or OLS estimation methods. The default ENDOGENOUS list consists of all the dependent variables in the MODEL and IDENTITY statements that do not appear in the INSTRUMENTS statement.

IDENTITY Statement

IDENTITY *equation* **;**

The IDENTITY statement specifies linear relationships among variables to write to the OUTEST= data set. It provides extra information in the OUTEST= data set but does not create or compute variables. The OUTEST= data set can be processed by the SIMLIN procedure in a later step.

The IDENTITY statement is also used to compute reduced form coefficients when the REDUCED option in the PROC SYSLIN statement is specified. See the section "Reduced Form Estimates" on page 2583 for details.

The *equation* given by the IDENTITY statement has the same form as equations in the MODEL statement. A label can be specified for an IDENTITY statement as follows:

label **: IDENTITY . . . ;**

INSTRUMENTS Statement

INSTRUMENTS *variables* **;**

The INSTRUMENTS statement declares the variables used in obtaining first-stage predicted values. All the instruments specified are used in each first-stage regression. The INSTRUMENTS statement is required for the 2SLS, 3SLS, IT3SLS, LIML, MELO, and K-class estimation methods. The INSTRUMENTS statement is not needed for the SUR, ITSUR, OLS, or FIML estimation methods.

MODEL Statement

MODEL *response = regressors / options* **;**

The MODEL statement regresses the response variable on the left side of the equal sign against the regressors listed on the right side.

Models can be given labels. Model labels are used in the printed output to identify the results for different models. Model labels are also used in SRESTRICT and STEST statements to refer to parameters in different models. If no label is specified, the response variable name is used as the label for the model. The model label is specified as follows:

label **: MODEL . . . ;**

The following options can be used in the MODEL statement after a slash (/).

ALL
specifies the CORRB, COVB, DW, I, OVERID, PLOT, STB, and XPX options.

ALPHA=*value*
specifies the α parameter for Fuller's modification to the LIML estimation method. See the section "Fuller's Modification to LIML" on page 2584 for details.

CORRB
prints the matrix of estimated correlations between the parameter estimates.

COVB
prints the matrix of estimated covariances between the parameter estimates.

DW
prints Durbin-Watson statistics and autocorrelation coefficients for the residuals. If there are missing values, d' is calculated according to Savin and White (1978). Use the DW option only if the data set to be analyzed is an ordinary SAS data set with time series observations sorted in time order. The Durbin-Watson test is not valid for models with lagged dependent regressors.

I
prints the inverse of the crossproducts matrix for the model, $(\mathbf{X}'\mathbf{X})^{-1}$. If restrictions are specified, the crossproducts matrix printed is adjusted for the restrictions. See the section "Computational Details" on page 2582 for details.

K=*value*
specifies K-class estimation.

NOINT
suppresses the intercept parameter from the model.

NOPRINT
suppresses the normal printed output.

OVERID
prints Basmann's (1960) test for over identifying restrictions. See the section "Overidentification Restrictions" on page 2584 for details.

PLOT
plots residual values against regressors. A plot of the residuals for each regressor is printed.

STB
prints standardized parameter estimates. Sometimes known as a standard partial regression coefficient, a standardized parameter estimate is a parameter estimate multiplied by the standard deviation of the associated regressor and divided by the standard deviation of the response variable.

UNREST
prints parameter estimates computed before restrictions are applied. The UNREST option is valid only if a RESTRICT statement is specified.

XPX

prints the model crossproducts matrix, $X'X$. See the section "Computational Details" on page 2582 for details.

OUTPUT Statement

OUTPUT < *PREDICTED=variable* > < *RESIDUAL=variable* > **;**

The OUTPUT statement writes predicted values and residuals from the preceding model to the data set specified by the OUT= option in the PROC SYSLIN statement. An OUTPUT statement must come after the MODEL statement to which it applies. The OUT= option must be specified in the PROC SYSLIN statement.

The following options can be specified in the OUTPUT statement:

PREDICTED=*variable*

names a new variable to contain the predicted values for the response variable. The PREDICTED= option can be abbreviated as PREDICT=, PRED=, or P=.

RESIDUAL=*variable*

names a new variable to contain the residual values for the response variable. The RESIDUAL= option can be abbreviated as RESID= or R=.

For example, the following statements create an output data set named B. In addition to the variables in the input data set, the data set B contains the variable YHAT, with values that are predicted values of the response variable Y, and YRESID, with values that are the residual values of Y.

```
proc syslin data=a out=b;
   model y = x1 x2;
   output p=yhat r=yresid;
run;
```

For example, the following statements create an output data set named PRED. In addition to the variables in the input data set, the data set PRED contains the variables Q_DEMAND and Q_SUPPLY, with values that are predicted values of the response variable Q for the demand and supply equations respectively, and R_DEMAND and R_SUPPLY, with values that are the residual values of the demand and supply equations respectively.

```
proc syslin data=in out=pred;
   demand: model q = p y s;
   output p=q_demand r=r_demand;
   supply: model q = p u;
   output p=q_supply r=r_supply;
run;
```

See the section "OUT= Data Set" on page 2585 for details.

RESTRICT Statement

RESTRICT *equation , . . . , equation* **;**

The RESTRICT statement places restrictions on the parameter estimates for the preceding MODEL statement. Any number of RESTRICT statements can follow a MODEL statement. Each restriction is written as a linear equation. If more than one restriction is specified in a single RESTRICT statement, the restrictions are separated by commas.

Parameters are referred to by the name of the corresponding regressor variable. Each name used in the equation must be a regressor in the preceding MODEL statement. The keyword INTERCEPT is used to refer to the intercept parameter in the model.

RESTRICT statements can be given labels. The labels are used in the printed output to distinguish results for different restrictions. Labels are specified as follows:

label **: RESTRICT . . . ;**

The following is an example of the use of the RESTRICT statement, in which the coefficients of the regressors X1 and X2 are required to sum to 1.

```
proc syslin data=a;
   model y = x1 x2;
   restrict x1 + x2 = 1;
run;
```

Variable names can be multiplied by constants. When no equal sign appears, the linear combination is set equal to 0. Note that the parameters associated with the variables are restricted, not the variables themselves. Here are some examples of valid RESTRICT statements:

```
restrict x1 + x2 = 1;
restrict x1 + x2 - 1;
restrict 2 * x1 = x2 + x3 , intercept + x4 = 0;
restrict x1 = x2 = x3 = 1;
restrict 2 * x1 - x2;
```

Restricted parameter estimates are computed by introducing a Lagrangian parameter λ for each restriction (Pringle and Rayner 1971). The estimates of these Lagrangian parameters are printed in the "Parameter Estimates" table. If a restriction cannot be applied, its parameter value and degrees of freedom are listed as 0.

The Lagrangian parameter λ measures the sensitivity of the sum of squared errors (SSE) to the restriction. If the restriction is changed by a small amount ϵ, the SSE is changed by $2\lambda\epsilon$.

The *t* ratio tests the significance of the restrictions. If λ is zero, the restricted estimates are the same as the unrestricted.

Any number of restrictions can be specified on a RESTRICT statement, and any number of RESTRICT statements can be used. The estimates are computed subject to all restrictions specified. However, restrictions should be consistent and not redundant.

NOTE: The RESTRICT statement is not supported for the FIML estimation method.

SRESTRICT Statement

SRESTRICT *equation , . . . , equation* **;**

The SRESTRICT statement imposes linear restrictions that involve parameters in two or more MODEL statements. The SRESTRICT statement is like the RESTRICT statement but is used to impose restrictions across equations, whereas the RESTRICT statement applies only to parameters in the immediately preceding MODEL statement.

Each restriction is written as a linear equation. Parameters are referred to as *label.variable,* where *label* is the model label and *variable* is the name of the regressor to which the parameter is attached. (If the MODEL statement does not have a label, you can use the dependent variable name as the label for the model, provided the dependent variable uniquely labels the model.) Each variable name used must be a regressor in the indicated MODEL statement. The keyword INTERCEPT is used to refer to intercept parameters.

SRESTRICT statements can be given labels. The labels are used in the printed output to distinguish results for different restrictions. Labels are specified as follows:

label **: SRESTRICT . . . ;**

The following is an example of the use of the SRESTRICT statement, in which the coefficient for the regressor X2 is constrained to be the same in both models.

```
proc syslin data=a 3sls;
   endogenous y1 y2;
   instruments x1 x2;
   model y1 = y2 x1 x2;
   model y2 = y1 x2;
   srestrict y1.x2 = y2.x2;
run;
```

When no equal sign is used, the linear combination is set equal to 0. Thus, the restriction in the preceding example can also be specified as

```
   srestrict y1.x2 - y2.x2;
```

Any number of restrictions can be specified on an SRESTRICT statement, and any number of SRESTRICT statements can be used. The estimates are computed subject to all restrictions specified. However, restrictions should be consistent and not redundant.

When a system restriction is requested for a single equation estimation method (such as OLS or 2SLS), PROC SYSLIN produces the restricted estimates by actually using a corresponding system method. For example, when SRESTRICT is specified along with OLS, PROC SYSLIN produces the restricted OLS estimates via a two-step process equivalent to using SUR estimation with the SDIAG option. First, the unrestricted OLS results are produced. Then, the GLS (SUR) estimation with the system restriction is performed, using the diagonal of the covariance matrix of the residuals. When SRESTRICT is specified along with 2SLS, PROC SYSLIN produces the restricted 2SLS estimates via a multistep process equivalent to using 3SLS estimation with the SDIAG option. First, the unrestricted 2SLS results are produced. Then, the GLS (3SLS) estimation with the system restriction is performed, using the diagonal of the covariance matrix of the residuals.

The results of the SRESTRICT statements are printed after the parameter estimates for all the models in the system. The format of the SRESTRICT statement output is the same as the "Parameter Estimates" table. In this output the parameter estimate is the Lagrangian parameter λ used to impose the restriction.

The Lagrangian parameter λ measures the sensitivity of the system sum of square errors to the restriction. The system SSE is the system MSE shown in the printed output multiplied by the degrees of freedom. If the restriction is changed by a small amount ϵ, the system SSE is changed by $2\lambda\epsilon$.

The *t* ratio tests the significance of the restriction. If λ is zero, the restricted estimates are the same as the unrestricted estimates.

The model degrees of freedom are not adjusted for the cross-model restrictions imposed by SRESTRICT statements.

NOTE: The SRESTRICT statement is not supported for the LIML and the FIML estimation methods.

STEST Statement

STEST *equation , . . . , equation / options* **;**

The STEST statement performs an *F* test for the joint hypotheses specified in the statement.

The hypothesis is represented in matrix notation as

$$\mathbf{L}\beta = \mathbf{c}$$

and the *F* test is computed as

$$\frac{(\mathbf{L}b - \mathbf{c})'(\mathbf{L}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{L}')^{-1}(\mathbf{L}b - \mathbf{c})}{m\hat{\sigma}^2}$$

where b is the estimate of β, m is the number of restrictions, and $\hat{\sigma}^2$ is the system weighted mean squared error. See the section "Computational Details" on page 2582 for information about the matrix $\mathbf{X}'\mathbf{X}$.

Each hypothesis to be tested is written as a linear equation. Parameters are referred to as *label.variable,* where *label* is the model label and *variable* is the name of the regressor to which the parameter is attached. (If the MODEL statement does not have a label, you can use the dependent variable name as the label for the model, provided the dependent variable uniquely labels the model.) Each variable name used must be a regressor in the indicated MODEL statement. The keyword INTERCEPT is used to refer to intercept parameters.

STEST statements can be given labels. The label is used in the printed output to distinguish different tests. Any number of STEST statements can be specified. Labels are specified as follows:

label **: STEST . . . ;**

The following is an example of the STEST statement:

```
proc syslin data=a 3sls;
   endogenous y1 y2;
   instruments x1 x2;
   model y1 = y2 x1 x2;
   model y2 = y1 x2;
   stest y1.x2 = y2.x2;
run;
```

The test performed is exact only for ordinary least squares, given the OLS assumptions of the linear model. For other estimation methods, the *F* test is based on large sample theory and is only approximate in finite samples.

If RESTRICT or SRESTRICT statements are used, the tests computed by the STEST statement are conditional on the restrictions specified. The validity of the tests can be compromised if incorrect restrictions are imposed on the estimates.

The following are examples of STEST statements:

```
stest a.x1 + b.x2 = 1;
stest 2 * b.x2 = c.x3 + c.x4 ,
      a.intercept + b.x2 = 0;
stest a.x1 = c.x2 = b.x3 = 1;
stest 2 * a.x1 - b.x2 = 0;
```

The PRINT option can be specified in the STEST statement after a slash (/):

PRINT
prints intermediate calculations for the hypothesis tests.

NOTE: The STEST statement is not supported for the FIML estimation method.

TEST Statement

TEST *equation , . . . , equation / options* **;**

The TEST statement performs *F* tests of linear hypotheses about the parameters in the preceding MODEL statement. Each equation specifies a linear hypothesis to be tested. If more than one equation is specified, the equations are separated by commas.

Variable names must correspond to regressors in the preceding MODEL statement, and each name represents the coefficient of the corresponding regressor. The keyword INTERCEPT is used to refer to the model intercept.

TEST statements can be given labels. The label is used in the printed output to distinguish different tests. Any number of TEST statements can be specified. Labels are specified as follows:

label **: TEST . . . ;**

The following is an example of the use of TEST statement, which tests the hypothesis that the coefficients of X1 and X2 are the same:

```
proc syslin data=a;
   model y = x1 x2;
   test x1 = x2;
run;
```

The following statements perform *F* tests for the hypothesis that the coefficients of X1 and X2 are equal, for the hypothesis that the sum of the X1 and X2 coefficients is twice the intercept, and for the joint hypothesis.

```
proc syslin data=a;
   model y = x1 x2;
   x1eqx2:  test x1 = x2;
   sumeq2i: test x1 + x2 = 2 * intercept;
   joint:   test x1 = x2, x1 + x2 = 2 * intercept;
run;
```

The following are additional examples of TEST statements:

```
test x1 + x2 = 1;
test x1 = x2 = x3 = 1;
test 2 * x1 = x2 + x3, intercept + x4 = 0;
test 2 * x1 - x2;
```

The TEST statement performs an F test for the joint hypotheses specified. The hypothesis is represented in matrix notation as follows:

$$\mathbf{L}\beta = \mathbf{c}$$

The F test is computed as

$$\frac{(\mathbf{L}b - \mathbf{c})'(\mathbf{L}(\mathbf{X}'\mathbf{X})^{-}\mathbf{L}')^{-1}(\mathbf{L}b - \mathbf{c})}{m\hat{\sigma}^2}$$

where b is the estimate of β, m is the number of restrictions, and $\hat{\sigma}^2$ is the model mean squared error. See the section "Computational Details" on page 2582 for information about the matrix $\mathbf{X}'\mathbf{X}$.

The test performed is exact only for ordinary least squares, given the OLS assumptions of the linear model. For other estimation methods, the F test is based on large sample theory and is only approximate in finite samples.

If RESTRICT or SRESTRICT statements are used, the tests computed by the TEST statement are conditional on the restrictions specified. The validity of the tests can be compromised if incorrect restrictions are imposed on the estimates.

The PRINT option can be specified in the TEST statement after a slash (/):

PRINT
prints intermediate calculations for the hypothesis tests.

NOTE: The TEST statement is not supported for the FIML estimation method.

VAR Statement

VAR *variables* **;**

The VAR statement is used to include variables in the crossproducts matrix that are not specified in any MODEL statement. This statement is rarely used with PROC SYSLIN and is used only with the OUTSSCP= option in the PROC SYSLIN statement.

WEIGHT Statement

WEIGHT *variable* **;**

The WEIGHT statement is used to perform weighted regression. The WEIGHT statement names a variable in the input data set whose values are relative weights for a weighted least squares fit. If the weight value is proportional to the reciprocal of the variance for each observation, the weighted estimates are the best linear unbiased estimates (BLUE).

Details: SYSLIN Procedure

Input Data Set

PROC SYSLIN does not compute new values for regressors. For example, if you need a lagged variable, you must create it with a DATA step. No values are computed by IDENTITY statements; all values must be in the input data set.

Special TYPE= Input Data Sets

The input data set for most applications of the SYSLIN procedure contains standard rectangular data. However, PROC SYSLIN can also process input data in the form of a crossproducts, covariance, or correlation matrix. Data sets that contain such matrices are identified by values of the TYPE= data set option.

These special kinds of input data sets can be used to save computer time. It takes nk^2 operations, where n is the number of observations and k is the number of variables, to calculate cross products; the regressions are of the order k^3. When n is in the thousands and k is much smaller, you can save most of the computer time in later runs of PROC SYSLIN by reusing the SSCP matrix rather than recomputing it.

The SYSLIN procedure can process TYPE=CORR, COV, UCORR, UCOV, or SSCP data sets. TYPE=CORR and TYPE=COV data sets, usually created by the CORR procedure, contain means and standard deviations, and correlations or covariances. TYPE=SSCP data sets, usually created in previous runs of PROC SYSLIN, contain sums of squares and cross products. For more information about special SAS data sets, see *SAS/STAT User's Guide*.

When special SAS data sets are read, you must specify the TYPE= data set option. PROC CORR and PROC SYSLIN automatically set the type for output data sets; however, if you create the data set by some other means, you must specify its type with the TYPE= data set option.

When the special data sets are used, the DW (Durbin-Watson test) and PLOT options in the MODEL statement cannot be performed, and the OUTPUT statements are not valid.

Estimation Methods

A brief description of the methods used by the SYSLIN procedure follows. For more information about these methods, see the references at the end of this chapter.

There are two fundamental methods of estimation for simultaneous equations: least squares and maximum likelihood. There are two approaches within each of these categories: single equation methods (also referred to as limited information methods) and system methods (also referred to as full information methods). System methods take into account cross-equation correlations of the disturbances in estimating parameters, while single equation methods do not.

OLS, 2SLS, MELO, K-class, SUR, ITSUR, 3SLS, and IT3SLS use the least squares method; LIML and FIML use the maximum likelihood method.

OLS, 2SLS, MELO, K-class, and LIML are single equation methods. The system methods are SUR, ITSUR, 3SLS, IT3SLS, and FIML.

Single Equation Estimation Methods

Single equation methods do not take into account correlations of errors across equations. As a result, these estimators are not asymptotically efficient compared to full information methods; however, there are instances in which they may be preferred. (See the section "Choosing a Method for Simultaneous Equations" on page 2580 for details.)

Let y_i be the dependent endogenous variable in equation i, and X_i and Y_i be the matrices of exogenous and endogenous variables appearing as regressors in the same equation.

The 2SLS method owes its name to the fact that, in a first stage, the instrumental variables are used as regressors to obtain a projected value $\hat{Y}_i$ that is uncorrelated with the residual in equation i. In a second stage, $\hat{Y}_i$ replaces Y_i on the right-hand side to obtain consistent least squares estimators.

Normally, the predetermined variables of the system are used as the instruments. It is possible to use variables other than predetermined variables from your system as instruments; however, the estimation might not be as efficient. For consistent estimates, the instruments must be uncorrelated with the residual and correlated with the endogenous variables.

The LIML method results in consistent estimates that are equal to the 2SLS estimates when an equation is exactly identified. LIML can be viewed as a least-variance ratio estimation or as a maximum likelihood estimation. LIML involves minimizing the ratio $\lambda = (rvar\_eq)/(rvar\_sys)$, where $rvar\_eq$ is the residual variance associated with regressing the weighted endogenous variables on all predetermined variables that appear in that equation, and $rvar\_sys$ is the residual variance associated with regressing weighted endogenous variables on all predetermined variables in the system.

The MELO method computes the minimum expected loss estimator. MELO estimators "minimize the posterior expectation of generalized quadratic loss functions for structural coefficients of linear structural models" (Judge et al. 1985, p. 635).

K-class estimators are a class of estimators that depends on a user-specified parameter k. A k value less than 1 is recommended but not required. The parameter k can be deterministic or stochastic, but its probability limit must equal 1 for consistent parameter estimates. When all the predetermined variables are listed as instruments, they include all the other single equation estimators supported by PROC SYSLIN. The instance when some of the predetermined variables are not listed among the instruments is not supported by PROC SYSLIN for the general K-class estimation. However, it is supported for the other methods.

For $k = 1$, the K-class estimator is the 2SLS estimator, while for $k = 0$, the K-class estimator is the OLS estimator. The K-class interpretation of LIML is that $k = \lambda$. Note that k is stochastic in the LIML method, unlike for OLS and 2SLS.

MELO is a Bayesian K-class estimator. It yields estimates that can be expressed as a matrix-weighted average of the OLS and 2SLS estimates. MELO estimators have finite second moments and hence finite risk. Other frequently used K-class estimators might not have finite moments under some commonly encountered circumstances, and hence there can be infinite risk relative to quadratic and other loss functions.

One way of comparing K-class estimators is to note that when k =1, the correlation between regressor and the residual is completely corrected for. In all other cases, it is only partially corrected for.

See "Computational Details" on page 2582 for more details about K-class estimators.

SUR and 3SLS Estimation Methods

SUR might improve the efficiency of parameter estimates when there is contemporaneous correlation of errors across equations. In practice, the contemporaneous correlation matrix is estimated using OLS residuals. Under two sets of circumstances, SUR parameter estimates are the same as those produced by OLS: when there is no contemporaneous correlation of errors across equations (the estimate of the contemporaneous correlation matrix is diagonal) and when the independent variables are the same across equations.

Theoretically, SUR parameter estimates are always at least as efficient as OLS in large samples, provided that your equations are correctly specified. However, in small samples the need to estimate the covariance matrix from the OLS residuals increases the sampling variability of the SUR estimates. This effect can cause SUR to be less efficient than OLS. If the sample size is small and the cross-equation correlations are small, then OLS is preferred to SUR. The consequences of specification error are also more serious with SUR than with OLS.

The 3SLS method combines the ideas of the 2SLS and SUR methods. Like 2SLS, the 3SLS method uses $\hat{Y}$ instead of Y for endogenous regressors, which results in consistent estimates. Like SUR, the 3SLS method takes the cross-equation error correlations into account to improve large sample efficiency. For 3SLS, the 2SLS residuals are used to estimate the cross-equation error covariance matrix.

The SUR and 3SLS methods can be iterated by recomputing the estimate of the cross-equation covariance matrix from the SUR or 3SLS residuals and then computing new SUR or 3SLS estimates based on this updated covariance matrix estimate. Continuing this iteration until convergence produces ITSUR or IT3SLS estimates.

FIML Estimation Method

The FIML estimator is a system generalization of the LIML estimator. The FIML method involves minimizing the determinant of the covariance matrix associated with residuals of the reduced form of the equation system. From a maximum likelihood standpoint, the LIML method involves assuming that the errors are normally distributed and then maximizing the likelihood function subject to restrictions on a particular equation. FIML is similar, except that the likelihood function is maximized subject to restrictions on all of the parameters in the model, not just those in the equation being estimated.

NOTE: The RESTRICT, SRESTRICT, TEST, and STEST statements are not supported when the FIML method is used.

Choosing a Method for Simultaneous Equations

A number of factors should be taken into account in choosing an estimation method. Although system methods are asymptotically most efficient in the absence of specification error, system methods are more sensitive to specification error than single equation methods.

In practice, models are never perfectly specified. It is a matter of judgment whether the misspecification is serious enough to warrant avoidance of system methods.

Another factor to consider is sample size. With small samples, 2SLS might be preferred to 3SLS. In general, it is difficult to say much about the small sample properties of K-class estimators because the results depend on the regressors used.

LIML and FIML are invariant to the normalization rule imposed but are computationally more expensive than 2SLS or 3SLS.

If the reason for contemporaneous correlation among errors across equations is a common, omitted variable, it is not necessarily best to apply SUR. SUR parameter estimates are more sensitive to specification error than OLS. OLS might produce better parameter estimates under these circumstances. SUR estimates are also affected by the sampling variation of the error covariance matrix. There is some evidence from Monte Carlo studies that SUR is less efficient than OLS in small samples.

ANOVA Table for Instrumental Variables Methods

In the instrumental variables methods (2SLS, LIML, K-class, MELO), first-stage predicted values are substituted for the endogenous regressors. As a result, the regression sum of squares (RSS) and the error sum of squares (ESS) do not sum to the total corrected sum of squares for the dependent variable (TSS). The analysis-of-variance table included in the second-stage results gives these sums of squares and the mean squares that are used for the F test, but this table is not a variance decomposition in the usual sense.

The F test shown in the instrumental variables case is a valid test of the no-regression hypothesis that the true coefficients of all regressors are 0. However, because of the first-stage projection of the regression mean square, this is a Wald-type test statistic, which is asymptotically F but not exactly F-distributed in finite samples. Thus, for small samples the F test is only approximate when instrumental variables are used.

The R-Square Statistics

As explained in the section "ANOVA Table for Instrumental Variables Methods" on page 2581, when instrumental variables are used, the regression sum of squares (RSS) and the error sum of squares (ESS) do not sum to the total corrected sum of squares. In this case, there are several ways that the R^2 statistic can be defined.

The definition of R^2 used by the SYSLIN procedure is

$$\mathrm{R}^2 = \frac{RSS}{RSS + ESS}$$

This definition is consistent with the F test of the null hypothesis that the true coefficients of all regressors are zero. However, this R^2 might not be a good measure of the goodness of fit of the model.

System Weighted R-Square and System Weighted Mean Squared Error

The system weighted R^2, printed for the 3SLS, IT3SLS, SUR, ITSUR, and FIML methods, is computed as follows.

$$\mathrm{R}^2 = \mathbf{Y}'\mathbf{W}\mathbf{R}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{R}'\mathbf{W}\mathbf{Y}/\mathbf{Y}'\mathbf{W}\mathbf{Y}$$

In this equation, the matrix $\mathbf{X}'\mathbf{X}$ is $\mathbf{R}'\mathbf{W}\mathbf{R}$ and $\mathbf{W}$ is the projection matrix of the instruments:

$$\mathbf{W} = \mathbf{S}^{-1} \otimes \mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'$$

The matrix $\mathbf{Z}$ is the instrument set, $\mathbf{R}$ is the regressor set, and $\mathbf{S}$ is the estimated cross-model covariance matrix.

The system weighted MSE, printed for the 3SLS, IT3SLS, SUR, ITSUR, and FIML methods, is computed as follows:

$$MSE = \frac{1}{tdf}(\mathbf{Y}'\mathbf{W}\mathbf{Y} - \mathbf{Y}'\mathbf{W}\mathbf{R}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{R}'\mathbf{W}\mathbf{Y})$$

In this equation, *tdf* is the sum of the error degrees of freedom for the equations in the system.

Computational Details

This section discusses various computational details.

Computation of Least Squares-Based Estimators

Let the system be composed of G equations and let the ith equation be expressed in this form:

$$y_i = Y_i\boldsymbol{\beta}_i + X_i\boldsymbol{\gamma}_i + \mathbf{u}$$

where

y_i is the vector of observations on the dependent variable

Y_i is the matrix of observations on the endogenous variables included in the equation

$\boldsymbol{\beta}_i$ is the vector of parameters associated with Y_i

X_i is the matrix of observations on the predetermined variables included in the equation

$\boldsymbol{\gamma}_i$ is the vector of parameters associated with X_i

$\mathbf{u}$ is a vector of errors

Let $\hat{V}_i = Y_i - \hat{Y}_i$, where $\hat{Y}_i$ is the projection of Y_i onto the space spanned by the instruments matrix $\mathbf{Z}$.

Let

$$\boldsymbol{\delta}_i = \begin{bmatrix} \boldsymbol{\beta}_i \\ \boldsymbol{\gamma}_i \end{bmatrix}$$

be the vector of parameters associated with both the endogenous and exogenous variables.

The K-class of estimators (Theil 1971) is defined by

$$\hat{\boldsymbol{\delta}}_{i,k} = \begin{bmatrix} Y_i'Y_i - k\hat{V}_i'\hat{V}_i & Y_i'X_i \\ X_i'Y_i & X_i'X_i \end{bmatrix}^{-1} \begin{bmatrix} (Y_i - kV_i)'y_i \\ X_i'y_i \end{bmatrix}$$

where k is a user-defined value.

Let

$$\mathbf{R} = [Y_i \, X_i]$$

and

$$\hat{\mathbf{R}} = [\hat{Y}_i \; X_i]$$

The 2SLS estimator is defined as

$$\hat{\boldsymbol{\delta}}_{i,2SLS} = [\hat{R}_i' \; \hat{R}_i]^{-1} \hat{R}_i' y_i$$

Let $\mathbf{y}$ and $\boldsymbol{\delta}$ be the vectors obtained by stacking the vectors of dependent variables and parameters for all G equations, and let $\mathbf{R}$ and $\hat{\mathbf{R}}$ be the block diagonal matrices formed by R_i and $\hat{R}_i$, respectively.

The SUR and ITSUR estimators are defined as

$$\hat{\boldsymbol{\delta}}_{(IT)SUR} = \left[\mathbf{R}' \left(\hat{\Sigma}^{-1} \otimes \mathbf{I}\right) \mathbf{R}\right]^{-1} \mathbf{R}' \left(\hat{\Sigma}^{-1} \otimes \mathbf{I}\right) \mathbf{y}$$

while the 3SLS and IT3SLS estimators are defined as

$$\hat{\boldsymbol{\delta}}_{(IT)3SLS} = \left[\hat{\mathbf{R}}' \left(\hat{\Sigma}^{-1} \otimes \mathbf{I}\right) \hat{\mathbf{R}}\right]^{-1} \hat{\mathbf{R}}' \left(\hat{\Sigma}^{-1} \otimes \mathbf{I}\right) \mathbf{y}$$

where $\mathbf{I}$ is the identity matrix, and $\hat{\Sigma}$ is an estimator of the cross-equation correlation matrix. For 3SLS, $\hat{\Sigma}$ is obtained from the 2SLS estimation, while for SUR it is derived from the OLS estimation. For IT3SLS and ITSUR, it is obtained iteratively from the previous estimation step, until convergence.

Computation of Standard Errors

The VARDEF= option in the PROC SYSLIN statement controls the denominator used in calculating the cross-equation covariance estimates and the parameter standard errors and covariances. The values of the VARDEF= option and the resulting denominator are as follows:

N	uses the number of nonmissing observations.
DF	uses the number of nonmissing observations less the degrees of freedom in the model.
WEIGHT	uses the sum of the observation weights given by the WEIGHTS statement.
WDF	uses the sum of the observation weights given by the WEIGHTS statement less the degrees of freedom in the model.

The VARDEF= option does not affect the model mean squared error, root mean squared error, or R^2 statistics. These statistics are always based on the error degrees of freedom, regardless of the VARDEF= option. The VARDEF= option also does not affect the dependent variable coefficient of variation (CV).

Reduced Form Estimates

The REDUCED option in the PROC SYSLIN statement computes estimates of the reduced form coefficients. The REDUCED option requires that the equation system be square. If there are fewer models than endogenous variables, IDENTITY statements can be used to complete the equation system.

The reduced form coefficients are computed as follows. Represent the equation system, with all endogenous variables moved to the left-hand side of the equations and identities, as

$$\mathbf{B}\mathbf{Y} = \mathbf{\Gamma}\mathbf{X}$$

Here $\mathbf{B}$ is the estimated coefficient matrix for the endogenous variables $\mathbf{Y}$, and $\mathbf{\Gamma}$ is the estimated coefficient matrix for the exogenous (or predetermined) variables $\mathbf{X}$.

The system can be solved for $\mathbf{Y}$ as follows, provided $\mathbf{B}$ is square and nonsingular:

$$\mathbf{Y} = \mathbf{B}^{-1}\mathbf{\Gamma}\mathbf{X}$$

The reduced form coefficients are the matrix $\mathbf{B}^{-1}\mathbf{\Gamma}$.

Uncorrelated Errors across Equations

The SDIAG option in the PROC SYSLIN statement computes estimates by assuming uncorrelated errors across equations. As a result, when the SDIAG option is used, the 3SLS estimates are identical to 2SLS estimates, and the SUR estimates are the same as the OLS estimates.

Overidentification Restrictions

The OVERID option in the MODEL statement can be used to test for overidentifying restrictions on parameters of each equation. The null hypothesis is that the predetermined variables that do not appear in any equation have zero coefficients. The alternative hypothesis is that at least one of the assumed zero coefficients is nonzero. The test is approximate and rejects the null hypothesis too frequently for small sample sizes.

The formula for the test is given as follows. Let $y_i = \beta_i \mathbf{Y}_i + \gamma_i \mathbf{Z}_i + e_i$ be the ith equation. $\mathbf{Y}_i$ are the endogenous variables that appear as regressors in the ith equation, and $\mathbf{Z}_i$ are the instrumental variables that appear as regressors in ith equation. Let N_i be the number of variables in $\mathbf{Y}_i$ and $\mathbf{Z}_i$.

Let $v_i = y_i - \mathbf{Y}_i\hat{\beta}_i$. Let $\mathbf{Z}$ represent all instrumental variables, T be the total number of observations, and K be the total number of instrumental variables. Define $\hat{l}$ as follows:

$$\hat{l} = \frac{v'_i(\mathbf{I} - \mathbf{Z}_i(\mathbf{Z}'_i\mathbf{Z}_i)^{-1}\mathbf{Z}'_i)v_i}{v'_i(\mathbf{I} - \mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}')v_i}$$

Then the test statistic

$$\frac{T-K}{K-N_i}(\hat{l} - 1)$$

is distributed approximately as an F with $K - N_i$ and $T - K$ degrees of freedom. See Basmann (1960) for more information.

Fuller's Modification to LIML

The ALPHA= option in the PROC SYSLIN and MODEL statements parameterizes Fuller's modification to LIML. This modification is $k = \gamma - (\alpha/(n-g))$, where α is the value of the ALPHA= option, γ is the LIML k value,n is the number of observations, and g is the number of predetermined variables. Fuller's modification is not used unless the ALPHA= option is specified. See Fuller (1977) for more information.

Missing Values

Observations that have a missing value for any variable in the analysis are excluded from the computations.

OUT= Data Set

The output SAS data set produced by the OUT= option in the PROC SYSLIN statement contains all the variables in the input data set and the variables that contain predicted values and residuals specified by OUTPUT statements.

The residuals are computed as actual values minus predicted values. Predicted values never use lags of other predicted values, as would be desirable for dynamic simulation. For these applications, PROC SIMLIN is available to predict or simulate values from the estimated equations.

OUTEST= Data Set

The OUTEST= option produces a TYPE=EST output SAS data set that contains estimates from the regressions. The variables in the OUTEST= data set are as follows:

BY variables	identifies the BY statement variables that are included in the OUTEST= data set.
TYPE	identifies the estimation type for the observations. The _TYPE_ value INST indicates first-stage regression estimates. Other values indicate the estimation method used: 2SLS indicates two-stage least squares results, 3SLS indicates three-stage least squares results, LIML indicates limited information maximum likelihood results, and so forth. Observations added by IDENTITY statements have the _TYPE_ value IDENTITY.
STATUS	identifies the convergence status of the estimation. _STATUS_ equals 0 when convergence criteria are met. Otherwise, _STATUS_ equals 1 when the estimation converges with a note, 2 when it converges with a warning, or 3 when it fails to converge.
MODEL	identifies the model label. The model label is the label specified in the MODEL statement or the dependent variable name if no label is specified. For first-stage regression estimates, _MODEL_ has the value FIRST.
DEPVAR	identifies the name of the dependent variable for the model.
NAME	identifies the names of the regressors for the rows of the covariance matrix, if the COVOUT option is specified. _NAME_ has a blank value for the parameter estimates observations. The _NAME_ variable is not included in the OUTEST= data set unless the COVOUT option is used to output the covariance of parameter estimates matrix.
SIGMA	contains the root mean squared error for the model, which is an estimate of the standard deviation of the error term. The _SIGMA_ variable contains the same values reported as Root MSE in the printed output.
INTERCEPT	identifies the intercept parameter estimates.
regressors	identifies the regressor variables from all the MODEL statements that are included in the OUTEST= data set. Variables used in IDENTIFY statements are also included in the OUTEST= data set.

The parameter estimates are stored under the names of the regressor variables. The intercept parameters are stored in the variable INTERCEPT. The dependent variable of the model is given a coefficient of –1. Variables that are not in a model have missing values for the OUTEST= observations for that model.

Some estimation methods require computation of preliminary estimates. All estimates computed are output to the OUTEST= data set. For each BY group and each estimation, the OUTEST= data set contains one observation for each MODEL or IDENTITY statement. Results for different estimations are identified by the _TYPE_ variable.

For example, consider the following statements:

```
proc syslin data=a outest=est 3sls;
   by b;
   endogenous y1 y2;
   instruments x1-x4;
   model y1 = y2 x1 x2;
   model y2 = y1 x3 x4;
   identity x1 = x3 + x4;
run;
```

The 3SLS method requires both a preliminary 2SLS stage and preliminary first-stage regressions for the endogenous variable. The OUTEST= data set thus contains three different kinds of estimates. The observations for the first-stage regression estimates have the _TYPE_ value INST. The observations for the 2SLS estimates have the _TYPE_ value 2SLS. The observations for the final 3SLS estimates have the _TYPE_ value 3SLS.

Since there are two endogenous variables in this example, there are two first-stage regressions and two _TYPE_=INST observations in the OUTEST= data set. Since there are two model statements, there are two OUTEST= observations with _TYPE_=2SLS and two observations with _TYPE_=3SLS. In addition, the OUTEST= data set contains an observation with the _TYPE_ value IDENTITY that contains the coefficients specified by the IDENTITY statement. All these observations are repeated for each BY group in the input data set defined by the values of the BY variable B.

When the COVOUT option is specified, the estimated covariance matrix for the parameter estimates is included in the OUTEST= data set. Each observation for parameter estimates is followed by observations that contain the rows of the parameter covariance matrix for that model. The row of the covariance matrix is identified by the variable _NAME_. For observations that contain parameter estimates, _NAME_ is blank. For covariance observations, _NAME_ contains the regressor name for the row of the covariance matrix and the regressor variables contain the covariances.

See Example 36.1 for an example of the OUTEST= data set.

OUTSSCP= Data Set

The OUTSSCP= option produces a TYPE=SSCP output SAS data set that contains sums of squares and cross products. The data set contains all variables used in the MODEL, IDENTITY, and VAR statements. Observations are identified by the variable _NAME_.

The OUTSSCP= data set can be useful when a large number of observations are to be explored in many different SYSLIN runs. The sum-of-squares-and-crossproducts matrix can be saved with the OUTSSCP=

option and used as the DATA= data set on subsequent SYSLIN runs. This is much less expensive computationally because PROC SYSLIN never reads the original data again. In the step that creates the OUTSSCP= data set, include in the VAR statement all the variables you expect to use.

Printed Output

The printed output produced by the SYSLIN procedure is as follows:

1. If the SIMPLE option is used, a table of descriptive statistics is printed that shows the sum, mean, sum of squares, variance, and standard deviation for all the variables used in the models.
2. If the FIRST option is specified and an instrumental variables method is used, first-stage regression results are printed. The results show the regression of each endogenous variable on the variables in the INSTRUMENTS list.
3. The results of the second-stage regression are printed for each model. (See the following section "Printed Output for Each Model" on page 2587 for details.)
4. If a systems method like 3SLS, SUR, or FIML is used, the cross-equation error covariance matrix is printed. This matrix is shown four ways: the covariance matrix itself, the correlation matrix form, the inverse of the correlation matrix, and the inverse of the covariance matrix.
5. If a systems method like 3SLS, SUR, or FIML is used, the system weighted mean squared error and system weighted R^2 statistics are printed. The system weighted MSE and R^2 measure the fit of the joint model obtained by stacking all the models together and performing a single regression with the stacked observations weighted by the inverse of the model error variances.
6. If a systems method like 3SLS, SUR, or FIML is used, the final results are printed for each model.
7. If the REDUCED option is used, the reduced form coefficients are printed. These consist of the structural coefficient matrix for the endogenous variables, the structural coefficient matrix for the exogenous variables, the inverse of the endogenous coefficient matrix, and the reduced form coefficient matrix. The reduced form coefficient matrix is the product of the inverse of the endogenous coefficient matrix and the exogenous structural coefficient matrix.

Printed Output for Each Model

The results printed for each model include the analysis-of-variance table, the "Parameter Estimates" table, and optional items requested by TEST statements or by options in the MODEL statement.

The printed output produced for each model is described in the following.

The analysis-of-variance table includes the following:

- the model degrees of freedom, sum of squares, and mean square
- the error degrees of freedom, sum of squares, and mean square. The error mean square is computed by dividing the error sum of squares by the error degrees of freedom and is not affected by the VARDEF= option.

- the corrected total degrees of freedom and total sum of squares. Note that for instrumental variables methods, the model and error sums of squares do not add to the total sum of squares.
- the *F* ratio, labeled "F Value," and its significance, labeled "PROB>F," for the test of the hypothesis that all the nonintercept parameters are 0
- the root mean squared error. This is the square root of the error mean square.
- the dependent variable mean
- the coefficient of variation (CV) of the dependent variable
- the R^2 statistic. This R^2 is computed consistently with the calculation of the *F* statistic. It is valid for hypothesis tests but might not be a good measure of fit for models estimated by instrumental variables methods.
- the R^2 statistic adjusted for model degrees of freedom, labeled "Adj R-SQ"

The "Parameter Estimates" table includes the following:

- estimates of parameters for regressors in the model and the Lagrangian parameter for each restriction specified
- a degrees of freedom column labeled DF. Estimated model parameters have 1 degree of freedom. Restrictions have a DF of –1. Regressors or restrictions dropped from the model due to collinearity have a DF of 0.
- the standard errors of the parameter estimates
- the *t* statistics, which are the parameter estimates divided by the standard errors
- the significance of the *t* tests for the hypothesis that the true parameter is 0, labeled "Pr > |t|." As previously noted, the significance tests are strictly valid in finite samples only for OLS estimates but are asymptotically valid for the other methods.
- the standardized regression coefficients, if the STB option is specified. This is the parameter estimate multiplied by the ratio of the standard deviation of the regressor to the standard deviation of the dependent variable.
- the labels of the regressor variables or restriction labels

In addition to the analysis-of-variance table and the "Parameter Estimates" table, the results printed for each model can include the following:

- If TEST statements are specified, the test results are printed.
- If the DW option is specified, the Durbin-Watson statistic and first-order autocorrelation coefficient are printed.
- If the OVERID option is specified, the results of Basmann's test for overidentifying restrictions are printed.

- If the PLOT option is used, plots of residual against each regressor are printed.
- If the COVB or CORRB options are specified, the results for each model also include the covariance or correlation matrix of the parameter estimates. For systems methods like 3SLS and FIML, the COVB and CORB output is printed for the whole system after the output for the last model, instead of separately for each model.

The third-stage output for 3SLS, SUR, IT3SLS, ITSUR, and FIML does not include the analysis-of-variance table. When a systems method is used, the second-stage output does not include the optional output, except for the COVB and CORRB matrices.

ODS Table Names

PROC SYSLIN assigns a name to each table it creates. You can use these names to reference the table when you use the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in the following table. If the estimation method used is 3SLS, IT3SLS, ITSUR or SUR, you can obtain tables by specifying ODS OUTPUT CorrResiduals, InvCorrResiduals, InvCovResiduals.

Table 36.2 ODS Tables Produced in PROC SYSLIN

ODS Table Name	Description	Option
ANOVA	Summary of the SSE, MSE for the equations	default
AugXPXMat	Model crossproducts	XPX or USSCP
AutoCorrStat	Autocorrelation statistics	DW
ConvergenceStatus	Convergence status	default
CorrB	Correlations of parameters	CORRB
CorrResiduals	Correlations of residuals	
CovB	Covariance of parameters	COVB
CovResiduals	Covariance of residuals	
EndoMat	Endogenous variables	REDUCED
ExogMat	Exogenous variables	REDUCED
FitStatistics	Statistics of fit	default
InvCorrResiduals	Inverse correlations of residuals	
InvCovResiduals	Inverse covariance of residuals	
InvEndoMat	Inverse endogenous variables	REDUCED
InvXPX	$X'X$ inverse for system	I
IterHistory	Iteration printing	ITPRINT
MissingValues	Missing values generated by the program	default
ModelVars	Name and label for the model	default
ParameterEstimates	Parameter estimates	default
RedMat	Reduced form	REDUCED
SimpleStatistics	Descriptive statistics	SIMPLE
SSCP	Model crossproducts	XPX or USSCP
TestResults	Test for overidentifying restrictions	
Weight	Weighted model statistics	

ODS Graphics

This section describes the use of ODS for creating graphics with the SYSLIN procedure.

ODS Graph Names

PROC SYSLIN assigns a name to each graph it creates using ODS. You can use these names to reference the graphs when you use ODS. The names are listed in Table 36.3.

To request these graphs, you must specify the ODS GRAPHICS statement.

Table 36.3 ODS Graphics Produced by PROC SYSLIN

ODS Graph Name	Plot Description
DiagnosticsPanel	All applicable plots listed below
ActualByPredicted	Predicted versus actual plot
QQPlot	Q-Q plot of residuals
ResidualHistogram	Histogram of the residuals
ResidualPlot	Residual plot

Examples: SYSLIN Procedure

Example 36.1: Klein's Model I Estimated with LIML and 3SLS

This example uses PROC SYSLIN to estimate the classic Klein Model I. For a discussion of this model, see Theil (1971). The following statements read the data.

```
*-------------------------Klein's Model I--------------------------*
| By L.R. Klein, Economic Fluctuations in the United States, 1921-1941 |
| (1950), NY: John Wiley.   A macro-economic model of the U.S. with    |
| three behavioral equations, and several identities. See Theil, p.456.|
*-----------------------------------------------------------------------*;
data klein;
input year c p w i x wp g t k wsum;
   date=mdy(1,1,year);
   format date monyy.;
   y   =c+i+g-t;
   yr  =year-1931;
   klag=lag(k);
   plag=lag(p);
   xlag=lag(x);
   label year='Year'
         date='Date'
         c   ='Consumption'
         p   ='Profits'
         w   ='Private Wage Bill'
         i   ='Investment'
```

```
          k   ='Capital Stock'
          y   ='National Income'
          x   ='Private Production'
          wsum='Total Wage Bill'
          wp  ='Govt Wage Bill'
          g   ='Govt Demand'
          i   ='Taxes'
          klag='Capital Stock Lagged'
          plag='Profits Lagged'
          xlag='Private Product Lagged'
          yr  ='YEAR-1931';
datalines;
1920     .  12.7     .    .  44.9    .     .     .  182.8     .
1921  41.9  12.4  25.5 -0.2  45.6  2.7   3.9   7.7  182.6  28.2
1922  45.0  16.9  29.3  1.9  50.1  2.9   3.2   3.9  184.5  32.2
1923  49.2  18.4  34.1  5.2  57.2  2.9   2.8   4.7  189.7  37.0
1924  50.6  19.4  33.9  3.0  57.1  3.1   3.5   3.8  192.7  37.0
1925  52.6  20.1  35.4  5.1  61.0  3.2   3.3   5.5  197.8  38.6
1926  55.1  19.6  37.4  5.6  64.0  3.3   3.3   7.0  203.4  40.7
1927  56.2  19.8  37.9  4.2  64.4  3.6   4.0   6.7  207.6  41.5
1928  57.3  21.1  39.2  3.0  64.5  3.7   4.2   4.2  210.6  42.9
1929  57.8  21.7  41.3  5.1  67.0  4.0   4.1   4.0  215.7  45.3
1930  55.0  15.6  37.9  1.0  61.2  4.2   5.2   7.7  216.7  42.1

   ... more lines ...
```

The following statements estimate the Klein model using the limited information maximum likelihood method. In addition, the parameter estimates are written to a SAS data set with the OUTEST= option.

```
proc syslin data=klein outest=b liml;
   endogenous c p w i x wsum k y;
   instruments klag plag xlag wp g t yr;
   consume: model c = p plag  wsum;
   invest:  model i = p plag  klag;
   labor:   model w = x xlag  yr;
run;

proc print data=b;
run;
```

The PROC SYSLIN estimates are shown in Output 36.1.1 through Output 36.1.3.

Output 36.1.1 LIML Estimates for Consumption

The SYSLIN Procedure
Limited-Information Maximum Likelihood Estimation

Model	CONSUME
Dependent Variable	c
Label	Consumption

Output 36.1.1 *continued*

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	3	854.3541	284.7847	118.42	<.0001
Error	17	40.88419	2.404952		
Corrected Total	20	941.4295			

Root MSE	1.55079	R-Square	0.95433
Dependent Mean	53.99524	Adj R-Sq	0.94627
Coeff Var	2.87209		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	Variable Label
Intercept	1	17.14765	2.045374	8.38	<.0001	Intercept
p	1	-0.22251	0.224230	-0.99	0.3349	Profits
plag	1	0.396027	0.192943	2.05	0.0558	Profits Lagged
wsum	1	0.822559	0.061549	13.36	<.0001	Total Wage Bill

Output 36.1.2 LIML Estimates for Investments

The SYSLIN Procedure
Limited-Information Maximum Likelihood Estimation

Model	INVEST
Dependent Variable	i
Label	Taxes

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	3	210.3790	70.12634	34.06	<.0001
Error	17	34.99649	2.058617		
Corrected Total	20	252.3267			

Root MSE	1.43479	R-Square	0.85738
Dependent Mean	1.26667	Adj R-Sq	0.83221
Coeff Var	113.27274		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	Variable Label
Intercept	1	22.59083	9.498146	2.38	0.0294	Intercept
p	1	0.075185	0.224712	0.33	0.7420	Profits
plag	1	0.680386	0.209145	3.25	0.0047	Profits Lagged
klag	1	-0.16826	0.045345	-3.71	0.0017	Capital Stock Lagged

Output 36.1.3 LIML Estimates for Labor

The SYSLIN Procedure
Limited-Information Maximum Likelihood Estimation

Model	LABOR
Dependent Variable	w
Label	Private Wage Bill

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	3	696.1485	232.0495	393.62	<.0001
Error	17	10.02192	0.589525		
Corrected Total	20	794.9095			

Root MSE	0.76781	**R-Square**	0.98581
Dependent Mean	36.36190	**Adj R-Sq**	0.98330
Coeff Var	2.11156		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	Variable Label
Intercept	1	1.526187	1.320838	1.16	0.2639	Intercept
x	1	0.433941	0.075507	5.75	<.0001	Private Production
xlag	1	0.151321	0.074527	2.03	0.0583	Private Product Lagged
yr	1	0.131593	0.035995	3.66	0.0020	YEAR-1931

The OUTEST= data set is shown in part in Output 36.1.4. Note that the data set contains the parameter estimates and root mean squared errors, _SIGMA_, for the first-stage instrumental regressions as well as the parameter estimates and σ for the LIML estimates for the three structural equations.

Output 36.1.4 The OUTEST= Data Set

Obs	_TYPE_	_STATUS_	_MODEL_	_DEPVAR_	_SIGMA_	Intercept	klag	plag	xlag	wp
1	LIML	0 Converged	CONSUME	c	1.55079	17.1477	.	0.39603	.	.
2	LIML	0 Converged	INVEST	i	1.43479	22.5908	-0.16826	0.68039	.	.
3	LIML	0 Converged	LABOR	w	0.76781	1.5262	.	.	0.15132	.

Obs	g	t	yr	c	p	w	i	x	wsum	k	y
1	.	.	.	-1	-0.22251	.	.	.	0.82256	.	.
2	.	.	.	.	0.07518	.	-1	.	.	.	.
3	.	.	0.13159	.	.	-1	.	0.43394	.	.	.

The following statements estimate the model using the 3SLS method. The reduced form estimates are produced by the REDUCED option; IDENTITY statements are used to make the model complete.

```
proc syslin data=klein 3sls reduced;
   endogenous c p w i x wsum k y;
   instruments klag plag xlag wp g t yr;
   consume: model    c = p plag wsum;
   invest:  model    i = p plag klag;
```

```
   labor:   model    w = x xlag yr;
   product: identity x = c + i + g;
   income:  identity y = c + i + g - t;
   profit:  identity p = y - w;
   stock:   identity k = klag + i;
   wage:    identity wsum = w + wp;
run;
```

The preliminary 2SLS results and estimated cross-model covariance matrix are not shown. The 3SLS estimates are shown in Output 36.1.5 through Output 36.1.7. The reduced form estimates are shown in Output 36.1.8 through Output 36.1.11.

Output 36.1.5 3SLS Estimates for Consumption

The SYSLIN Procedure
Three-Stage Least Squares Estimation

System Weighted MSE	5.9342
Degrees of freedom	51
System Weighted R-Square	0.9550

Model	CONSUME
Dependent Variable	c
Label	Consumption

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	Variable Label
Intercept	1	16.44079	1.449925	11.34	<.0001	Intercept
p	1	0.124890	0.120179	1.04	0.3133	Profits
plag	1	0.163144	0.111631	1.46	0.1621	Profits Lagged
wsum	1	0.790081	0.042166	18.74	<.0001	Total Wage Bill

Output 36.1.6 3SLS Estimates for Investments

Model	INVEST
Dependent Variable	i
Label	Taxes

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	Variable Label
Intercept	1	28.17785	7.550853	3.73	0.0017	Intercept
p	1	-0.01308	0.179938	-0.07	0.9429	Profits
plag	1	0.755724	0.169976	4.45	0.0004	Profits Lagged
klag	1	-0.19485	0.036156	-5.39	<.0001	Capital Stock Lagged

Output 36.1.7 3SLS Estimates for Labor

Model	LABOR
Dependent Variable	w
Label	Private Wage Bill

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	Variable Label
Intercept	1	1.797218	1.240203	1.45	0.1655	Intercept
x	1	0.400492	0.035359	11.33	<.0001	Private Production
xlag	1	0.181291	0.037965	4.78	0.0002	Private Product Lagged
yr	1	0.149674	0.031048	4.82	0.0002	YEAR-1931

Output 36.1.8 Reduced Form Estimates

Endogenous Variables

	c	p	w	i	x	wsum	k	y
CONSUME	1	-0.12489	0	0	0	-0.79008	0	0
INVEST	0	0.013079	0	1	0	0	0	0
LABOR	0	0	1	0	-0.40049	0	0	0
PRODUCT	-1	0	0	-1	1	0	0	0
INCOME	-1	0	0	-1	0	0	0	1
PROFIT	0	1	1	0	0	0	0	-1
STOCK	0	0	0	-1	0	0	1	0
WAGE	0	0	-1	0	0	1	0	0

Output 36.1.9 Reduced Form Estimates

Exogenous Variables

	Intercept	plag	klag	xlag	yr	g	t	wp
CONSUME	16.44079	0.163144	0	0	0	0	0	0
INVEST	28.17785	0.755724	-0.19485	0	0	0	0	0
LABOR	1.797218	0	0	0.181291	0.149674	0	0	0
PRODUCT	0	0	0	0	0	1	0	0
INCOME	0	0	0	0	0	1	-1	0
PROFIT	0	0	0	0	0	0	0	0
STOCK	0	0	1	0	0	0	0	0
WAGE	0	0	0	0	0	0	0	1

Output 36.1.10 Reduced Form Estimates

Inverse Endogenous Variables								
	CONSUME	INVEST	LABOR	PRODUCT	INCOME	PROFIT	STOCK	WAGE
c	1.634654	0.634654	1.095657	0.438802	0.195852	0.195852	0	1.291509
p	0.972364	0.972364	-0.34048	-0.13636	1.108721	1.108721	0	0.768246
w	0.649572	0.649572	1.440585	0.576943	0.072629	0.072629	0	0.513215
i	-0.01272	0.987282	0.004453	0.001783	-0.0145	-0.0145	0	-0.01005
x	1.621936	1.621936	1.10011	1.440585	0.181351	0.181351	0	1.281461
wsum	0.649572	0.649572	1.440585	0.576943	0.072629	0.072629	0	1.513215
k	-0.01272	0.987282	0.004453	0.001783	-0.0145	-0.0145	1	-0.01005
y	1.621936	1.621936	1.10011	0.440585	1.181351	0.181351	0	1.281461

Output 36.1.11 Reduced Form Estimates

Reduced Form								
	Intercept	plag	klag	xlag	yr	g	t	wp
c	46.7273	0.746307	-0.12366	0.198633	0.163991	0.634654	-0.19585	1.291509
p	42.77363	0.893474	-0.18946	-0.06173	-0.05096	0.972364	-1.10872	0.768246
w	31.57207	0.596871	-0.12657	0.261165	0.215618	0.649572	-0.07263	0.513215
i	27.6184	0.744038	-0.19237	0.000807	0.000667	-0.01272	0.014501	-0.01005
x	74.3457	1.490345	-0.31603	0.19944	0.164658	1.621936	-0.18135	1.281461
wsum	31.57207	0.596871	-0.12657	0.261165	0.215618	0.649572	-0.07263	1.513215
k	27.6184	0.744038	0.80763	0.000807	0.000667	-0.01272	0.014501	-0.01005
y	74.3457	1.490345	-0.31603	0.19944	0.164658	1.621936	-1.18135	1.281461

Example 36.2: Grunfeld's Model Estimated with SUR

The following example was used by Zellner in his classic 1962 paper on seemingly unrelated regressions. Different stock prices often move in the same direction at a given point in time. The SUR technique might provide more efficient estimates than OLS in this situation.

The following statements read the data. (The prefix GE stands for General Electric and WH stands for Westinghouse.)

```
*---------Zellner's Seemingly Unrelated Technique------------*
| A. Zellner, "An Efficient Method of Estimating Seemingly   |
| Unrelated Regressions and Tests for Aggregation Bias,"     |
| JASA 57(1962) pp.348-364                                   |
|                                                            |
| J.C.G. Boot, "Investment Demand: an Empirical Contribution |
| to the Aggregation Problem," IER 1(1960) pp.3-30.          |
|                                                            |
| Y. Grunfeld, "The Determinants of Corporate Investment,"   |
| Unpublished thesis, Chicago, 1958                          |
*------------------------------------------------------------*;

data grunfeld;
   input year ge_i ge_f ge_c wh_i wh_f wh_c;
   label ge_i = 'Gross Investment, GE'
```

```
         ge_c = 'Capital Stock Lagged, GE'
         ge_f = 'Value of Outstanding Shares Lagged, GE'
         wh_i = 'Gross Investment, WH'
         wh_c = 'Capital Stock Lagged, WH'
         wh_f = 'Value of Outstanding Shares Lagged, WH';
datalines;
1935     33.1      1170.6    97.8      12.93     191.5     1.8

   ... more lines ...
```

The following statements compute the SUR estimates for the Grunfeld model.

```
proc syslin data=grunfeld sur;
   ge:      model ge_i = ge_f ge_c;
   westing: model wh_i = wh_f wh_c;
run;
```

The PROC SYSLIN output is shown in Output 36.2.1 through Output 36.2.5.

Output 36.2.1 PROC SYSLIN Output for SUR

The SYSLIN Procedure
Ordinary Least Squares Estimation

Model	GE
Dependent Variable	ge_i
Label	Gross Investment, GE

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	2	31632.03	15816.02	20.34	<.0001
Error	17	13216.59	777.4463		
Corrected Total	19	44848.62			

Root MSE	27.88272	**R-Square**	0.70531
Dependent Mean	102.29000	**Adj R-Sq**	0.67064
Coeff Var	27.25850		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	Variable Label
Intercept	1	-9.95631	31.37425	-0.32	0.7548	Intercept
ge_f	1	0.026551	0.015566	1.71	0.1063	Value of Outstanding Shares Lagged, GE
ge_c	1	0.151694	0.025704	5.90	<.0001	Capital Stock Lagged, GE

Output 36.2.2 PROC SYSLIN Output for SUR

The SYSLIN Procedure
Ordinary Least Squares Estimation

Model	WESTING
Dependent Variable	wh_i
Label	Gross Investment, WH

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	2	5165.553	2582.776	24.76	<.0001
Error	17	1773.234	104.3079		
Corrected Total	19	6938.787			

Root MSE	10.21312	**R-Square**	0.74445
Dependent Mean	42.89150	**Adj R-Sq**	0.71438
Coeff Var	23.81153		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	Variable Label
Intercept	1	-0.50939	8.015289	-0.06	0.9501	Intercept
wh_f	1	0.052894	0.015707	3.37	0.0037	Value of Outstanding Shares Lagged, WH
wh_c	1	0.092406	0.056099	1.65	0.1179	Capital Stock Lagged, WH

Output 36.2.3 PROC SYSLIN Output for SUR

The SYSLIN Procedure
Seemingly Unrelated Regression Estimation

Cross Model Covariance

	GE	WESTING
GE	777.446	207.587
WESTING	207.587	104.308

Cross Model Correlation

	GE	WESTING
GE	1.00000	0.72896
WESTING	0.72896	1.00000

Cross Model Inverse Correlation

	GE	WESTING
GE	2.13397	-1.55559
WESTING	-1.55559	2.13397

Output 36.2.3 *continued*

Cross Model Inverse Covariance		
	GE	**WESTING**
GE	0.002745	-.005463
WESTING	-.005463	0.020458

Output 36.2.4 PROC SYSLIN Output for SUR

System Weighted MSE	0.9719
Degrees of freedom	34
System Weighted R-Square	0.6284

Model	GE
Dependent Variable	ge_i
Label	Gross Investment, GE

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	Variable Label
Intercept	1	-27.7193	29.32122	-0.95	0.3577	Intercept
ge_f	1	0.038310	0.014415	2.66	0.0166	Value of Outstanding Shares Lagged, GE
ge_c	1	0.139036	0.024986	5.56	<.0001	Capital Stock Lagged, GE

Output 36.2.5 PROC SYSLIN Output for SUR

Model	WESTING
Dependent Variable	wh_i
Label	Gross Investment, WH

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	Variable Label
Intercept	1	-1.25199	7.545217	-0.17	0.8702	Intercept
wh_f	1	0.057630	0.014546	3.96	0.0010	Value of Outstanding Shares Lagged, WH
wh_c	1	0.063978	0.053041	1.21	0.2443	Capital Stock Lagged, WH

Example 36.3: Illustration of ODS Graphics

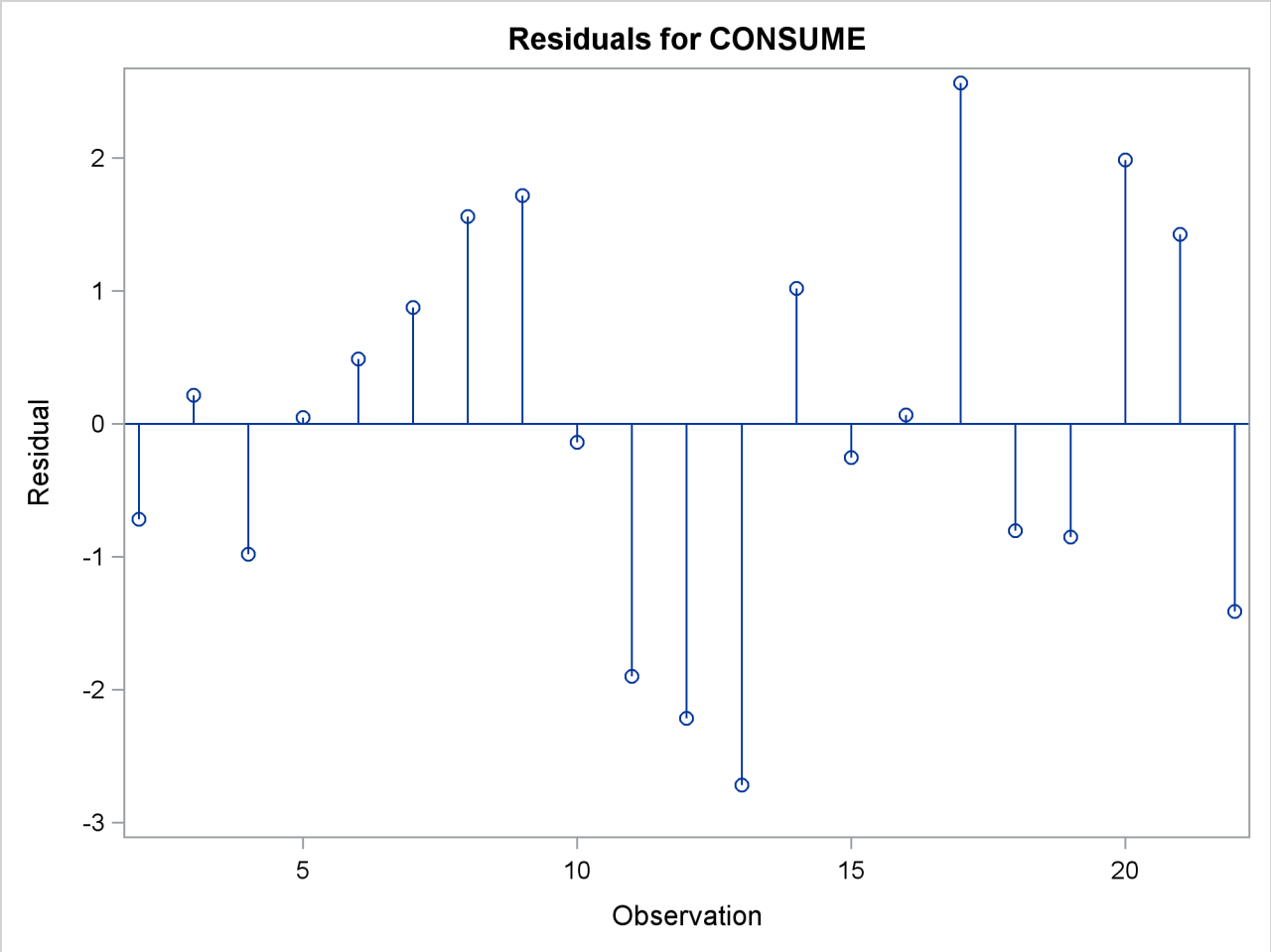
This example illustrates the use of ODS graphics. This is a continuation of the section "Example 36.1: Klein's Model I Estimated with LIML and 3SLS" on page 2590. These graphical displays are requested by specifying the ODS GRAPHICS statement before running PROC SYSLIN. For information about the graphics available in the SYSLIN procedure, see the section "ODS Graphics" on page 2590.

The following statements show how to generate ODS graphics plots with the SYSLIN procedure. The plots of residuals for each one of the equations in the model are displayed in Figure 36.3.1 through Figure 36.3.3.

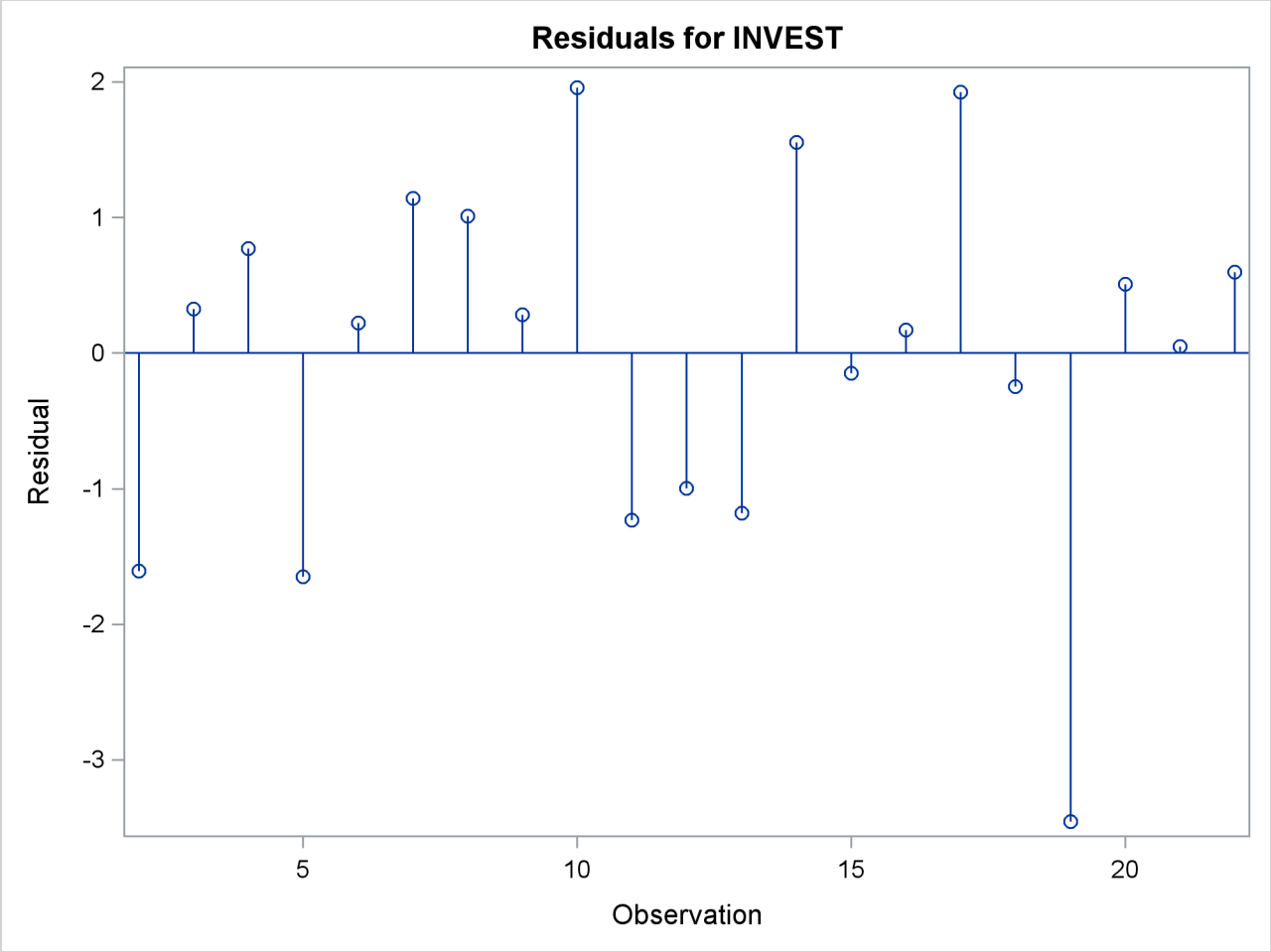
```
*---------------------------Klein's Model I----------------------------*
| By L.R. Klein, Economic Fluctuations in the United States, 1921-1941 |
| (1950), NY: John Wiley.   A macro-economic model of the U.S. with    |
| three behavioral equations, and several identities. See Theil, p.456.|
*--------------------------------------------------------------------------*;
data klein;
input year c p w i x wp g t k wsum;
   date=mdy(1,1,year);
   format date monyy.;
   y   =c+i+g-t;
   yr  =year-1931;
   klag=lag(k);
   plag=lag(p);
   xlag=lag(x);
   label year='Year'
         date='Date'
         c   ='Consumption'
         p   ='Profits'
         w   ='Private Wage Bill'
         i   ='Investment'
         k   ='Capital Stock'
         y   ='National Income'
         x   ='Private Production'
         wsum='Total Wage Bill'
         wp  ='Govt Wage Bill'
         g   ='Govt Demand'
         i   ='Taxes'
         klag='Capital Stock Lagged'
         plag='Profits Lagged'
         xlag='Private Product Lagged'
         yr  ='YEAR-1931';
datalines;
1920     .  12.7     .    .  44.9    .     .     .  182.8     .
1921  41.9  12.4  25.5 -0.2  45.6  2.7   3.9   7.7  182.6  28.2
1922  45.0  16.9  29.3  1.9  50.1  2.9   3.2   3.9  184.5  32.2
1923  49.2  18.4  34.1  5.2  57.2  2.9   2.8   4.7  189.7  37.0
1924  50.6  19.4  33.9  3.0  57.1  3.1   3.5   3.8  192.7  37.0
1925  52.6  20.1  35.4  5.1  61.0  3.2   3.3   5.5  197.8  38.6
1926  55.1  19.6  37.4  5.6  64.0  3.3   3.3   7.0  203.4  40.7
1927  56.2  19.8  37.9  4.2  64.4  3.6   4.0   6.7  207.6  41.5
1928  57.3  21.1  39.2  3.0  64.5  3.7   4.2   4.2  210.6  42.9
1929  57.8  21.7  41.3  5.1  67.0  4.0   4.1   4.0  215.7  45.3
1930  55.0  15.6  37.9  1.0  61.2  4.2   5.2   7.7  216.7  42.1

   ... more lines ...
```

```
ods graphics on;

proc syslin data=klein outest=b liml plots(unpack only)=residual ;
   endogenous c p w i x wsum k y;
   instruments klag plag xlag wp g t yr;
   consume: model c = p plag  wsum;
   invest:  model i = p plag  klag;
   labor:   model w = x xlag  yr;
run;
```

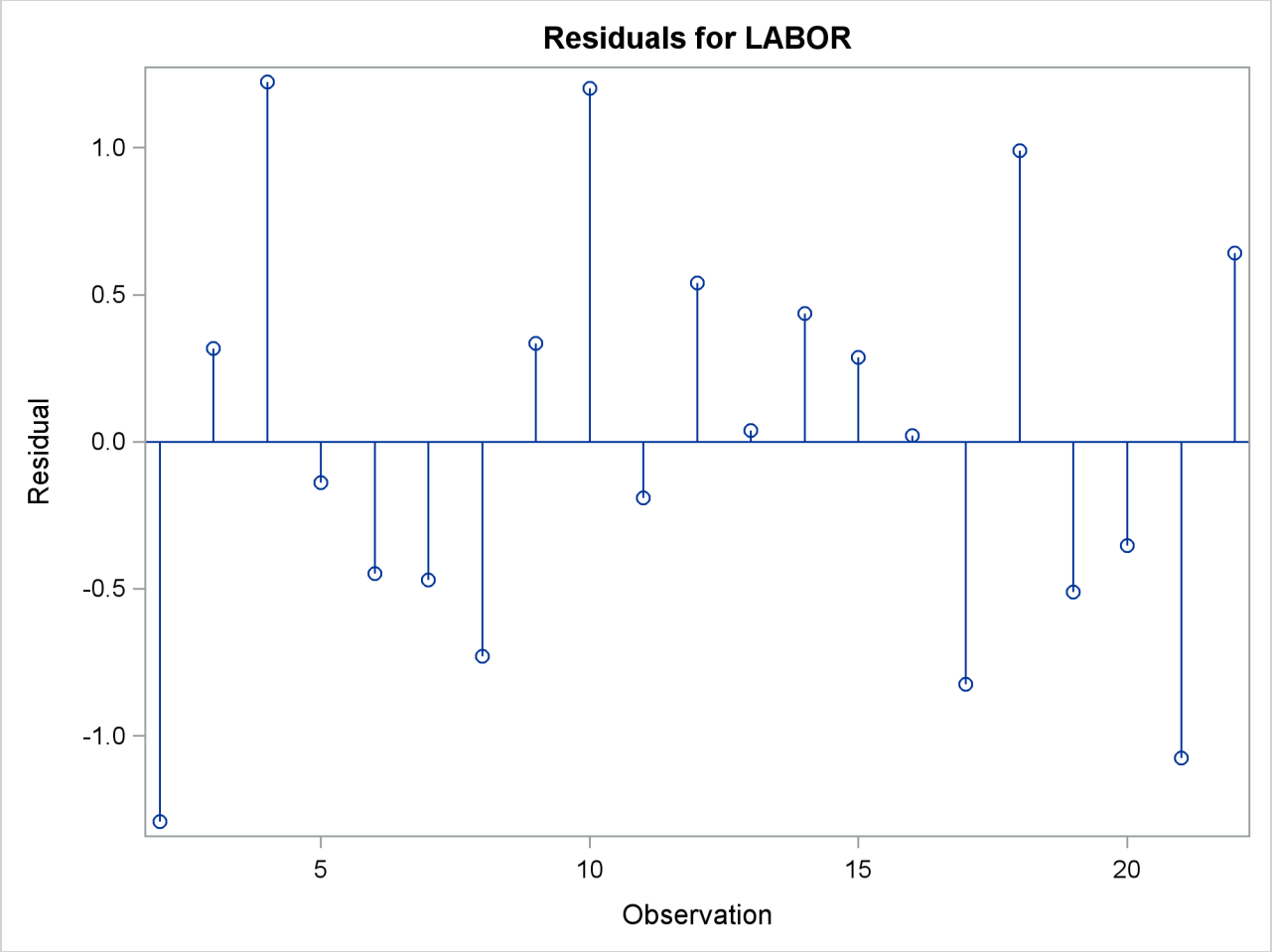
Output 36.3.1 Residuals Diagnostic Plots for Consumption



Output 36.3.2 Residuals Diagnostic Plots for Investments



Output 36.3.3 Residuals Diagnostic Plots for Labor



References

Basmann, R. L. (1960). "On Finite Sample Distributions of Generalized Classical Linear Identifiability Test Statistics." *Journal of the American Statistical Association* 55:650–659.

Fuller, W. A. (1977). "Some Properties of a Modification of the Limited Information Estimator." *Econometrica* 45:939–952.

Hausman, J. A. (1975). "An Instrumental Variable Approach to Full Information Estimators for Linear and Certain Nonlinear Econometric Models." *Econometrica* 43:727–738.

Johnston, J. (1984). *Econometric Methods*. 3rd ed. New York: McGraw-Hill.

Judge, G. G., Griffiths, W. E., Hill, R. C., Lütkepohl, H., and Lee, T.-C. (1985). *The Theory and Practice of Econometrics*. 2nd ed. New York: John Wiley & Sons.

Maddala, G. S. (1977). *Econometrics*. New York: McGraw-Hill.

Park, S.-B. (1982). "Some Sampling Properties of Minimum Expected Loss (MELO) Estimators of Structural Coefficients." *Journal of Econometrics* 18:295–311.

Pindyck, R. S., and Rubinfeld, D. L. (1981). *Econometric Models and Econometric Forecasts*. 2nd ed. New York: McGraw-Hill.

Pringle, R. M., and Rayner, A. A. (1971). *Generalized Inverse Matrices with Applications to Statistics*. New York: Hafner Publishing.

Rao, P. (1974). "Specification Bias in Seemingly Unrelated Regressions." In *Econometrics and Economic Theory: Essays in Honour of Jan Tinbergen*, edited by W. Sellekaerts, 101–113. White Plains, NY: International Arts and Sciences Press.

Savin, N. E., and White, K. J. (1978). "Testing for Autocorrelation with Missing Observations." *Econometrica* 46:59–67.

Theil, H. (1971). *Principles of Econometrics*. New York: John Wiley & Sons.

Zellner, A. (1962). "An Efficient Method of Estimating Seemingly Unrelated Regressions and Tests for Aggregation Bias." *Journal of the American Statistical Association* 57:348–368.

Zellner, A. (1978). "Estimation of Functions of Population Means and Regression Coefficients: A Minimum Expected Loss (MELO) Approach." *Journal of Econometrics* 8:127–158.

Zellner, A., and Park, S.-B. (1979). "Minimum Expected Loss (MELO) Estimators for Functions of Parameters and Structural Coefficients of Econometric Models." *Journal of the American Statistical Association* 74:185–193.

Chapter 37
The TIMEDATA Procedure

Contents

Overview: TIMEDATA Procedure **2606**
Getting Started: TIMEDATA Procedure **2606**
Syntax: TIMEDATA Procedure **2608**
Functional Summary 2608
PROC TIMEDATA Statement 2610
BY Statement 2612
FCMPOPT Statement 2612
ID Statement 2613
OUTARRAYS Statements 2615
OUTSCALARS Statements 2615
VAR Statements 2616
Program Statements 2617
Details: TIMEDATA Procedure **2617**
Accumulation 2617
Missing Value Interpretation 2619
Time Series Transformation 2619
Time Series Differencing 2620
Summary Statistics 2620
Programming Statements 2620
Predefined Symbols 2620
Auxiliary Data Sets 2621
Data Set Output 2625
OUT= Data Set 2625
OUTARRAY= Data Set 2625
OUTPROCINFO= Data Set 2626
OUTSCALAR= Data Set 2626
OUTSUM= Data Set 2626
STATUS Variable Values 2627
Printed Output 2627
ODS Table Names 2628
ODS Graphics Names 2628
Examples: TIMEDATA Procedure **2629**
Example 37.1: Accumulating Transactional Data into Time Series Data 2629
Example 37.2: Using User-Defined Functions and Subroutines 2630
Example 37.3: Using Auxiliary Data Sets with PROC TIMEDATA 2631
References **2633**

Overview: TIMEDATA Procedure

The TIMEDATA procedure analyzes time-stamped transactional data with respect to time and accumulates the data into a time series format.

After the transactional data are accumulated to form a time series and any missing values are interpreted, the accumulated time series can be functionally transformed using log, square root, logistic, or Box-Cox transformations. The time series can be further transformed using simple differencing, seasonal differencing, or both. After functional and difference transformations have been applied, the accumulated and transformed time series can be stored in an output data set. This working time series can then be analyzed further using various time series analysis techniques provided by this procedure or other SAS/ETS procedures.

The TIMEDATA procedure is very similar to the TIMESERIES procedure. However, unlike the TIMESERIES procedure (which enables you to perform a variety of standard time series analysis techniques), the TIMEDATA procedure enables you to define your own analyses using SAS programming statements.

By default, the TIMEDATA procedure provides no further analyses.

The TIMEDATA procedure forms time series vectors and then provides these vectors as SAS data arrays for subsequent processing by your SAS programming statements. Your programming statements are processed independently for each BY group. The TIMEDATA procedure is like the SAS DATA step for time series data. The SAS DATA step processes data by each row; the TIMEDATA procedure processes time series vectors.

As part of your SAS programming statements, you can include user-defined functions and subroutines created by the FCMP procedure. Additionally, you can use the RUN_MACRO subroutine provided by the FCMP procedure to submit SAS statements that use any SAS procedures.

All results of the transactional or time series analysis can be stored in output data sets or printed using the Output Delivery System (ODS).

Getting Started: TIMEDATA Procedure

This section outlines the use of the TIMEDATA procedure and gives a cursory description of some of the analysis techniques that you can perform on time-stamped transactional data.

Given an input data set that contains numerous transaction variables recorded over time at no specific frequency, the TIMEDATA procedure can form time series as follows:

```
PROC TIMEDATA DATA=<input-data-set>
                 OUT=<output-data-set>;
   BY <list-of-BY-variables>;
   ID <time-ID-variable> INTERVAL=<frequency>
                         ACCUMULATE=<statistic>;
   VAR <time-series-variables>;
   /* programming statements */
RUN;
```

The TIMEDATA procedure forms time series from the input time-stamped transactional data. It can provide results in output data sets or in other output formats by using the Output Delivery System (ODS).

Time-stamped transactional data are recorded at no fixed interval. Analysts often want to use time series analysis techniques that require fixed-time intervals. Therefore, the transactional data must be accumulated to form a fixed-interval time series, such as daily, weekly, or monthly.

Suppose that a bank wants to analyze the transactions that are associated with each of its customers over time. Further, suppose that the data set Work.Transactions contains four variables that are related to these transactions: Customer, Date, Withdrawals, and Deposits. The following examples illustrate possible ways to analyze these transactions by using the TIMEDATA procedure.

The following TIMEDATA procedure statements accumulate the time-stamped transactional data to form a daily time series based on the accumulated daily totals of each type of transaction (Withdrawals and Deposits):

```
proc timedata data=transactions
              out=timeseries
              outarray=arrays;
   by customer;
   id date interval=day accumulate=total;
   var withdrawals deposits;
   outarrays balance;

   balance[1] = deposits[1] - withdrawals[1];
   do t = 2 to _LENGTH_;
      balance[t] = balance[t-1] + (deposits[t] - withdrawals[t]);
   end;

run;
```

The OUT=TIMESERIES option specifies that the resulting time series data for each customer are to be stored in the data set Work.Transactions. The OUTARRAY=ARRAYS option specifies that the resulting time series data along with a newly created variable, Balance, are to be stored in the data set Work.Arrays. The INTERVAL=DAY option specifies that the transactions are to be accumulated on a daily basis. The ACCUMULATE=TOTAL option specifies that the sum of the transactions is to be calculated. After the transactional data are accumulated into a time series format, many of the procedures provided with SAS/ETS software can be used to analyze the resulting time series data.

For example, the following statements use the ARIMA procedure to model and forecast each customer's balance data by using an ARIMA$(1,0,0)(0,1,0)_s$ model (where the number of seasons is s=7 days in a week):

```
proc arima data=arrays;
   by customer;
   identify var=balance(7) noprint;
   estimate p=(1) outest=estimates noprint;
   forecast id=date interval=day out=forecasts;
quit;
```

The OUTEST=ESTIMATES data set contains the parameter estimates of the model specified. The OUT=FORECASTS data set contains forecasts based on the model specified. See the SAS/ETS ARIMA procedure for more detail.

By default, the TIMEDATA procedure produces no printed output.

Syntax: TIMEDATA Procedure

The following statements are available in the TIMEDATA Procedure:

PROC TIMEDATA *options* ;
BY *variables* ;
ID *variable* **INTERVAL**= *interval-option* ;
FCMPOPT *options* ;
OUTARRAYS *array-name-list* ;
OUTSCALARS *scalar-name-list* ;
VAR *variable-list / options* ;
Programming Statements ;

Functional Summary

Table 37.1 summarizes the statements and options that control the TIMEDATA procedure.

Table 37.1 TIMEDATA Functional Summary

Description	Statement	Option
Statements		
Specifies BY-group processing	BY	
Specifies variables to analyze	VAR	
Specifies the time ID variable	ID	
Specifies the FCMP options	FCMPOPT	
Specifies the arrays to output	OUTARRAYS	
Specifies the scalars to output	OUTSCALARS	
Data Set Options		
Specifies the auxiliary input data sets	PROC TIMEDATA	AUXDATA=
Specifies the input data set	PROC TIMEDATA	DATA=
Specifies the output data set	PROC TIMEDATA	OUT=
Specifies the array output data set	PROC TIMEDATA	OUTARRAY=
Specifies the run status data set	PROC TIMEDATA	OUTPROCINFO=
Specifies the scalar output data set	PROC TIMEDATA	OUTSCALAR=
Specifies the summary statistics output data set	PROC TIMEDATA	OUTSUM=

Description	Statement	Option
User-Defined Functions and Subroutine Options		
Specifies FCMP quiet mode	FCMPOPT	QUIET=
Specifies FCMP trace mode	FCMPOPT	TRACE=
Accumulation and Seasonality Options		
Specifies the accumulation frequency	ID	INTERVAL=
Specifies the length of seasonal cycle	PROC TIMEDATA	SEASONALITY=
Specifies the type of life-cycle indexing	PROC TIMEDATA	CYCLETYPE=
Specifies the interval alignment	ID	ALIGN=
Specifies that time ID variable values not be sorted	ID	NOTSORTED
Specifies the starting time ID value	ID	START=
Specifies the ending time ID value	ID	END=
Specifies the accumulation statistic	ID, VAR	ACCUMULATE=
Specifies missing value interpretation	ID, VAR	SETMISSING=
Specifies the zero value interpretation	ID, VAR	ZEROMISS=
Time Series Transformation Options		
Specifies simple differencing	VAR	DIF=
Specifies seasonal differencing	VAR	SDIF=
Specifies transformation	VAR	TRANSFORM=
Printing Control Options		
Specifies the time ID format	ID	FORMAT=
Specifies which output to print	PROC TIMEDATA	PRINT=
Miscellaneous Options		
Specifies the forecast horizon or lead used to extend the data set	PROC TIMEDATA	LEAD=
Limits error and warning messages	PROC TIMEDATA	MAXERROR=
ODS Graphics Options		
Specifies the variable and array graphical output	PROC TIMEDATA	PLOTS=

PROC TIMEDATA Statement

PROC TIMEDATA *options* **;**

The following *options* can be used in the PROC TIMEDATA statement:

AUXDATA=*SAS-data-set*
names a SAS data set that contains auxiliary input data for the procedure to use for supplying time series variables. See section "Auxiliary Data Sets" on page 2621 for more information.

CYCLETYPE=*option*
specifies the indexing of each time series with respect to life-cycle. By default, CYCLETYPE=BOL.

The following CYCLETYPE= *options* are available:

BOL indexes the time series by the beginning of life. The first time value is 1. The following values are incremented by 1.

MOL indexes the time series by the middle of life. The middle time value is zero. The preceding values are decremented by 1. The following values are incremented by 1.

EOL indexes the time series by the end of life. The last time value is 1. The preceding values are incremented by 1.

The CYCLETYPE= option specifies the indexing of the _CYCLE_ variable contained in the OUTARRAY= data set and the predefined array _CYCLE_.

DATA=*SAS-data-set*
names the SAS data set that contains the input data from which the procedure creates the time series. If the DATA= option is not specified, the most recently created SAS data set is used.

LEAD=*n*
specifies the number of periods ahead to forecast (forecast lead or horizon) used to extend the data set. The default is LEAD=0.

The LEAD= value is relative to the last observation in the input data set and not to the last nonmissing observation of a particular series.

MAXERROR=*number*
limits the number of warning and error messages that are produced during the execution of the procedure to the specified *number*. The default is MAXERRORS=50. This option is particularly useful in BY-group processing where it can be used to suppress recurring messages.

OUT=*SAS-data-set*
names the output data set to contain the time series variables specified in the subsequent VAR statements. If BY variables are specified, they are also included in the OUT= data set. If an ID variable is specified, it is also included in the OUT= data set. The values are accumulated based on the INTERVAL= option or the ACCUMULATE= option or both in the ID statement. The OUT= data set is particularly useful when you want to further analyze, model, or forecast the resulting time series with other SAS/ETS procedures.

OUTARRAY=*SAS-data-set*
names the output data set to contain the time series vectors listed in the VAR and OUTARRAYS statements.

The OUTARRAY= data set contains the variables specified in the BY, ID, and VAR statements in addition to the arrays that are specified in the OUTARRAYS statements.

OUTSCALAR=*SAS-data-set*
names the output data set to contain the scalar names listed in the OUTSCALARS statements.

The OUTSCALAR= data set contains the variables specified in the BY statement and the scalars that are specified in the OUTSCALARS statements.

OUTPROCINFO=*SAS-data-set*
names the output data set to summarize information in the SAS log, specifically the number of notes, errors, and warnings and the number of series processed, analyses requested, and analyses failed.

OUTSUM=*SAS-data-set*
names the output data set to contain the descriptive statistics. The descriptive statistics are based on the accumulated time series when the ACCUMULATE= option, the SETMISSING= option, or both are specified in the ID or VAR statements. The OUTSUM= data set is particularly useful when analyzing large numbers of series and a summary of the results is needed.

PLOTS=*option* | **(** *options* **)**
specifies the univariate graphical output desired. By default, the TIMEDATA procedure produces no graphical output. The PLOTS= option produces results that are similar to the data sets shown in parentheses next to the following *options*:

ARRAYS	plots the time series (OUT= data set).
ALL	same as PLOTS=(ARRAYS).

For example, PLOTS=ARRAYS plots the time series. The PLOTS= option produces graphical output for these results by using the Output Delivery System (ODS).

PRINT=*option* | **(** *options* **)**
specifies the printed output desired. By default, the TIMEDATA procedure produces no printed output. The PRINT= option produces results that are similar to the data sets shown in parentheses next to the following *options*:

ARRAYS	prints the arrays table (OUTARRAY= data set).
SCALARS	prints the scalars table (OUTSCALAR= data set).
SUMMARY	prints the descriptive statistics table for all time series (OUTSUM= data set).
ALL	same as PRINT=(ARRAYS SCALARS SUMMARY).

For example, PRINT=SCALARS prints the scalars specified in the OUTSCALARS statement. The PRINT= option produces printed output for these results by using the Output Delivery System (ODS).

SEASONALITY=*number*

specifies the length of the seasonal cycle. For example, SEASONALITY=3 means that every group of three time periods forms a seasonal cycle. By default, the length of the seasonal cycle is 1 (no seasonality) or the length implied by the INTERVAL= option specified in the ID statement. For example, INTERVAL=MONTH implies that the length of the seasonal cycle is 12.

BY Statement

You can include a BY statement with PROC TIMEDATA to obtain separate dummy variable definitions for groups of observations defined by the BY variables.

When a BY statement appears, the procedure expects the input data set to be sorted in order of the BY variables. If your input data set is not sorted in ascending order, use one of the following alternatives:

- Sort the data by using the SORT procedure with a similar BY statement.
- Specify the option NOTSORTED or DESCENDING in the BY statement for the TIMEDATA procedure. The NOTSORTED option does not mean that the data are unsorted but rather that the data are arranged in groups (according to values of the BY variables) and that these groups are not necessarily in alphabetical or increasing numeric order.
- Create an index on the BY variables by using the DATASETS procedure.

For more information about the BY statement, see *SAS Language Reference: Concepts*. For more information about the DATASETS procedure, see the discussion in the *Base SAS Procedures Guide.*

FCMPOPT Statement

FCMPOPT *options* **;**

The FCMPOPT statement specifies the following *options* that are related to user-defined functions and subroutines:

QUIET=ON | OFF

specifies whether the nonfatal errors and warnings that are generated by the user-defined SAS language functions and subroutines are printed to the log. Nonfatal errors are usually associated with operations with missing values. The default is QUIET=ON.

TRACE=ON | OFF

specifies whether the user-defined SAS language functions and subroutines tracings are printed to the log. Tracings are the results of every operation executed. This option is generally used for debugging. The default is TRACE=OFF.

ID Statement

ID *variable* **INTERVAL=***interval* < *options* > **;**

The ID statement names a numeric variable that identifies observations in the input and output data sets. The ID variable's values are assumed to be SAS date or datetime values. In addition, the ID statement specifies the (desired) frequency associated with the time series. The ID statement *options* also specify how the observations are accumulated and how the time ID values are aligned to form the time series. The information specified affects all variables listed in subsequent VAR statements. If the ID statement is specified, the INTERVAL= must also be used. If an ID statement is not specified, the observation number, with respect to the BY group, is used as the time ID.

You can specify following *options* in the ID statement:

ACCUMULATE= *option*

specifies how the data set observations are to be accumulated within each time period. The frequency (width of each time interval) is specified by the INTERVAL= option. The ID variable contains the time ID values. Each time ID variable value corresponds to a specific time period. The accumulated values form the time series, which is used in subsequent analysis.

The ACCUMULATE= option is useful when there are zero or more than one input observations that coincide with a particular time period (for example, time-stamped transactional data). The EXPAND procedure offers additional frequency conversions and transformations that can also be useful in creating a time series.

The following *options* determine how the observations are accumulated within each time period based on the ID variable and the frequency specified by the INTERVAL= option:

NONE	No accumulation occurs; the ID variable values must be equally spaced with respect to the frequency. This is the default. Observations are accumulated based on the:
TOTAL	total sum of their values.
AVERAGE | AVG	average of their values.
MINIMUM | MIN	minimum of their values.
MEDIAN | MED	median of their values.
MAXIMUM | MAX	maximum of their values.
N	number of nonmissing observations.
NMISS	number of missing observations.
NOBS	number of observations.
FIRST	first of their values.
LAST	last of their values.
STDDEV |STD	standard deviation of their values.
CSS	corrected sum of squares of their values.
USS	uncorrected sum of squares of their values.

If the ACCUMULATE= option is specified, the SETMISSING= option is useful for specifying how accumulated missing values are to be treated. If missing values should be interpreted as zero, then SETMISSING=0 should be used. The section "Details: TIMEDATA Procedure" on page 2617 describes accumulation in greater detail.

ALIGN=*option*

controls the alignment of SAS dates that are used to identify output observations. The ALIGN= option accepts the following values: BEGINNING | BEG | B, MIDDLE | MID | M, and ENDING | END | E. BEGINNING is the default.

END=*option*

specifies a SAS date or datetime value that represents the end of the data. If the last time ID variable value is less than the END= value, the series is extended with missing values. If the last time ID variable value is greater than the END= value, the series is truncated. For example, END="&sysdate"D uses the automatic macro variable SYSDATE to extend or truncate the series to the current date. You can specify the START= and END= options to ensure that the data that are associated within each BY group contain the same number of observations.

FORMAT=*format*

specifies the SAS format for the time ID values. If the FORMAT= option is not specified, the default format is inferred from the INTERVAL= option.

INTERVAL=*interval*

specifies the frequency of the accumulated time series. For example, if the input data set consists of quarterly observations, then INTERVAL=QTR should be used. If the SEASONALITY= option is not specified in the PROC TIMEDATA statement, the length of the seasonal cycle is implied from the INTERVAL= option. For example, INTERVAL=QTR implies a seasonal cycle of length 4. If the ACCUMULATE= option is also specified, the INTERVAL= option determines the time periods for the accumulation of observations. The INTERVAL= option is required and must be specified in the ID statement.

NOTSORTED

specifies that the time ID values not be in sorted order. The TIMEDATA procedure sorts the data with respect to the time ID prior to analysis.

SETMISSING=*option* | *number*

specifies how missing values (either actual or accumulated) are to be interpreted in the accumulated time series. If a *number* is specified, missing values are set to the *number*. If a missing value indicates an unknown value, specify SETMISSING=MISSING. If a missing value indicates a zero value, specify SETMISSING=0. You would typically use SETMISSING=0 for transactional data because no recorded data usually implies no activity. You can use the following *options* to determine how missing values are assigned. Missing values are set to:

MISSING	a missing value. This is the default.
AVERAGE | AVG	the accumulated average value.
MINIMUM | MIN	the accumulated minimum value.
MEDIAN | MED	the accumulated median value.
MAXIMUM | MAX	the accumulated maximum value.

FIRST	the accumulated first nonmissing value.
LAST	the accumulated last nonmissing value.
PREVIOUS | PREV	the previous period's accumulated nonmissing value. Missing values at the beginning of the accumulated series remain missing.
NEXT	the next period's accumulated nonmissing value. Missing values at the end of the accumulated series remain missing.

START=*option*

specifies a SAS date or datetime value that represents the beginning of the data. If the first time ID variable value is greater than the START= value, missing values are added at the beginning of the series. If the first time ID variable value is less than the START= value, the series is truncated. You can specify the START= and END= options to ensure that data associated with each BY group contain the same number of observations.

ZEROMISS=*option*

specifies how beginning and ending zero values (either actual or accumulated) are interpreted in the accumulated time series. The following *options* can also be used to determine how beginning and ending zero values are assigned:

NONE	Beginning and ending zeros are unchanged. This is the default.
LEFT	Beginning zeros are set to missing.
RIGHT	Ending zeros are set to missing.
BOTH	Both beginning and ending zeros are set to missing.

If the accumulated series is all missing or zero, the series is not changed.

OUTARRAYS Statements

OUTARRAYS *array-name-list* **;**

Each array name listed in an OUTARRAYS statement specifies a numeric output array variable to be stored in the OUTARRAY= data set. You can include any number of OUTARRAYS statements.

Your programming statements can create and use any number of arrays. Only arrays that are listed in the OUTARRAYS statement are predefined and included in your output. The arrays are initialized to missing values.

OUTSCALARS Statements

OUTSCALARS *scalar-name-list* **;**

Each scalar name listed in an OUTSCALARS statement specifies a numeric output scalar variable to be stored in the OUTSCALAR= data set. You can include any number of OUTSCALARS statements.

Your programming statements can create and use any number of scalars. Only scalars that are listed in the OUTSCALARS statement are predefined and included in your output. The scalars are initialized to missing values.

VAR Statements

VAR *variable-list* < */ options* > **;**

The VAR statements list the numeric variables in the DATA= data set whose values are to be accumulated to form the time series.

An input data set variable can be specified in only one VAR statement. You can specify any number of VAR statements. You can also specify the following *options* in the VAR statements:

ACCUMULATE=*option*

specifies how the data set observations are to be accumulated within each time period for the variables listed in the VAR statement. If the ACCUMULATE= option is not specified in the VAR statement, accumulation is determined by the ACCUMULATE= option in the ID statement. See the ACCUMULATE= option in the ID statement for more details.

DIF=*(numlist)*

specifies the differencing to be applied to the accumulated time series. The list of differencing orders must be separated by spaces or commas. For example, DIF=(1,3) specifies first then third order differencing. Differencing is applied after time series transformation. The TRANSFORM= option is applied before the DIF= option.

SDIF=*(numlist)*

specifies the seasonal differencing to be applied to the accumulated time series. The list of seasonal differencing orders must be separated by spaces or commas. For example, SDIF=(1,3) specifies first then third order seasonal differencing. Differencing is applied after time series transformation. The TRANSFORM= option is applied before the SDIF= option.

SETMISS=*option* | *number*

SETMISSING= *option* | *number*

specifies how missing values (either actual or accumulated) are to be interpreted in the accumulated time series for variables listed in the VAR statement. If the SETMISSING= option is not specified in the VAR statement, missing values are set based on the SETMISSING= option in the ID statement. See the SETMISSING= option in the ID statement for more details.

TRANSFORM=*option*

specifies the time series transformation to be applied to the accumulated time series. You can specify the following transformation *options*:

NONE	No transformation is applied. This option is the default.
LOG	Logarithmic transformation
SQRT	Square-root transformation
LOGISTIC	Logistic transformation
BOXCOX(*n*)	Box-Cox transformation with parameter number where *n* is between –5 and 5

When the TRANSFORM= option is specified, the time series must be strictly positive.

ZEROMISS=*option*

specifies how beginning and ending zero values (either actual or accumulated) are interpreted in the accumulated time series or ordered sequence for variables listed in the VAR statement. If the ZEROMISS= option is not specified in the VAR statement, beginning and ending zero values are set based on the ZEROMISS= option of the ID statement. If the ZEROMISS= option is not specified in the ID statement or the VAR statement, no zero value interpretation is performed. See the ID statement ZEROMISS= option for more details.

Program Statements

Program Statements ;

You can use most of the programming statements that are allowed in the SAS DATA step.

Details: TIMEDATA Procedure

The TIMEDATA procedure forms time series data from transactional data. The accumulated time series can then be processed using SAS programming statements. The resulting time series can then be analyzed using time series techniques. The data are analyzed using the following steps (the relevant option is listed to the left):

1.	accumulation	ACCUMULATE= option in the ID or VAR statement
2.	missing value interpretation	SETMISSING= option in the ID or VAR statement
3.	time series transformation	TRANSFORM= option in the VAR statement
4.	time series differencing	DIF= and SDIF= options in the VAR statement
5.	program execution	SAS programming statements
6.	descriptive statistics	OUTSUM= option

Accumulation

If the ACCUMULATE= option in the ID or VAR statement is specified, data set observations are accumulated within each time period. The frequency (width of each time interval) is specified by the INTERVAL= option in the ID statement. The ID variable contains the time ID values. Each time ID value corresponds to a specific time period. Accumulation is useful when the input data set contains transactional data, whose observations are not spaced with respect to any particular time interval. The accumulated values form the time series, which is used in subsequent analyses.

For example, suppose a data set contains the following observations:

```
19MAR1999    10
19MAR1999    30
11MAY1999    50
12MAY1999    20
23MAY1999    20
```

If the INTERVAL=MONTH is specified, all of the preceding observations fall within a three-month period of time between March 1999 and May 1999. The observations are accumulated within each time period as follows:

If the ACCUMULATE=NONE option is specified, an error is generated because the ID variable values are not equally spaced with respect to the specified frequency (MONTH).

If the ACCUMULATE=TOTAL option is specified, the resulting time series is:

```
O1MAR1999    40
O1APR1999    .
O1MAY1999    90
```

If the ACCUMULATE=AVERAGE option is specified, the resulting time series is:

```
O1MAR1999    20
O1APR1999    .
O1MAY1999    30
```

If the ACCUMULATE=MINIMUM option is specified, the resulting time series is:

```
O1MAR1999    10
O1APR1999    .
O1MAY1999    20
```

If the ACCUMULATE=MEDIAN option is specified, the resulting time series is:

```
O1MAR1999    20
01APR1999    .
O1MAY1999    20
```

If the ACCUMULATE=MAXIMUM option is specified, the resulting time series is:

```
O1MAR1999    30
O1APR1999    .
O1MAY1999    50
```

If the ACCUMULATE=FIRST option is specified, the resulting time series is:

```
O1MAR1999    10
O1APR1999    .
O1MAY1999    50
```

If the ACCUMULATE=LAST option is specified, the resulting time series is:

```
O1MAR1999    30
O1APR1999    .
O1MAY1999    20
```

If the ACCUMULATE=STDDEV option is specified, the resulting time series is:

```
O1MAR1999    14.14
O1APR1999    .
O1MAY1999    17.32
```

As you can see from the preceding examples, the accumulated time series can have missing values even though the data set observations contain no missing values.

Missing Value Interpretation

Sometimes missing values should be interpreted as unknown values. But sometimes missing values are known, such as when missing values are created from accumulation and no observations should be interpreted as no value—that is, zero. In the former case, the SETMISSING= option can be used to interpret how missing values are treated. Specify SETMISSING=0 when missing observations are to be treated as no (zero) values. In other cases, missing values should be interpreted as global values, such as minimum or maximum values of the accumulated series. The accumulated and interpreted time series is used in subsequent analyses.

Time Series Transformation

Four transformations are available for strictly positive series only. Let $y_t > 0$ be the original time series, and let w_t be the transformed series. The transformations are defined as follows:

Log — is the logarithmic transformation.

$$w_t = \ln(y_t)$$

Logistic — is the logistic transformation.

$$w_t = \ln(c y_t / (1 - c y_t))$$

where the scaling factor c is

$$c = (1 - 10^{-6}) 10^{-\text{ceil}(\log_{10}(max(y_t)))}$$

and $\text{ceil}(x)$ is the smallest integer greater than or equal to x.

Square root — is the square root transformation.

$$w_t = \sqrt{y_t}$$

Box Cox is the Box-Cox transformation.

$$w_t = \begin{cases} \frac{y_t^{\lambda}-1}{\lambda}, & \lambda \neq 0 \\ \ln(y_t), & \lambda = 0 \end{cases}$$

More complex time series transformations can be performed by using the SAS/ETS EXPAND procedure.

Time Series Differencing

After you optionally transform the series, you can simply or seasonally difference the accumulated series by using the VAR statement DIF= and SDIF= options. For example, suppose y_t is a monthly time series. The following examples of the DIF= and SDIF= options demonstrate how to simply and seasonally difference the time series.

```
dif=(1) sdif=(1)
dif=(1,12)
```

Additionally, when y_t is strictly positive and the TRANSFORM=, DIF=, and SDIF= options are combined in the VAR statements, the transformation operation is performed before the differencing operations.

Summary Statistics

You can compute summary statistics from the working series by specifying the OUTSUM= option or PRINT=SUMMARY.

Programming Statements

You can typically use most of the SAS programming statements and SAS functions that you can use in a DATA step for defining the FCMP functions and subroutines. However, there are a few differences in the capabilities of the DATA step and the FCMP procedure. Refer to the documentation of the FCMP procedure to learn more.

All variables listed in the ID and VAR statements are assigned as predefined arrays for subsequent processing. Additionally, all of the array names listed in the OUTARRAYS statements and all of the scalars names listed in the OUTSCALARS statements are assigned as predefined symbols for subsequent processing.

Predefined Symbols

In addition to both the predefined arrays listed in the OUTARRAYS statements and also the predefined scalars listed in the OUTSCALARS statements, the TIMEDATA procedure creates the following predefined symbols for use in the program statements:

Predefined Scalar Values

FORMAT	time format either implied by the INTERVAL= option or specified by the FORMAT= option in the ID statement
INTERVAL	time interval specified by the INTERVAL= option in the ID statement
LEAD	forecast horizon or lead specified by the LEAD= option in the PROC TIMEDATA statement
LENGTH	length of the time series associated with the current BY group
SERIES	series index or BY group counter
SEASONALITY	length of the seasonal cycle specified by the SEASONALITY= option PROC TIMEDATA statement or implied by the INTERVAL= option in the ID statement

Predefined Array Values

TIMEID	time ID values
SEASON	season index values
CYCLE	life-cycle index values

Auxiliary Data Sets

Auxiliary data set support enables the TIMEDATA procedure to use auxiliary data sets to contribute input variables to the run of the procedure step. This functionality creates a virtual data source that enables some of the input variables to physically reside in different data sets with some defined in the primary data set defined by the DATA= option and others defined in the data sets that are specified by one or more AUXDATA= options. For example, this functionality enables sharing of common time series data across multiple projects.

Furthermore, auxiliary data set support enables more than the simple separation of shared data. It also facilitates the elimination of redundancy in these auxiliary data sources by performing partial matching on BY-group qualification. Duplication of time series for the full BY-group hierarchy is no longer required for the auxiliary data sets.

Finally, this functionality permits more than one auxiliary data source to be used concurrently to materialize the virtual time series vectors across a given BY-group hierarchy. So variables that have naturally different levels of BY-group qualification can be isolated into separate data sets and supplied with separate AUXDATA= options to optimize data management and performance.

AUXDATA Functionality

When used, this option declares the presence of an auxiliary data set to optionally provide time series variables to satisfy various declaration statements in the respective procedure steps.

There are two classes of time series data set sources:

- a primary data set from the DATA=DataSet option
- auxiliary data sources from AUXDATA=DataSet options

You can specify zero or more AUXDATA= options in the PROC TIMEDATA statement. Each AUXDATA= option establishes an auxiliary data set source to supply variables declared in subsequent statements that comprise the procedure step.

Variables referenced in the PROC TIMEDATA invocation fall into three classes:

- those that must be physically present in the primary data set
- those that must be physically present in each auxiliary data set
- those that can reside in either the primary or an auxiliary data set

If you specify an ID variable for PROC TIMEDATA, it must be present in the primary data set and all of the auxiliary data sets that you specify. Variables that you specify in the BY statement must be present in the primary data set. A leftmost subset of those BY variables can be present in each of the auxiliary data sets that you specify, and it is not required that all auxiliary data sets contain the same subset. Partial BY group matching is performed for each auxiliary data set independent of the others.

The time series variables that you specify in VAR statements can be resolved from either the primary data set or an auxiliary data set. Variable resolution proceeds in reverse order from the last AUXDATA= option in the PROC TIMEDATA statement to the first. If the variable in question is not found in any of those, the variable must be present in the primary data set for the procedure step to be successful.

AUXDATA Alignment across BY Groups

All BY statement variables must be physically present in the primary data set. However, it is not necessary to have the BY variables present in any of the auxiliary data sets. All, some, or none of the BY variables can be present in any auxiliary data set, as your requirements dictate. Partial BY-group matching is performed between the primary data set and the auxiliary data sets based on the number of BY statement variables that are present in the respective auxiliary data sets.

For example, suppose you have a hierarchy of (REGION, PRODUCT) in the primary data set, which holds the time series variables for monthly sales metrics. Suppose you have an auxiliary data set with time series qualified by REGION for pertinent explanatory variables and another with time series for other explanatory variables to be applied across all (REGION, PRODUCT) groupings of the primary data set. In this scenario, each (REGION, PRODUCT) group in the primary data set seeks a match with a corresponding REGION from the first auxiliary data set to materialize the time series for its variables, but no matching is performed on the second auxiliary data set to materialize the time series for its variables. So if ('SOUTH','EDSEL') is a BY group from the primary data set, the 'SOUTH' BY group series from the first auxiliary data set are used, and the series from the second auxiliary data set are supplied without qualification. If the next primary BY group is ('SOUTH', 'HUDSON'), then the 'SOUTH' BY group is again used to supply the time series from the first auxiliary data set, and the unqualified series are supplied from the second auxiliary data set. So on it goes, each auxiliary data set performing a partial match on the BY variables it holds within the BY group from the primary data set.

AUXDATA Alignment over the Time Dimension

The series from each BY group of the primary data set defines a reference time span for the auxiliary data sets. Only the intersection of the time interval for each auxiliary series with the reference span is materialized. Head or tail missing values are inserted into the auxiliary series for start or stop times that lie inside the

reference span. More generally, missing value semantics apply to the head and tail regions that require filling to materialize the full reference time span.

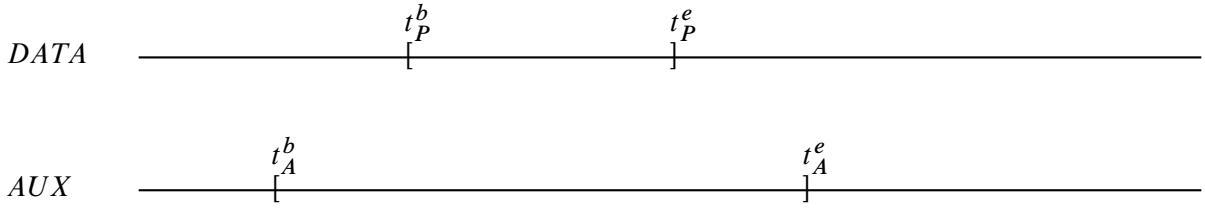
With time series materialized from a single primary data set, there is no latitude for different time ID ranges between the different variables because each observation read contains not only the time ID but also the associated values for all of the variables. With some series materialized from the primary data set and some materialized from auxiliary data sets, the possibility exists for the reference time span to have an arbitrary intersection with the time span of the corresponding series from the auxiliary sources. The intent is to materialize the portion of the auxiliary series time span that intersects with the reference time span and to handle head and tail shortages via missing value semantics as needed.

For the previous usage scenario with a primary data set and two auxiliary data sets, when data is read over a sequence of primary BY groups it might be necessary to materialize various spans of the auxiliary series with appropriate missing value semantics applied as needed to resolve head and tail shortages even though the actual time series contributed from the auxiliary data sets does not physically change. The following discussion breaks this down into several cases depending on intersection possibilities between the reference time span and the auxiliary time span.

Legend:

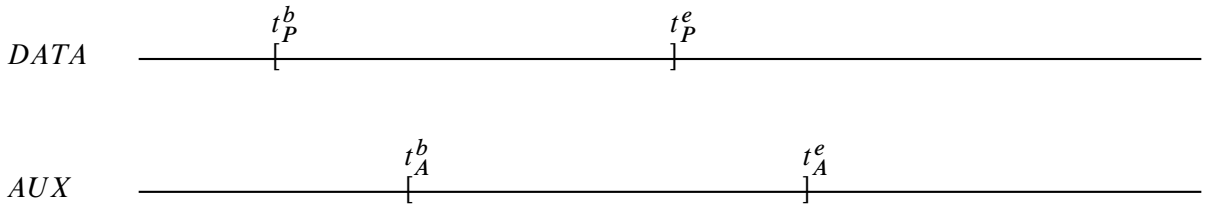
- t_P^b denotes the begin time ID of the primary (DATA=) series.
- t_P^e denotes the end time ID of the primary (DATA=) series.
- t_A^b denotes the begin time ID of the AUXDATA series.
- t_A^e denotes the end time ID of the AUXDATA series.
- $[t_P^b, t_P^e]$ denotes the time span for the primary (DATA=) series (also known as the reference time span).
- $[t_A^b, t_A^e]$ denotes the time span for the AUXDATA series.

Case 1:



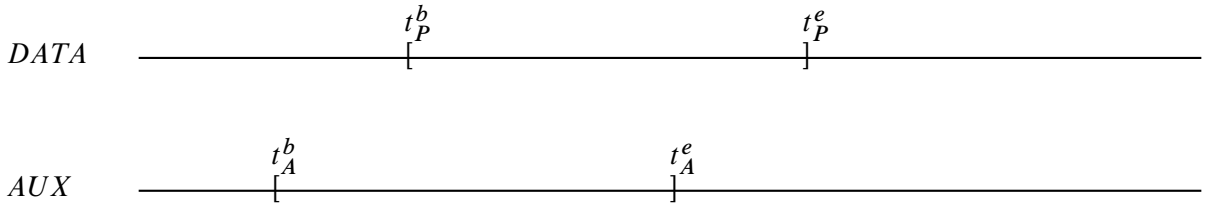
Here $[t_P^b, t_P^e] \subseteq [t_A^b, t_A^e]$. The auxiliary time span includes the reference span as a subset. Values in the AUXDATA series to the left of t_P^b and values to the right of t_P^e are truncated from the AUXDATA series that is materialized in connection with the primary series.

Case 2:



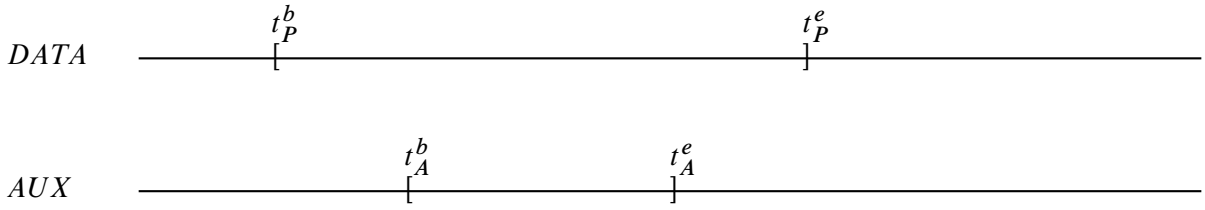
Here $[t_P^b, t_P^e] = [t_P^b, t_A^b) \cup [t_A^b, t_P^e]$. The reference time span leads the auxiliary time span with a non-empty intersection. AUXDATA series values in $[t_P^b, t_A^b)$ are materialized with missing value semantics. AUXDATA series values in $[t_A^b, t_P^e]$ are materialized as actual subject to missing value semantics.

Case 3:



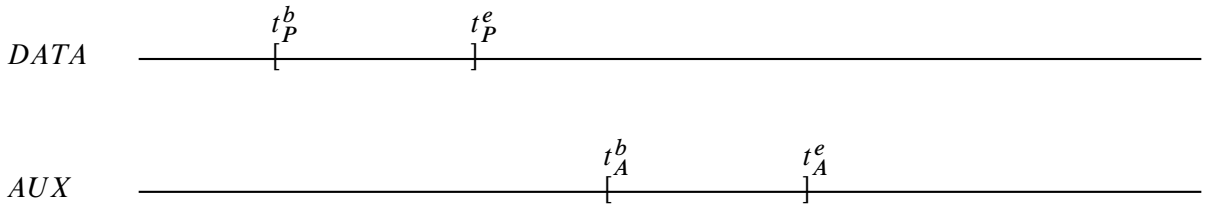
Here $[t_P^b, t_P^e] = [t_P^b, t_A^e] \cup (t_A^e, t_P^e]$. The reference time span lags the auxiliary time span with a non-empty intersection. AUXDATA series values in $[t_P^b, t_A^e]$ are materialized as actual subject to missing value semantics. AUXDATA series values in $(t_A^e, t_P^e]$ are materialized with missing value semantics.

Case 4:



Here $[t_A^b, t_A^e] \subset [t_P^b, t_P^e]$. The auxiliary time span is a subset of the reference time span. AUXDATA series values in $[t_P^b, t_A^b)$ and values in $(t_A^e, t_P^e]$ are materialized with missing value semantics. AUXDATA series values in $[t_A^b, t_A^e]$ are materialized as actual subject to missing value semantics.

Case 5:



Here $[t_P^b, t_P^e] \cap [t_A^b, t_A^e] = \oslash$. The auxiliary time span does not intersect the reference time span at all. In this case all AUXDATA series values are materialized with missing value semantics.

Data Set Output

The TIMEDATA procedure can create the OUT=, OUTARRAY=, OUTPROCINFO=, OUTSCALAR=, and OUTSUM= data sets. In general, these data sets contain the variables listed in the BY statement. If an analysis step that is related to an output step fails, the values of this step are not recorded or are set to missing in the related output data set but appropriate error or warning messages (or both) are recorded in the log.

OUT= Data Set

The OUT= data set contains the variables specified in the BY, ID, or VAR statements. If the ID statement is specified, the ID variable values are aligned and extended based on the ALIGN= and INTERVAL= options. The values of the variables specified in the VAR statements are accumulated based on the ACCUMULATE= option, and missing values are interpreted based on the SETMISSING= option.

OUTARRAY= Data Set

The OUTARRAY= data set contains the variables specified in the BY, ID, or VAR statements. If the ID statement is specified, the ID variable values are aligned and extended based on the ALIGN= and INTERVAL= options. The values of the variables specified in the VAR statements are accumulated based on the ACCUMULATE= option, and missing values are interpreted based on the SETMISSING= option. Additionally, the OUTARRAY= data set contains the variables that are specified in the OUTARRAYS statements and the following variables:

STATUS	status flag that indicates whether the requested analyses were successful
SERIES	series index or BY group index
TIMEID	time ID values
SEASON	season index values
CYCLE	life-cycle index values
Array-Variable-Names	variables listed in the OUTARRAYS statement

The OUTARRAY= data set contains the arrays that are related to the (accumulated) time series.

OUTPROCINFO= Data Set

The OUTPROCINFO= data set contains information about the run of the TIMEDATA procedure. The following variables are present:

SOURCE	name of the procedure, in this case TIMEDATA
NAME	name of the item being reported
LABEL	descriptive label for the item in _NAME_
STAGE	current stage of the procedure (for TIMEDATA this is set to ALL)
VALUE	value of the item specified in _NAME_

OUTSCALAR= Data Set

The OUTSCALAR= data set contains the variables specified in the BY statement. Additionally, the OUTSCALAR= data set contains the variables that are specified in the OUTSCALARS statements and following variables:

STATUS	status flag that indicates whether the requested analyses were successful
SERIES	series index or BY group counter
Scalar-Variable-Names	variables listed in the OUTSCALARS statement.

The OUTSCALAR= data set contains the scalars that are related to the (accumulated) time series.

OUTSUM= Data Set

The OUTSUM= data set contains the variables that are specified in the BY statement as and the variables in the following list. The OUTSUM= data set records the descriptive statistics for each variable specified in a VAR statement. Variables related to descriptive statistics are based on the ACCUMULATE= and SETMISSING= options in the ID and VAR statements:

NAME	variable name
STATUS	status flag that indicates whether the requested analyses were successful
SERIES	count of the series processed in each BY group
START	the starting date of each series
END	the ending date of each series
STARTOBS	the beginning observation number of each series
ENDOBS	the ending observation number of each series
NOBS	number of observations
N	number of nonmissing observations

NMISS	number of missing observations
MINIMUM	minimum value
MAXIMUM	maximum value
AVG	average value
STDDEV	standard deviation

The OUTSUM= data set contains the descriptive statistics of the (accumulated) time series.

STATUS Variable Values

The _STATUS_ variable that appears in the OUTSUM= data set contains a value that specifies whether the analysis has been successful or not. The _STATUS_ variable can take the following values:

0	Analysis was successful.
3000	Accumulation failed.
4000	Missing value interpretation failed.
6000	Series is all missing.
7000	Transformation failed.
8000	Differencing failed.
9000	Descriptive statistics could not be computed.

Printed Output

The TIMEDATA procedure optionally produces printed output by using the Output Delivery System (ODS). By default, the procedure produces no printed output. All output is controlled by the PRINT= option associated with the PROC TIMEDATA statement. In general, if an analysis step related to printed output fails, the values of this step are not printed and appropriate error or warning messages or both are recorded in the log. The printed output is similar to the output data set as follows.

PRINT=ARRAYS	prints the arrays similar to the OUTARRAY= data set.
PRINT=SCALARS	prints the scalars similar to the OUTSCALAR= data set.
PRINT=SUMMARY	prints the summary statistics similar to the OUTSUM= data set.

ODS Table Names

Table 37.2 relates the PRINT= options to ODS tables:

Table 37.2 ODS Tables Produced in PROC TIMEDATA

ODS Table Name	Description	Statement	Option
Arrays	Arrays Table	PRINT	ARRAYS
Scalars	Scalars Table	PRINT	SCALARS
StatisticsSummary	Statistics summary	PRINT	SUMMARY

The tables are related to a all series within a BY group.

Arrays Table

The arrays table (Arrays) illustrate the arrays in tabular form with respect to the Time ID values.

Scalars Table

The scalars table (Scalars) illustrate the scalars in tabular form.

Statistics Summary Table

The summary statistics table (StatisticsSummary) illustrate the summary statistics for each array in tabular form.

ODS Graphics Names

This section describes the graphical output produced by the TIMEDATA procedure. PROC TIMEDATA assigns a name to each graph it creates. These names are listed in Table 37.3.

Table 37.3 ODS Graphics Produced by PROC TIMEDATA

ODS Graph Name	Plot Description	Statement	Option
ArrayPlot	Array Plot	PLOTS	ARRAY

The graphs are related to a single series within a BY group.

Array Plots

The array plots (ArraryPlot) illustrate time series associated with each array. The horizontal axis represents the time ID values, and the vertical axis represents the time series values.

Examples: TIMEDATA Procedure

Example 37.1: Accumulating Transactional Data into Time Series Data

This example uses the TIMEDATA procedure to accumulate time-stamped transactional data that has been recorded at no particular frequency into time series data at a specific frequency. After the time series is created, the various SAS/ETS procedures related to time series analysis, seasonal adjustment and decomposition, modeling, and forecasting can be used to further analyze the time series data.

Suppose that the input data set Work.Retail contains variables Store and Timestamp and numerous other numeric transaction variables. The BY variable Store contains values that break up the transactions into groups (BY groups). The time ID variable Timestamp contains SAS date values recorded at no particular frequency. The other data set variables contain the numeric transaction values to be analyzed. It is further assumed that the input data set is sorted by the variables Store and Timestamp. The following statements form monthly time series from the transactional data based on the median value (ACCUMULATE=MEDIAN) of the transactions recorded with each time period. Also, the accumulated time series values for time periods with no transactions are set to zero instead of to missing (SETMISS=0) and only transactions recorded between the first day of 1998 (START='01JAN1998'D) and last day of 2000 (END='31JAN2000'D) are considered and, if needed, extended to include this range.

```
proc timedata data=retail out=mseries;
   by store;
   id timestamp interval=month
                accumulate=median
                setmiss=0
                start='01jan1998'd
                end  ='31dec2000'd;
   var item1-item8;
run;
```

The monthly time series data are stored in the data Work.Mseries. Each BY group associated with the BY variable Store contains an observation for each of the 36 months associated with the years 1998, 1999, and 2000. Each observation contains the values Store, Timestamp, and each of the analysis variables in the input data set.

After each set of transactions has been accumulated to form a corresponding time series, accumulated time series can be analyzed using various time series analysis techniques. For example, exponentially weighted moving averages can be used to smooth each series. The following statements use the EXPAND procedure to smooth the analysis variable named Storeitem:

```
proc expand data=mseries out=smoothed from=month;
   by store;
   id date;
   convert storeitem=smooth / transform=(ewma 0.1);
run;
```

The smoothed series are stored in the data set Work.Smoothed. The variable Smooth contains the smoothed series.

If the time ID variable Timestamp contains SAS datetime values instead of SAS date values, the INTERVAL=, START=, and END= options must be changed accordingly and the following statements could be used:

```
proc timedata data=retail out=tseries;
   by store;
   id timestamp interval=dtmonth
                accumulate=median
                setmiss=0
                start='01jan1998:00:00:00'dt
                end  ='31dec2000:00:00:00'dt;
   var _numeric_;
run;
```

The monthly time series data are stored in the data Work.Tseries, and the time ID values use a SAS datetime representation.

Example 37.2: Using User-Defined Functions and Subroutines

This example uses the TIMEDATA procedure with a user-defined function and subroutine created by the FCMP procedure.

The following statements use the FCMP procedure to create a user-defined subroutine and a user-defined function. Mylog is a subroutine that log-transforms a time series. Mymean is a function that compute the mean of a time series. The subroutine and function definitions are stored in the data set Work.Timefnc. The OPTIONS statement loads the subroutine and function definitions.

```
proc fcmp outlib=work.timefnc.funcs;

   subroutine mylog(actual[*], transform[*]);
   outargs transform;
   actlen  = DIM(actual);
   do t = 1 to actlen;
      transform[t] = log(actual[t]);
   end;
   endsub;

   function mymean(actual[*]);
   actlen  = DIM(actual);
   sum = 0;
   do t = 1 to actlen;
      sum = sum + actual[t];
   end;
   return( sum / actlen );
   endsub;

run;
quit;

options cmplib = work.timefnc;
```

The input data set Sashelp.Air contains the variables Air and Date. The time series is recorded monthly.

The following statements form quarterly time series from the monthly series based on the median value (ACCUMULATE=TOTAL) of the transactions recorded with each time period and assign the SAS time format (FORMAT=YYMMDD.). The OUTARRAYS statement specifies the Logair and Myair arrays as output. The OUTSCALARS statement specifies the Mystats scalars as output. The other arrays and scalars are not part of the output. The subsequent programming statements create the output arrays and scalars. The PRINT=(ARRAYS SCALARS) prints the output arrays and scalars.

```
proc timedata data=sashelp.air out=work.air
              print=(scalars arrays);
              id date interval=qtr acc=t format=yymmdd.;
              vars air;
              outarrays logair myair;
              outscalars mystats;

       call mylog(air,logair);
       do t = 1 to dim(air);
       myair[t] = air[t] - logair[t];
       end;
       mystats= mymean(air);

run;
```

Example 37.3: Using Auxiliary Data Sets with PROC TIMEDATA

This example demonstrates the use of the AUXDATA= option in PROC TIMEDATA. The data set Sashelp.Gulfoil contains oil and gas production data from the Gulf of Mexico. The variables RegionName and ProtractionName can be used to define a time series hierarchy of interest. Suppose you want to generate two new series that contain the protraction's share of oil and gas production for its associated region at each time index.

You first use PROC TIMESERIES to perform temporal aggregation (accumulation) of the time series for the RegionName level.

```
proc timeseries data=sashelp.gulfoil
                out=byregion(rename=(oil=roil gas=rgas));
   by regionname;
   id date interval=month accumulate=total notsorted;
   var oil gas;
run;
```

You can then use PROC TIMEDATA with the AUXDATA= option to compute the share of oil and gas production contributed by each protraction within its associated region. PROC TIMEDATA reads a monthly time series for each (RegionName, ProtractionName) group for the variables Oil and Gas from Sashelp.Gulfoil. Two new series are produced in the variables Oilshare and Gasshare that respectively contain the protraction's share of the oil and gas production at the region level of the hierarchy (given by variables Roil and Rgas). Those share variables are specified in the OUTARRAY statement for inclusion in the OUTARRAY= data set (Work.Shares). This example relies on the capability of the AUXDATA= feature to perform partial BY group matching. The time series that are acquired for the variables Roil and Rgas are the result of matching

on the RegionName BY variable from the data set Work.Byregion with the RegionName variable from the BY groups that are acquired from the Sashelp.Gulfoil data set.

```
proc timedata data=sashelp.gulfoil
              auxdata=byregion
              out=_null_
              outarray=shares;
   by regionname protractionname;
   outarray oilshare gasshare;
   var oil gas roil rgas;
   id date interval=month accumulate=total;
   do i=1 to _length_;
      oilshare[i] = oil[i] / roil[i];
      gasshare[i] = gas[i] / rgas[i];
   end;
run;
```

The following code demonstrates that the computed shares sum to 1 for each time index in the resulting Oilshare and Gasshare series. PROC TIMESERIES is used to accumulate the shares for these respective variables from the data set Work.Shares and the accumulated share series at the RegionName level are stored to the data set Work.Rshares with variable names Oilsum and Gassum, respectively. The summary from PROC MEANS for the distinct values of RegionName shows that per-time totals for both share series sums to 1.

```
proc timeseries data=shares
                out=rshares(rename=(oilshare=oilsum gasshare=gassum));
   by regionname;
   id date interval=month accumulate=total notsorted;
   var oilshare gasshare;
run;
proc means data=rshares;
   by regionname;
   var oilsum gassum;
run;
```

Output 37.3.1 Validation of Oil and Gas Shares by Region

The MEANS Procedure

Region within Gulf of Mexico=Central

Variable	N	Mean	Std Dev	Minimum	Maximum
oilsum	123	1.0000000	0	1.0000000	1.0000000
gassum	123	1.0000000	0	1.0000000	1.0000000

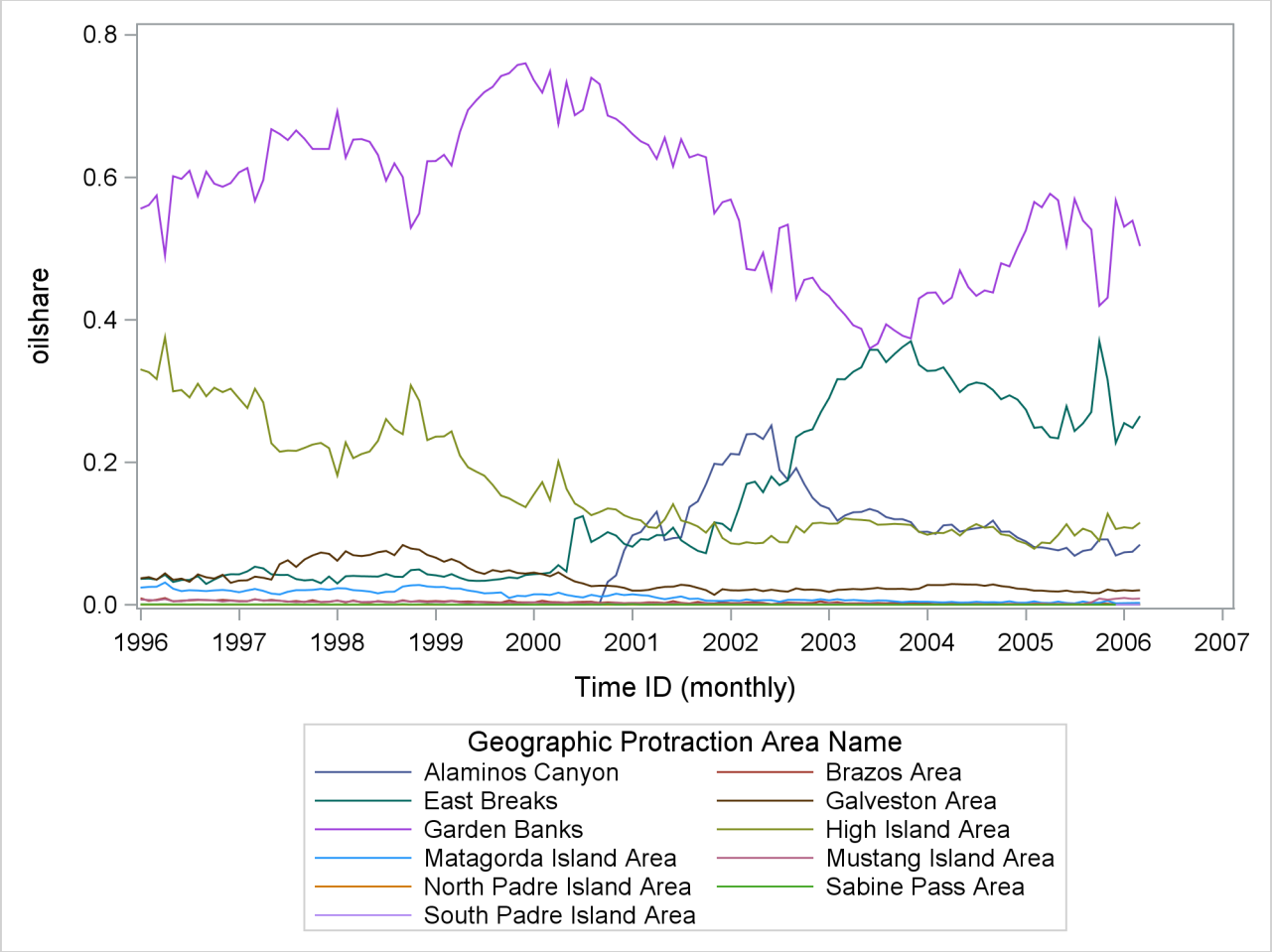
Region within Gulf of Mexico=Western

Variable	N	Mean	Std Dev	Minimum	Maximum
oilsum	123	1.0000000	0	1.0000000	1.0000000
gassum	123	1.0000000	0	1.0000000	1.0000000

You might also want to plot the share series. The following code produces a graph that overlays the protraction level share series for oil production for the Western region.

```
proc sgplot data=shares(where=(RegionName='Western'));
   series x=Date y=OilShare/group=ProtractionName;
run;
```

Output 37.3.2 Protraction Share of Oil Production for Western Region



References

Keogh, E., Chu, S., Hart, D., and Pazzani, M. (2004). "Segmenting Time Series: A Survey and Novel Approach." In *Data Mining in Time Series Databases*, edited by M. Last, A. Kandel, and H. Bunke, 1–22. London: World Scientific. http://www.worldscientific.com/worldscibooks/10.1142/5210.

Chapter 38
The TIMEID Procedure

Contents

Overview: TIMEID Procedure **2635**
Getting Started: TIMEID Procedure **2636**
Syntax: TIMEID Procedure **2636**
Functional Summary 2637
PROC TIMEID Statement 2638
BY Statement 2639
ID Statement 2640
Details: TIMEID Procedure **2641**
Time ID Diagnostics 2641
Diagnostic Output Representation 2641
Inferring Time Intervals and Alignments 2643
Data Set Output 2644
Printed Tabular Output 2647
ODS Graphics 2647
Examples: TIMEID Procedure **2648**
Example 38.1: Examining a Weekly Time ID Variable 2648
Example 38.2: Inferring a Date Interval 2655
Example 38.3: Examining Multiple BY Groups 2656

Overview: TIMEID Procedure

The TIMEID procedure evaluates a variable in an input data set for its suitability as a time ID variable in SAS procedures and solutions that are used for time series analysis. PROC TIMEID assesses how well a time interval specification fits SAS date or datetime values, or observation numbers used to index a time series. The time interval used in this analysis can be either specified explicitly as input to PROC TIMEID or inferred by the procedure based on values of the time ID variable. The TIMEID procedure produces diagnostic information in the form of data sets and ODS tabular and plotted output. These diagnostic results summarize characteristics of the time ID variable that can help determine its use as an index in other time series procedures and solutions.

PROC TIMEID is intended for use as a tool to either identify the time interval of a variable or prepare problematic data sets for use in subsequent time series analyses. In particular, this procedure can be used to investigate inconsistencies between time ID values and the ID statement options used in other SAS procedures and solutions.

Getting Started: TIMEID Procedure

When a data set contains a time ID variable with corrupted, missing, or duplicate values, PROC TIMEID can help isolate and identify these problematic observations. For a data set with a small number of ID variable anomalies and a known time interval, a graphical depiction of the problem areas can be created using the following statements:

```
proc timeid data=<input-dataset> plot=values;
   id <time-ID-variable>  interval=<frequency>;
run;
```

For larger data sets whose quality is unknown, it can be useful to get a general overview of the relative number of observations with problematic time ID values. The following statements graphically summarize the prevalence of anomalous time ID values:

```
proc timeid data=<input-dataset> plot=(intervalcounts offsets spans);
   id <time-ID-variable>  interval=<frequency>;
run;
```

When prior knowledge of the time interval that separates observations is incomplete, PROC TIMEID can be used to infer the interval by omitting the INTERVAL= option from the ID statement as in the following statements:

```
proc timeid data=<input-dataset> outinterval=<output-dataset>;
   id <time-ID-variable>;
run;
```

Syntax: TIMEID Procedure

The TIMEID procedure uses the following statements:

PROC TIMEID *options* **;**
BY *variables* **;**
ID *variable* < *options* > **;**

Functional Summary

The statements and options that control the TIMEID procedure are summarized in Table 38.1.

Table 38.1 Syntax Summary

Description	Statement	Option
Statements		
Specifies data sets and options	PROC TIMEID	
Specifies BY-group processing	BY	
Specifies the time ID variable	ID	
Data Set Options		
Specifies the input data set	PROC TIMEID	DATA=
Specifies the maximum number of ID values to analyze	PROC TIMEID	NBYOBS=
Specifies the output frequency count data set	PROC TIMEID	OUTFREQ=
Specifies the output interval data set	PROC TIMEID	OUTINTERVAL=
Specifies the detailed output interval data set	PROC TIMEID	OUTINTERVALDETAILS=
Time ID Options		
Specifies the interval alignment	ID	ALIGN=
Specifies that duplicate time ID values can be present in DATA= data set	ID	DUPLICATES
Specifies the time interval between observations	ID	INTERVAL=
Specifies that time ID variable values are not sorted	ID	NOTSORTED
Printing and Plotting Options		
Specifies the time ID format	ID	FORMAT=
Specifies the types of graphical output	PROC TIMEID	PLOT=
Specifies the types of printed output	PROC TIMEID	PRINT=
Miscellaneous Options		
Limits error and warning messages	PROC TIMEID	MAXERROR=

PROC TIMEID Statement

PROC TIMEID *options* **;**

The following options can be used in the PROC TIMEID statement:

DATA=*SAS-data-set*
names the SAS data set that contains the input data for the procedure. If the DATA= option is not specified, the most recently created SAS data set is used.

MAXERROR=*number*
limits the number of warning and error messages produced during the execution of the procedure to the specified value. The default is MAXERRORS=50. This option is particularly useful in BY-group processing where it can be used to suppress recurring messages.

NBYOBS=*number*
limits the number of observations that are used to analyze the time ID variable. The NBYOBS= option should be used instead of the OBS= data set option when BY variables are specified. The NBYOBS= option excludes observations from incomplete BY groups in the analysis. This option guarantees that any truncation of the DATA= data set occurs at a BY-group boundary. Only BY groups that are completely contained within the first *number* of observations are processed. When the NBYOBS= option is omitted, all observations are processed.

OUTFREQ=*SAS-data-set*
names the output data set to contain the frequency counts of each unique value of the time ID variable. The frequency counts are performed on time ID values that are recorded in the DATA= data set. The time ID values are not aligned with respect to an interval prior to computation of the frequency counts. See the section "OUTFREQ= Data Set" on page 2644 for details.

OUTINTERVAL=*SAS-data-set*
names the output data set to contain the time ID interval information that is summarized across all BY groups in the DATA= data set. See the section "OUTINTERVAL= Data Set" on page 2644 for details.

OUTINTERVALDETAILS=*SAS-data-set*
names the output data set to contain the time ID interval information for each BY group. See the section "OUTINTERVALDETAILS= Data Set" on page 2645 for details.

PLOT*(global-option)***=***request-option* | *(request-options)*
specifies the graphical output desired. By default, the TIMEID procedure produces no graphical output. The following *global-options* are available:

UNPACK | UNPACKPANELS suppresses paneling.

By default, multiple plots can appear in some output panels. Specify UNPACKPANELS to get each plot in a separate panel. The following plot *request-options* are available:

COUNTS | INTCNTS | INTERVALCOUNTS
plots a histogram of the time ID interval counts.

OFFSETS plots a histogram of the time offsets for the time ID values.

PERIODS | SPANS	plots a histogram of the spans between adjacent time ID values.
VALUES	plots a panel of the counts, offsets, and spans for each of the time ID values.
ALL	is equivalent to specifying PLOT=(INTERVALCOUNTS SPANS OFFSETS VALUES).

See the section "Time ID Diagnostics" on page 2641 for details.

PRINT=*option* | **(***options***)**

specifies the printed output desired. By default, the TIMEID procedure produces no printed output. The following printing options are available:

COUNTS | INTCNTS | INTERVALCOUNTS	prints a table that contains the counts of time ID values per interval.
INTERVAL	prints a summary of information about the time interval.
OFFSETS	prints a table that contains the time offsets for the time ID values.
PERIODS | SPANS	prints tables that contain statistics on the spans between adjacent time ID values.
VALUES	prints tables that contain offset span and count information for the time ID values.
ALL	is equivalent to specifying PRINT=(INTERVALCOUNTS SPANS INTERVAL OFFSETS VALUES).

See the section "Time ID Diagnostics" on page 2641 for details.

BY Statement

BY *variables* **;**

A BY statement can be used with PROC TIMEID to obtain separate analyses for groups of observations defined by the BY variables.

When a BY statement appears, the procedure expects the input data set to be sorted in order of the BY variables.

If your input data set is not sorted in ascending order, use one of the following alternatives:

- Sort the data by using the SORT procedure with a similar BY statement.
- Specify the option NOTSORTED or DESCENDING in the BY statement for the TIMESERIES procedure. The NOTSORTED option does not mean that the data are unsorted but rather that the data are arranged in groups (according to values of the BY variables) and that these groups are not necessarily in alphabetical or increasing numeric order.
- Create an index on the BY variables by using the DATASETS procedure.

For more information about the BY statement, see *SAS Language Reference: Concepts*. For more information about the DATASETS procedure, see the discussion in the *Base SAS Procedures Guide.*

ID Statement

ID *variable* < *options* > **;**

The ID statement names a numeric variable that identifies observations in the input and output data sets. The ID variable's values are assumed to be SAS date or datetime values. The ID statement options specify how the time ID values are spaced and aligned relative to a SAS date or datetime interval. The INTERVAL= option specifies the fundamental spacing that is used as the basis for counting intervals, offsets, and spans in the data. Specification of the ID variable in an ID statement is required.

ALIGN=*alignment*

specifies the alignment of the identifying SAS date or datetime that is used to represent intervals. The value of the ALIGN= option is used in the analysis of the time ID variable. The ALIGN= option accepts the following values: BEGINNING | BEG | B, MIDDLE | MID | M, ENDING | END | E, and INFER. For example, ALIGN=BEGIN specifies that the identifying date for the interval is the beginning date in the interval. If the ALIGN= option is not specified, then the default alignment is BEGIN. ALIGN=INFER specifies that the alignment of values within time intervals be inferred from the time ID values.

DUPLICATES

specifies that multiple observations in the DATA= data set can fall within the same time interval as defined by the time ID variable. When this option is omitted and multiple time ID values are encountered in a single time interval, error messages are written to the SAS log.

FORMAT=*format*

specifies the SAS format used for time ID values in the data sets and in printed and plotted output that is generated by PROC TIMEID. If the FORMAT= option is not specified, the format applied to the input time ID variable is used. If neither of these formats is specified, the format is inferred from the INTERVAL= option.

INTERVAL=*interval*

specifies the proposed time interval and shift that describe the time ID values in the input data set. See Chapter 5, "Date Intervals, Formats, and Functions," for more information about the intervals that can be specified. See the section "Time ID Diagnostics" on page 2641 for more information about how the INTERVAL= option determines the nature of diagnostic information reported by the TIMEID procedure.

If no interval is specified, the procedure attempts to infer an interval from the input time ID values. See the section "Inferring Time Intervals and Alignments" on page 2643 for details about how the time interval is inferred.

NOTSORTED

specifies that the observations in the DATA= data set are not sorted by the time ID variable. When this option is omitted, error messages are generated for time ID values that are not sorted in ascending order.

Details: TIMEID Procedure

Time ID Diagnostics

For a specified time interval, PROC TIMEID decomposes the raw time ID values in an input data set into the following three quantities, whose values are represented by nonnegative integers at each unique time ID value in the input series:

interval counts the number of observations that share each time interval in the data set.

offsets the numerical difference between a time ID value and the aligned value for that time interval. The unit of measure used to express this distance is days for date values and seconds for datetime values. The offset is computed for each time ID value, t_i, by using the following SAS expression:

$$\text{offset}_i = t_i - \text{INTNX}(\text{interval}, t_i, 0, \text{alignment})$$

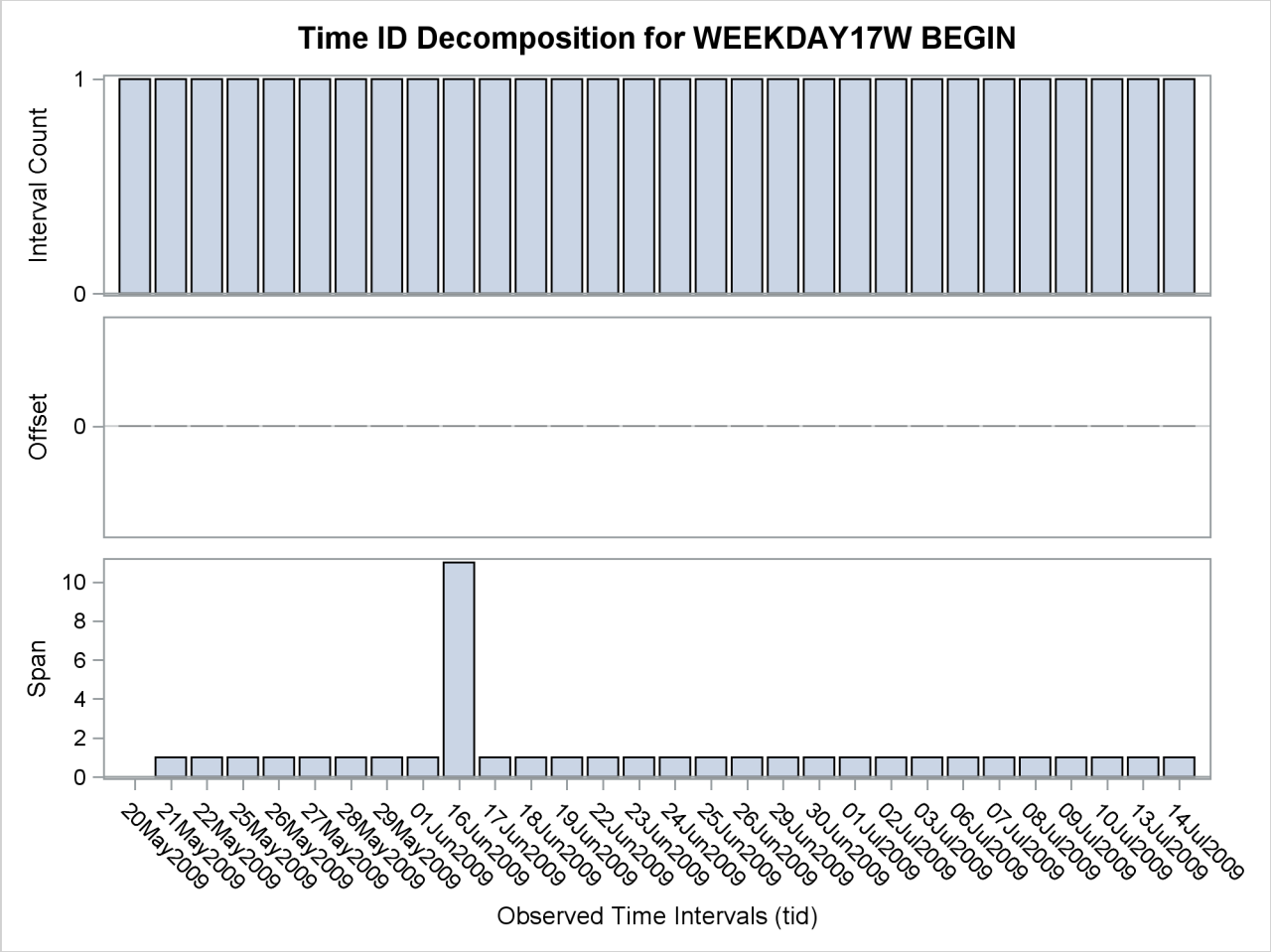
spans the number of intervals between each time ID value and the previous time ID value. The spans value is equivalent to the number returned by the following SAS expression:

$$\text{spans}_i = \text{INTCK}(\text{interval}, t_{i-1}, t_i)$$

Diagnostic Output Representation

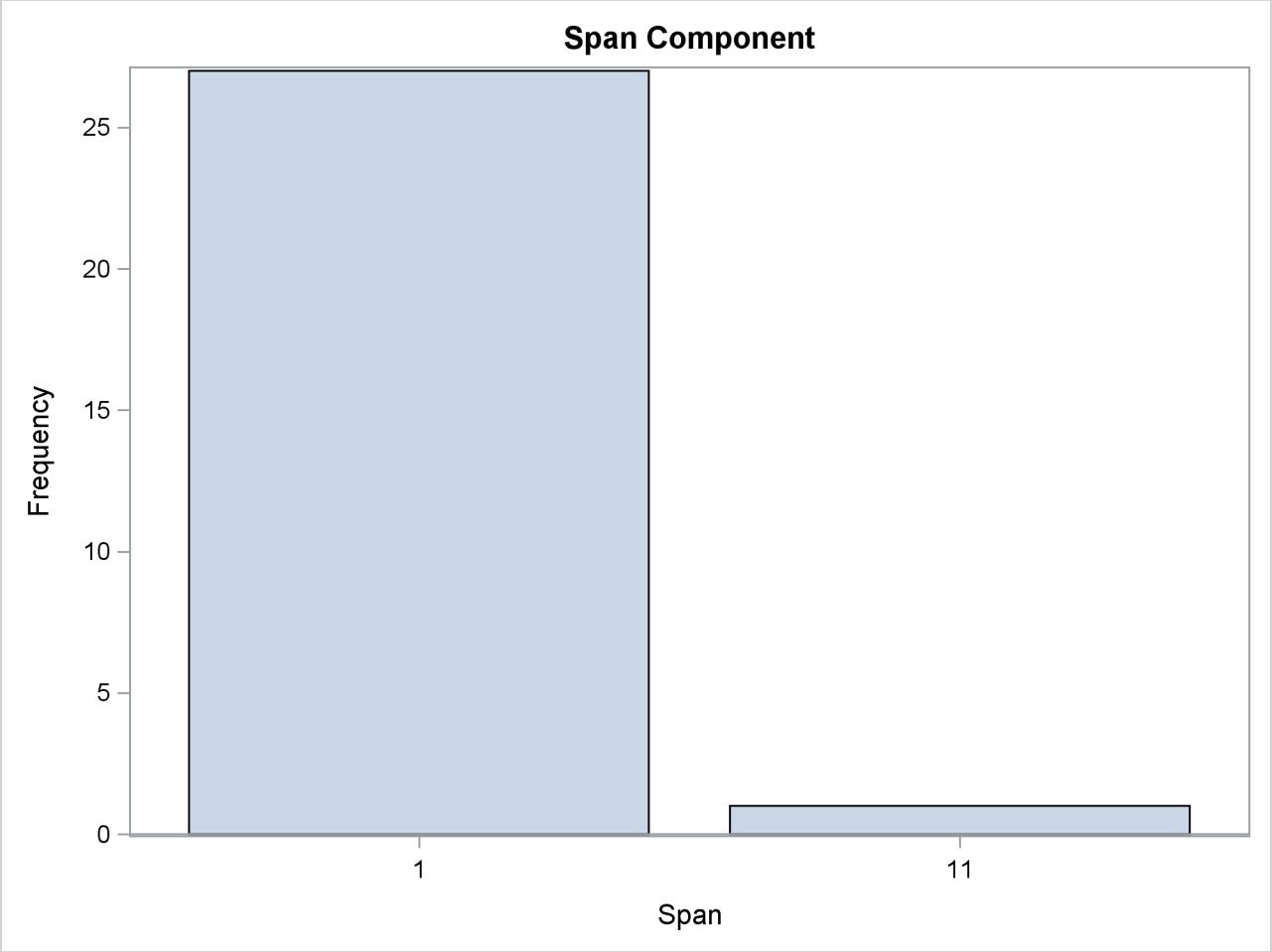
The TIMEID procedure produces time ID diagnostics as both time-ID-based and count-based frequency distributions to expose many of the possible problems that can occur in a time ID variable. The time-ID-based frequency distributions that are generated with the PLOT= option provide a detailed view of time ID values that can isolate problems with specific ID values. Figure 38.1 shows a time series that has a span of 10 observations in a weekday series based on the results of the PLOT=(VALUES SPANS) option. The single large bar in the spans plot shows where data are omitted.

Figure 38.1 Time ID Decomposition



The count-based frequency distributions summarize features of the time ID variable. Individual printed and plotted outputs are available to describe the distribution of the number of spans, offsets, and interval counts that occur in the time ID variable. Figure 38.2 illustrates a count-based frequency distribution of the spans within the weekday series.

Figure 38.2 Span Count Distribution



The large bar at the span of 1 shows that most of the observations are correctly separated by one interval. The bar at 11 indicates that one observation is separated by 11 intervals from the preceding value of the time ID variable. This further illustrates a span of 10 omitted observations.

Inferring Time Intervals and Alignments

When the INTERVAL= option is not specified in the ID statement, a time interval is inferred from the time ID values in the input data set. The technique used to infer a time interval involves searching for the interval that fits the greatest number of time ID values. First, time ID values are sampled from the input data set to generate a set of candidate intervals. Then the candidate interval that is consistent with greatest number of time ID values is chosen to represent the time series.

When the ALIGN=INFER option is specified, the convention that is used to specify time interval alignment is inferred from the time ID variable values by using a similar technique. When both the time interval and its alignment are to be inferred, each of the possible alignments, BEGIN, MIDDLE, and END, are considered in the search. Precedence in the search is given to intervals with the BEGIN alignment.

Data Set Output

The TIMEID procedure creates the OUTFREQ=, OUTINTERVAL=, and OUTINTERVALDETAILS= data sets. The OUTFREQ= and OUTINTERVALDETAILS= data sets contain the variables that are specified in the BY statement along with variables that characterize the time ID values. The OUTINTERVAL= option creates a data set without BY variables. The information in this data set summarizes time ID diagnostic information across all BY groups in the DATA= data set.

OUTFREQ= Data Set

The OUTFREQ= data set contains a single observation for each value of the time ID variable in the input data set for each BY group. Additionally, the following variables are written to the OUTFREQ= data set:

COUNT	number of the occurrences of the time ID value
PERCENT	percentage of all time ID values

OUTINTERVAL= Data Set

The OUTINTERVAL= data set contains information that is similar to the variables written to the OUTINTERVALDETAILS= data set; however, the OUTINTERVAL= data set summarizes the information across all BY groups into a single observation. The following variables are written to the OUTINTERVAL= data set:

TIMEID	time ID variable
START	smallest time ID interval
END	largest time ID interval
STARTSHARED	largest starting time ID interval
ENDSHARED	smallest ending time ID interval
NOBS	number of observations
N	number of nonmissing observations
NMISS	number of missing observations
NBY	number of BY groups
NINVALID	number of invalid observations
STATUS	status flag that indicates whether the requested analyses were successful:

0	The analysis completed successfully.
1	interval consistent but data contains gaps
2	interval not consistent with data
10	missing or invalid values found
20	ID values not sorted
21	duplicate ID values detected
30	fewer than 3 values found

4000	Inference of a time interval from the data set failed.
5000	Diagnosis of the DATA= data set for the specified time interval failed.

MSG	a message that provides further details when the STATUS variable is not zero
INTERVAL	time interval that is specified or recommended
INTNAME	time interval base name that is specified or recommended
MULTIPLIER	time interval multiplier that is specified or recommended
SHIFT_INDEX	time interval shift index that is specified or recommended
ALIGNMENT	time interval alignment that is specified or recommended
SEASONALITY	seasonality determined from specified or recommended time interval
TOTALSEASONCYCLES	total number of seasonal cycles spanned by all the observations
SEASONCYCLESSHARED	number of seasonal cycles that are shared among all BY groups
FORMAT	format of the time ID variable

The START, END, STARTSHARED, and ENDSHARED variables are reported using the interval and alignment specified in the ID statement or inferred from the time ID values.

OUTINTERVALDETAILS= Data Set

The OUTINTERVALDETAILS= data set contains statistics about the time interval that is specified in the ID statement or inferred from the time ID values for each BY group. The following variables represent these statistics:

TIMEID	time ID variable name
START	starting time ID interval
END	ending time ID interval
NOBS	number of observations
N	number of nonmissing observations
NMISS	number of missing observations
NINVALID	number of invalid observations
NINTCNTS	number of unique interval count values
PCTINTCNTS	percentage of interval counts greater than one
MININTCNT	minimum of interval counts
MAXINTCNT	maximum of interval counts
MEANINTCNT	mean of interval counts
STDINTCNT	standard deviation of interval counts
MEDINTCNT	median of interval counts
NOFFSETS	number of time ID offset
PCTOFFSETS	percentage of time ID offset

MINOFFSET	minimum of time ID offsets
MAXOFFSET	maximum of time ID offsets
MEANOFFSET	mean of time ID offsets
STDOFFSET	standard deviation of time ID offsets
MEDOFFSET	median of time ID offsets
NSPANS	number of spans between time ID values
PCTSPANS	percentage of spans between time ID values
MINSPAN	maximum of spans between time ID values
MAXSPAN	minimum of spans between time ID values
MEANSPAN	mean of spans between time ID values
STDSPAN	standard deviation of spans between time ID values
MEDSPAN	median of spans between time ID values
STATUS	status flag that indicates whether the requested analyses were successful:

0	The analysis completed successfully.
1	interval consistent but data contains gaps
2	interval not consistent with data
10	missing or invalid values found
20	ID values not sorted
21	duplicate ID values detected
30	fewer than 3 values found
4000	Inference of a time interval from the data set failed .
5000	Diagnosis of the DATA= data set for specified time interval failed.

MSG	a message that provides further details when the STATUS variable is not zero
INTERVAL	time interval specified or recommended
INTNAME	time interval base name specified or recommended
MULTIPLIER	time interval multiplier specified or recommended
SHIFT_INDEX	time interval shift index specified or recommended
ALIGNMENT	time interval alignment specified or recommended
SEASONALITY	seasonality determined from specified or recommended time interval
NSEASONCYCLES	number of seasonal cycles spanned by the time ID values
FORMAT	format of the time ID variable

The START and END variables are reported using the interval and alignment specified in the ID statement or inferred from the time ID values.

Printed Tabular Output

The TIMEID procedure optionally produces printed output by using the Output Delivery System (ODS). By default, the procedure produces no printed output. The appearance of the printed tabular output is controlled by the PRINT= option in the PROC TIMEID statement.

Table 38.2 relates the PRINT= options to the names of the ODS tables.

Table 38.2 ODS Tables Produced in PROC TIMEID

ODS Name	Description	PRINT= Option
DataSet	Information about the input data set	ALL
Decomposition	Time ID counts, offsets, and spans	VALUES
Interval	Information about the time interval	INTERVAL
IntervalCountsComponent	Frequency distribution of interval counts	INTERVALCOUNTS
IntervalCountsStatistics	Statistics on interval count frequency distribution	INTERVALCOUNTS
OffsetsComponent	Frequency distribution of offsets	OFFSETS
OffsetStatistics	Statistics on offset frequency distribution	OFFSETS
SpansComponent	Frequency distribution of spans	SPANS
SpanStatistics	Statistics on the span frequency distribution	SPANS
Values	Time ID value counts	VALUES
ValueSummary	Summary of the number of valid observations	VALUES

ODS Graphics

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*).

Before you create graphs, ODS Graphics must be enabled (for example, with the ODS GRAPHICS ON statement). For more information about enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" in that chapter.

The overall appearance of graphs is controlled by ODS styles. Styles and other aspects of using ODS Graphics are discussed in the section "A Primer on ODS Statistical Graphics" in that chapter.

The TIMEID procedure uses ODS Graphics to produce plotted output as specified by the PLOT= option. Table 38.3 relates the PLOT= options to the names of the ODS Graphics objects.

Table 38.3 ODS Graphics Produced by the PLOT= Option in PROC TIMEID

ODS Graph Name	Plot Description	PLOT= Option
DecompositionPlot	Panel of spans, offsets, and counts for each time interval	VALUES
IntervalCountsComponentPlot	Histogram of interval counts	INTERVALCOUNTS
IntervalCountsPlot	Plot of counts for each time interval value	VALUES
OffsetComponentPlot	Histogram of time ID offsets	OFFSETS
OffsetsPlot	Plot of offsets for each time interval value	VALUES
SpanComponentPlot	Histogram of span sizes between time ID values	SPANS
SpansPlot	Plot of spans for each time interval value	VALUES
ValuesPlot	Plot of counts of each time ID value	VALUES

Examples: TIMEID Procedure

Example 38.1: Examining a Weekly Time ID Variable

This example illustrates how problems in a weekly time series can be visualized and quantified using the TIMEID procedure's diagnostic capabilities.

The following DATA step creates a data set that contains time values spaced in three week intervals where some weeks have been skipped or duplicated and some have been recorded on different weekdays.

```
data triweek;
   format date date.;
   input date : date. @@;
datalines;
28DEC48 18JAN49 08FEB49 01MAR49 22MAR49 12APR49 03MAY49 24MAY49
17JUN49 05JUL49 26JUL49 16AUG49 06SEP49 27SEP49 18OCT49 08NOV49
29NOV49 20DEC49 10JAN50 04FEB50 21FEB50 14MAR50 04APR50 25APR50

   ... more lines ...
```

The following TIMEID procedure statements generate an ODS display of the time series that characterizes interval counts, offsets, and spans in the time ID variable.

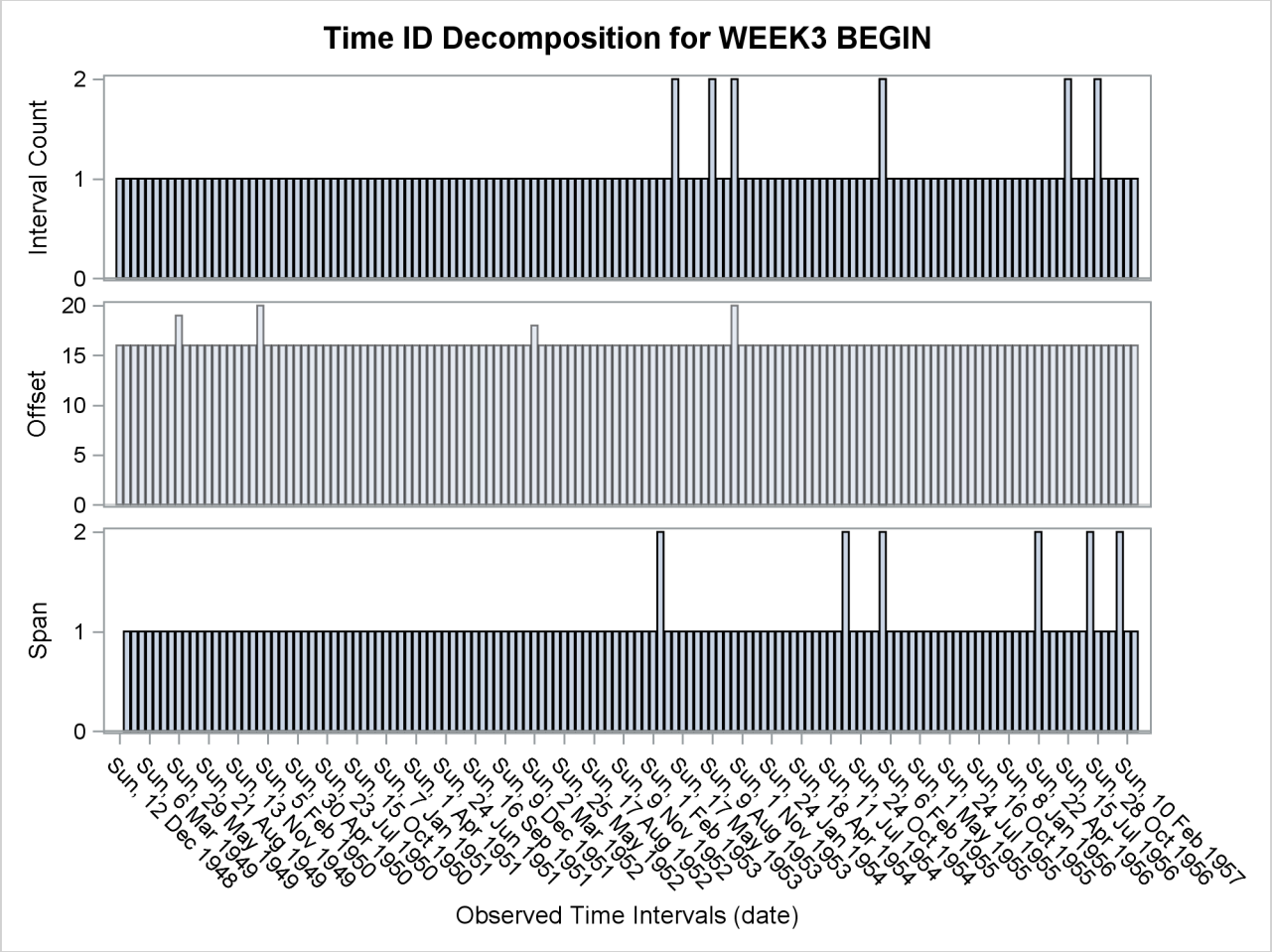
```
proc timeid data=triweek print=all plot=all;
  id date interval=week3;
run;
```

The Time ID decomposition listing and plot shown in Output 38.1.1 and Output 38.1.2 summarize how well the WEEK3 interval fits the time ID values by showing the number of counts, offsets, and spans for each time interval that is represented by the DATE variable. The listing in Output 38.1.1 has been truncated to include only the first 10 observations. The Time ID plots in Output 38.1.2 indicate that there are duplicated time ID values for a three-week time interval in the Counts plot. The duplicated time intervals have a Count value of 2. The Offsets plot shows which days in the 21 day cycle have been used to record each time interval in the series. The Spans plot records values of 2 for six time intervals where no observations were recorded in the previous interval. The three component plots are histogram summaries of the diagnostic quantities plotted against individual intervals in the decomposition plots. The component plots can be useful in diagnosing time series that contain many time intervals.

Output 38.1.1 Time ID Decomposition Listing

	Time Component			
Value Index	date	Offset	Span	Interval Count
1	Sun, 12 Dec 1948	16	.	1
2	Sun, 2 Jan 1949	16	1	1
3	Sun, 23 Jan 1949	16	1	1
4	Sun, 13 Feb 1949	16	1	1
5	Sun, 6 Mar 1949	16	1	1
6	Sun, 27 Mar 1949	16	1	1
7	Sun, 17 Apr 1949	16	1	1
8	Sun, 8 May 1949	16	1	1
9	Sun, 29 May 1949	19	1	1
10	Sun, 19 Jun 1949	16	1	1

Output 38.1.2 Time ID Decomposition Plot



Output 38.1.3 and Output 38.1.4 describe the distribution of counts of duplicated WEEK3 intervals in the TriWeek data set. For this data set there are 134 intervals that contain one DATE value, and 10 intervals that contain two DATE values.

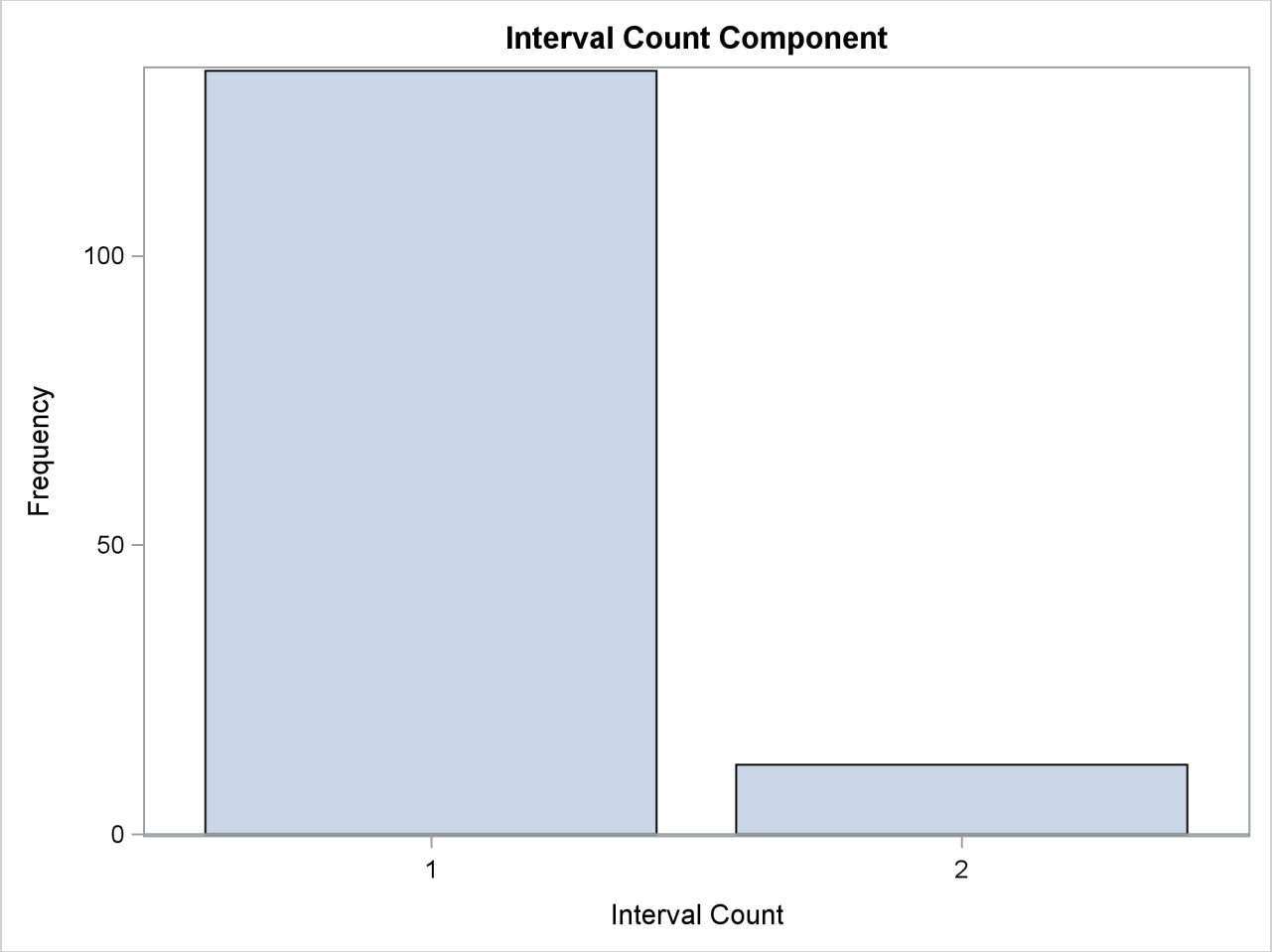
Output 38.1.3 Time ID Interval Counts Listings

The TIMEID Procedure

Component			
Value Index	Interval Count	Frequency	Percentage
1	1	132	91.666667
2	2	12	8.333333

Statistics Summary			
Minimum	Maximum	Mean	Standard Deviation
1	2	1.0833333	1.3008873

Output 38.1.4 Time ID Interval Counts Histogram



The offsets diagnostics Output 38.1.5 and Output 38.1.6 show the distribution of days in the 21-day WEEK3 interval used to record the time intervals in the series. The observations in the TriWeek data set represent intervals with five different offsets from the beginning of the WEEK3 interval: 0, 16, 18, 19 and 20. The high prevalence of intervals with offset 16 indicates that the TriWeek data set would be represented better using the WEEK3.17 interval.

Output 38.1.5 Time ID Offsets Listings

The TIMEID Procedure

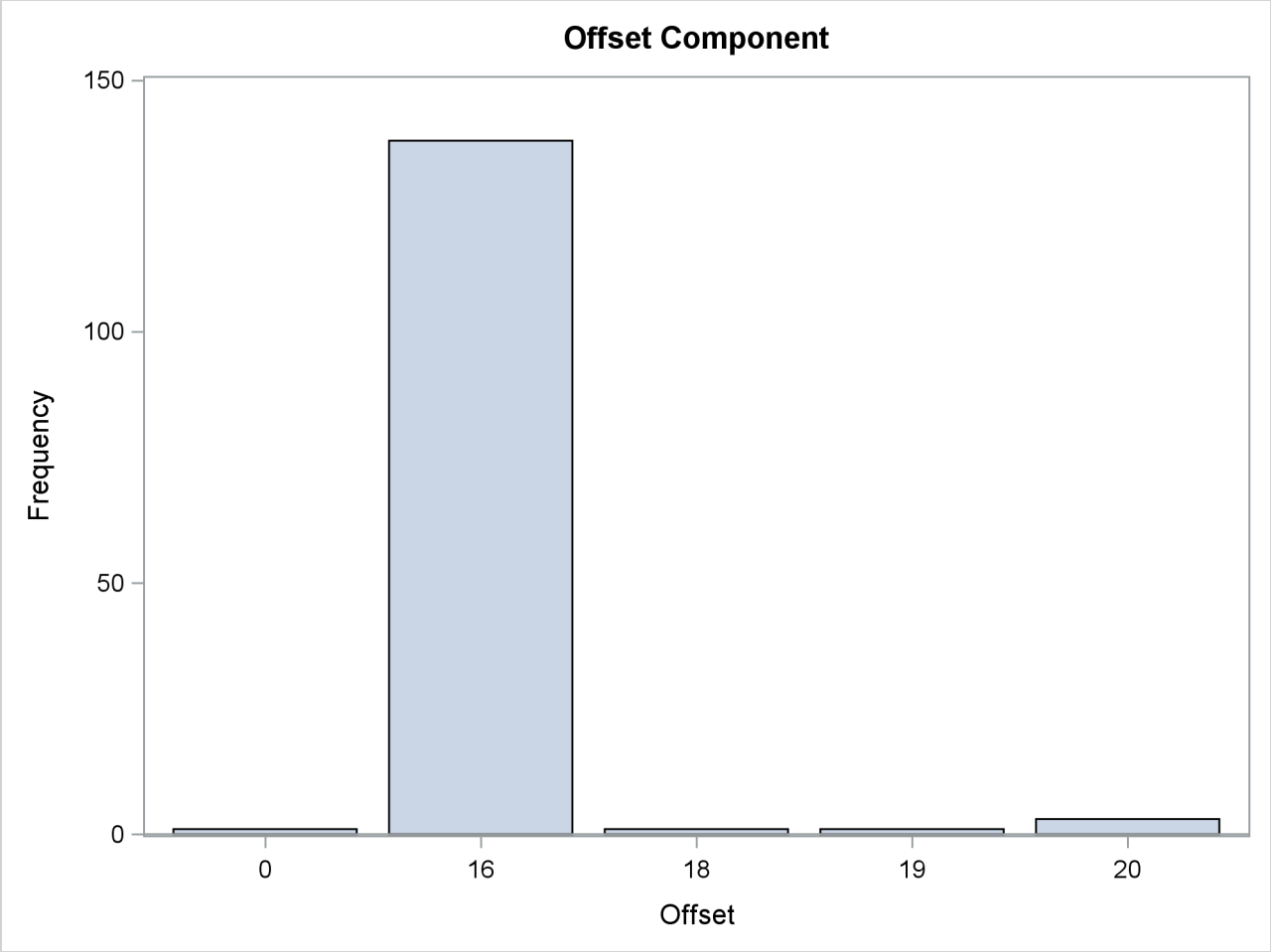
	Component		
Value Index	Offset	Frequency	Percentage
1	0	1	0.694444
2	16	138	95.833333
3	18	1	0.694444
4	19	1	0.694444
5	20	3	2.083333

Output 38.1.5 *continued*

Statistics Summary

Minimum	Maximum	Mean	Standard Deviation
0	20	16.006944	1.7006205

Output 38.1.6 Time ID Offsets Histogram



The span diagnostics Output 38.1.7 and Output 38.1.8 show the distribution of the span sizes between successive DATE values. The TriWeek data set has three different span sizes of widths 0, 1 and 2. Here one span corresponds to the width of a WEEK3 interval.

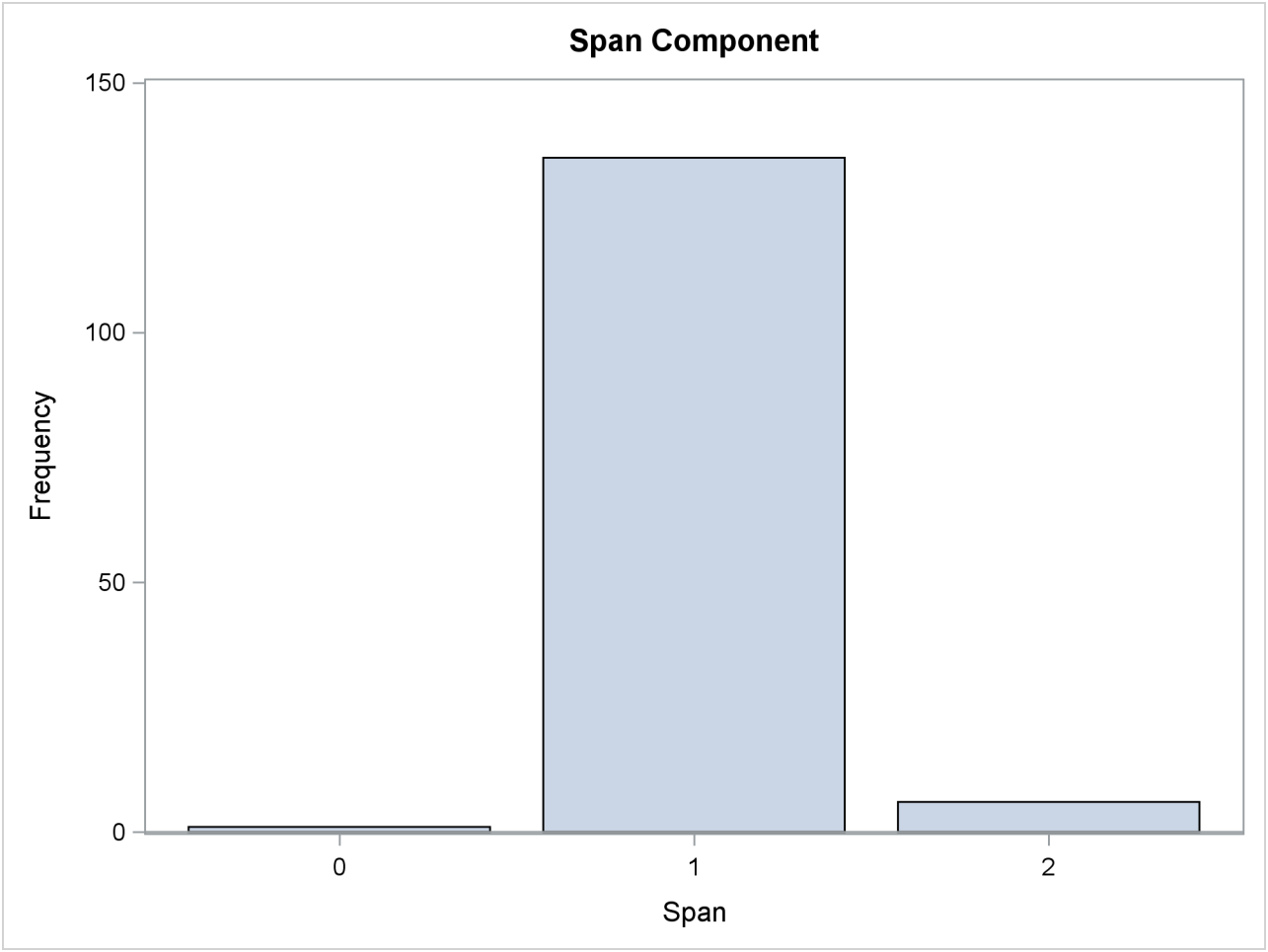
Output 38.1.7 Time ID Span Listings

The TIMEID Procedure

Component			
Value Index	Span	Frequency	Percentage
1	0	1	0.704225
2	1	135	95.070423
3	2	6	4.225352

Statistics Summary			
Minimum	Maximum	Mean	Standard Deviation
0	2	1.0352113	0.6367974

Output 38.1.8 Time ID Span Histogram

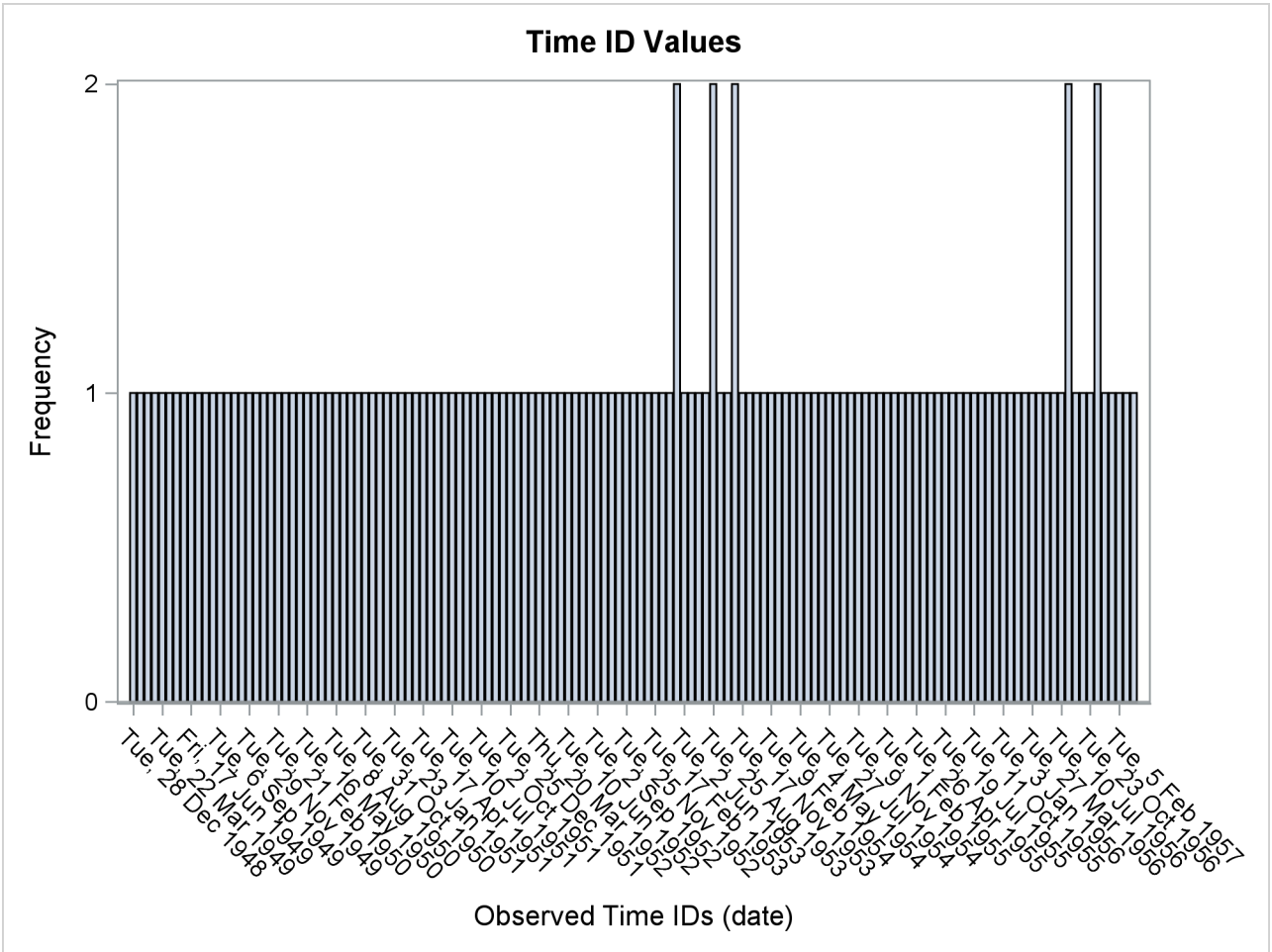


Output 38.1.9 and Output 38.1.10 show the distribution of time ID values before alignment to the WEEK3 interval. The listing in Output 38.1.9 has been truncated to include only the first 10 observations.

Output 38.1.9 Unaligned Time ID Listings

Time ID Values for DATE			
Value Index	date	Frequency	Percentage
1	Tue, 28 Dec 1948	1	0.694444
2	Tue, 18 Jan 1949	1	0.694444
3	Tue, 8 Feb 1949	1	0.694444
4	Tue, 1 Mar 1949	1	0.694444
5	Tue, 22 Mar 1949	1	0.694444
6	Tue, 12 Apr 1949	1	0.694444
7	Tue, 3 May 1949	1	0.694444
8	Tue, 24 May 1949	1	0.694444
9	Fri, 17 Jun 1949	1	0.694444
10	Tue, 5 Jul 1949	1	0.694444

Output 38.1.10 Unaligned Time ID Histogram



Example 38.2: Inferring a Date Interval

This example illustrates how a time ID variable can be inferred from a data set when a sufficient number of observations are present.

```
data workdays;
   format day weekdate.;
   input day : date. @@;
   datalines;
01AUG09 06AUG09 11AUG09 14AUG09 19AUG09 22AUG09
27AUG09 01SEP09 04SEP09 09SEP09 12SEP09 17SEP09
;

proc timeid data=workdays print=interval;
   id day;
run;
```

The 12 observations in the WorkDays data set are enough to determine that the DAY time ID variable is represented by the WEEKDAY12W3 interval. The WEEKDAY12W3 interval corresponds to every third day of the week excluding Sundays and Mondays. Characteristics of this interval are shown in Output 38.2.1.

Output 38.2.1 Inferred Time Interval Information

The TIMEID Procedure

Time Interval Analysis Summary	
Time ID Variable	day
Time Interval	WEEKDAY12W3
Base Name	WEEKDAY
Multiplier	3
Shift	0
Length of Seasonal Cycle	5
Time ID Format	DATE9.
Start	01AUG2009
End	17SEP2009

Example 38.3: Examining Multiple BY Groups

This example illustrates how a time ID variable can be examined independently over each BY group and summarized over all observations in the DATA= data set.

```
data bygroups;
   format tid date.;
   input tid : date. by @@;
datalines;
24NOV09 1 25NOV09 1 26NOV09 1 27NOV09 1 30NOV09 1 01DEC09 1 02DEC09 1 03DEC09 1

   ... more lines ...
```

The following TIMEID procedure statements generate two data sets that summarize a data set with four BY groups.

```
proc timeid data=bygroups outintervaldetails=int outinterval=intsum;
   id tid;
   by by;
run;
```

The summarized information in Output 38.3.1 shows that BY groups 2, 3, and 4 in the ByGroups data set contain some duplicate values and spans, and group 1 conforms exactly to the WEEKDAY17W interval. This listing also shows that the date ranges in these two BY groups start and end on different days and that they overlap between December 7, 2009, and December 28, 2009.

Output 38.3.1 Selected Variables in the Combined OUTINTERVALDETAILS= OUTINTERVAL= Data Sets

by	N	NINTCNTS	PCTINTCNTS	NOFFSETS	PCTOFFSETS	NSPANS	PCTSPANS	STATUS
1	25	1	0.00	1	0	1	0.00000	0
2	25	2	0.08	1	0	2	0.00000	0
3	25	2	0.16	1	0	2	0.04348	1
4	25	2	0.24	1	0	2	0.13043	1
.	100	.	.	.	.	.	.	1

INTERVAL	START	END	SEASONALITY	NSEASONCYCLES	STARTSHARED	ENDSHARED
WEEKDAY17W	24NOV09	28DEC09	5	5	.	.
WEEKDAY17W	27NOV09	31DEC09	5	5	.	.
WEEKDAY17W	02DEC09	05JAN10	5	5	.	.
WEEKDAY17W	07DEC09	08JAN10	5	4	.	.
WEEKDAY17W	24NOV09	08JAN10	5	.	07DEC09	28DEC09

NBY	TOTALSEASONCYCLES	SEASONCYCLESSHARED
.	.	.
.	.	.
.	.	.
.	.	.
4	6	3

Chapter 39
The TIMESERIES Procedure

Contents

Overview: TIMESERIES Procedure **2658**
Getting Started: TIMESERIES Procedure **2659**
Syntax: TIMESERIES Procedure **2662**
- Functional Summary 2662
- PROC TIMESERIES Statement 2665
- BY Statement 2669
- CORR Statement 2669
- COUNT Statement (Experimental) 2671
- CROSSCORR Statement 2672
- DECOMP Statement 2673
- ID Statement 2675
- SEASON Statement 2678
- SPECTRA Statement 2679
- SSA Statement 2681
- TREND Statement 2684
- VAR and CROSSVAR Statements 2685

Details: TIMESERIES Procedure **2686**
- Accumulation 2687
- Missing Value Interpretation 2689
- Time Series Transformation 2690
- Time Series Differencing 2690
- Descriptive Statistics 2691
- Seasonal Decomposition 2691
- Correlation Analysis 2693
- Cross-Correlation Analysis 2694
- Spectral Density Analysis 2695
- Singular Spectrum Analysis 2699
- Data Set Output 2701
- OUT= Data Set 2701
- OUTCORR= Data Set 2702
- OUTCROSSCORR= Data Set 2703
- OUTDECOMP= Data Set 2704
- OUTFREQ= Data Set 2705
- OUTPROCINFO= Data Set 2705
- OUTSEASON= Data Set 2705
- OUTSPECTRA= Data Set 2706

OUTSSA= Data Set 2707
OUTSUM= Data Set 2707
OUTTREND= Data Set 2708
STATUS Variable Values 2709
Printed Output 2709
ODS Table Names 2710
ODS Graphics Names 2710
Examples: TIMESERIES Procedure **2712**
Example 39.1: Accumulating Transactional Data into Time Series Data 2712
Example 39.2: Trend and Seasonal Analysis 2714
Example 39.3: Illustration of ODS Graphics 2718
Example 39.4: Illustration of Spectral Analysis 2723
Example 39.5: Illustration of Singular Spectrum Analysis 2725
References **2727**

Overview: TIMESERIES Procedure

The TIMESERIES procedure analyzes time-stamped transactional data with respect to time and accumulates the data into a time series format. The procedure can perform trend and seasonal analysis on the transactions. After the transactional data are accumulated, time domain and frequency domain analysis can be performed on the accumulated time series.

For seasonal analysis of the transaction data, various statistics can be computed for each season. For trend analysis of the transaction data, various statistics can be computed for each time period. The analysis is similar to applying the MEANS procedure of Base SAS software to each season or time period of concern.

After the transactional data are accumulated to form a time series and any missing values are interpreted, the accumulated time series can be functionally transformed using log, square root, logistic, or Box-Cox transformations. The time series can be further transformed using simple and/or seasonal differencing. After functional and difference transformations have been applied, the accumulated and transformed time series can be stored in an output data set. This working time series can then be analyzed further using various time series analysis techniques provided by this procedure or other SAS/ETS procedures.

Time series analyses performed by the TIMESERIES procedure include:

- descriptive (global) statistics
- seasonal decomposition/adjustment analysis
- correlation analysis
- cross-correlation analysis
- spectral analysis

All results of the transactional or time series analysis can be stored in output data sets or printed using the Output Delivery System (ODS).

The TIMESERIES procedure can process large amounts of time-stamped transactional data. Therefore, the analysis results are useful for large-scale time series analysis or (temporal) data mining. All of the results can be stored in output data sets in either a time series format (default) or in a coordinate format (transposed). The time series format is useful for preparing the data for subsequent analysis with other SAS/ETS procedures. For example, the working time series can be further analyzed, modeled, and forecast with other SAS/ETS procedures. The coordinate format is useful when using this procedure with SAS/STAT procedures or SAS Enterprise Miner. For example, clustering time-stamped transactional data can be achieved by using the results of this procedure with the clustering procedures of SAS/STAT and the nodes of SAS Enterprise Miner.

The EXPAND procedure can be used for the frequency conversion and transformations of time series output from this procedure.

Getting Started: TIMESERIES Procedure

This section outlines the use of the TIMESERIES procedure and gives a cursory description of some of the analysis techniques that can be performed on time-stamped transactional data.

Given an input data set that contains numerous transaction variables recorded over time at no specific frequency, the TIMESERIES procedure can form time series as follows:

```
PROC TIMESERIES DATA=<input-data-set>
                OUT=<output-data-set>;
   ID <time-ID-variable> INTERVAL=<frequency>
                         ACCUMULATE=<statistic>;
   VAR <time-series-variables>;
RUN;
```

The TIMESERIES procedure forms time series from the input time-stamped transactional data. It can provide results in output data sets or in other output formats by using the Output Delivery System (ODS).

Time-stamped transactional data are often recorded at no fixed interval. Analysts often want to use time series analysis techniques that require fixed-time intervals. Therefore, the transactional data must be accumulated to form a fixed-interval time series.

Suppose that a bank wants to analyze the transactions associated with each of its customers over time. Further, suppose that the data set WORK.TRANSACTIONS contains four variables that are related to these transactions: CUSTOMER, DATE, WITHDRAWAL, and DEPOSITS. The following examples illustrate possible ways to analyze these transactions by using the TIMESERIES procedure.

To accumulate the time-stamped transactional data to form a daily time series based on the accumulated daily totals of each type of transaction (WITHDRAWALS and DEPOSITS), the following TIMESERIES procedure statements can be used:

```
proc timeseries data=transactions
                out=timeseries;
   by customer;
   id date interval=day accumulate=total;
   var withdrawals deposits;
run;
```

The OUT=TIMESERIES option specifies that the resulting time series data for each customer is to be stored in the data set WORK.TIMESERIES. The INTERVAL=DAY option specifies that the transactions are to be accumulated on a daily basis. The ACCUMULATE=TOTAL option specifies that the sum of the transactions is to be calculated. After the transactional data is accumulated into a time series format, many of the procedures provided with SAS/ETS software can be used to analyze the resulting time series data.

For example, the ARIMA procedure can be used to model and forecast each customer's withdrawal data by using an ARIMA$(0,1,1)(0,1,1)_s$ model (where the number of seasons is s=7 days in a week) using the following statements:

```
proc arima data=timeseries;
   identify var=withdrawals(1,7) noprint;
   estimate q=(1)(7) outest=estimates noprint;
   forecast id=date interval=day out=forecasts;
quit;
```

The OUTEST=ESTIMATES data set contains the parameter estimates of the model specified. The OUT=FORECASTS data set contains forecasts based on the model specified. See the SAS/ETS ARIMA procedure for more detail.

A single set of transactions can be very large and must be summarized in order to analyze them effectively. Analysts often want to examine transactional data for trends and seasonal variation. To analyze transactional data for trends and seasonality, statistics must be computed for each time period and season of concern. For each observation, the time period and season must be determined and the data must be analyzed based on this determination.

The following statements illustrate how to use the TIMESERIES procedure to perform trend and seasonal analysis of time-stamped transactional data.

```
proc timeseries data=transactions out=out
                outseason=season outtrend=trend;
   by customer;
   id date interval=day accumulate=total;
   var withdrawals deposits;
run;
```

Since the INTERVAL=DAY option is specified, the length of the seasonal cycle is seven (7) where the first season is Sunday and the last season is Saturday. The output data set specified by the OUTSEASON=SEASON option contains the seasonal statistics for each day of the week by each customer. The output data set specified by the OUTTREND=TREND option contains the trend statistics for each day of the calendar by each customer.

Often it is desired to seasonally decompose into seasonal, trend, cycle, and irregular components or to seasonally adjust a time series. The following techniques describe how the changing seasons influence the time series.

The following statements illustrate how to use the TIMESERIES procedure to perform seasonal adjustment/decomposition analysis of time-stamped transactional data.

```
proc timeseries data=transactions
                out=out
                outdecomp=decompose;
   by customer;
   id date interval=day accumulate=total;
   var withdrawals deposits;
run;
```

The output data set specified by the OUTDECOMP=DECOMPOSE data set contains the decomposed/adjusted time series for each customer.

A single time series can be very large. Often, a time series must be summarized with respect to time lags in order to be efficiently analyzed using time domain techniques. These techniques help describe how a current observation is related to the past observations with respect to the time (season) lag.

The following statements illustrate how to use the TIMESERIES procedure to perform time domain analysis of time-stamped transactional data.

```
proc timeseries data=transactions
                out=out
                outcorr=timedomain;
   by customer;
   id date interval=day accumulate=total;
   var withdrawals deposits;
run;
```

The output data set specified by the OUTCORR=TIMEDOMAIN data set contains the time domain statistics, such as sample autocorrelations and partial autocorrelations, by each customer.

Sometimes time series data contain underlying patterns that can be identified using spectral analysis techniques. Two kinds of spectral analyses on univariate data can be performed using the TIMESERIES procedure. They are singular spectrum analysis and Fourier spectral analysis.

Singular spectrum analysis (SSA) is a technique for decomposing a time series into additive components and categorizing these components based on the magnitudes of their contributions. SSA uses a single parameter, the window length, to quantify patterns in a time series without relying on prior information about the series' structure. The window length represents the maximum lag that is considered in the analysis, and it corresponds to the dimensionality of the principle components analysis (PCA) on which SSA is based. The components are combined into groups to categorize their roles in the SSA decomposition.

Fourier spectral analysis decomposes a time series into a sum of harmonics. In the discrete Fourier transform, the contribution of components at evenly spaced frequencies are quantified in a periodogram and summarized in spectral density estimates.

The following statements illustrate how to use the TIMESERIES procedure to analyze time-stamped transactional data without prior information about the series' structure.

```
proc timeseries data=transactions
                outssa=ssa
                outspectra=spectra;
   by customer;
   id date interval=day accumulate=total;
   var withdrawals deposits;
run;
```

The output data set specified by the OUTSSA=SSA data set contains a singular spectrum analysis of the withdrawals and deposits data. The data set specified by OUTSPECTRA=SPECTRA contains a Fourier spectral decomposition of the same data.

By default, the TIMESERIES procedure produces no printed output.

Syntax: TIMESERIES Procedure

The TIMESERIES Procedure uses the following statements:

PROC TIMESERIES *options* ;
BY *variables* ;
CORR *statistics-list* / *options* ;
CROSSCORR *statistics-list* / *options* ;
CROSSVAR *variable-list* / *options* ;
COUNT / *options* ;
DECOMP *component-list* / *options* ;
ID *variable* **INTERVAL**= *interval-option* ;
SEASON *statistics-list* / *options* ;
SPECTRA *statistics-list* / *options* ;
SSA / *options* ;
TREND *statistics-list* / *options* ;
VAR *variable-list* / *options* ;

Functional Summary

Table 39.1 summarizes the statements and options that control the TIMESERIES procedure.

Table 39.1 TIMESERIES Functional Summary

Description	Statement	Option
Statements		
Specifies BY-group processing	BY	
Specifies variables to analyze	VAR	
Specifies cross variables to analyze	CROSSVAR	
Specifies the time ID variable	ID	
Specifies correlation options	CORR	
Specifies cross-correlation options	CROSSCORR	
Specifies discrete distribution analysis options	COUNT	
Specifies decomposition analysis options	DECOMP	
Specifies seasonal statistics options	SEASON	
Specifies spectral analysis options	SPECTRA	
Specifies SSA options	SSA	
Specifies trend statistics options	TREND	
Data Set Options		
Specifies the input data set	PROC TIMESERIES	DATA=
Specifies the output data set	PROC TIMESERIES	OUT=
Specifies the correlations output data set	PROC TIMESERIES	OUTCORR=
Specifies the cross-correlations output data set	PROC TIMESERIES	OUTCROSSCORR=
Specifies the decomposition output data set	PROC TIMESERIES	OUTDECOMP=
Specifies the frequency (count) output data set	PROC TIMESERIES	OUTFREQ=

Description	Statement	Option
Specifies the SAS log output data set	PROC TIMESERIES	OUTPROCINFO=
Specifies the seasonal statistics output data set	PROC TIMESERIES	OUTSEASON=
Specifies the spectral analysis output data set	PROC TIMESERIES	OUTSPECTRA=
Specifies the SSA output data set	PROC TIMESERIES	OUTSSA=
Specifies the summary statistics output data set	PROC TIMESERIES	OUTSUM=
Specifies the trend statistics output data set	PROC TIMESERIES	OUTTREND=
Accumulation and Seasonality Options		
Specifies the accumulation frequency	ID	INTERVAL=
Specifies the length of seasonal cycle	PROC TIMESERIES	SEASONALITY=
Specifies the interval alignment	ID	ALIGN=
Specifies the interval boundary alignment	ID	BOUNDARYALIGN=
Specifies that time ID variable values not be sorted	ID	NOTSORTED
Specifies the starting time ID value	ID	START=
Specifies the ending time ID value	ID	END=
Specifies the accumulation statistic	ID, VAR, CROSSVAR	ACCUMULATE=
Specifies missing value interpretation	ID, VAR, CROSSVAR	SETMISSING=
Time-Stamped Data Seasonal Statistics Options		
Specifies the form of the output data set	SEASON	TRANSPOSE=
Fourier Spectral Analysis Options		
Specifies whether to adjust to the series mean	SPECTRA	ADJUSTMEAN=
Specifies confidence limits	SPECTRA	ALPHA=
Specifies the kernel weighting function	SPECTRA	PARZEN | BARTLETT | TUKEY | TRUNC | QS
Specifies the domain where kernel functions apply	SPECTRA	DOMAIN=
Specifies the constant kernel scale parameter	SPECTRA	C=
Specifies the exponent kernel scale parameter	SPECTRA	EXPON=
Specifies the periodogram weights	SPECTRA	WEIGHTS
Singular Spectrum Analysis Options		
Specifies whether to adjust to the series mean	SSA	ADJUSTMEAN=
Specifies the grouping of principal components	SSA	GROUPS=
Specifies the window length	SSA	LENGTH=
Specifies the number of time periods in the transposed output	SSA	NPERIODS=
Specifies the division between principal component groupings	SSA	THRESHOLDPCT
Specifies that the output be transposed	SSA	TRANSPOSE=

Description	Statement	Option
Time-Stamped Data Trend Statistics Options		
Specifies the form of the output data set	TREND	TRANSPOSE=
Specifies the number of time periods to be stored	TREND	NPERIODS=
Time Series Transformation Options		
Specifies simple differencing	VAR, CROSSVAR	DIF=
Specifies seasonal differencing	VAR, CROSSVAR	SDIF=
Specifies transformation	VAR, CROSSVAR	TRANSFORM=
Time Series Correlation Options		
Specifies the list of lags	CORR	LAGS=
Specifies the number of lags	CORR	NLAG=
Specifies the number of parameters	CORR	NPARMS=
Specifies the form of the output data set	CORR	TRANSPOSE=
Time Series Cross-Correlation Options		
Specifies the list of lags	CROSSCORR	LAGS=
Specifies the number of lags	CROSSCORR	NLAG=
Specifies the form of the output data set	CROSSCORR	TRANSPOSE=
Time Series Decomposition Options		
Specifies the mode of decomposition	DECOMP	MODE=
Specifies the Hodrick-Prescott filter parameter	DECOMP	LAMBDA=
Specifies the number of time periods to be stored	DECOMP	NPERIODS=
Specifies the form of the output data set	DECOMP	TRANSPOSE=
Time Series Discrete Distribution Analysis Options		
Specifies the confidence limit size	COUNT	ALPHA=
Specifies the discrete distribution selection criterion	COUNT	CRITERION=
Specifies one or more discrete distributions	COUNT	DISTRIBUTION=
Printing Control Options		
Specifies the time ID format	ID	FORMAT=
Specifies which output to print	PROC TIMESERIES	PRINT=
Specifies that detailed output be printed	PROC TIMESERIES	PRINTDETAILS
Miscellaneous Options		
Specifies that analysis variables be processed in sorted order	PROC TIMESERIES	SORTNAMES
Limits error and warning messages	PROC TIMESERIES	MAXERROR=

Description	Statement	Option
ODS Graphics Options		
Specifies the count series graphical output	PROC TIMESERIES	COUNTPLOTS=
Specifies the cross-variable graphical output	PROC TIMESERIES	CROSSPLOTS=
Specifies the variable graphical output	PROC TIMESERIES	PLOTS=
Specifies the vector time series graphical output	PROC TIMESERIES	VECTORPLOTS=

PROC TIMESERIES Statement

PROC TIMESERIES *options* **;**

You can specify the following *options*:

DATA=*SAS-data-set*
names the SAS data set that contains the input data for the procedure to create the time series. If the DATA= option is not specified, the most recently created SAS data set is used.

COUNTPLOTS=*option* **|** **(** *options* **)** (Experimental)
specifies the count series graphical output to be produced. You can specify the following plotting *options*:

COUNTS	plots the counts of the discrete values of the time series (OUTFREQ= data set).
CHISQPROB | CHISQ	plots the chi-square probabilities.
DISTRIBUTION | DIST	plots the discrete probability distribution.
VALUES	plots the distinct values of the time series (OUTFREQ= data set).
ALL	is equivalent to PLOTS=(COUNTS CHISQPROB DISTRIBUTION VALUES).

The COUNTPLOTS= option produces graphical results similar to the information contained in the data sets that are listed in parentheses next to the options.

By default, the TIMESERIES procedure produces no graphical output.

CROSSPLOTS=*option* **|** **(** *options* **)**
specifies the cross-variable graphical output to be produced. You can specify the following plotting *options*:

SERIES	plots the two time series (OUT= data set).
CCF	plots the cross-correlation functions (OUTCROSSCORR= data set).
ALL	is equivalent to PLOTS=(SERIES CCF).

The CROSSPLOTS= option produces results similar to the information contained in the data sets that are listed in parentheses next to the options.

By default, the TIMESERIES procedure produces no graphical output.

MAXERROR=*number*
limits the number of warning and error messages that are produced during the execution of the procedure to the specified value. This option is particularly useful in BY-group processing, where it can be used to suppress the recurring messages. By default, MAXERRORS=50.

MAXVARLENGTH
specifies that processed variables be set to eight bytes in length. This option exists principally for use when data storage might be a concern.

OUT=*SAS-data-set*
names the output data set to contain the time series variables that are specified in the subsequent VAR and CROSSVAR statements. If BY variables are specified, they are also included in the OUT= data set. If an ID variable is specified, it is also included in the OUT= data set. The values are accumulated based on the INTERVAL= or the ACCUMULATE= option (or both) in the ID statement. The OUT= data set is particularly useful when you want to further analyze, model, or forecast the resulting time series with other SAS/ETS procedures.

OUTCORR=*SAS-data-set*
names the output data set to contain the univariate time domain statistics.

OUTCROSSCORR=*SAS-data-set*
names the output data set to contain the cross-correlation statistics.

OUTDECOMP=*SAS-data-set*
names the output data set to contain the decomposed or seasonally adjusted time series (or both).

OUTFREQ=*SAS-data-set*
names the output data set to contain the frequency (count) analysis.

OUTPROCINFO=*SAS-data-set*
names the output data set to contain information in the SAS log, specifically the number of notes, errors, and warnings and the number of series processed, number of analyses requested, and number of analyses failed.

OUTSEASON=*SAS-data-set*
names the output data set to contain the seasonal statistics. The statistics are computed for each season as specified by the INTERVAL= option in the ID statement or the SEASONALITY= option in the PROC TIMESERIES statement. The OUTSEASON= data set is particularly useful when analyzing transactional data for seasonal variations.

OUTSPECTRA=*SAS-data-set*
names the output data set to contain the univariate frequency domain analysis results.

OUTSSA=*SAS-data-set*
names the output data set to contain the singular spectrum analysis result series.

OUTSUM=*SAS-data-set*
names the output data set to contain the descriptive statistics. The descriptive statistics are based on the accumulated time series when the ACCUMULATE= or SETMISSING= options are specified in the ID or VAR statements. The OUTSUM= data set is particularly useful when you analyze large numbers of series and you need a summary of the results.

OUTTREND=*SAS-data-set*
names the output data set to contain the trend statistics. The statistics are computed for each time period as specified by the INTERVAL= option in the ID statement . The OUTTREND= data set is particularly useful when you analyze transactional data for trends.

PLOTS=*option* | **(** *options* **)**
specifies the univariate graphical output desired. By default, the TIMESERIES procedure produces no graphical output. You can specify the following plotting *options*:

SERIES plots the time series (OUT= data set).

RESIDUAL plots the residual time series (OUT= data set).

HISTOGRAM plots a histogram of the time series values

CYCLES plots the seasonal cycles (OUT= data set).

CORR plots the correlation panel (OUTCORR= data set).

ACF plots the autocorrelation function (OUTCORR= data set).

PACF plots the partial autocorrelation function (OUTCORR= data set).

IACF plots the inverse autocorrelation function (OUTCORR= data set).

WN plots the white noise probabilities (OUTCORR= data set).

DECOMP plots the seasonal adjustment panel (OUTDECOMP= data set).

TCS plots the trend-cycle-seasonal component (OUTDECOMP= data set).

TCC plots the trend-cycle component (OUTDECOMP= data set).

SIC plots the seasonal-irregular component (OUTDECOMP= data set).

SC plots the seasonal component (OUTDECOMP= data set).

SA plots the seasonal adjusted component (OUTDECOMP= data set).

PCSA plots the percent change in the seasonal adjusted component (OUTDECOMP= data set).

IC plots the irregular component (OUTDECOMP= data set).

TC plots the trend component (OUTDECOMP= data set).

CC plots the cycle component (OUTDECOMP= data set).

PERIODOGRAM< *(suboption)* > plots the periodogram (OUTSPECTRA= data set). You can specify the following *suboptions*:

MAXFREQ=*number* specifies the maximum frequency in radians to include in the plot.

MINPERIOD=*number* specifies the minimum period to include in the plot.

SPECTRUM< *(suboption)* > plots the spectral density estimate (OUTSPECTRA= data set). You can specify the following *suboptions*:

MAXFREQ=*number* specifies the maximum frequency in radians to include in the plot.

MINPERIOD=*number* specifies the minimum period to include in the plot.

SSA plots the singular spectrum analysis results (OUTSSA= data set).

ALL is equivalent to PLOTS=(SERIES HISTOGRAM ACF PACF IACF WN SSA PERIODOGRAM SPECTRUM).

BASIC is equivalent to PLOTS=(SERIES HISTOGRAM CYCLES CORR DECOMP)

The PLOTS= option produces graphical output for these results by using the Output Delivery System (ODS). The PLOTS= option produces results similar to the data sets listed in parentheses next to the preceding options.

PRINT=*option* | **(** *options* **)**

specifies the printed output desired. By default, the TIMESERIES procedure produces no printed output. You can specify the following printing *options*:

COUNTS prints the discrete distribution analysis (OUTFREQ= data set). This *PRINT=COUNTS* is experimental.

DECOMP prints the seasonal decomposition/adjustment table (OUTDECOMP= data set).

SEASONS prints the seasonal statistics table (OUTSEASON= data set).

DESCSTATS prints the descriptive statistics for the accumulated time series (OUTSUM= data set).

SUMMARY prints the descriptive statistics table for all time series (OUTSUM= data set).

TRENDS prints the trend statistics table (OUTTREND= data set).

SSA prints the singular spectrum analysis results (OUTSSA= data set).

ALL is equivalent to PRINT=(DESCSTATS SUMMARY).

The PRINT= option produces printed output for these results by using the Output Delivery System (ODS). The PRINT= option produces results similar to the data sets listed in parentheses next to the preceding options.

PRINTDETAILS

requests that output specified in the PRINT= option be printed in greater detail.

SEASONALITY=*number*

specifies the length of the seasonal cycle. For example, SEASONALITY=3 means that every group of three time periods forms a seasonal cycle. By default, the length of the seasonal cycle is one (no seasonality) or the length implied by the INTERVAL= option specified in the ID statement. For example, INTERVAL=MONTH implies that the length of the seasonal cycle is 12.

SORTNAMES
requests that the variables specified in the VAR and CROSSVAR statements be processed in sorted order by the variable names. This option enables the output data sets to be presorted by the variable names.

VECTORPLOTS=*option* | **(** *options* **)**
specifies the vector time series graphical output to be produced. You can specify the following plotting *options*:

SCALED	plots each time series scaled between 0 and 1.
SERIES	plots each time series on a common axis without scaling.
STACKED	plots each time series on stacked thumbnail plots.
ALL	is equivalent to PLOTS=(SCALED SERIES STACKED).

By default, the TIMESERIES procedure produces no graphical output.

BY Statement

You can use a BY statement to obtain separate dummy variable definitions for groups of observations that are defined by the BY variables.

When a BY statement appears, the procedure expects the input data set to be sorted in order of the BY variables.

If your input data set is not sorted in ascending order, use one of the following alternatives:

- Sort the data by using the SORT procedure with a similar BY statement.
- Specify the option NOTSORTED or DESCENDING in the BY statement for the TIMESERIES procedure. The NOTSORTED option does not mean that the data are unsorted but rather that the data are arranged in groups (according to values of the BY variables) and that these groups are not necessarily in alphabetical or increasing numeric order.
- Create an index on the BY variables by using the DATASETS procedure.

For more information about the BY statement, see *SAS Language Reference: Concepts*. For more information about the DATASETS procedure, see *Base SAS Procedures Guide.*

CORR Statement

CORR *statistics* < / *options* > **;**

You can use a CORR statement to specify options that are related to time domain analysis of the accumulated time series. Only one CORR statement is allowed.

You can specify the following time domain *statistics*:

LAG	time lag
N	number of variance products
ACOV	autocovariances
ACF	autocorrelations
ACFSTD	autocorrelation standard errors
ACF2STD	an indicator of whether autocorrelations are less than (–1), greater than (1), or within (0) two standard errors of zero
ACFNORM	normalized autocorrelations
ACFPROB	autocorrelation probabilities
ACFLPROB	autocorrelation log probabilities
PACF	partial autocorrelations
PACFSTD	partial autocorrelation standard errors
PACF2STD	an indicator of whether partial autocorrelation are less than (–1), greater than (1), or within (0) two standard errors of zero
PACFNORM	partial normalized autocorrelations
PACFPROB	partial autocorrelation probabilities
PACFLPROB	partial autocorrelation log probabilities
IACF	inverse autocorrelations
IACFSTD	inverse autocorrelation standard errors
IACF2STD	an indicator of whether the inverse autocorrelation is less than (–1), greater than (1) or within (0) two standard errors of zero
IACFNORM	normalized inverse autocorrelations
IACFPROB	inverse autocorrelation probabilities
IACFLPROB	inverse autocorrelation log probabilities
WN	white noise test statistics
WNPROB	white noise test probabilities
WNLPROB	white noise test log probabilities

If you do not specify any *statistics*, then the default is as follows:

```
corr lag n acov acf acfstd pacf pacfstd iacf iacfstd wn wnprob;
```

You can specify the following *options* after a slash (/):

LAGS=*(numlist)*
specifies the list of lags to be stored in OUTCORR= data set or to be plotted. The list of lags must be separated by spaces or commas. For example, LAGS=(1,3) specifies the first then third lag.

NLAG=*number*
specifies the number of lags to be stored in the OUTCORR= data set or to be plotted. The default is 24 or three times the length of the seasonal cycle, whichever is smaller. The LAGS= option takes precedence over the NLAG= option.

NPARMS=*number*
specifies the number of parameters that are used in the model that created the residual time series. The number of parameters determines the degrees of freedom associated with the Ljung-Box statistics. This option is useful when you analyze the residuals of a time series model whose number of parameters is specified by *number*. By default, NPARMS=0.

TRANSPOSE=NO | YES
specifies which values are recorded as column names in the OUTCORR= data set. You can specify the following values:

NO specifies that correlation statistics be recorded as the column names. This option is useful for graphing the correlation results with SAS/GRAPH procedures.

YES specifies that lags be recorded as the column names instead of correlation statistics as the column names. This option is useful for analyzing the correlation results with other SAS procedures such as the CLUSTER procedure in SAS/STAT or with SAS Enterprise Miner software.

By default, TRANSPOSE=NO.

COUNT Statement (Experimental)

COUNT < */ options* > **;**

You can use a COUNT statement to specify options that are related to the discrete distribution analysis of the accumulated time series. Only one COUNT statement is allowed.

You can specify the following *options* after a slash (/):

ALPHA=*number*
specifies the confidence limit size. The *number* must be between 0 and 1; the default is 0.05.

CRITERION=LOGLIK | AIC | BIC
specifies the discrete distribution selection criterion. The default is CRITERION=LOGLIK.

You can specify the following selection criteria:

AIC specifies the Akaike's information criterion.

BIC specifies the Bayesian information criterion.

LOGLIK specifies the log-likelihood as the criterion.

By default, CRITERION=LOGLIK.

DISTRIBUTION= *option* | **(** *options* **)**
specifies one or more discrete distributions for automatic selection. You can specify one or more of the following *options*:

BINOMIAL	specifies the binomial distribution.
ZMBINOMIAL	specifies the zero-modified binomial distribution.
GEOMETRIC	specifies the geometric distribution.
ZMGEOMETRIC	specifies the zero-modified geometric distribution.
POISSON	specifies the Poisson distribution.
ZMPOISSON	specifies the zero-modified Poisson distribution.
NEGBINOMIAL | NEGBIN	specifies the negative binomial distribution.

CROSSCORR Statement

CROSSCORR *statistics* < */ options* > **;**

You can use a CROSSCORR statement to produce statistics that are related to cross-correlation analysis of the accumulated time series. Only one CROSSCORR statement is allowed.

You can specify the following time domain *statistics*:

LAG	time lag
N	number of variance products
CCOV	cross covariances
CCF	cross-correlations
CCFSTD	cross-correlation standard errors
CCF2STD	an indicator of whether cross-correlations are less than (–1), greater than (1), or within (0) two standard errors of zero
CCFNORM	normalized cross-correlations
CCFPROB	cross-correlation probabilities
CCFLPROB	cross-correlation log probabilities

If do not specify any *statistics*, the default is as follows:

```
crosscorr lag n ccov ccf ccfstd;
```

You can also specify the following *options* after a slash (/):

NLAG=*number*
specifies the number of lags to be stored in the OUTCROSSCORR= data set or to be plotted. The default is 24 or three times the length of the seasonal cycle, whichever is smaller. The LAGS= option takes precedence over the NLAG= option.

LAGS=*(numlist)*
specifies a list of lags to be stored in OUTCROSSCORR= data set or to be plotted. The list of lags must be separated by spaces or commas. For example, LAGS=(1,3) specifies the first then third lag.

TRANSPOSE=NO | YES
specifies which values are recorded as column names in the OUTCROSSCORR= data set. You can specify the following values:

NO specifies that cross-correlation statistics be recorded as the column names. This option is useful for graphing the cross-correlation results with SAS/GRAPH procedures.

YES specifies that lags instead of cross-correlation statistics be recorded as the column names. This option is useful for analyzing the cross-correlation results with other procedures such as the CLUSTER procedure in SAS/STAT or with SAS Enterprise Miner software.

By default, TRANSPOSE=NO.

DECOMP Statement

DECOMP *components* < */ options* > **;**

You can use a DECOMP statement to specify *options* that are related to classical seasonal decomposition of the time series data. Only one DECOMP statement is allowed. The *options* affect all variables that are listed in the VAR statements. Decomposition can be performed only when the length of the seasonal cycle specified by the SEASONALITY= option in the PROC TIMESERIES statement or implied by the INTERVAL= option in the ID statement is greater than 1.

You can specify the following seasonal decomposition *components*:

ORIG | ORIGINAL	original series
TCC | TRENDCYCLE	trend-cycle component
SIC | SEASONIRREGULAR	seasonal-irregular component
SC | SEASONAL	seasonal component
SCSTD	seasonal component standard errors
TCS | TRENDCYCLESEASON	trend-cycle-seasonal component
IC | IRREGULAR	irregular component
SA | ADJUSTED	seasonally adjusted series
PCSA	percent change seasonally adjusted series
TC	trend component
CC | CYCLE	cycle component

If you do not specify any components, then the default is as follows:

```
decomp orig tcc sc ic sa;
```

You can also specify the following *options* after a slash (/):

MODE=*option*
specifies the type of decomposition to be used to decompose the time series. You can specify the following *options*:

ADD | ADDITIVE	uses additive decomposition.
MULT | MULTIPLICATIVE	uses multiplicative decomposition.
LOGADD | LOGADDITIVE	uses log-additive decomposition.
PSEUDOADD | PSEUDOADDITIVE	uses pseudo-additive decomposition.
MULTORADD	uses multiplicative decomposition when the accumulated time series contains only positive values, uses pseudo-additive decomposition when the accumulated time series contains only nonnegative values, and uses additive decomposition otherwise.

Multiplicative and log additive decomposition require strictly positive time series. If the accumulated time series contains nonpositive values and MODE=MULT or MODE=LOGADD, an error results. Pseudo-additive decomposition requires a nonnegative-valued time series. If the accumulated time series contains negative values and the MODE=PSEUDOADD option is specified, an error results.

By default, MODE=MULTORADD.

LAMBDA=*number*
specifies the Hodrick-Prescott filter parameter for trend-cycle decomposition. Filtering applies when the trend component or the cycle component is requested. If filtering is not specified, this option is ignored. By default, LAMBDA=1600.

NPERIODS=*number*
specifies the *number* of time periods to be stored in the OUTDECOMP= data set when the TRANSPOSE=YES option is specified. If TRANSPOSE=NO, the NPERIODS= option is ignored. If *number* is positive, the first or beginning time periods are recorded. If *number* is negative, the last or ending time periods are recorded. The NPERIODS= option specifies the number of OUTDECOMP= data set variables to contain the seasonal decomposition and is therefore limited to the maximum allowed number of SAS variables. If the number of time periods exceeds this limit, a warning is printed in the log and the number of periods stored is reduced to the limit.

If the NPERIODS= option is not specified, all of the periods specified between the ID statement START= and END= options are stored. If at least one of the START= or END= options is not specified, the default magnitude is the seasonality specified in the SEASONALITY= option in the PROC TIMESERIES statement or implied by the INTERVAL= option in the ID statement. If only the START= option or both the START= and END= options are specified and the seasonality is zero, the default is NPERIODS=5. If only the END= option or neither the START= nor END= option is specified and the seasonality is zero, the default is NPERIODS=–5.

TRANSPOSE= NO | YES
specifies which values are recorded as column names in the OUTDECOMP= data set.

NO specifies that decomposition components be recorded as the column names. This option is useful for analyzing or displaying the decomposition results with SAS/GRAPH procedures.

YES specifies that the time periods be recorded as the column names instead of decomposition components. The first and last time periods stored in the OUTDECOMP= data set correspond to the period specified in the START= option and END= option, respectively, in the ID statement. If only the END= option is specified, the last time ID value of each accumulated time series corresponds to the last time period column. If only the START= option is specified, the first time ID value of each accumulated time series corresponds to the first time period column. If neither the START= option nor the END= option is specified in the ID statement, the first time ID value of each accumulated time series corresponds to the first time period column. This option is useful for analyzing the decomposition results with other SAS procedures or with SAS Enterprise Miner software.

By default, TRANSPOSE=NO.

ID Statement

ID *variable* **INTERVAL=***interval* < *options* > **;**

The ID statement names a numeric variable that identifies observations in the input and output data sets. The ID variable's values are assumed to be SAS date or datetime values. In addition, the ID statement specifies the frequency to be associated with the time series. The ID statement *option* also specify how the observations are accumulated and how the time ID values are aligned to form the time series. The specified information affects all variables that are listed in subsequent VAR statements. If you do not specify an ID statement, the observation number, with respect to the BY group, is used as the time ID.

You must specify the following argument:

INTERVAL=*interval*
specifies the frequency of the accumulated time series. For example, if the input data set consists of quarterly observations, then specify INTERVAL=QTR. If the PROC TIMESERIES statement SEASONALITY= option is not specified, the length of the seasonal cycle is implied from the INTERVAL= option. For example, INTERVAL=QTR implies a seasonal cycle of length 4. If the ACCUMULATE= option is also specified, the INTERVAL= option determines the time periods for the accumulation of observations. The INTERVAL= option is required and must be the first option specified in the ID statement.

You can also specify the following *options*:

ACCUMULATE=*option*
specifies how the data set observations are to be accumulated within each time period. The frequency (width of each time interval) is specified by the INTERVAL= *interval*. The ID variable contains the

time ID values. Each time ID variable value corresponds to a specific time period. The accumulated values form the time series, which is used in subsequent analysis.

This option is useful when there are zero or more than one input observations that coincide with a particular time period (for example, time-stamped transactional data). The EXPAND procedure offers additional frequency conversions and transformations that can also be useful in creating a time series.

You can specify the following *options*, which determine how the observations are accumulated within each time period based on the ID variable and on the frequency specified in INTERVAL= *interval*:

NONE	does not accumulate observations; the ID variable values must be equally spaced with respect to the frequency.
TOTAL	accumulates observations based on the total sum of their values.
AVERAGE | AVG	accumulates observations based on the average of their values.
MINIMUM | MIN	accumulates observations based on the minimum of their values.
MEDIAN | MED	accumulates observations based on the median of their values.
MAXIMUM | MAX	accumulates observations based on the maximum of their values.
N	accumulates observations based on the number of nonmissing observations.
NMISS	accumulates observations based on the number of missing observations.
NOBS	accumulates observations based on the number of observations.
FIRST	accumulates observations based on the first of their values.
LAST	accumulates observations based on the last of their values.
STDDEV |STD	accumulates observations based on the standard deviation of their values.
CSS	accumulates observations based on the corrected sum of squares of their values.
USS	accumulates observations based on the uncorrected sum of squares of their values.

If you specify the ACCUMULATE= option, the SETMISSING= option is useful for specifying how accumulated missing values are to be treated. If missing values are to be interpreted as 0, then specify SETMISSING=0. For more information about accumulation, see the section "Details: TIMESERIES Procedure" on page 2686.

By default, ACCUMULATE=NONE.

ALIGN=*option*

controls the alignment of SAS dates that are used to identify output observations. The ALIGN= option accepts the following values: BEGINNING | BEG | B, MIDDLE | MID | M, and ENDING | END | E. BEGINNING is the default.

BOUNDARYALIGN=*option*

controls how the ACCUMULATE= option is processed for the two boundary time intervals, which include the START= and END= time ID values. Some time ID values might fall inside the first and last accumulation intervals but fall outside the START= and END= boundaries. In these cases the BOUNDARYALIGN= option determines which values to include in the accumulation operation. You can specify the following *options*:

NONE	does not accumulate any values outside the START= and END= boundaries.
START	accumulates all observations in the first time interval.
END	accumulates all observations in the last time interval.
BOTH	accumulates all observations in the first and last.

For more information, see the section "Details: TIMESERIES Procedure" on page 2686. By default, BOUNDARYALIGN=NONE.

END=*value*
specifies a SAS date or datetime value that represents the end of the data. If the last time ID variable value is less than *value*, the series is extended with missing values. If the last time ID variable value is greater than *value*, the series is truncated. For example, END="&sysdate"D uses the automatic macro variable SYSDATE to extend or truncate the series to the current date. You can use the START= and END= options to ensure that data associated within each BY group contains the same number of observations.

FORMAT=*format*
specifies the SAS format for the time ID values. The default format is implied from the INTERVAL= option.

NOTSORTED
specifies that the time ID values might not be in sorted order. Prior to analysis, the TIMESERIES procedure sorts the data with respect to the time ID.

SETMISSING=*option* | *number*
specifies how missing values (either actual or accumulated) are to be interpreted in the accumulated time series. If you specify a *number*, missing values are set to the *number*. If a missing value indicates an unknown value, this option should not be used. If a missing value indicates no value, specify SETMISSING=0. You would typically use SETMISSING=0 for transactional data because no recorded data usually implies no activity. Instead of specifying a *number*, you can specify one of the following *options* to determine how missing values are assigned:

MISSING	sets missing values to missing.
AVERAGE | AVG	sets missing values to the accumulated average value.
MINIMUM | MIN	sets missing values to the accumulated minimum value.
MEDIAN | MED	sets missing values to the accumulated median value.
MAXIMUM | MAX	sets missing values to the accumulated maximum value.
FIRST	sets missing values to the accumulated first nonmissing value.
LAST	sets missing values to the accumulated last nonmissing value.
PREVIOUS | PREV	sets missing values to the previous period's accumulated nonmissing value. Missing values at the beginning of the accumulated series remain missing.
NEXT	sets missing values to the next period's accumulated nonmissing value. Missing values at the end of the accumulated series remain missing.

By default, SETMISSING=MISSING.

START=*value*
specifies a SAS date or datetime value that represents the beginning of the data. If the first time ID variable value is greater than *value*, missing values are added to the beginning of the series. If the first time ID variable value is less than *value*, the series is truncated. You can specify the START= and END= options to ensure that data associated with each BY group contains the same number of observations.

SEASON Statement

SEASON *statistics* < / *options* > **;**

You can use a SEASON statement to specify seasonal *statistics* and *options* that are related to seasonal analysis of the time-stamped transactional data. Only one SEASON statement is allowed. The *options* affect all variables specified in the VAR statements. Seasonal analysis can be performed only when the length of the seasonal cycle specified by the SEASONALITY= option in the PROC TIMESERIES statement or implied by the INTERVAL= option in the ID statement is greater than 1.

You can specify the following seasonal *statistics*:

NOBS	number of observations
N	number of nonmissing observations
NMISS	number of missing observations
MINIMUM	minimum value
MAXIMUM	maximum value
RANGE	range value
SUM	summation value
MEAN	mean value
STDDEV	standard deviation
CSS	corrected sum of squares
USS	uncorrected sum of squares
MEDIAN	median value

If you do not specify any of the seasonal *statistics*, then the default is as follows:

```
season n min max mean std;
```

You can also specify the following *options* after a slash (/):

TRANSPOSE= NO | YES
specifies which values are recorded as column names in the OUTSEASON= data set. You can specify the following values:

NO specifies that the seasonal statistics be recorded as the column names. This option is useful for graphing the seasonal analysis results with SAS/GRAPH procedures.

YES specifies that the seasonal indices instead of the seasonal statistics be recorded as the column names. This option is useful for analyzing the seasonal analysis results with SAS procedures or with SAS Enterprise Miner software.

By default, TRANSPOSE=NO.

SPECTRA Statement

SPECTRA *statistics* < / *options* > **;**

You can use a SPECTRA statement to specify which *statistics* appear in the OUTSPECTRA= data set. The SPECTRA statement *options* are used in performing a spectral analysis on the variables listed in the VAR statement. These *options* affect values that are produced in the PROC TIMESERIES statement's OUTSPECTRA= data set, and in the periodogram and spectral density estimate. Only one SPECTRA statement is allowed.

You can request the following univariate frequency domain *statistics*:

FREQ frequency in radians from 0 to π

PERIOD period or wavelength

COS cosine transform

SIN sine transform

P periodogram

S spectral density estimates

If you do not specify any frequency domain *statistics*, then the default is as follows:

```
spectra period p;
```

You can also specify the following *options* after a slash (/):

C=*coefficient*
specifies the scale coefficient for the kernel function. For more information, see the section "Kernel Option Details" on page 2681.

E=*exponent*

EXP=*exponent*

EXPON=*exponent*
specifies the exponent for the kernel function. For more information, see the section "Kernel Option Details" on page 2681.

ADJUSTMEAN=NO | YES

CENTER=NO | YES

specifies whether the series is to be adjusted by its mean prior to performing the Fourier decomposition. This adjustment sets the first periodogram ordinate to 0 rather than to $2n$ times the squared mean. This option is commonly used when the periodograms are to be plotted to prevent a large first periodogram ordinate from distorting the scale of the plot.

NO	specifies that no adjustment of the series be performed.
YES	specifies that the series be transformed by subtracting its mean.

By default, ADJUSTMEAN=NO.

ALPHA=*num*

specifies the width of a window that is drawn around the spectral density estimate in a spectral density versus frequency plot. Based on approximations proposed by Brockwell and Davis (1991), periodogram ordinates fall within this window with a confidence level of $1 - num$. The value *num* must be between 0 and 1; the default is 0.05.

DOMAIN=*domain*

specifies how the smoothing function is interpreted. You can specify the following *domain* values:

FREQUENCY	smooths the periodogram ordinates.
TIME	applies the kernel as a filter to the time series autocovariance function.

By default DOMAIN=FREQUENCY, and smoothing is applied in the same manner as weights are applied when you specify the WEIGHTS= option.

kernel

specifies the smoothing function to use to calculate a spectral density estimate as the moving average of periodogram ordinates. The kernel function is an alternative smoothing method to using the WEIGHTS= option. You can specify the following *kernel* values:

PARZEN	Parzen kernel
BARTLETT	Bartlett kernel
TUKEY	Tukey-Hanning kernel
TRUNC | TRUNCAT	truncated kernel
QS | QUADR	quadratic spectral kernel

If neither a WEIGHTS= option nor a *kernel* function is specified, the spectral density estimate is identical to the unmodified periodogram.

WEIGHTS=*numlist*

specifies the relative weights to use to compute a spectral density estimate as the moving average smoothing of periodogram ordinates. If neither a WEIGHTS= option nor a kernel function is specified, the spectral density estimate is identical to the unmodified periodogram. The following SPECTRA statement uses the WEIGHTS= option to specify equal weighting for each of the three adjacent periodogram ordinates that are centered on each spectral density estimate:

```
spectra / weights 1 1 1;
```

For information about how the weights are applied, see the section "Using Specification of Weight Constants" on page 2699.

Kernel Option Details

You can further parameterize each of the kernel functions with a kernel scale factor by using the C= and E= options. The default values of the kernel scale parameters, c and e, that are associated with each of the kernel functions together with their kernel scale factor values, M, for a series with 100 periodogram ordinates are listed in Table 39.2. The formula that is used to generate the table entries is $M = cK^e$, where K is the number of Fourier component frequencies.

Table 39.2 Default Kernel Scale Factor Parameters

Kernel	c	e	M
Bartlett	1/2	1/3	2.32
Parzen	1	1/5	2.51
Quadratic	1/2	1/5	1.26
Tukey-Hanning	2/3	1/5	1.67
Truncated	1/4	1/5	0.63

For example, to apply the truncated kernel by using default scale factor parameters in the frequency domain, you could use the following SPECTRA statement:

```
spectra / truncat;
```

For more information about the kernel function parameterization and the DOMAIN= option, see the section "Using Kernel Specifications" on page 2697.

SSA Statement

SSA < / *options* > **;**

The SSA statement requests singular spectrum analysis (SSA) of the accumulated time series. Only one SSA statement is allowed.

You can also specify the following *options* after a slash (/):

ADJUSTMEAN=NO | YES

CENTER=NO | YES

specifies whether the series should be adjusted by its mean prior to performing the singular spectrum analysis. You can specify the following values:

NO specifies that no adjustment of the series be performed.

YES specifies that the series be transformed by subtracting its mean.

By default, ADJUSTMEAN=NO.

GROUPS=*(numlist)...(numlist)*

specifies the lists that categorize window lags into groups. The window lags must be separated by spaces or commas. For example, GROUPS=(1,3) (2,4) specifies that the first and third window lags form the first group and the second and fourth window lags form the second group. If you do not specify the GROUPS= option, the window lags are divided into two groups based on the THRESHOLDPCT= value.

For example, the following SSA statement specifies three groups:

```
ssa / groups=(1 3)(2 4 5)(6);
```

The first group contains the first and third principal components; the second group contains the second, fourth, and fifth principal components; and the third group contains the sixth principal component.

By default, the first group contains the principal components whose contributions to the series sum to greater than the THRESHOLDPCT= value of 90%, and the second group contains the remaining components.

LENGTH=*number*

specifies the window length to be used in the analysis. The window length represents the maximum lag to be used in the SSA autocovariance calculations, where *number* must be greater than 1. When the SEASONALITY= option is provided or implied by the INTERVAL= option in the ID statement, the default window length is the smaller of two times the length of the seasonal cycle and one-half the length of the time series. When no seasonality value is available, the default window length is the smaller of 12 and one half the length of the time series.

For example, the following SSA statement specifies a window length of 10:

```
ssa / length=10;
```

If the specified *number* is greater than one-half the length of the accumulated time series, the window length is reduced and a warning message is printed to the log. If you do not specify the window length option and the INTERVAL=MONTH or SEASONALITY=12 options are specified, a window length of 24 is used.

NPERIODS=*number*

specifies the *number* of time periods to be stored in the OUTSSA= data set when the TRANSPOSE=YES option is specified. If the TRANSPOSE option is not specified, the NPERIODS= option is ignored. The NPERIODS= option specifies the number of OUTSSA= data set variables to contain the groups.

If you do not specify this option, all the periods specified between the START= and END= options are stored of the ID statement. If at least one of the START= or END= options is not specified, the default magnitude is the seasonality specified by the SEASONALITY= option in the PROC TIMESERIES statement or implied by the INTERVAL= option in the ID statement. If only the START= option or both

the START= and END= options are specified and the seasonality is zero, the default is NPERIODS=5. If only the END= option or neither the START= nor END= option is specified and the seasonality is zero, the default is NPERIODS=–5.

THRESHOLDPCT=*percentage*

specifies a *percentage* to be used to divide the SSA components into two groups based on the cumulative percentage of their singular values. The *percentage* must be between 0 and 100, inclusive. By default, THRESHOLDPCT=90.

For example, the following SSA statement specifies 80%:

```
ssa / THRESHOLDPCT=80;
```

The size of the second group must be at least 1, and it must be less than the window length. The *percentage* is adjusted to achieve this requirement.

For example, the following SSA statement specifies THRESHOLDPCT=0, which effectively sets the size of the second group to one less than the window length:

```
ssa / THRESHOLDPCT = 0;
```

The following SSA statement specifies 100%, which implies that the size of the last group is one:

```
ssa / THRESHOLDPCT= 100;
```

TRANSPOSE=NO | YES

specifies which values are recorded as column names in the OUTSSA= data set.

NO specifies that the specified groups be recorded as the column names. This option is useful for displaying the SSA results.

YES specifies that the time periods instead of the specified groups be recorded as the column names. The first and last time periods stored in the OUTSSA= data set correspond to the periods that are specified in the START= and END= options, respectively, in the ID statement. If only the END= option is specified in the ID statement, the last time ID value of each accumulated time series corresponds to the last time period column. If only the START= option is specified in the ID statement, the first time ID value of each accumulated time series corresponds to the first time period column. If neither the START= option nor the END= option is specified in the ID statement, the first time ID value of each accumulated time series corresponds to the first time period column. This option is useful for analyzing the SSA results using SAS Enterprise Miner software.

By default, TRANSPOSE=NO.

TREND Statement

TREND *statistics* < / *options* > **;**

You can use a TREND statement to specify *statistics* and related *options* for trend analysis of the time-stamped transactional data. Only one TREND statement is allowed. The specified *options* affect all variables that are listed in the VAR statements.

You can specify the following trend *statistics*:

NOBS	number of observations
N	number of nonmissing observations
NMISS	number of missing observations
MINIMUM	minimum value
MAXIMUM	maximum value
RANGE	range value
SUM	summation value
MEAN	mean value
STDDEV	standard deviation
CSS	corrected sum of squares
USS	uncorrected sum of squares
MEDIAN	median value

If you do not specify any trend *statistics*, the default is as follows:

```
trend n min max mean std;
```

You can also specify the following *options* after a slash (/):

NPERIODS=*number*

specifies the number of time periods to be stored in the OUTTREND= data set when the TRANSPOSE=YES option is specified. If the TRANSPOSE option is not specified, the NPERIODS= option is ignored. The NPERIODS= option specifies the number of OUTTREND= data set variables to contain the trend statistics and is therefore limited to the maximum allowed number of SAS variables.

If you do not specify this option, all the periods that are specified between the START= and END= options in the ID statement are stored. If at least one of the START= or END= options is not specified, the default magnitude is the seasonality that is specified in the SEASONALITY= option in the PROC TIMESERIES statement or implied by the INTERVAL= option in the ID statement. If only the START= option or both the START= and END= options are specified and the seasonality is zero, the default is NPERIODS=5. If only the END= option or neither the START= nor END= option is specified and the seasonality is zero, the default is NPERIODS=–5.

TRANSPOSE=NO | YES
specifies which values are recorded as column names in the OUTTREND= data set.

NO specifies that the specified groups be recorded as the column names. This option is useful for displaying the SSA results.

YES specifies that the time periods instead of the specified groups be recorded as the column names. The first and last time periods stored in the OUTSSA= data set correspond to the periods that are specified in the START= and END= options, respectively, in the ID statement. If only the END= option is specified in the ID statement, the last time ID value of each accumulated time series corresponds to the last time period column. If only the START= option is specified in the ID statement, the first time ID value of each accumulated time series corresponds to the first time period column. If neither the START= option nor the END= option is specified in the ID statement, the first time ID value of each accumulated time series corresponds to the first time period column. This option is useful for analyzing the SSA results using SAS Enterprise Miner software.

By default, TRANSPOSE=NO.

VAR and CROSSVAR Statements

VAR *variable-list* < / *options* > **;**

CROSSVAR *variable-list* < / *options* > **;**

The VAR and CROSSVAR statements list the numeric variables in the DATA= data set whose values are to be accumulated to form the time series.

An input data set variable can be specified in only one VAR or CROSSVAR statement. You can specify any number of VAR and CROSSVAR statements. You can specify the following *options* for either the VAR and CROSSVAR statement:

ACCUMULATE=*option*
specifies how the data set observations are to be accumulated within each time period for the variables in the *variable-list*. If you do not specify the ACCUMULATE= option in the VAR or CROSSVAR statement, accumulation is determined by the ACCUMULATE= option in the ID statement. For more information, see the ACCUMULATE= option in the ID statement.

DIF=*(numlist)*
specifies the differencing to be applied to the accumulated time series. The list of differencing orders must be separated by spaces or commas. For example, DIF=(1,3) specifies first then third order differencing. Differencing is applied after time series transformation. The TRANSFORM= option is applied before the DIF= option.

SDIF=*(numlist)*
specifies the seasonal differencing to be applied to the accumulated time series. The list of seasonal differencing orders must be separated by spaces or commas. For example, SDIF=(1,3) specifies first then third order seasonal differencing. Differencing is applied after time series transformation. The TRANSFORM= option is applied before the SDIF= option.

SETMISS=*option* | *number*

SETMISSING=*option* | *number*

specifies how missing values (either actual or accumulated) are to be interpreted in the accumulated time series for variables in the *variable-list*. If the SETMISSING= option is not specified in the VAR or CROSSVAR statement, missing values are set based on the SETMISSING= option of the ID statement. For more information, see the SETMISSING= option in the ID statement.

TRANSFORM=*transformation*

specifies the time series transformation to be applied to the accumulated time series. When you specify the TRANSFORM= option, the time series must be strictly positive. You can specify the following *transformations*:

NONE	does not apply any transformation.
LOG	logarithmic transformation
SQRT	square-root transformation
LOGISTIC	logistic transformation
BOXCOX(*n*)	Box-Cox transformation with parameter *n* where the *n* is between –5 and 5

By default, TRANSFORM=NONE.

Details: TIMESERIES Procedure

The TIMESERIES procedure can be used to perform trend and seasonal analysis on transactional data. For trend analysis, various sample statistics are computed for each time period defined by the time ID variable and INTERVAL= option. For seasonal analysis, various sample statistics are computed for each season defined by the INTERVAL= or the SEASONALITY= option. For example, if the transactional data ranges from June 1990 to January 2000 and the data are to be accumulated on a monthly basis, then the trend statistics are computed for every month: June 1990, July 1990, . . . , January 2000. The seasonal statistics are computed for each season: January, February, . . . , December.

The TIMESERIES procedure can be used to form time series data from transactional data. The accumulated time series can then be analyzed using time series techniques. The data is analyzed in the following order:

1.	accumulation	ACCUMULATE= option in the ID, VAR, or CROSSVAR statement
2.	missing value interpretation	SETMISSING= option in the ID, VAR, or CROSSVAR statement
3.	time series transformation	TRANSFORM= option in the VAR or CROSSVAR statement
4.	time series differencing	DIF= and SDIF= options in the VAR or CROSSVAR statement
5.	descriptive statistics	OUTSUM= option and the PRINT=DESCSTATS option
6.	seasonal decomposition	DECOMP statement or the OUTDECOMP= option in the PROC TIMESERIES statement

7.	correlation analysis	CORR statement or the OUTCORR= option in the PROC TIMESERIES statement
8.	singular spectrum analysis	SSA statement or the OUTSSA= option in the PROC TIMESERIES statement
9.	Fourier spectral analysis	SPECTRA statement or the OUTSPECTRA= option in the PROC TIMESERIES statement
10.	cross-correlation analysis	CROSSCORR statement or the OUTCROSSCORR= option in the PROC TIMESERIES statement

Accumulation

If the ACCUMULATE= option in the ID, VAR, or CROSSVAR statement is specified, data set observations are accumulated within each time period. The frequency (width of each time interval) is specified by the ID statement INTERVAL= option. The ID variable contains the time ID values. Each time ID value corresponds to a specific time period. Accumulation is useful when the input data set contains transactional data, whose observations are not spaced with respect to any particular time interval. The accumulated values form the time series, which is used in subsequent analyses.

For example, suppose a data set contains the following observations:

```
19MAR1999    10
19MAR1999    30
11MAY1999    50
12MAY1999    20
23MAY1999    20
```

If the INTERVAL=MONTH is specified, all of the above observations fall within a three-month period of time between March 1999 and May 1999. The observations are accumulated within each time period as follows:

If the ACCUMULATE=NONE option is specified, an error is generated because the ID variable values are not equally spaced with respect to the specified frequency (MONTH).

If the ACCUMULATE=TOTAL option is specified, the resulting time series is:

```
01MAR1999    40
01APR1999    .
01MAY1999    90
```

If the ACCUMULATE=AVERAGE option is specified, the resulting time series is:

```
01MAR1999    20
01APR1999    .
01MAY1999    30
```

If the ACCUMULATE=MINIMUM option is specified, the resulting time series is:

```
O1MAR1999    10
O1APR1999    .
O1MAY1999    20
```

If the ACCUMULATE=MEDIAN option is specified, the resulting time series is:

```
O1MAR1999    20
01APR1999    .
O1MAY1999    20
```

If the ACCUMULATE=MAXIMUM option is specified, the resulting time series is:

```
O1MAR1999    30
O1APR1999    .
O1MAY1999    50
```

If the ACCUMULATE=FIRST option is specified, the resulting time series is:

```
O1MAR1999    10
O1APR1999    .
O1MAY1999    50
```

If the ACCUMULATE=LAST option is specified, the resulting time series is:

```
O1MAR1999    30
O1APR1999    .
O1MAY1999    20
```

If the ACCUMULATE=STDDEV option is specified, the resulting time series is:

```
O1MAR1999    14.14
O1APR1999    .
O1MAY1999    17.32
```

As can be seen from the above examples, even though the data set observations contain no missing values, the accumulated time series can have missing values.

Boundary Alignment

When the BOUNDARYALIGN= option is used to qualify the START= or END= options, additional time series values can be incorporated into the accumulation operation. For instance, if a data set contains the following observations

```
01JAN1999  10
01FEB1999  10
01MAR1999  10
01APR1999  10
01MAY1999  10
01JUN1999  10
```

and the options START='01FEB1999'd, END='01APR1999'd, INTERVAL=QUARTER, and ACCUMULATE=TOTAL are specified, using the BOUNDARYALIGN= option results in the following accumulated time series:

If BOUNDARYALIGN=START is specified, the accumulated time series is

```
01JAN1999  30
01APR1999  10
```

If BOUNDARYALIGN=END is specified, the accumulated time series is

```
01JAN1999  20
01APR1999  30
```

If BOUNDARYALIGN=BOTH is specified, the accumulated time series is

```
01JAN1999  30
01APR1999  30
```

If BOUNDARYALIGN=NONE is specified, the accumulated time series is

```
01JAN1999  20
01APR1999  10
```

Missing Value Interpretation

Sometimes missing values should be interpreted as unknown values. But sometimes missing values are known, such as when missing values are created from accumulation and no observations should be interpreted as no value—that is, zero. In the former case, the SETMISSING= option can be used to interpret how missing values are treated. The SETMISSING=0 option should be used when missing observations are to be treated as no (zero) values. In other cases, missing values should be interpreted as global values, such as minimum or maximum values of the accumulated series. The accumulated and interpreted time series is used in subsequent analyses.

Time Series Transformation

There are four transformations available for strictly positive series only. Let $y_t > 0$ be the original time series, and let w_t be the transformed series. The transformations are defined as follows:

Log is the logarithmic transformation.

$$w_t = \ln(y_t)$$

Logistic is the logistic transformation.

$$w_t = \ln(c y_t / (1 - c y_t))$$

where the scaling factor c is

$$c = (1 - 10^{-6}) 10^{-\text{ceil}(\log_{10}(max(y_t)))}$$

and $\text{ceil}(x)$ is the smallest integer greater than or equal to x.

Square root is the square root transformation.

$$w_t = \sqrt{y_t}$$

Box Cox is the Box-Cox transformation.

$$w_t = \begin{cases} \frac{y_t^{\lambda} - 1}{\lambda}, & \lambda \neq 0 \\ \ln(y_t), & \lambda = 0 \end{cases}$$

More complex time series transformations can be performed by using the EXPAND procedure of SAS/ETS.

Time Series Differencing

After optionally transforming the series, the accumulated series can be simply or seasonally differenced by using the VAR and CROSSVAR statement DIF= and SDIF= options. For example, suppose y_t is a monthly time series. The following examples of the DIF= and SDIF= options demonstrate how to simply and seasonally difference the time series.

```
dif=(1) sdif=(1)
dif=(1,12)
```

Additionally, when y_t is strictly positive and the TRANSFORM=, DIF=, and SDIF= options are combined in the VAR and CROSSVAR statements, the transformation operation is performed before the differencing operations.

Descriptive Statistics

Descriptive statistics can be computed from the working series by specifying the OUTSUM= option or PRINT=DESCSTATS.

Seasonal Decomposition

Seasonal decomposition/analysis can be performed on the working series by specifying the OUTDECOMP= option, the PRINT=DECOMP option, or one of the PLOTS= options associated with decomposition in the PROC TIMESERIES statement. The DECOMP statement enables you to specify options related to decomposition. The TIMESERIES procedure uses classical decomposition. More complex seasonal decomposition/adjustment analysis can be performed by using the X11 or the X12 procedure of SAS/ETS.

The DECOMP statement MODE= option determines the mode of the seasonal adjustment decomposition to be performed. There are four modes: multiplicative (MODE=MULT), additive (MODE=ADD), pseudo-additive (MODE=PSEUDOADD), and log-additive (MODE=LOGADD) decomposition. The default is MODE=MULTORADD which specifies MODE=MULT for series that are strictly positive, MODE=PSEUDOADD for series that are nonnegative, and MODE=ADD for series that are not nonnegative.

When MODE=LOGADD is specified, the components are exponentiated to the original metric.

The DECOMP statement LAMBDA= option specifies the Hodrick-Prescott filter parameter (Hodrick and Prescott 1980). The default is LAMBDA=1600. The Hodrick-Prescott filter is used to decompose the trend-cycle component into the trend component and cycle component in an additive fashion. A smaller parameter assigns less significance to the cycle; that is, LAMBDA=0 implies no cycle component.

The notation and keywords associated with seasonal decomposition/adjustment analysis are defined in Table 39.3.

Table 39.3 Seasonal Adjustment Formulas

Component	Keyword	MODE= Option	Formula
original series	ORIGINAL	MULT	$O_t = TC_t S_t I_t$
		ADD	$O_t = TC_t + S_t + I_t$
		LOGADD	$log(O_t) = TC_t + S_t + I_t$
		PSEUDOADD	$O_t = TC_t(S_t + I_t - 1)$
trend-cycle component	TCC	MULT	centered moving average of O_t
		ADD	centered moving average of O_t
		LOGADD	centered moving average of $log(O_t)$
		PSEUDOADD	centered moving average of O_t
seasonal-irregular component	SIC	MULT	$SI_t = S_t I_t = O_t / TC_t$
		ADD	$SI_t = S_t + I_t = O_t - TC_t$
		LOGADD	$SI_t = S_t + I_t = log(O_t) - TC_t$
		PSEUDOADD	$SI_t = S_t + I_t - 1 = O_t / TC_t$
seasonal component	SC	MULT	seasonal Averages of SI_t
		ADD	seasonal Averages of SI_t
		LOGADD	seasonal Averages of SI_t
		PSEUDOADD	seasonal Averages of SI_t
irregular component	IC	MULT	$I_t = SI_t / S_t$
		ADD	$I_t = SI_t - S_t$
		LOGADD	$I_t = SI_t - S_t$
		PSEUDOADD	$I_t = SI_t - S_t + 1$
trend-cycle-seasonal component	TCS	MULT	$TCS_t = TC_t S_t = O_t / I_t$
		ADD	$TCS_t = TC_t + S_t = O_t - I_t$
		LOGADD	$TCS_t = TC_t + S_t = O_t - I_t$
		PSEUDOADD	$TCS_t = TC_t S_t$
trend component	TC	MULT	$T_t = TC_t - C_t$
		ADD	$T_t = TC_t - C_t$
		LOGADD	$T_t = TC_t - C_t$
		PSEUDOADD	$T_t = TC_t - C_t$
cycle component	CC	MULT	$C_t = TC_t - T_t$
		ADD	$C_t = TC_t - T_t$
		LOGADD	$C_t = TC_t - T_t$
		PSEUDOADD	$C_t = TC_t - T_t$
seasonally adjusted series	SA	MULT	$SA_t = O_t / S_t = TC_t I_t$
		ADD	$SA_t = O_t - S_t = TC_t + I_t$
		LOGADD	$SA_t = O_t / exp(S_t) = exp(TC_t + I_t)$
		PSEUDOADD	$SA_t = TC_t I_t$

When s is odd the trend-cycle component is computed from the s-period centered moving average as follows:

$$TC_t = \sum_{k=-\lfloor s/2 \rfloor}^{\lfloor s/2 \rfloor} y_{t+k}/s$$

When s is even the trend-cycle component is computed from the s-period centered moving average as follows:

$$TC_t = \sum_{k=-s/2}^{s/2-1} (y_{t+k} + y_{t+1+k})/2s$$

The seasonal component is obtained by averaging the seasonal-irregular component for each season.

$$S_{k+js} = \sum_{t=k \bmod s} \frac{SI_t}{T/s}$$

where $0 \le j \le T/s$ and $1 \le k \le s$. The seasonal components are normalized to sum to one (multiplicative) or zero (additive).

Correlation Analysis

Correlation analysis can be performed on the working series by specifying the OUTCORR= option or one of the PLOTS= options that are associated with correlation. The CORR statement enables you to specify options that are related to correlation analysis.

Autocovariance Statistics

LAGS	$h \in \{0, \ldots, H\}$
N	N_h is the number of observed products at lag h, ignoring missing values
ACOV	$\hat{\gamma}(h) = \frac{1}{T} \sum_{t=h+1}^{T} (y_t - \overline{y})(y_{t-h} - \overline{y})$
ACOV	$\hat{\gamma}(h) = \frac{1}{N_h} \sum_{t=h+1}^{T} (y_t - \overline{y})(y_{t-h} - \overline{y})$ when embedded missing values are present

Autocorrelation Statistics

ACF	$\hat{\rho}(h) = \hat{\gamma}(h)/\hat{\gamma}(0)$
ACFSTD	$Std(\hat{\rho}(h)) = \sqrt{\frac{1}{T}\left(1 + 2\sum_{j=1}^{h-1} \hat{\rho}(j)^2\right)}$
ACFNORM	$Norm(\hat{\rho}(h)) = \hat{\rho}(h)/Std(\hat{\rho}(h))$
ACFPROB	$Prob(\hat{\rho}(h)) = 2\,(1 - \Phi\,(\lvert Norm(\hat{\rho}(h))\rvert))$
ACFLPROB	$LogProb(\hat{\rho}(h)) = -\log_{10}(Prob(\hat{\rho}(h))$
ACF2STD	$Flag(\hat{\rho}(h)) = \begin{cases} 1 & \hat{\rho}(h) > 2Std(\hat{\rho}(h)) \\ 0 & -2Std(\hat{\rho}(h)) < \hat{\rho}(h) < 2Std(\hat{\rho}(h)) \\ -1 & \hat{\rho}(h) < -2Std(\hat{\rho}(h)) \end{cases}$

Partial Autocorrelation Statistics

PACF	$\hat{\varphi}(h) = \Gamma_{(0,h-1)}\{\gamma_j\}_{j=1}^{h}$
PACFSTD	$Std(\hat{\varphi}(h)) = 1/\sqrt{N_0}$
PCFNORM	$Norm(\hat{\varphi}(h)) = \hat{\varphi}(h)/Std(\hat{\varphi}(h))$
PACFPROB	$Prob(\hat{\varphi}(h)) = 2\,(1 - \Phi\,(\lvert Norm(\hat{\varphi}(h))\rvert))$
PACFLPROB	$LogProb(\hat{\varphi}(h)) = -\log_{10}(Prob(\hat{\varphi}(h))$
PACF2STD	$Flag(\hat{\varphi}(h)) = \begin{cases} 1 & \hat{\varphi}(h) > 2Std(\hat{\varphi}(h)) \\ 0 & -2Std(\hat{\varphi}(h)) < \hat{\varphi}(h) < 2Std(\hat{\varphi}(h)) \\ -1 & \hat{\varphi}(h) < -2Std(\hat{\varphi}(h)) \end{cases}$

Inverse Autocorrelation Statistics

IACF	$\hat{\theta}(h)$
IACFSTD	$Std(\hat{\theta}(h)) = 1/\sqrt{N_0}$
IACFNORM	$Norm(\hat{\theta}(h)) = \hat{\theta}(h)/Std(\hat{\theta}(h))$
IACFPROB	$Prob(\hat{\theta}(h)) = 2\left(1 - \Phi\left(\lvert Norm(\hat{\theta}(h))\rvert\right)\right)$
IACFLPROB	$LogProb(\hat{\theta}(h)) = -\log_{10}(Prob(\hat{\theta}(h))$
IACF2STD	$Flag(\hat{\theta}(h)) = \begin{cases} 1 & \hat{\theta}(h) > 2Std(\hat{\theta}(h)) \\ 0 & -2Std(\hat{\theta}(h)) < \hat{\theta}(h) < 2Std(\hat{\theta}(h)) \\ -1 & \hat{\theta}(h) < -2Std(\hat{\theta}(h)) \end{cases}$

White Noise Statistics

WN	$Q(h) = T(T+2)\sum_{j=1}^{h} \rho(j)^2/(T-j)$
WN	$Q(h) = \sum_{j=1}^{h} N_j \rho(j)^2$ when embedded missing values are present
WNPROB	$Prob(Q(h)) = \chi_{\max(1,h-p)}(Q(h))$
WNLPROB	$LogProb(Q(h)) = -\log_{10}(Prob(Q(h))$

Cross-Correlation Analysis

Cross-correlation analysis can be performed on the working series by specifying the OUTCROSSCORR= option or one of the CROSSPLOTS= options that are associated with cross-correlation. The CROSSCORR statement enables you to specify options that are related to cross-correlation analysis.

Cross-Correlation Statistics

The cross-correlation statistics for the variable *x* supplied in a VAR statement and variable *y* supplied in a CROSSVAR statement are:

LAGS	$h \in \{0, \ldots, H\}$
N	N_h is the number of observed products at lag h, ignoring missing values
CCOV	$\hat{\gamma}_{x,y}(h) = \frac{1}{T}\sum_{t=h+1}^{T}(x_t - \overline{x})(y_{t-h} - \overline{y})$
CCOV	$\hat{\gamma}_{x,y}(h) = \frac{1}{N_h}\sum_{t=h+1}^{T}(x_t - \overline{x})(y_{t-h} - \overline{y})$ when embedded missing values are present
CCF	$\hat{\rho}_{x,y}(h) = \hat{\gamma}_{x,y}(h)/\sqrt{\hat{\gamma}_x(0)\hat{\gamma}_y(0)}$
CCFSTD	$Std(\hat{\rho}_{x,y}(h)) = 1/\sqrt{N_0}$
CCFNORM	$Norm(\hat{\rho}_{x,y}(h)) = \hat{\rho}_{x,y}(h)/Std(\hat{\rho}_{x,y}(h))$
CCFPROB	$Prob(\hat{\rho}_{x,y}(h)) = 2\left(1 - \Phi\left(\lvert Norm(\hat{\rho}_{x,y}(h))\rvert\right)\right)$
CCFLPROB	$LogProb(\hat{\rho}_{x,y}(h)) = -\log_{10}(Prob(\hat{\rho}_{x,y}(h))$
CCF2STD	$Flag(\hat{\rho}_{x,y}(h)) = \begin{cases} 1 & \hat{\rho}_{x,y}(h) > 2Std(\hat{\rho}_{x,y}(h)) \\ 0 & -2Std(\hat{\rho}_{x,y}(h)) < \hat{\rho}_{x,y}(h) < 2Std(\hat{\rho}_{x,y}(h)) \\ -1 & \hat{\rho}_{x,y}(h) < -2Std(\hat{\rho}_{x,y}(h)) \end{cases}$

Spectral Density Analysis

Spectral analysis can be performed on the working series by specifying the OUTSPECTRA= option or by specifying the PLOTS=PERIODOGRAM or PLOTS=SPECTRUM option in the PROC TIMESERIES statement. PROC TIMESERIES uses the finite Fourier transform to decompose data series into a sum of sine and cosine terms of different amplitudes and wavelengths. The finite Fourier transform decomposition of the series x_t is

$$x_t = \frac{a_0}{2} + \sum_{k=1}^{K-1} f_k(a_k \cos \omega_k t + b_k \sin \omega_k t)$$

$$f_k = \begin{cases} 1/2 & \text{if } T \text{ is even and } k = K-1 \\ 1 & \text{otherwise} \end{cases}$$

where

t	is the time subscript, $t = 0, 1, 2, \ldots, T-1$
x_t	are the equally spaced time series data
T	is the number of observations in the time series
K	is the number of frequencies in the Fourier decomposition: $K = \frac{T+2}{2}$ if T is even, $K = \frac{T+1}{2}$ if T is odd
k	is the frequency subscript, $k = 0, 1, 2, \ldots, K-1$
a_0	is the mean term: $a_0 = 2\overline{x}$
a_k	are the cosine coefficients
b_k	are the sine coefficients
ω_k	are the Fourier frequencies: $\omega_k = \frac{2\pi k}{T}$

The Fourier decomposition is performed after the ACCUMULATE=, DIF=, SDIF= and TRANSFORM= options in the ID and VAR statements have been applied.

Functions of the Fourier coefficients a_k and b_k can be plotted against frequency or against wavelength to form *periodograms*. The amplitude periodogram I_k is defined as follows:

$$I_k = \frac{T}{2}(a_k^2 + b_k^2)$$

Since the Fourier transform is an even, periodic function of frequency which repeats every T ordinates the periodogram is also. Values of I_k for all k therefore can be mapped to the unique values $I_k : k = 0 \ldots K-1$ using the equations

$$\begin{aligned} I_k &= I_{-k} && \text{for all } k \\ I_k &= I_{k+nT} && \text{for } n = \pm 1, \pm 2, \pm 3, \ldots \\ I_k &= I_{T-k} && \text{for } 0 \leq k \leq K-1 \end{aligned}$$

The periodogram, I_k, is an estimate at the discrete frequencies ω_k of the spectral density function which characterizes the series x_t. By smoothing the periodogram an improved spectral density estimate with reduced variance and bias can be achieved at these points. Smoothing can be accomplished either through use of a spectral window smoothing function or by applying a lag window filter to the series autocovariance function.

When the SPECTRA statement's DOMAIN=FREQUENCY option is in effect spectral density estimates are computed by smoothing the periodogram ordinates using the equation

$$S_k(M) = \sum_{\tau=K-T}^{K-1} \mathrm{w}\left(\frac{\tau}{M}\right) I_{k+\tau}$$

where $\mathrm{w}(\theta)$ is the spectral window function whose form is specified by either the KERNEL= option or the WEIGHTS option. M is the kernel scale parameter which acts as a frequency scaling factor in the spectral window smoothing function. Values of $I_{k+\tau}$ that fall outside of $0 \leq k+\tau \leq K-1$ are mapped to values inside this range by the equations presented previously.

When the DOMAIN=TIME option is specified spectral density values are estimated by applying a lag window filter, $\lambda(h, M)$, to the series autocovariance function. The spectral density estimate then can be computed from the filtered autocovariance function using the equation

$$S_k(M) = \sum_{h=-(T-1)}^{T-1} \lambda(h, M)\hat{\gamma}(h) \cos h\omega_k.$$

In this case the kernel scale parameter, M, serves as a scale factor for the lag length, h, in the time domain. In the lag window formulation the spectral density estimate is a consistent estimator as $T, M \rightarrow \infty$ under the conditions $\lambda(h, M) = 0$ for $|h| > M$, and $\lim_{T\rightarrow\infty} M/T = 0$. These conditions lead to the following parameterization of M provided by the SPECTRA statement:

$$M = cK^e$$

where the values $c > 0$ and $0 < e < 1$ satisfy the consistency conditions. To specify the kernel scale parameter explicitly, set $c =$ to the desired scale factor and $e = 0$.

For uniformity and computational efficiency all spectral density estimates are calculated using a spectral window weighting function, $\mathrm{w}(\theta)$, applied to the periodogram ordinates. In the case where the DOMAIN=TIME option is specified the effective spectral window weighting function is computed by the equation

$$\mathrm{w}_{\mathrm{TIME}}(\theta) = \sum_{h=-(T-1)}^{T-1} \lambda(h, M) \cos h\theta.$$

Because the kernel scale parameter, M, serves as a lag scale factor in the time domain and bandwidth scale factor in the frequency domain the impact of M on spectral density estimates depends on the value of the DOMAIN= option. When DOMAIN=FREQUENCY increasing values of M decrease variance and increase bias in the spectral density estimates whereas when DOMAIN=TIME increasing values of M increase variance and decrease bias.

Using Kernel Specifications

You can specify one of ten different kernel smoothing functions in the SPECTRA statement. Five smoothing functions are available as KERNEL= options and five complementary smoothing functions which correspond to lag window filters are available when the KERNEL= option is used in conjunction with the DOMAIN=TIME option.

For example, a Parzen kernel with a support of 11 periodogram ordinates in the frequency domain can be specified using the kernel option:

```
spectra / parzen c=5 expon=0;
```

The TIMESERIES procedure supports the following spectral window kernel functions in the frequency domain where $x = \tau/M$:

BARTLETT: Bartlett kernel

$$\mathrm{w}(x) = \begin{cases} 1 - |x| & |x| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

PARZEN: Parzen kernel

$$\mathrm{w}(x) = \begin{cases} 1 - 6|x|^2 + 6|x|^3 & 0 \leq |x| \leq \frac{1}{2} \\ 2(1 - |x|)^3 & \frac{1}{2} \leq |x| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

QS: quadratic spectral kernel

$$\mathrm{w}(x) = \frac{3}{(2\pi x)^2} \left(\frac{\sin 2\pi x}{2\pi x} - \cos 2\pi x \right)$$

TUKEY: Tukey-Hanning kernel

$$\mathrm{w}(x) = \begin{cases} (1+\cos(\pi x))/2 & |x| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

TRUNCAT: truncated kernel

$$\mathrm{w}(x) = \begin{cases} 1 & |x| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

When the DOMAIN=TIME option is specified the five kernel functions above are interpreted as lag window filters on the autocovariance function. The lag window kernel functions correspond to the following spectral window smoothing functions where $\theta = 2\pi\tau/T$:

BARTLETT: Bartlett equivalent lag window filter

$$\mathrm{w}(\theta) = \frac{1}{2\pi M}\left(\frac{\sin(M\theta/2)}{\sin(\theta/2)}\right)^2$$

PARZEN: Parzen equivalent lag window filter

$$\mathrm{w}(\theta) = \frac{6}{\pi M^3}\left(\frac{\sin(M\theta/4)}{\sin(\theta/2)}\right)^4\left(1-\frac{2}{3}\sin^2(\theta/2)\right)$$

QS: quadratic spectral equivalent lag window filter

$$\mathrm{w}(\theta) = \begin{cases} \frac{3M}{4\pi}(1-(M\theta/\pi)^2) & |\theta| \leq \pi/M \\ 0 & |\theta| > \pi/M \end{cases}$$

TUKEY: Tukey-Hanning equivalent lag window filter

$$\begin{aligned} \mathrm{w}(\theta) &= \frac{1}{4}D_M(\theta-\pi/M)+\frac{1}{2}D_M(\theta)+\frac{1}{4}D_M(\theta+\pi/M) \\ D_M(\theta) &= \frac{1}{2\pi}\frac{\sin[(M+1/2)\theta]}{\sin(\theta/2)} \end{aligned}$$

TRUNC: truncated equivalent lag window filter

$$\mathrm{w}(\theta) = D_M(\theta)$$

Using Specification of Weight Constants

Any number of weighting constants can be specified. The constants are interpreted symmetrically about the middle weight. The middle constant (or the constant to the right of the middle if an even number of weight constants is specified) is the relative weight of the current periodogram ordinate. The constant immediately following the middle one is the relative weight of the next periodogram ordinate, and so on. The actual weights used in the smoothing process are the weights specified in the WEIGHTS option scaled so that they sum to 1.

The moving average calculation reflects at each end of the periodogram to accommodate the periodicity of the periodogram function.

For example, a simple triangular weighting can be specified using the following WEIGHTS option:

```
spectra / weights 1 2 3 2 1;
```

Computational Method

If the number of observations, T, factors into prime integers that are less than or equal to 23, and the product of the square-free factors of T is less than 210, then the procedure uses the fast Fourier transform developed by Cooley and Tukey (1965) and implemented by Singleton (1969). If T cannot be factored in this way, then the procedure uses a Chirp-Z algorithm similar to that proposed by Monro and Branch (1977).

Missing Values

Missing values are replaced with an estimate of the mean to perform spectral analyses. This treatment of a series with missing values is consistent with the approach used by Priestley (1981).

Singular Spectrum Analysis

Given a time series, y_t, for $t = 1, \ldots, T$, and a window length, $2 \leq L < T/2$, singular spectrum analysis Golyandina, Nekrutkin, and Zhigljavsky (2001) decompose the time series into spectral groupings using the following steps:

Embedding Step

Using the time series, form a $K \times L$ trajectory matrix, $\mathbf{X}$, with elements

$$\mathbf{X} = \{x_{k,l}\}_{k=1,l=1}^{K,L}$$

such that $x_{k,l} = y_{k-l+1}$ for $k = 1, \ldots, K$ and $l = 1, \ldots, L$ and where $K = T - L + 1$. By definition $L \leq K < T$, because $2 \leq L < T/2$.

Decomposition Step

Using the trajectory matrix, $\mathbf{X}$, apply singular value decomposition to the trajectory matrix

$$\mathbf{X} = \mathbf{UQV}$$

where $\mathbf{U}$ represents the $K \times L$ matrix that contains the left-hand-side (LHS) eigenvectors, where $\mathbf{Q}$ represents the diagonal $L \times L$ matrix that contains the singular values, and where $\mathbf{V}$ represents the $L \times L$ matrix that contains the right-hand-side (RHS) eigenvectors.

Therefore,

$$\mathbf{X} = \sum_{l=1}^{L} \mathbf{X}^{(l)} = \sum_{l=1}^{L} \mathbf{u}_l q_l \mathbf{v}_l^T$$

where $\mathbf{X}^{(l)}$ represents the $K \times L$ principal component matrix, $\mathbf{u}_l$ represents the $K \times 1$ left-hand-side (LHS) eigenvector, q_l represents the singular value, and $\mathbf{v}_l$ represents the $L \times 1$ right-hand-side (RHS) eigenvector associated with the lth window index.

Grouping Step

For each group index, $m = 1, \ldots, M$, define a group of window indices $I_m \subset \{1, \ldots, L\}$. Let

$$\mathbf{X}_{I_m} = \sum_{l \in I_m} \mathbf{X}^{(l)} = \sum_{l \in I_m} \mathbf{u}_l q_l \mathbf{v}_l^T$$

represent the grouped trajectory matrix for group I_m. If groupings represent a spectral partition,

$$\bigcup_{m=1}^{M} I_m = \{1, \ldots, L\} \qquad \text{and} \qquad I_m \cap I_n = \emptyset \quad \text{for} \quad m \neq n$$

then according to the singular value decomposition theory,

$$\mathbf{X} = \sum_{m=1}^{M} \mathbf{X}_{I_m}$$

Averaging Step

For each group index, $m = 1, \ldots, M$, compute the diagonal average of $\mathbf{X}_{I_m}$,

$$\tilde{x}_t^{(m)} = \frac{1}{n_t} \sum_{l=s_t}^{e_t} x_{t-l+1,l}^{(m)}$$

where

$$\begin{array}{llll} s_t = 1, & e_t = t, & n_t = t & \text{for} \quad 1 \leq t < L \\ s_t = 1, & e_t = L, & n_t = L & \text{for} \quad L \leq t \leq T - L + 1 \\ s_t = T - t - 1, & e_t = L, & n_t = T - t + 1 & \text{for} \quad T - L + 1 < t \leq T \end{array}$$

If the groupings represent a spectral partition, then by definition

$$y_t = \sum_{m=1}^{M} \tilde{x}_t^{(m)}$$

Hence, singular spectrum analysis additively decomposes the original time series, y_t, into m component series $\tilde{x}_t^{(m)}$ for $m = 1, \ldots, M$.

Specifying the Window Length

You can explicitly specify the maximum window length, $2 \leq L \leq 1000$, using the LENGTH= option or implicitly specify the window length using the INTERVAL= option in the ID statement or the SEASONALITY= option in the PROC TIMESERIES statement.

Either way the window length is reduced based on the accumulated time series length, T, to enforce the requirement that $2 \leq L \leq T/2$.

Specifying the Groups

You can use the GROUPS= option to explicitly specify the composition and number of groups, $I_m \subset \{1, \ldots, L\}$ or use the THRESHOLDPCT= option in the SSA statement to implicitly specify the grouping. The THRESHOLDPCT= option is useful for removing noise or less dominant patterns from the accumulated time series.

Let $0 < \alpha < 1$ be the cumulative percent singular value THRESHOLDPCT=. Then the last group, $I_M = \{l_\alpha, \ldots, L\}$, is determined by the smallest value such that

$$\left(\sum_{l=1}^{l_\alpha - 1} q_l \Big/ \sum_{l=1}^{L} q_l \right) \geq \alpha \qquad \text{where } 1 < l_\alpha \leq L$$

Using this rule, the last group, I_M, describes the least dominant patterns in the time series and the size of the last group is at least one and is less than the window length, $L \geq 2$.

The magnitudes of the principal components which are plotted using the PLOT=SSA option and selected by the THRESHOLDPCT= option are based on the singular values which appear on the diagonal of $\mathbf{Q}$. Alternatively, each principal component's contribution to variation in the series can be quantified by using the squares of the singular values. The relative contributions of the principal components to variation in the series are included in the printed tabular output produced by the PRINT=SSA option.

Data Set Output

The TIMESERIES procedure can create the OUT=, OUTCORR=, OUTCROSSCORR=, OUTDECOMP=, OUTFREQ=, OUTSEASON=, OUTSPECTRA=, OUTSSA=, OUTSUM=, and OUTTREND= data sets. In general, these data sets contain the variables listed in the BY statement. If an analysis step that is related to an output data step fails, the values of this step are not recorded or are set to missing in the related output data set and appropriate error and/or warning messages are recorded in the log.

OUT= Data Set

The OUT= data set contains the variables specified in the BY, ID, VAR, and CROSSVAR statements. If the ID statement is specified, the ID variable values are aligned and extended based on the ALIGN= and INTERVAL= options. The values of the variables specified in the VAR and CROSSVAR statements are accumulated based on the ACCUMULATE= option, and missing values are interpreted based on the SETMISSING= option.

OUTCORR= Data Set

The OUTCORR= data set contains the variables specified in the BY statement as well as the variables listed below. The OUTCORR= data set records the correlations for each variable specified in a VAR statement (not the CROSSVAR statement).

When the CORR statement TRANSPOSE=NO option is omitted or specified explicitly, the variable *names* are related to correlation statistics specified in the CORR statement options and the variable *values* are related to the NLAG= or LAGS= option.

NAME	variable name
LAG	time lag
N	number of variance products
ACOV	autocovariances
ACF	autocorrelations
ACFSTD	autocorrelation standard errors
ACF2STD	an indicator of whether autocorrelations are less than (–1), greater than (1), or within (0) two standard errors of zero
ACFNORM	normalized autocorrelations
ACFPROB	autocorrelation probabilities
ACFLPROB	autocorrelation log probabilities
PACF	partial autocorrelations
PACFSTD	partial autocorrelation standard errors
PACF2STD	an indicator of whether partial autocorrelations are less than (–1), greater than (1), or within (0) two standard errors of zero
PACFNORM	partial normalized autocorrelations
PACFPROB	partial autocorrelation probabilities
PACFLPROB	partial autocorrelation log probabilities
IACF	inverse autocorrelations
IACFSTD	an indicator of whether inverse autocorrelations are less than (–1), greater than (1), or within (0) two standard errors of zero
IACF2STD	two standard errors beyond inverse autocorrelation
IACFNORM	normalized inverse autocorrelations
IACFPROB	inverse autocorrelation probabilities
IACFLPROB	inverse autocorrelation log probabilities
WN	white noise test Statistics
WNPROB	white noise test probabilities
WNLPROB	white noise test log probabilities

The preceding correlation statistics are computed for each specified time lag.

When the CORR statement TRANSPOSE=YES option is specified, the variable *values* are related to correlation statistics specified in the CORR statement and the variable *names* are related to the NLAG= or LAGS= options.

NAME	variable name
STAT	correlation statistic name
LABEL	correlation statistic label
LAG*h*	correlation statistics for lag *h*

OUTCROSSCORR= Data Set

The OUTCROSSCORR= data set contains the variables specified in the BY statement as well as the variables listed below. The OUTCROSSCORR= data set records the cross-correlations for each variable specified in a VAR and the CROSSVAR statements.

When the CROSSCORR statement TRANSPOSE=NO option is omitted or specified explicitly, the variable *names* are related to cross-correlation statistics specified in the CROSSCORR statement options and the variable *values* are related to the NLAG= or LAGS= option.

NAME	variable name
CROSS	cross variable name
LAG	time lag
N	number of variance products
CCOV	cross-covariances
CCF	cross-correlations
CCFSTD	cross-correlation standard errors
CCF2STD	an indicator of whether cross-correlations are less than (–1), greater than (1), or within (0) two standard errors of zero
CCFNORM	normalized cross-correlations
CCFPROB	cross-correlation probabilities
CCFLPROB	cross-correlation log probabilities

The preceding cross-correlation statistics are computed for each specified time lag.

When the CROSSCORR statement TRANSPOSE=YES option is specified, the variable *values* are related to cross-correlation statistics specified in the CROSSCORR statement and the variable *names* are related to the NLAG= or LAGS= options.

NAME	variable name
CROSS	cross variable name
STAT	cross-correlation statistic name
LABEL	cross-correlation statistic label
LAG*h*	cross-correlation statistics for lag *h*

OUTDECOMP= Data Set

The OUTDECOMP= data set contains the variables specified in the BY statement as well as the variables listed below. The OUTDECOMP= data set records the seasonal decomposition/adjustments for each variable specified in a VAR statement (not the CROSSVAR statement).

When the DECOMP statement TRANSPOSE=NO option is omitted or specified explicitly, the variable *names* are related to decomposition/adjustments specified in the DECOMP statement and the variable *values* are related to the ID statement INTERVAL= option and the PROC TIMESERIES statement SEASONALITY= option.

NAME	variable name
MODE	mode of decomposition
TIMEID	time ID values
SEASON	seasonal index
ORIGINAL	original series values
TCC	trend-cycle component
SIC	seasonal-irregular component
SC	seasonal component
SCSTD	seasonal component standard errors
TCS	trend-cycle-seasonal component
IC	irregular component
SA	seasonally adjusted series
PCSA	percent change seasonally adjusted series
TC	trend component
CC	cycle component

The preceding decomposition components are computed for each time period.

When the DECOMP statement TRANSPOSE=YES option is specified, the variable *values* are related to decomposition/adjustments specified in the DECOMP statement and the variable *names* are related to the ID statement INTERVAL= option, the PROC TIMESERIES statement SEASONALITY= option, and the DECOMP statement NPERIODS= option.

NAME	variable name
MODE	mode of decomposition name
COMP	decomposition component name
LABEL	decomposition component label
PERIOD*t*	decomposition component value for time period *t*

OUTFREQ= Data Set

The OUTFREQ= data set contains the variables specified in the BY statement as well as the variables listed below. The OUTFREQ= data set records the counts of the discrete values of the time series for each variable specified in a VAR statement (not the CROSSVAR statement).

NAME	variable name
VALUES	distinct series values
COUNTS	counts of the discrete values
PERCENT	percentage of the total counts

OUTPROCINFO= Data Set

The OUTPROCINFO= data set contains information about the run of the TIMESERIES procedure. The following variables are present:

SOURCE	set to the name of the procedure, in this case TIMESERIES
NAME	name of the item being reported
LABEL	descriptive label for the item in _NAME_
STAGE	set to the current stage of the procedure; for TIMESERIES this is set to ALL
VALUE	value of the item specified in _NAME_

OUTSEASON= Data Set

The OUTSEASON= data set contains the variables specified in the BY statement as well as the variables listed below. The OUTSEASON= data set records the seasonal statistics for each variable specified in a VAR statement (not the CROSSVAR statement).

When the SEASON statement TRANSPOSE=NO option is omitted or specified explicitly, the variable *names* are related to seasonal statistics specified in the SEASON statement and the variable *values* are related to the ID statement INTERVAL= option or the PROC TIMESERIES statement SEASONALITY= option.

NAME	variable name
TIMEID	time ID values
SEASON	seasonal index
NOBS	number of observations
N	number of nonmissing observations
NMISS	number of missing observations
MINIMUM	minimum value

MAXIMUM	maximum value
RANGE	range value
SUM	summation value
MEAN	mean value
STDDEV	standard deviation
CSS	corrected sum of squares
USS	uncorrected sum of squares
MEDIAN	median value

The preceding statistics are computed for each season.

When the SEASON statement TRANSPOSE=YES option is specified, the variable *values* are related to seasonal statistics specified in the SEASON statement and the variable *names* are related to the ID statement INTERVAL= option or the PROC TIMESERIES statement SEASONALITY= option.

NAME	variable name
STAT	season statistic name
LABEL	season statistic name
SEASON*s*	season statistic value for season *s*

OUTSPECTRA= Data Set

The OUTSPECTRA= data set contains the variables that are specified in the BY statement in addition to the variables listed below. The OUTSPECTRA= data set records the frequency domain analysis for each variable specified in a VAR statement (not the CROSSVAR statement).

The following variable names are related to correlation statistics specified in the SPECTRA statement options.

NAME	variable name
FREQ	frequency in radians from 0 to π
PERIOD	period or wavelength
COS	cosine transform
SIN	sine transform
P	periodogram
S	spectral density estimates

OUTSSA= Data Set

The OUTSSA= data set contains the variables that are specified in the BY statement in addition to the variables listed below. The OUTSSA= data set records the singular spectrum analysis (SSA) for each variable specified in a VAR statement (not the CROSSVAR statement).

When the SSA statement TRANSPOSE=NO option is omitted or specified explicitly, the variable *names* are related to singular spectrum analysis specified in the SSA statement, and the variable *values* are related to the INTERVAL= option in the ID statement and the SEASONALITY= option in the PROC TIMESERIES statement.

NAME	variable name
TIMEID	time ID values
CYCLE	cycle index
SEASON	seasonal index
ORIGINAL	original series values
GROUP*i*	SSA result groups

The _GROUP*i*_ decomposition components are computed for each time period.

When the SSA statement TRANSPOSE=YES option is specified, the variable *values* are related to singular spectrum analysis specified in the SSA statement, and the variable *names* are related to the INTERVAL= option in the ID statement, the SEASONALITY= option in the PROC TIMESERIES statement, or the NPERIODS= option in the SSA statement. The following variables are written to a transposed OUTSSA= data set:

NAME	variable name
GROUP	group number
PERIOD*t*	SSA group value for time period *t*

OUTSUM= Data Set

The OUTSUM= data set contains the variables specified in the BY statement as well as the variables listed below. The OUTSUM= data set records the descriptive statistics for each variable specified in a VAR statement (not the CROSSVAR statement).

Variables related to descriptive statistics are based on the ACCUMULATE= and SETMISSING= options in the ID and VAR statements:

NAME	variable name
STATUS	status flag that indicates whether the requested analyses were successful
NOBS	number of observations
N	number of nonmissing observations

NMISS	number of missing observations
START	the starting date of the time series
END	the ending date of the time series
STARTOBS	the beginning observation of the time series
ENDOBS	the ending observation of the time series
MINIMUM	minimum value
MAXIMUM	maximum value
AVG	average value
STDDEV	standard deviation

The OUTSUM= data set contains the descriptive statistics of the (accumulated) time series.

OUTTREND= Data Set

The OUTTREND= data set contains the variables specified in the BY statement as well as the variables listed below. The OUTTREND= data set records the trend statistics for each variable specified in a VAR statement (not the CROSSVAR statement).

When the TREND statement TRANSPOSE=NO option is omitted or explicitly specified, the variable *names* are related to trend statistics specified in the TREND statement and the variable *values* are related to the INTERVAL= option in the ID statement or the SEASONALITY= option in the PROC TIMESERIES statement.

NAME	variable name
TIMEID	time ID values
SEASON	seasonal index
NOBS	number of observations
N	number of nonmissing observations
NMISS	number of missing observations
MINIMUM	minimum value
MAXIMUM	maximum value
RANGE	range value
SUM	summation value
MEAN	mean value
STDDEV	standard deviation
CSS	corrected sum of squares
USS	uncorrected sum of squares
MEDIAN	median value

The preceding statistics are computed for each time period.

When the TREND statement TRANSPOSE=YES option is specified, the variable *values* related to trend statistics specified in the TREND statement and the variable *names* are related to the INTERVAL= option in the ID statement, the SEASONALITY= option in the PROC TIMESERIES statement, or the NPERIODS= option in the TREND statement. The following variables are written to the OUTTREND= data set:

NAME	variable name
STAT	trend statistic name
LABEL	trend statistic name
PERIOD*t*	trend statistic value for time period *t*

STATUS Variable Values

The _STATUS_ variable that appears in the OUTSUM= data set contains a code that specifies whether the analysis has been successful or not. The _STATUS_ variable can take the following values:

0	success
1000	transactional trend statistics failure
2000	transactional seasonal statistics failure
3000	accumulation failure
4000	missing value interpretation failure
6000	series is all missing
7000	transformation failure
8000	differencing failure
9000	unable to compute descriptive statistics
10000	seasonal decomposition failure
11000	correlation analysis failure
15000	singular spectrum analysis failure
16000	spectral analysis failure

Printed Output

The TIMESERIES procedure optionally produces printed output by using the Output Delivery System (ODS). By default, the procedure produces no printed output. All output is controlled by the PRINT= and PRINTDETAILS options associated with the PROC TIMESERIES statement. In general, if an analysis step related to printed output fails, the values of this step are not printed and appropriate error or warning messages or both are recorded in the log. The printed output is similar to the output data set, and these similarities are described below.

PRINT=COUNTS	prints the discrete distribution analysis.
PRINT=DECOMP	prints the seasonal decomposition similar to the OUTDECOMP= data set.
PRINT=DESCSTATS	prints a table of descriptive statistics for each variable.
PRINT=SEASONS	prints the seasonal statistics similar to the OUTSEASON= data set.
PRINT=SSA	prints the singular spectrum analysis similar to the OUTSSA= data set.
PRINT=SUMMARY	prints the summary statistics similar to the OUTSUM= data set.
PRINT=TRENDS	prints the trend statistics similar to the OUTTREND= data set.
PRINTDETAILS	prints each table with greater detail.

If PRINT=SEASONS and the PRINTDETAILS options are both specified, all seasonal statistics are printed.

ODS Table Names

Table 39.4 relates the PRINT= options to ODS tables:

Table 39.4 ODS Tables Produced in PROC TIMESERIES

ODS Table Name	Description	Statement	Option
CountStatistics	Discrete distribution	PRINT	COUNTS
DistributionSelection	Discrete distribution Selection	PRINT	COUNTS
DistributionParameterEstimates	Discrete distribution Parameter Estimates	PRINT	COUNTS
DistributionEstimates	Discrete distribution Estimates	PRINT	COUNTS
SeasonalDecomposition	Seasonal decomposition	PRINT	DECOMP
DescStats	Descriptive statistics	PRINT	DESCSTATS
GlobalStatistics	Global statistics	PRINT	SEASONS
SeasonStatistics	Season statistics	PRINT	SEASONS
StatisticsSummary	Statistics summary	PRINT	SUMMARY
TrendStatistics	Trend statistics	PRINT	TRENDS
GlobalStatistics	Global statistics	PRINT	TRENDS
SSASingularValues	SSA singular values	PRINT	SSA
SSAResults	SSA results	PRINT	SSA

The tables are related to a single series within a BY group.

ODS Graphics Names

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*).

Before you create graphs, ODS Graphics must be enabled (for example, with the ODS GRAPHICS ON statement). For more information about enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" in that chapter.

The overall appearance of graphs is controlled by ODS styles. Styles and other aspects of using ODS Graphics are discussed in the section "A Primer on ODS Statistical Graphics" in that chapter.

This section describes the graphical output produced by the TIMESERIES procedure. PROC TIMESERIES assigns a name to each graph it creates. These names are listed in Table 39.5.

Table 39.5 ODS Graphics Produced by PROC TIMESERIES

ODS Graph Name	Plot Description	Statement	Option
ACFPlot	Autocorrelation function	PLOTS	ACF
ACFNORMPlot	Normalized autocorrelation function	PLOTS	ACF
CCFNORMPlot	Normalized cross-correlation function	CROSSPLOTS	CCF
CCFPlot	Cross-correlation function	CROSSPLOTS	CCF
ChiSquareProbabilitiesPlot	Discrete distribution evaluation	COUNTPLOTS	CHISQPROB
ChiSquareLogProbabilitiesPlot	Discrete distribution evaluation	COUNTPLOTS	CHISQPROB
CorrelationPlots	Correlation graphics panel	PLOTS	CORR
CrossSeriesPlot	Cross series plot	CROSSPLOTS	SERIES
CycleComponentPlot	Cycle component	PLOTS	CC
CyclePlot	Seasonal cycles plot	PLOTS	CYCLES
DecompositionPlots	Decomposition graphics panel	PLOTS	DECOMP
DiscreteDistributionPlot	Discrete distribution analysis	COUNTPLOTS	DISTRIBUTION
FrequencyAnalysisPlot	Frequency analysis	COUNTPLOTS	COUNTS
FrequencyCountsPlot	Frequency analysis	COUNTPLOTS	COUNTS
FrequencyValuesPlot	Frequency analysis	COUNTPLOTS	VALUES
IACFPlot	Inverse autocorrelation function	PLOTS	IACF
IACFNORMPlot	Normalized inverse autocorrelation function	PLOTS	IACF
IrregularComponentPlot	Irregular component	PLOTS	IC
PACFPlot	Partial autocorrelation function	PLOTS	PACF
PACFNORMPlot	Standardized partial autocorrelation function	PLOTS	PACF
PercentChangeAdjustedPlot	Percent-change seasonally adjusted	PLOTS	PCSA
Periodogram	Periodogram versus period	PLOTS	PERIODOGRAM
ResidualPlot	Residual time series plot	PLOTS	RESIDUAL

Table 39.5 (continued)

ODS Graph Name	Plot Description	Statement	Option
SeasonallyAdjustedPlot	Seasonally adjusted	PLOTS	SA
SeasonalComponentPlot	Seasonal component	PLOTS	SC
SeasonalIrregularComponentPlot	Seasonal-irregular component	PLOTS	SIC
SeriesHistogram	Histogram of series values	PLOTS	HISTOGRAM
SeriesPlot	Time series plot	PLOTS	SERIES
SpectralDensityPlot	Spectral density versus period	PLOTS	SPECTRUM
SSASingularValuesPlot	SSA singular values	PLOTS	SSA
SSAResultsPlot	SSA results	PLOTS	SSA
TrendComponentPlot	Trend component	PLOTS	TC
TrendCycleComponentPlot	Trend-cycle component	PLOTS	TCC
TrendCycleSeasonalPlot	Trend-cycle-seasonal component	PLOTS	TCS
WhiteNoiseLogProbabilityPlot	White noise log probability	PLOTS	WN
WhiteNoiseProbabilityPlot	White noise probability	PLOTS	WN

Examples: TIMESERIES Procedure

Example 39.1: Accumulating Transactional Data into Time Series Data

This example illustrates using the TIMESERIES procedure to accumulate time-stamped transactional data that has been recorded at no particular frequency into time series data at a specific frequency. After the time series is created, the various SAS/ETS procedures related to time series analysis, seasonal adjustment/decomposition, modeling, and forecasting can be used to further analyze the time series data.

Suppose that the input data set WORK.RETAIL contains variables STORE and TIMESTAMP and numerous other numeric transaction variables. The BY variable STORE contains values that break up the transactions into groups (BY groups). The time ID variable TIMESTAMP contains SAS date values recorded at no particular frequency. The other data set variables contain the numeric transaction values to be analyzed. It is further assumed that the input data set is sorted by the variables STORE and TIMESTAMP. The following statements form monthly time series from the transactional data based on the median value (ACCUMULATE=MEDIAN) of the transactions recorded with each time period. Also, the accumulated time series values for time periods with no transactions are set to zero instead of to missing (SETMISS=0) and only transactions recorded between the first day of 1998 (START='01JAN1998'D) and last day of 2000 (END='31JAN2000'D) are considered and, if needed, extended to include this range.

```
proc timeseries data=retail out=mseries;
   by store;
   id timestamp interval=month
                accumulate=median
                setmiss=0
                start='01jan1998'd
                end  ='31dec2000'd;
   var item1-item8;
run;
```

The monthly time series data are stored in the data WORK.MSERIES. Each BY group associated with the BY variable STORE contains an observation for each of the 36 months associated with the years 1998, 1999, and 2000. Each observation contains the variable STORE, TIMESTAMP, and each of the analysis variables in the input data set.

After each set of transactions has been accumulated to form corresponding time series, accumulated time series can be analyzed using various time series analysis techniques. For example, exponentially weighted moving averages can be used to smooth each series. The following statements use the EXPAND procedure to smooth the analysis variable named STOREITEM.

```
proc expand data=mseries out=smoothed from=month;
   by store;
   id date;
   convert storeitem=smooth / transform=(ewma 0.1);
run;
```

The smoothed series are stored in the data set WORK.SMOOTHED. The variable SMOOTH contains the smoothed series.

If the time ID variable TIMESTAMP contains SAS datetime values instead of SAS date values, the INTERVAL=, START=, and END= options must be changed accordingly and the following statements could be used:

```
proc timeseries data=retail out=tseries;
   by store;
   id timestamp interval=dtmonth
                accumulate=median
                setmiss=0
                start='01jan1998:00:00:00'dt
                end  ='31dec2000:00:00:00'dt;
   var _numeric_;
run;
```

The monthly time series data are stored in the data WORK.TSERIES, and the time ID values use a SAS datetime representation.

Example 39.2: Trend and Seasonal Analysis

This example illustrates using the TIMESERIES procedure for trend and seasonal analysis of time-stamped transactional data.

Suppose that the data set SASHELP.AIR contains two variables: DATE and AIR. The variable DATE contains sorted SAS date values recorded at no particular frequency. The variable AIR contains the transaction values to be analyzed.

The following statements accumulate the transactional data on an average basis to form a quarterly time series and perform trend and seasonal analysis on the transactions.

```
proc timeseries data=sashelp.air
                out=series
                outtrend=trend
                outseason=season print=seasons;
   id date interval=qtr accumulate=avg;
   var air;
run;
```

The time series is stored in the data set WORK.SERIES, the trend statistics are stored in the data set WORK.TREND, and the seasonal statistics are stored in the data set WORK.SEASON. Additionally, the seasonal statistics are printed (PRINT=SEASONS) and the results of the seasonal analysis are shown in Output 39.2.1.

Output 39.2.1 Seasonal Statistics Table

The TIMESERIES Procedure

Season Statistics for Variable AIR

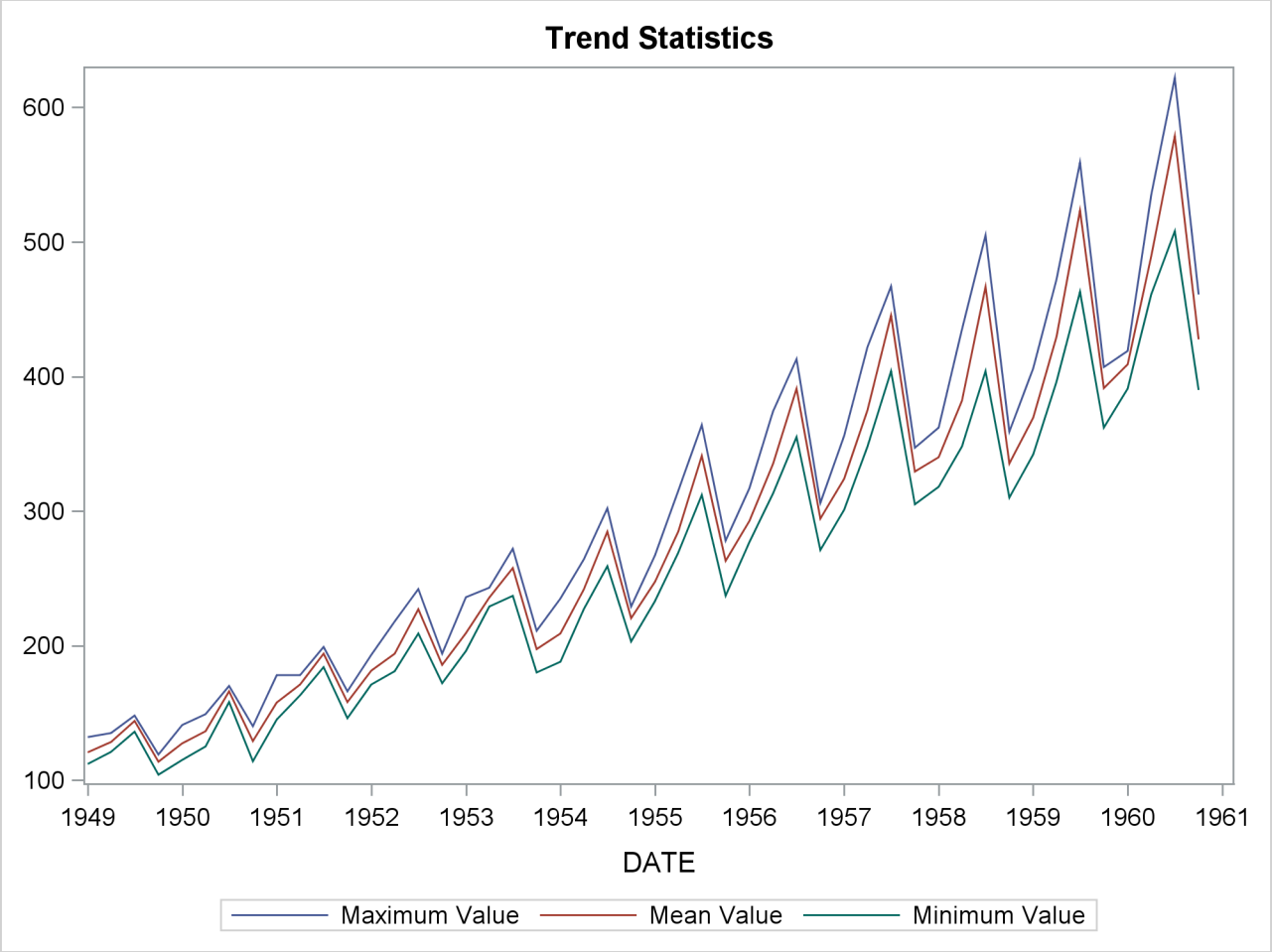
Season Index	N	Minimum	Maximum	Sum	Mean	Standard Deviation
1	36	112.0000	419.0000	8963.00	248.9722	95.65189
2	36	121.0000	535.0000	10207.00	283.5278	117.61839
3	36	136.0000	622.0000	12058.00	334.9444	143.97935
4	36	104.0000	461.0000	9135.00	253.7500	101.34732

Using the trend statistics stored in the WORK.TREND data set, the following statements plot various trend statistics associated with each time period over time.

```
title1 "Trend Statistics";
proc sgplot data=trend;
   series x=date y=max  / lineattrs=(pattern=solid);
   series x=date y=mean / lineattrs=(pattern=solid);
   series x=date y=min  / lineattrs=(pattern=solid);
   yaxis display=(nolabel);
   format date year4.;
run;
```

The results of this trend analysis are shown in Output 39.2.2.

Output 39.2.2 Trend Statistics Plot

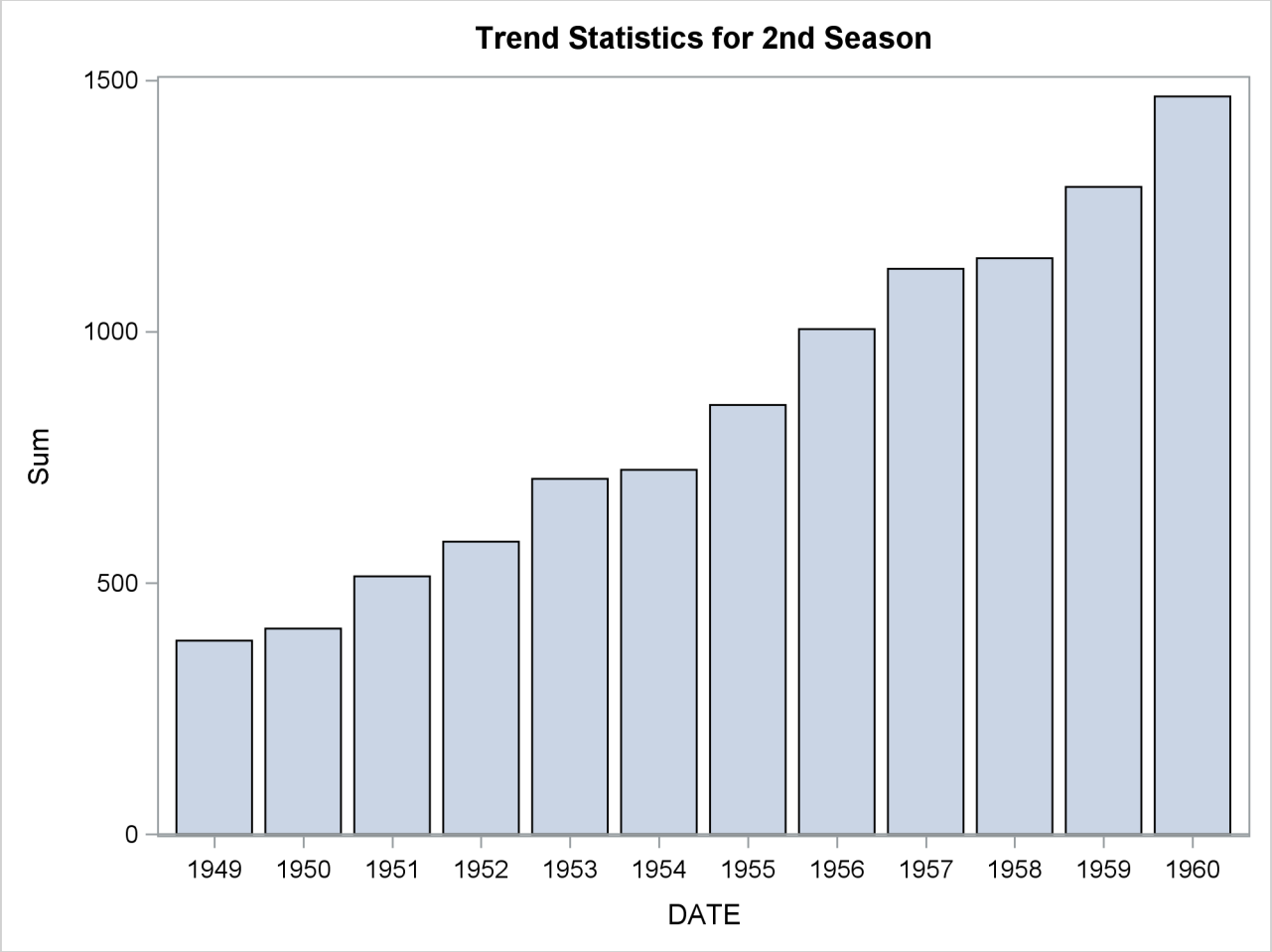


Using the trend statistics stored in the WORK.TREND data set, the following statements chart the sum of the transactions associated with each time period for the second season over time.

```
title1 "Trend Statistics for 2nd Season";
proc sgplot data=trend;
   where _season_ = 2;
   vbar date / freq=sum;
   format date year4.;
   yaxis label='Sum';
run;
```

The results of this trend analysis are shown in Output 39.2.3.

Output 39.2.3 Trend Statistics Bar Chart

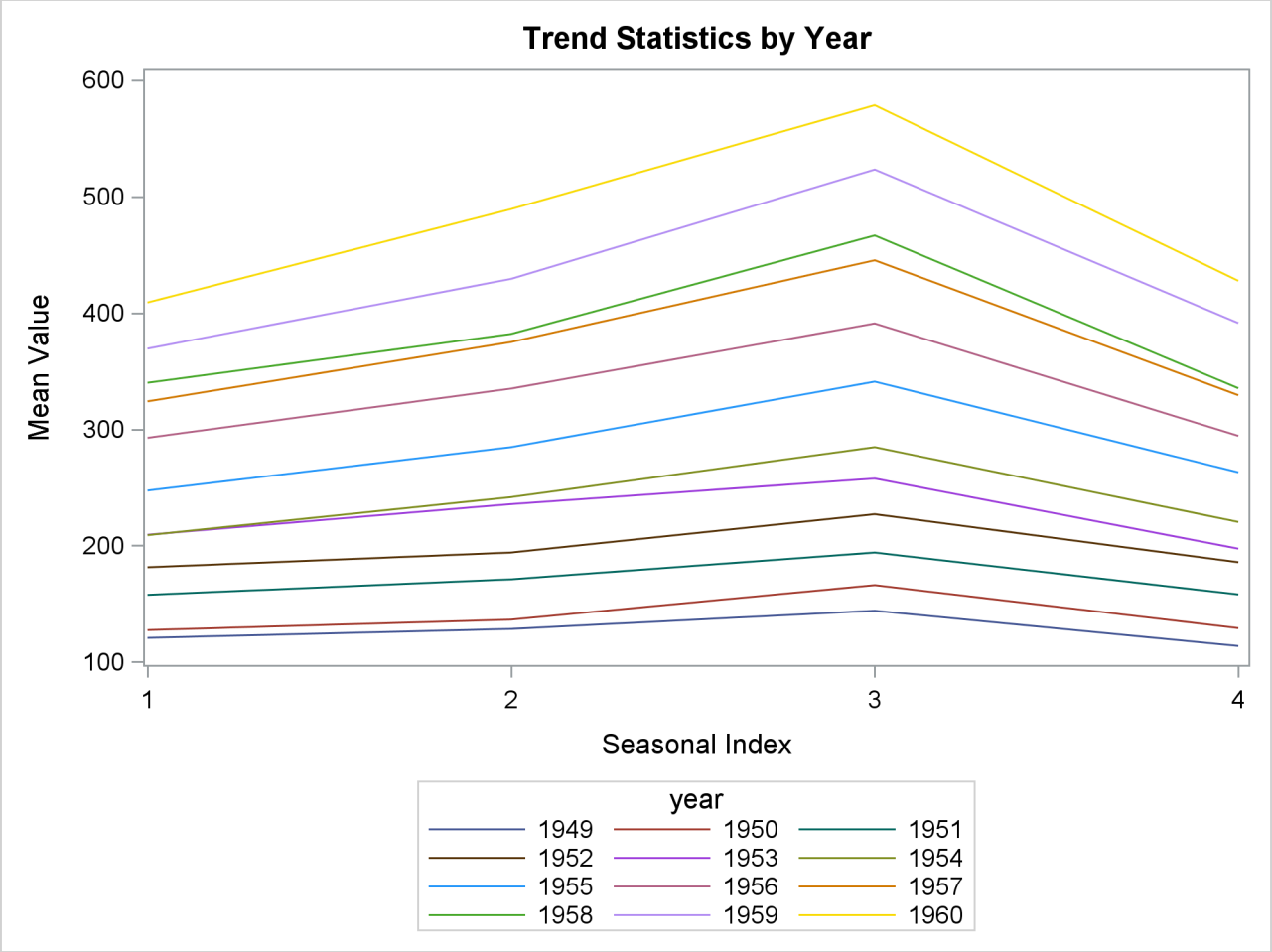


Using the trend statistics stored in the WORK.TREND data set, the following statements plot the mean of the transactions associated with each time period by each year over time.

```
data trend;
   set trend;
   year = year(date);
run;

title1 "Trend Statistics by Year";
proc sgplot data=trend;
   series x=_season_ y=mean / group=year lineattrs=(pattern=solid);
   xaxis values=(1 to 4 by 1);
run;
```

The results of this trend analysis are shown in Output 39.2.4.

Output 39.2.4 Trend Statistics

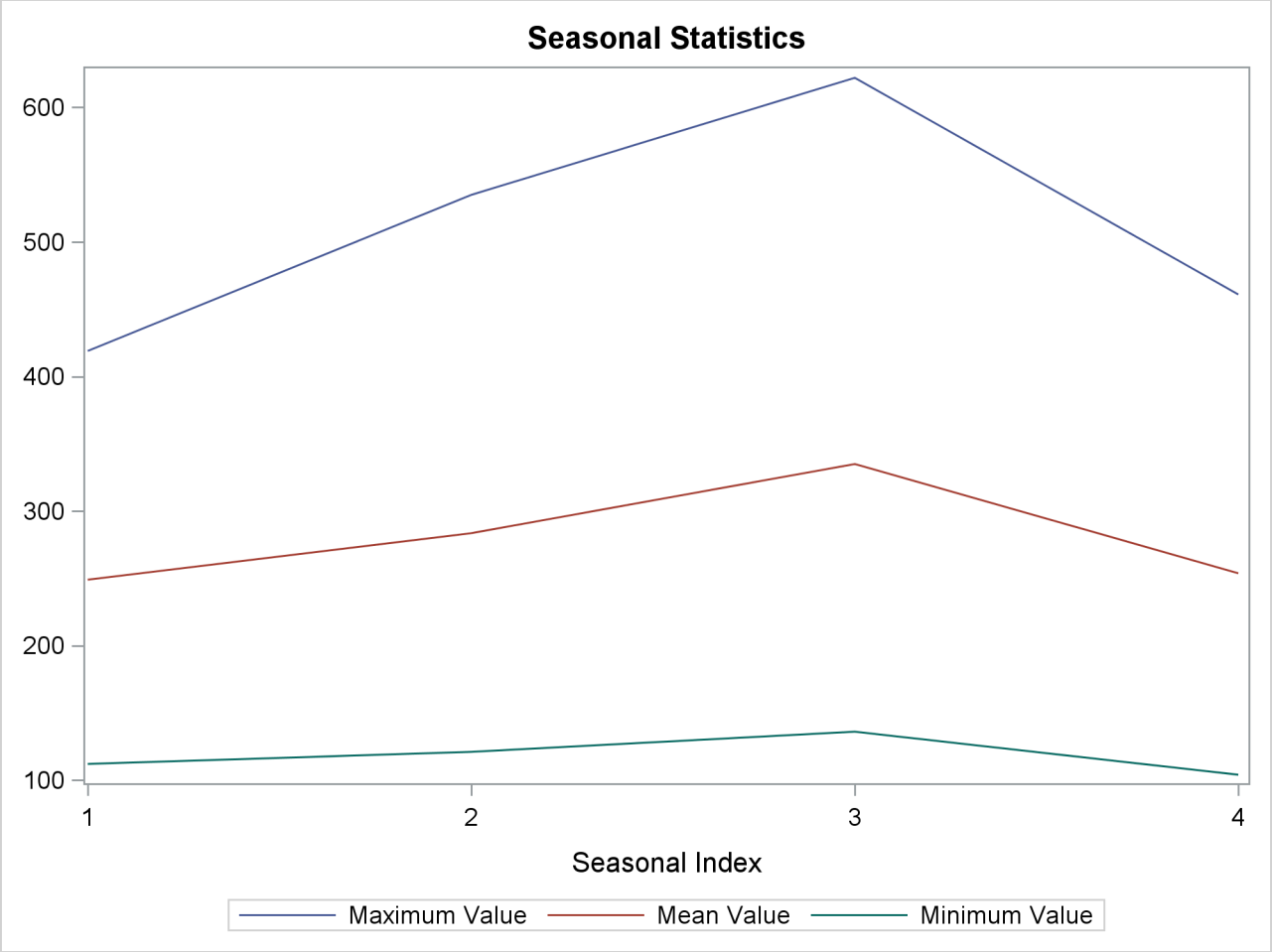


Using the season statistics stored in the WORK.SEASON data set, the following statements plot various season statistics for each season.

```
title1 "Seasonal Statistics";
proc sgplot data=season;
   series x=_season_ y=max  / lineattrs=(pattern=solid);
   series x=_season_ y=mean / lineattrs=(pattern=solid);
   series x=_season_ y=min  / lineattrs=(pattern=solid);
   yaxis display=(nolabel);
   xaxis values=(1 to 4 by 1);
run;
```

The results of this seasonal analysis are shown in Output 39.2.5.

Output 39.2.5 Seasonal Statistics Plot



Example 39.3: Illustration of ODS Graphics

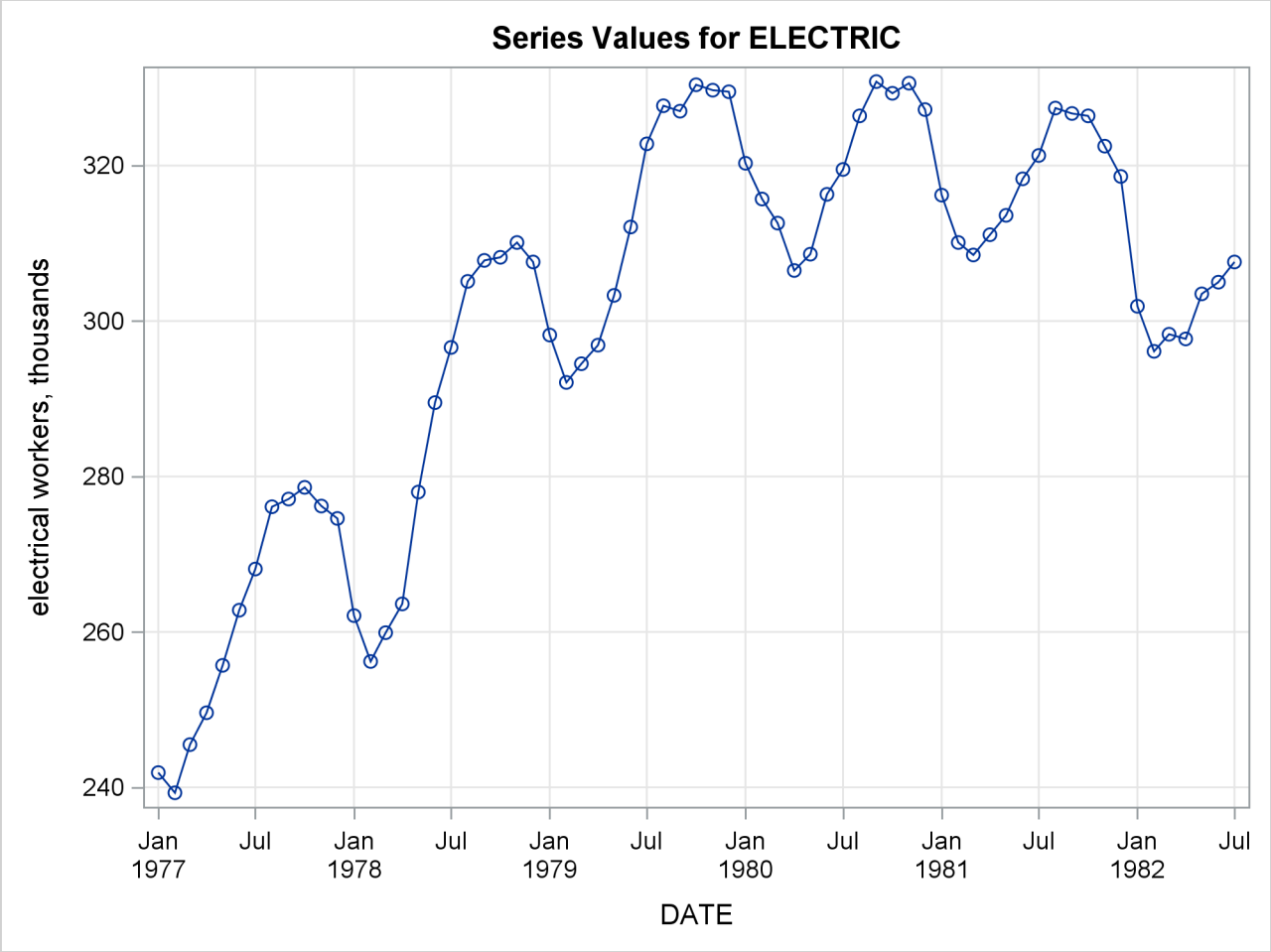
This example illustrates the use of ODS graphics.

The following statements use the SASHELP.WORKERS data set to study the time series of electrical workers and its interaction with the simply differenced series of masonry workers. The series plot, the correlation panel, the seasonal adjustment panel, and all cross-series plots are requested. Output 39.3.1 through Output 39.3.4 show a selection of the plots created.

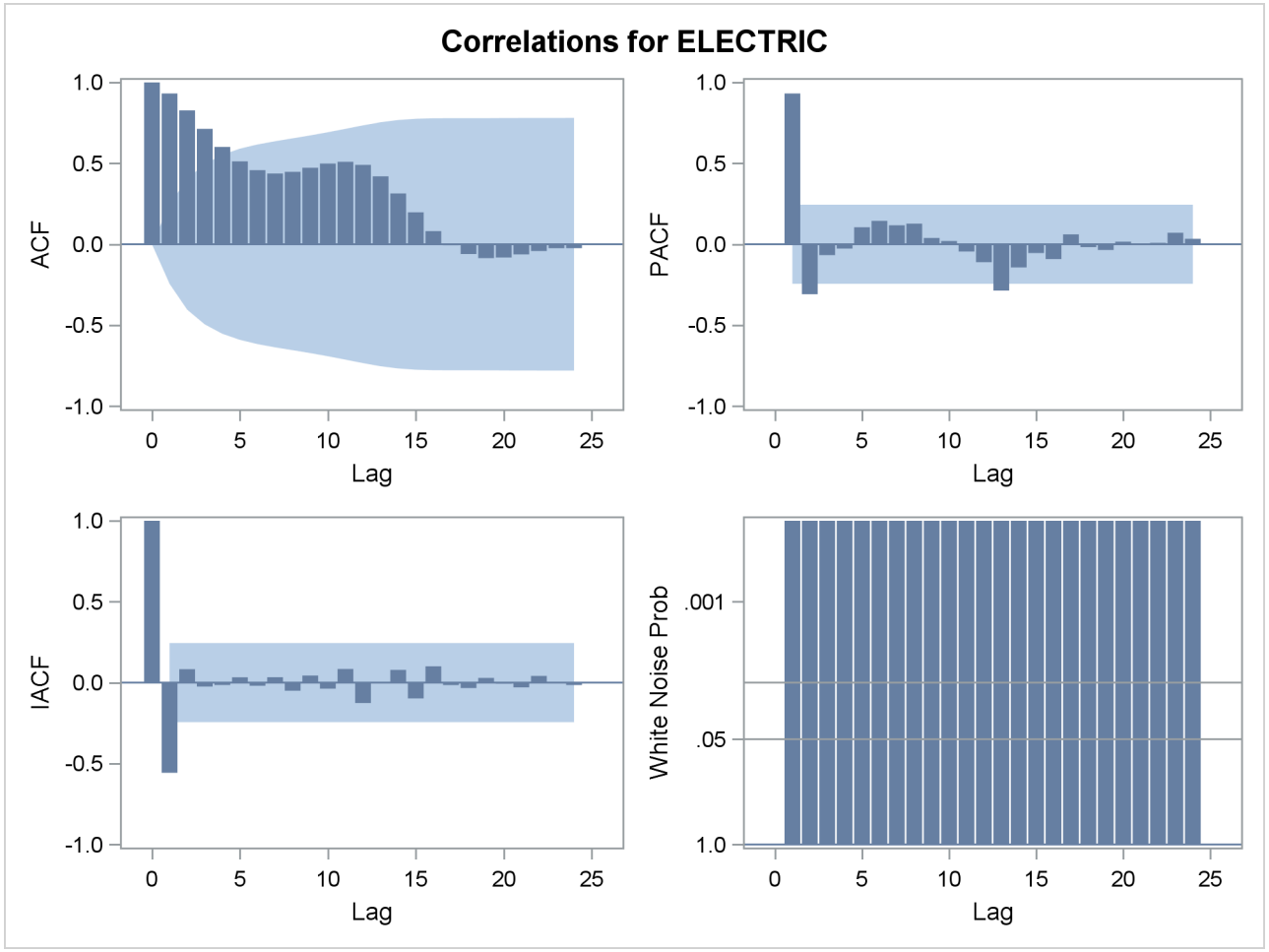
The graphical displays are requested by specifying the PLOTS= or CROSSPLOTS= options in the PROC TIMESERIES statement. For information about the graphics available in the TIMESERIES procedure, see the section "ODS Graphics Names" on page 2710.

```
title "Illustration of ODS Graphics";
proc timeseries data=sashelp.workers out=_null_
                plots=(series corr decomp)
                crossplots=all;
   id date interval=month;
   var electric;
   crossvar masonry / dif=(1);
run;
```

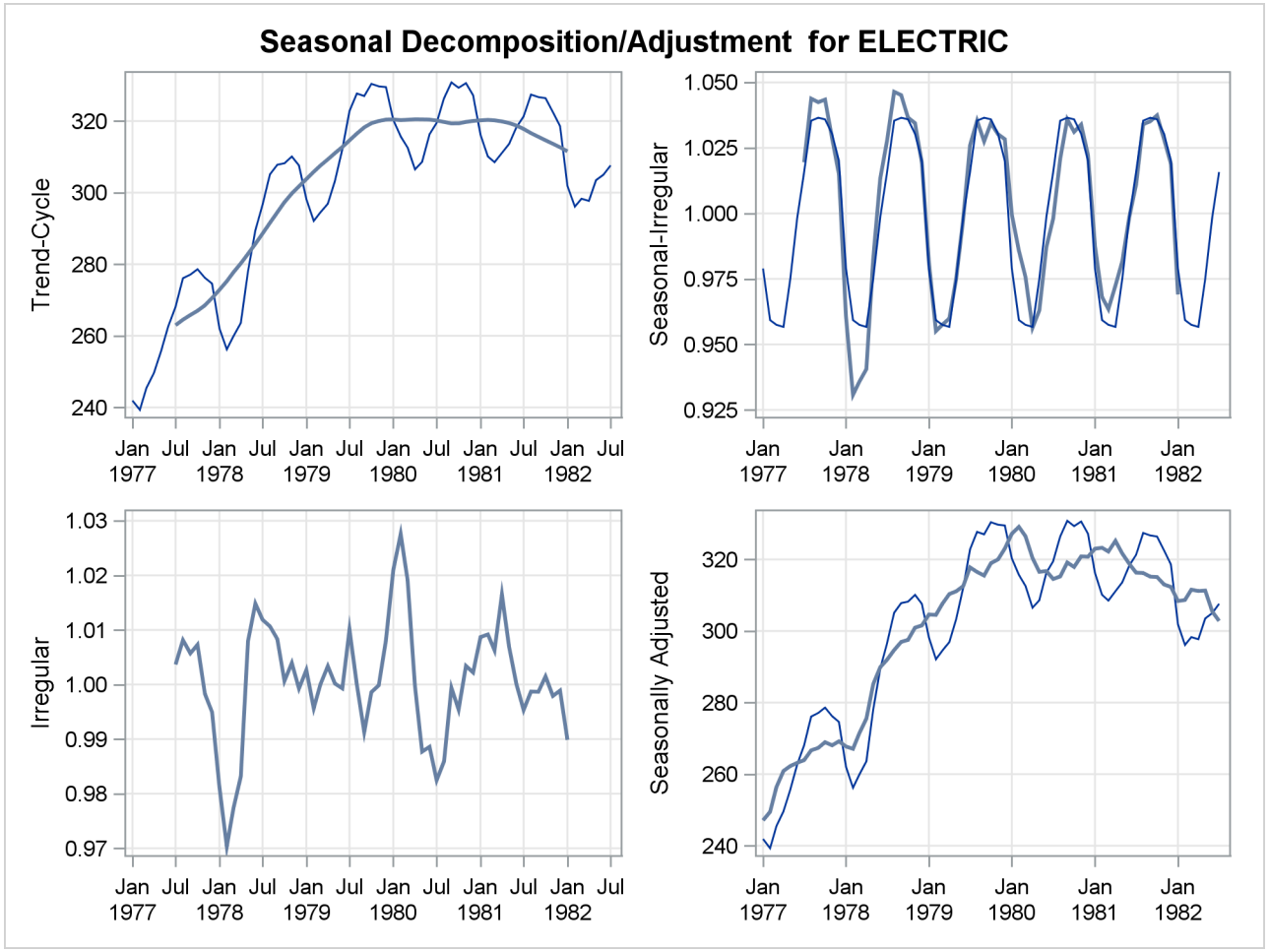
Output 39.3.1 Series Plot



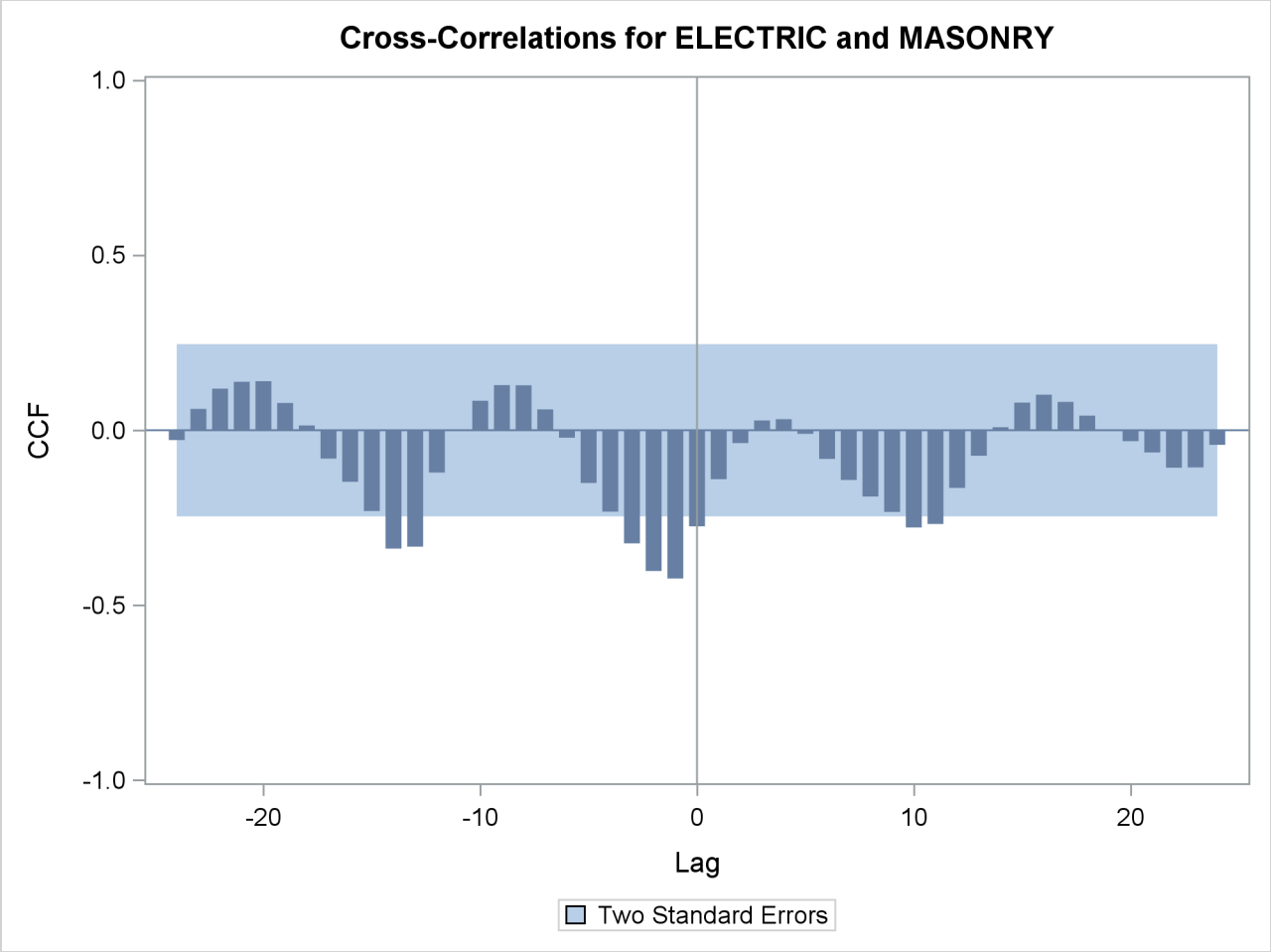
Output 39.3.2 Correlation Panel



Output 39.3.3 Seasonal Decomposition Panel



Output 39.3.4 Cross-Correlation Plot



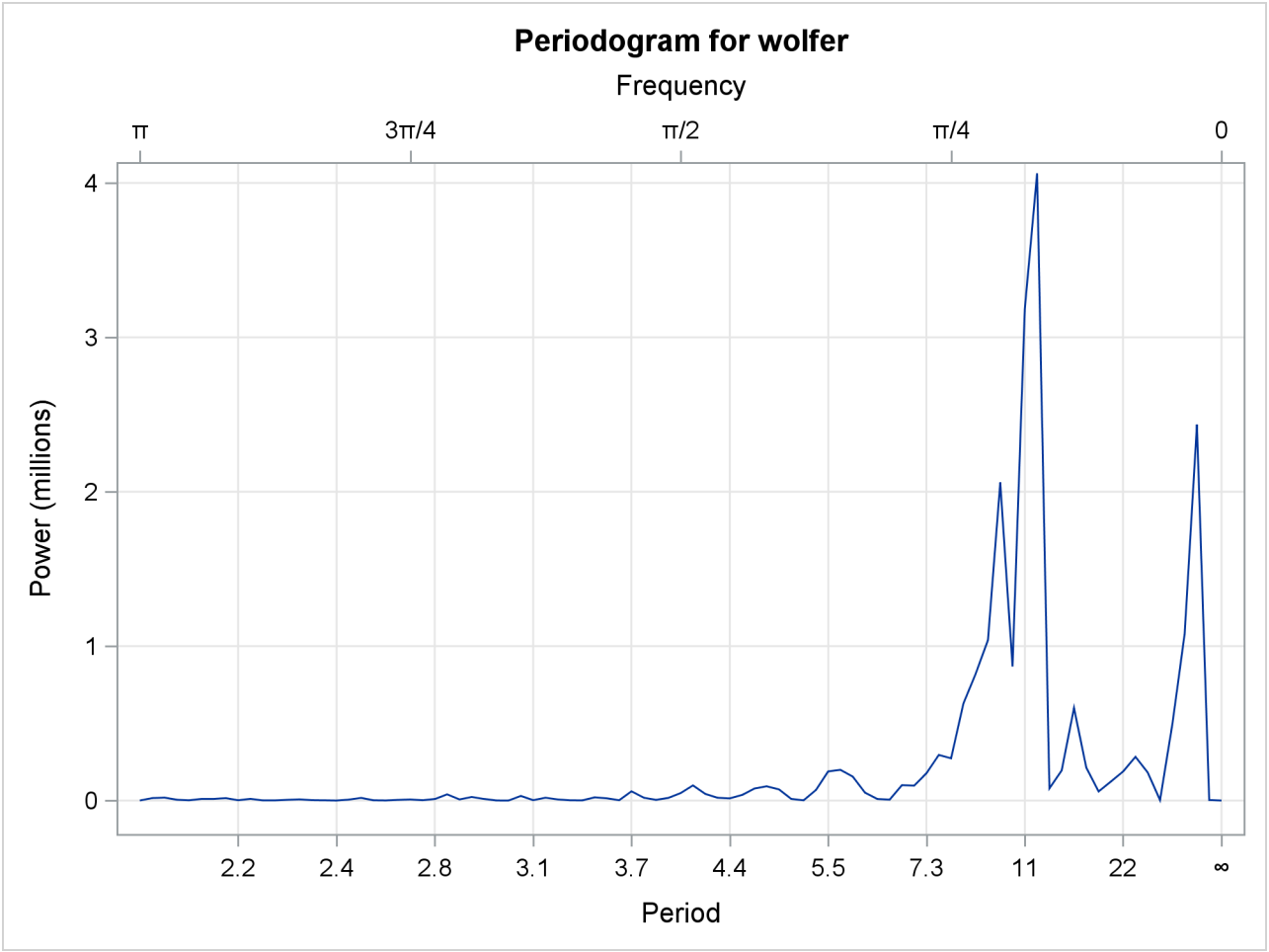
Example 39.4: Illustration of Spectral Analysis

This example illustrates the use of spectral analysis.

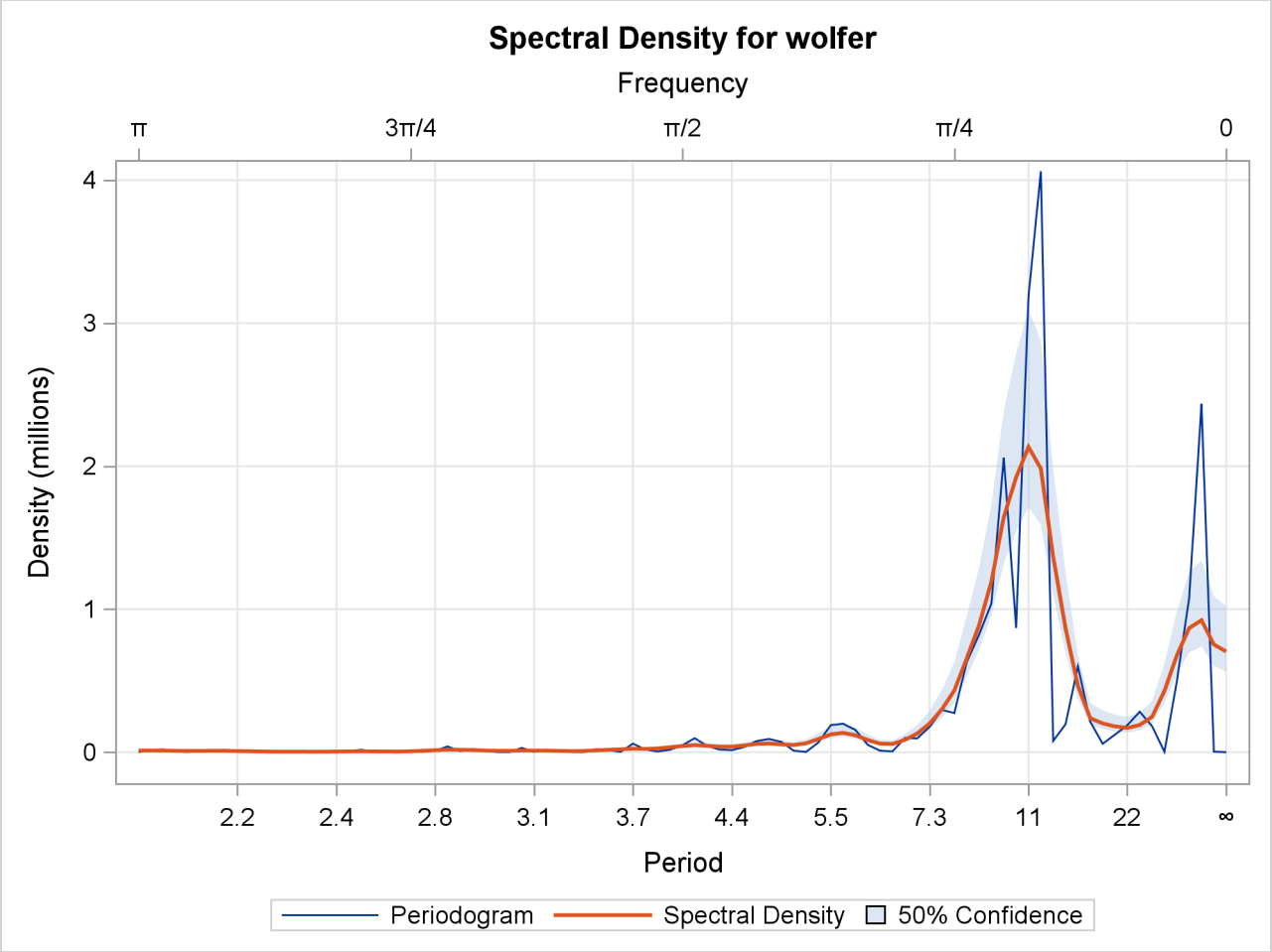
The following statements perform a spectral analysis on the SUNSPOT dataset. The periodogram is displayed as a function of the period and frequency in Output 39.4.1. The estimated spectral density together with its 50% confidence limits are displayed in Output 39.4.2.

```
title "Wolfer's Sunspot Data";

proc timeseries data=sunspot plot=(series periodogram spectrum);
   var wolfer;
   id year interval=year;
   spectra freq period p s / adjmean bart c=1.5 expon=0.2;
run;
```

Output 39.4.1 Periodogram



Output 39.4.2 Spectral Density Plot



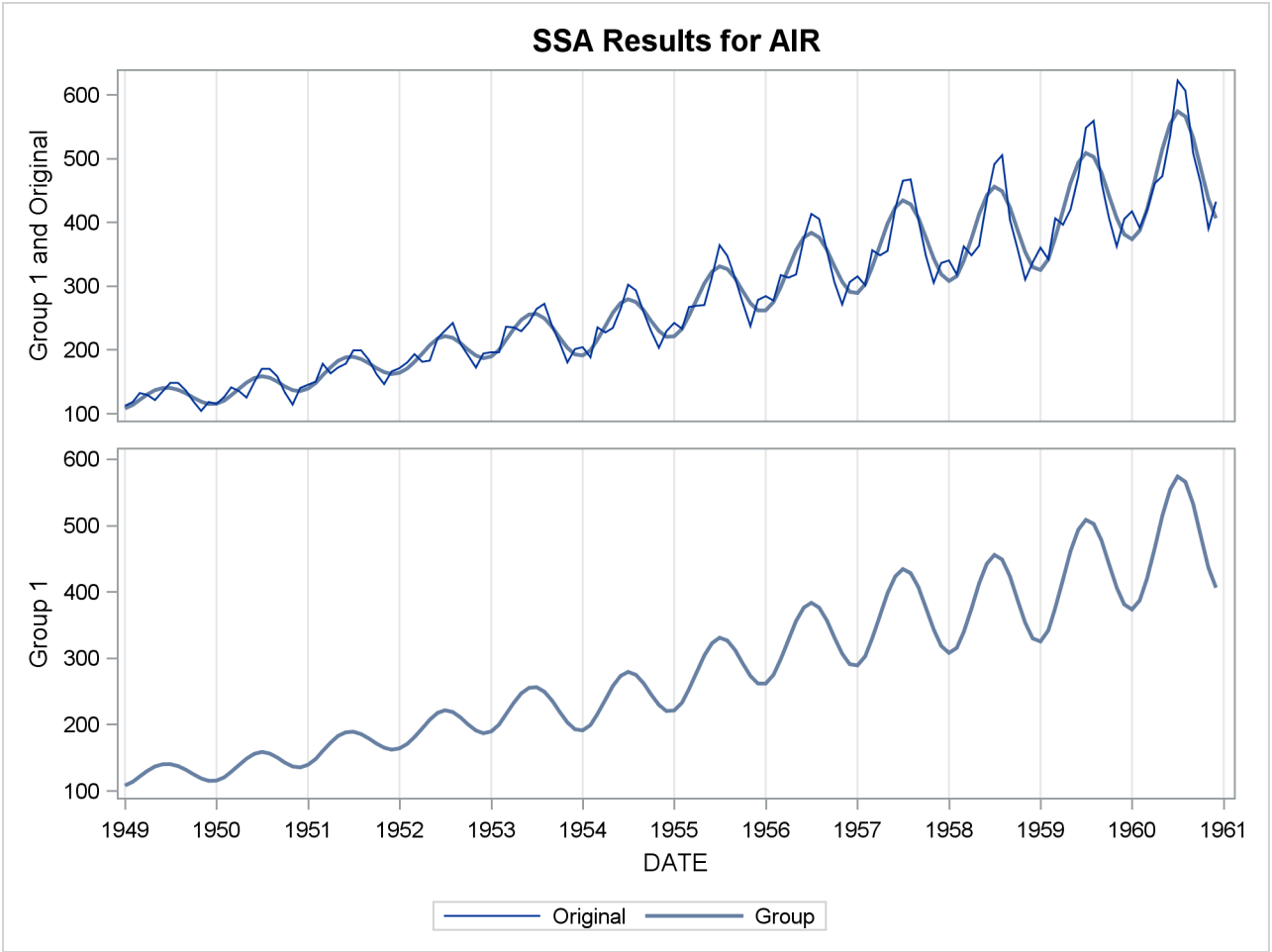
Example 39.5: Illustration of Singular Spectrum Analysis

This example illustrates the use of singular spectrum analysis.

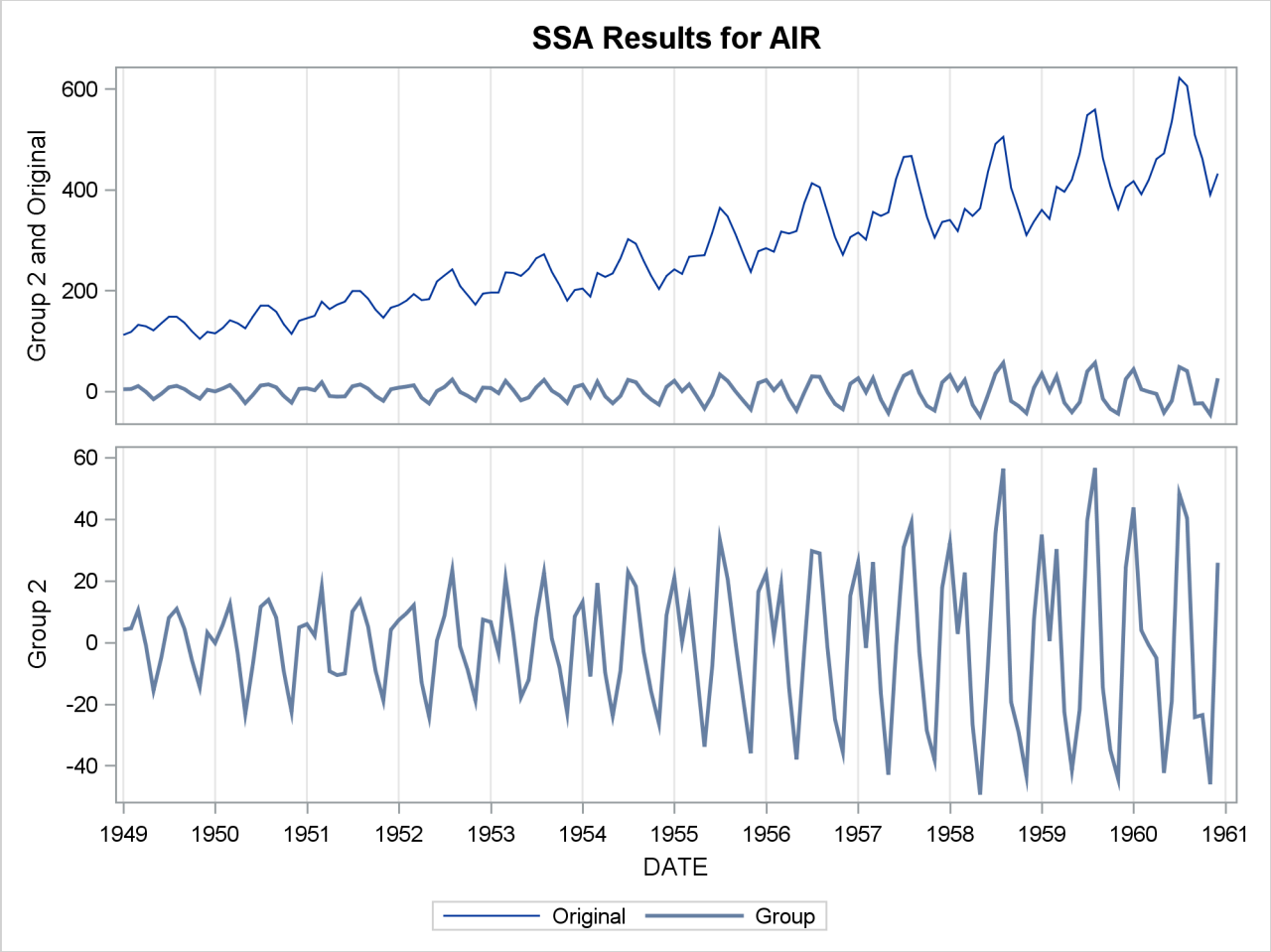
The following statements extract two additive components from the SASHELP.AIR time series by using the THRESHOLDPCT= option to specify that the first component represent 80% of the variability in the series. The resulting groupings, consisting of the first three and remaining nine singular value components, are presented in Output 39.5.1 through Output 39.5.3.

```
title "SSA of AIR data";

proc timeseries data=sashelp.air plot=ssa;
   id date interval=month;
   var air;
   ssa / length=12 THRESHOLDPCT=80;
run;
```

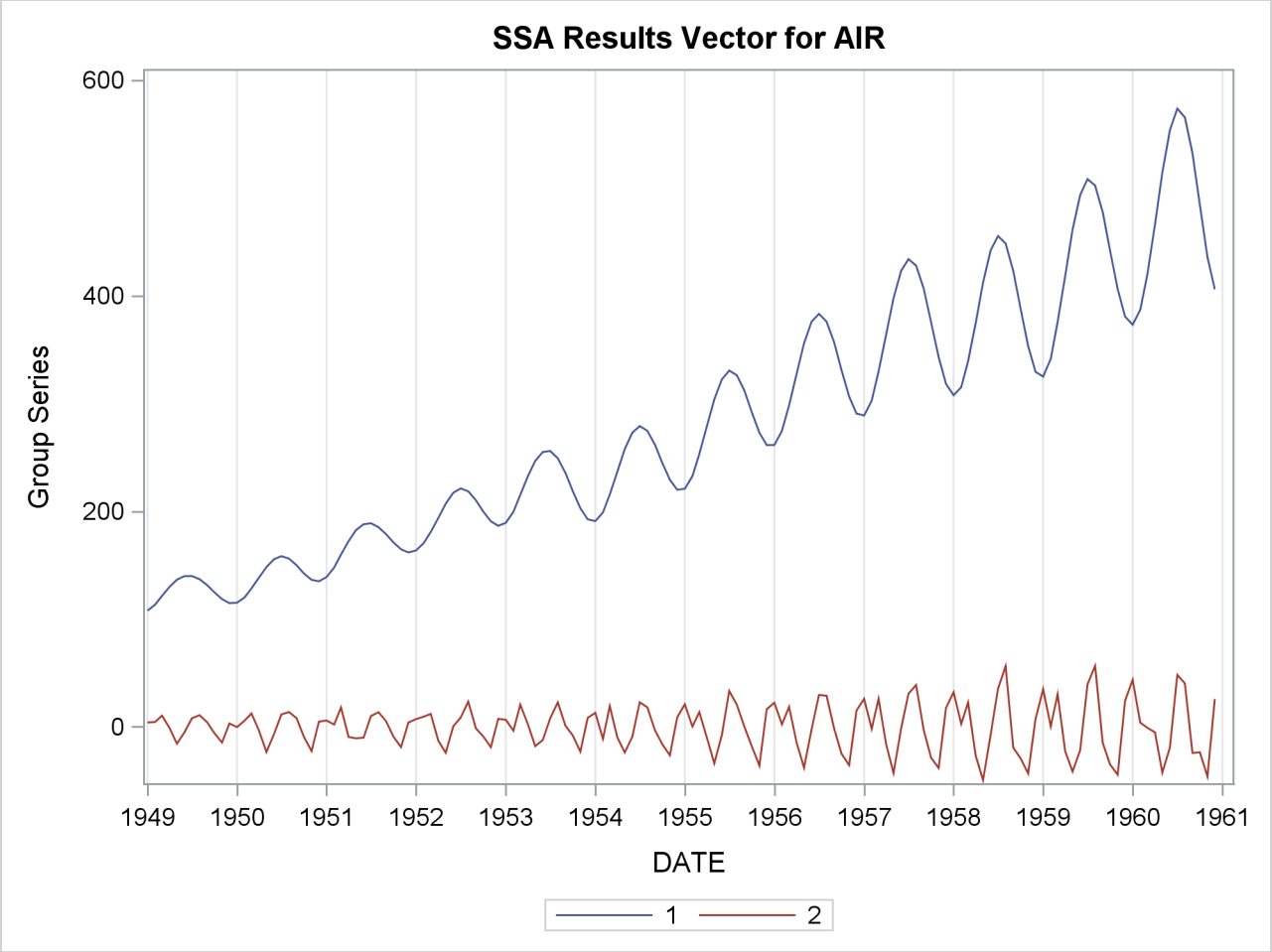
Output 39.5.1 Singular Value Grouping #1 Plot



Output 39.5.2 Singular Value Grouping #2 Plot



Output 39.5.3 Singular Value Components Plot



References

Brockwell, P. J., and Davis, R. A. (1991). *Time Series: Theory and Methods*. 2nd ed. New York: Springer-Verlag.

Cooley, J. W., and Tukey, J. W. (1965). "An Algorithm for the Machine Calculation of Complex Fourier Series." *Mathematics of Computation* 19:297–301.

Golyandina, N., Nekrutkin, V., and Zhigljavsky, A. (2001). *Analysis of Time Series Structure SSA and Related Techniques*. Boca Raton, FL: CRC Press.

Greene, W. H. (2000). *Econometric Analysis*. 4th ed. Upper Saddle River, NJ: Prentice-Hall.

Hodrick, R. J., and Prescott, E. C. (1980). "Post-war U.S. Business Cycles: An Empirical Investigation." Discussion Paper 451, Carnegie Mellon University.

Makridakis, S. G., and Wheelwright, S. C. (1978). *Interactive Forecasting: Univariate and Multivariate Methods*. 2nd ed. San Francisco: Holden-Day.

Monro, D. M., and Branch, J. L. (1977). "Algorithm AS 117: The Chirp Discrete Fourier Transform of General Length." *Journal of the Royal Statistical Society, Series C* 26:351–361.

Priestley, M. B. (1981). *Spectral Analysis and Time Series*. London: Academic Press.

Pyle, D. (1999). *Data Preparation for Data Mining*. San Francisco: Morgan Kaufmann.

Singleton, R. C. (1969). "An Algorithm for Computing the Mixed Radix Fast Fourier Transform." *IEEE Transactions on Audio and Electroacoustics* 17:93–103.

Stoffer, D. S., and Toloi, C. M. C. (1992). "A Note on the Ljung-Box-Pierce Portmanteau Statistic with Missing Data." *Statistics and Probability Letters* 13:391–396.

Wheelwright, S. C., and Makridakis, S. G. (1973). *Forecasting Methods for Management*. 3rd ed. New York: Wiley-Interscience.

Chapter 40
The TSCSREG Procedure

Contents

Overview: The TSCSREG Procedure **2729**
Getting Started: The TSCSREG Procedure **2730**
Specifying the Input Data 2730
Unbalanced Data 2730
Specifying the Regression Model 2731
Estimation Techniques 2732
Introductory Example 2733
Syntax: The TSCSREG Procedure **2734**
Functional Summary 2735
PROC TSCSREG Statement 2736
BY Statement 2736
ID Statement 2737
MODEL Statement 2738
TEST Statement 2739
Details: The TSCSREG Procedure **2740**
ODS Table Names 2740
Examples: The TSCSREG Procedure **2741**
References **2741**

Overview: The TSCSREG Procedure

The TSCSREG (time series cross section regression) procedure analyzes a class of linear econometric models that commonly arise when time series and cross-sectional data are combined. The TSCSREG procedure deals with panel data sets that consist of time series observations on each of several cross-sectional units.

The TSCSREG procedure is very similar to the PANEL procedure; for full description, syntax details, models, and estimation methods, see Chapter 27, "The PANEL Procedure." The TSCSREG procedure is no longer being updated, and it shares the code base with the PANEL procedure.

The original TSCSREG procedure was developed by Douglas J. Drummond and A. Ronald Gallant, and contributed to the Version 5 SUGI Supplemental Library in 1979. The original code was changed substantially over the years. Additional new methods as well as other new features are currently included in the PANEL PROCEDURE. SAS Institute would like to thank Dr. Drummond and Dr. Gallant for their contribution of the original version of the TSCSREG procedure.

Getting Started: The TSCSREG Procedure

Specifying the Input Data

The input data set used by the TSCSREG procedure must be sorted by cross section and by time within each cross section. Therefore, the first step in using PROC TSCSREG is to make sure that the input data set is sorted. Normally, the input data set contains a variable that identifies the cross section for each observation and a variable that identifies the time period for each observation.

To illustrate, suppose that you have a data set A that contains data over time for each of several states. You want to regress the variable Y on regressors X1 and X2. Cross sections are identified by the variable STATE, and time periods are identified by the variable DATE. The following statements sort the data set A appropriately:

```
proc sort data=a;
   by state date;
run;
```

The next step is to invoke the TSCSREG procedure and specify the cross section and time series variables in an ID statement. List the variables in the ID statement exactly as they are listed in the BY statement.

```
proc tscsreg data=a;
   id state date;
```

Alternatively, you can omit the ID statement and use the CS= and TS= options on the PROC TSCSREG statement to specify the number of cross sections in the data set and the number of time series observations in each cross section.

Unbalanced Data

In the case of fixed-effects and random-effects models, the TSCSREG procedure is capable of processing data with different numbers of time series observations across different cross sections. You must specify the ID statement to estimate models that use unbalanced data. The missing time series observations are recognized by the absence of time series ID variable values in some of the cross sections in the input data set. Moreover, if an observation with a particular time series ID value and cross-sectional ID value is present in the input data set, but one or more of the model variables are missing, that time series point is treated as missing for that cross section.

Specifying the Regression Model

Next, specify the linear regression model with a MODEL statement, as shown in the following statements.

```
proc tscsreg data=a;
   id state date;
   model y = x1 x2;
run;
```

The MODEL statement in PROC TSCSREG is specified like the MODEL statement in other SAS regression procedures: the dependent variable is listed first, followed by an equal sign, followed by the list of regressor variables.

The reason for using PROC TSCSREG instead of other SAS regression procedures is that you can incorporate a model for the structure of the random errors. It is important to consider what kind of error structure model is appropriate for your data and to specify the corresponding option in the MODEL statement.

The error structure options supported by the TSCSREG procedure are FIXONE, FIXTWO, RANONE, RANTWO, FULLER, PARKS, and DASILVA. See "Details: The TSCSREG Procedure" on page 2740 for more information about these methods and the error structures they assume.

By default, the two-way random-effects error model structure is used while Fuller-Battese and Wansbeek-Kapteyn methods are used for the estimation of variance components in balanced data and unbalanced data, respectively. Thus, the preceding example is the same as specifying the RANTWO option, as shown in the following statements:

```
proc tscsreg data=a;
   id state date;
   model y = x1 x2 / rantwo;
run;
```

You can specify more than one error structure option in the MODEL statement; the analysis is repeated using each method specified. You can use any number of MODEL statements to estimate different regression models or estimate the same model by using different options.

In order to aid in model specification within this class of models, the procedure provides two specification test statistics. The first is an F statistic that tests the null hypothesis that the fixed-effects parameters are all zero. The second is a Hausman m-statistic that provides information about the appropriateness of the random-effects specification. It is based on the idea that, under the null hypothesis of no correlation between the effects variables and the regressors, OLS and GLS are consistent, but OLS is inefficient. Hence, a test can be based on the result that the covariance of an efficient estimator with its difference from an inefficient estimator is zero. Rejection of the null hypothesis might suggest that the fixed-effects model is more appropriate.

The procedure also provides the Buse R-square measure, which is the most appropriate goodness-of-fit measure for models estimated by using GLS. This number is interpreted as a measure of the proportion of the transformed sum of squares of the dependent variable that is attributable to the influence of the independent variables. In the case of OLS estimation, the Buse R-square measure is equivalent to the usual R-square measure.

Estimation Techniques

If the effects are fixed, the models are essentially regression models with dummy variables that correspond to the specified effects. For fixed-effects models, ordinary least squares (OLS) estimation is equivalent to best linear unbiased estimation.

The output from TSCSREG is identical to what one would obtain from creating dummy variables to represent the cross-sectional and time (fixed) effects. The output is presented in this manner to facilitate comparisons to the least squares dummy variables estimator (LSDV). As such, the inclusion of a intercept term implies that one dummy variable must be dropped. The actual estimation of the fixed-effects models is not LSDV. LSDV is much too cumbersome to implement. Instead, TSCSREG operates in a two step fashion. In the first step, the following occurs:

- *One-way fixed-effects model*: In the one-way fixed-effects model, the data is transformed by removing the cross-sectional means from the dependent and independent variables. The following is true:

$$\tilde{y}_{it} = y_{it} - \bar{y}_{i\cdot}$$

$$\tilde{\mathbf{x}}_{it} = \mathbf{x}_{it} - \bar{\mathbf{x}}_{i\cdot}$$

- *Two-way fixed-effects model*: In the two-way fixed-effects model, the data is transformed by removing the cross-sectional and time means and adding back the overall means:

$$\tilde{y}_{it} = y_{it} - \bar{y}_{i\cdot} - \bar{y}_{\cdot t} + \bar{\bar{y}}$$

$$\tilde{\mathbf{x}}_{it} = \mathbf{x}_{it} - \bar{\mathbf{x}}_{i\cdot} - \bar{\mathbf{x}}_{\cdot t} + \bar{\bar{\mathbf{x}}}$$

where the symbols:

y_{it} and $\mathbf{x}_{it}$ are the dependent variable (a scalar) and the explanatory variables (a vector whose columns are the explanatory variables not including a constant), respectively

$\bar{y}_{i\cdot}$ and $\bar{\mathbf{x}}_{i\cdot}$ are cross section means

$\bar{y}_{\cdot t}$ and $\bar{\mathbf{x}}_{\cdot t}$ are time means

$\bar{\bar{y}}$ and $\bar{\bar{\mathbf{x}}}$ are the overall means

The second step consists of running OLS on the properly demeaned series, provided that the data are balanced. The unbalanced case is slightly more difficult, because the structure of the missing data must be retained. For this case, PROC TSCSREG uses a slight specialization on Wansbeek and Kapteyn.

The other alternative is to assume that the effects are random. In the one-way case, $E(v_i) = 0$, $E(v_i^2) = \sigma_v^2$, and $E(v_i v_j) = 0$ for $i \neq j$, and v_i is uncorrelated with ϵ_{it} for all i and t. In the two-way case, in addition to all of the preceding, $E(e_t) = 0$, $E(e_t^2) = \sigma_e^2$, and $E(e_t e_s) = 0$ for $t \neq s$, and the e_t are uncorrelated with the v_i and the ϵ_{it} for all i and t. Thus, the model is a variance components model, with the variance components σ_v^2, σ_e^2, and σ_ϵ^2, to be estimated. A crucial implication of such a specification is that the effects are independent of the regressors. For random-effects models, the estimation method is an estimated generalized least squares (EGLS) procedure that involves estimating the variance components in the first stage and using the estimated variance covariance matrix thus obtained to apply generalized least squares (GLS) to the data.

Introductory Example

The following example uses the cost function data from Greene (1990) to estimate the variance components model. The variable OUTPUT is the log of output in millions of kilowatt-hours, and COST is the log of cost in millions of dollars. See to Greene (1990) for details.

```
title1;
data greene;
   input firm year output cost @@;
   df1 = firm = 1;
   df2 = firm = 2;
   df3 = firm = 3;
   df4 = firm = 4;
   df5 = firm = 5;
   d60 = year = 1960;
   d65 = year = 1965;
   d70 = year = 1970;
datalines;
    1 1955   5.36598   1.14867  1 1960   6.03787   1.45185

... more lines ...
```

Usually you cannot explicitly specify all the explanatory variables that affect the dependent variable. The omitted or unobservable variables are summarized in the error disturbances. The TSCSREG procedure used with the RANTWO option specifies the two-way random-effects error model where the variance components are estimated by the Fuller-Battese method, because the data are balanced and the parameters are efficiently estimated by using the GLS method. The variance components model used by the Fuller-Battese method is

$$y_{it} = \sum_{k=1}^{K} X_{itk}\boldsymbol{\beta}_k + v_i + e_t + \epsilon_{it} i = 1, \ldots, N; t = 1, \ldots, T$$

The following statements fit this model.

```
proc sort data=greene;
   by firm year;
run;

proc tscsreg data=greene;
   model cost = output / rantwo;
   id firm year;
run;
```

The TSCSREG procedure output is shown in Figure 40.1. A model description is printed first; it reports the estimation method used and the number of cross sections and time periods. The variance components estimates are printed next. Finally, the table of regression parameter estimates shows the estimates, standard errors, and t tests.

Figure 40.1 The Variance Components Estimates

The TSCSREG Procedure
Fuller and Battese Variance Components (RanTwo)

Dependent Variable: cost

Model Description	
Estimation Method	RanTwo
Number of Cross Sections	6
Time Series Length	4

Fit Statistics			
SSE	0.3481	DFE	22
MSE	0.0158	Root MSE	0.1258
R-Square	0.8136		

Variance Component Estimates	
Variance Component for Cross Sections	0.046907
Variance Component for Time Series	0.00906
Variance Component for Error	0.008749

Hausman Test for Random Effects		
DF	m Value	Pr > m
1	26.46	<.0001

Parameter Estimates

Variable	DF	Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	-2.99992	0.6478	-4.63	0.0001
output	1	0.746596	0.0762	9.80	<.0001

Syntax: The TSCSREG Procedure

The following statements are used with the TSCSREG procedure.

PROC TSCSREG *options* ;
  BY *variables* ;
  ID *cross-section-id-variable time-series-id-variable* ;
  MODEL *dependent = regressor-variables / options* ;
  TEST *equation1 < ,equation2. . . >* ;

Functional Summary

The statements and options used with the TSCSREG procedure are summarized in the following table.

Table 40.1 Functional Summary

Description	Statement	Option
Data Set Options		
specify the input data set	TSCSREG	DATA=
write parameter estimates to an output data set	TSCSREG	OUTEST=
include correlations in the OUTEST= data set	TSCSREG	CORROUT
include covariances in the OUTEST= data set	TSCSREG	COVOUT
specify number of time series observations	TSCSREG	TS=
specify number of cross sections	TSCSREG	CS=
Declaring the Role of Variables		
specify BY-group processing	BY	
specify the cross section and time ID variables	ID	
Printing Control Options		
print correlations of the estimates	MODEL	CORRB
print covariances of the estimates	MODEL	COVB
suppress printed output	MODEL	NOPRINT
perform tests of linear hypotheses	TEST	
Model Estimation Options		
specify the one-way fixed-effects model	MODEL	FIXONE
specify the two-way fixed-effects model	MODEL	FIXTWO
specify the one-way random-effects model	MODEL	RANONE
specify the two-way random-effects model	MODEL	RANTWO
specify Fuller-Battese method	MODEL	FULLER
specify PARKS	MODEL	PARKS
specify Da Silva method	MODEL	DASILVA
specify order of the moving-average error process for Da Silva method	MODEL	M=
print Φ matrix for Parks method	MODEL	PHI
print autocorrelation coefficients for Parks method	MODEL	RHO
suppress the intercept term	MODEL	NOINT
control check for singularity	MODEL	SINGULAR=

PROC TSCSREG Statement

PROC TSCSREG *options* **;**

The following options can be specified in the PROC TSCSREG statement.

DATA=*SAS-data-set*
names the input data set. The input data set must be sorted by cross section and by time period within cross section. If you omit the DATA= option, the most recently created SAS data set is used.

TS=*number*
specifies the number of observations in the time series for each cross section. The TS= option value must be greater than 1. The TS= option is required unless an ID statement is used. Note that the number of observations for each time series must be the same for each cross section and must cover the same time period.

CS=*number*
specifies the number of cross sections. The CS= option value must be greater than 1. The CS= option is required unless an ID statement is used.

OUTEST=*SAS-data-set*
the parameter estimates. When the OUTEST= option is not specified, the OUTEST= data set is not created.

OUTCOV
COVOUT
writes the covariance matrix of the parameter estimates to the OUTEST= data set.

OUTCORR
CORROUT
writes the correlation matrix of the parameter estimates to the OUTEST= data set.

In addition, any of the following MODEL statement options can be specified in the PROC TSCSREG statement: CORRB, COVB, FIXONE, FIXTWO, RANONE, RANTWO, FULLER, PARKS, DASILVA, NOINT, NOPRINT, M=, PHI, RHO, and SINGULAR=. When specified in the PROC TSCSREG statement, these options are equivalent to specifying the options for every MODEL statement.

BY Statement

BY *variables* **;**

A BY statement can be used with PROC TSCSREG to obtain separate analyses on observations in groups defined by the BY variables. When a BY statement appears, the input data set must be sorted by the BY variables as well as by cross section and time period within the BY groups.

When both an ID statement and a BY statement are specified, the input data set must be sorted first with respect to BY variables and then with respect to the cross section and time series ID variables. For example,

```
proc sort data=a;
   by byvar1 byvar2 csid tsid;
run;

proc tscsreg data=a;
   by byvar1 byvar2;
   id csid tsid;
   ...
run;
```

When both a BY statement and an ID statement are used, the data set might have a different number of cross sections or a different number of time periods in each BY group. If no ID statement is used, the CS=*N* and TS=*T* options must be specified and each BY group must contain $N \times T$ observations.

ID Statement

ID *cross-section-id-variable time-series-id-variable* **;**

The ID statement is used to specify variables in the input data set that identify the cross section and time period for each observation.

When an ID statement is used, the TSCSREG procedure verifies that the input data set is sorted by the cross section ID variable and by the time series ID variable within each cross section. The TSCSREG procedure also verifies that the time series ID values are the same for all cross sections.

To make sure the input data set is correctly sorted, use PROC SORT with a BY statement with the variables listed exactly as they are listed in the ID statement to sort the input data set. For example,

```
proc sort data=a;
   by csid tsid;
run;

proc tscsreg data=a;
   id csid tsid;
   ... etc. ...
run;
```

If the ID statement is not used, the TS= and CS= options must be specified on the PROC TSCSREG statement. Note that the input data must be sorted by time within cross section, regardless of whether the cross section structure is given by an ID statement or by the options TS= and CS=.

If an ID statement is specified, the time series length *T* is set to the minimum number of observations for any cross section, and only the first *T* observations in each cross section are used. If both the ID statement and the TS= and CS= options are specified, the TS= and CS= options are ignored.

MODEL Statement

MODEL *response = regressors / options* **;**

The MODEL statement specifies the regression model and the error structure assumed for the regression residuals. The response variable on the left side of the equal sign is regressed on the independent variables listed after the equal sign. Any number of MODEL statements can be used. For each model statement, only one response variable can be specified on the left side of the equal sign.

The error structure is specified by the FIXONE, FIXTWO, RANONE, RANTWO, FULLER, PARKS, and DASILVA options. More than one of these options can be used, in which case the analysis is repeated for each error structure model specified.

Models can be given labels up to 32 characters in length. Model labels are used in the printed output to identify the results for different models. If no label is specified, the response variable name is used as the label for the model. The model label is specified as follows:

label: **MODEL** *response = regressors / options* **;**

The following options can be specified on the MODEL statement after a slash (/).

CORRB

CORR

prints the matrix of estimated correlations between the parameter estimates.

COVB

VAR

prints the matrix of estimated covariances between the parameter estimates.

FIXONE

specifies that a one-way fixed-effects model be estimated with the one-way model that corresponds to group effects only.

FIXTWO

specifies that a two-way fixed-effects model be estimated.

RANONE

specifies that a one-way random-effects model be estimated.

RANTWO

specifies that a two-way random-effects model be estimated.

FULLER

specifies that the model be estimated by using the Fuller-Battese method, which assumes a variance components model for the error structure.

PARKS

specifies that the model be estimated by using the Parks method, which assumes a first-order autoregressive model for the error structure.

DASILVA
specifies that the model be estimated by using the Da Silva method, which assumes a mixed variance-component moving-average model for the error structure.

M=*number*
specifies the order of the moving-average process in the Da Silva method. The M= value must be less than $T - 1$. The default is M=1.

PHI
prints the Φ matrix of estimated covariances of the observations for the Parks method. The PHI option is relevant only when the PARKS option is used.

RHO
prints the estimated autocorrelation coefficients for the Parks method.

NOINT
NOMEAN
suppresses the intercept parameter from the model.

NOPRINT
suppresses the normal printed output.

SINGULAR=*number*
specifies a singularity criterion for the inversion of the matrix. The default depends on the precision of the computer system.

TEST Statement

TEST *equation* < , *equation* . . . > < / *options* > ;

The TEST statement performs *F* tests of linear hypotheses about the regression parameters in the preceding MODEL statement. Each equation specifies a linear hypothesis to be tested. All hypotheses in one TEST statement are tested jointly. Variable names in the equations must correspond to regressors in the preceding MODEL statement, and each name represents the coefficient of the corresponding regressor. The keyword INTERCEPT refers to the coefficient of the intercept.

The following statements illustrate the use of the TEST statement:

```
proc tscsreg;
   model y = x1 x2 x3;
   test x1 = 0, x2 * .5 + 2 * x3= 0;
   test_int: test intercept=0, x3 = 0;
```

Note that a test of the following form is not permitted:

```
   test_bad: test x2 / 2 + 2 * x3= 0;
```

Do not use the division sign in test/restrict statements.

Details: The TSCSREG Procedure

Models, estimators, and methods are covered in detail in Chapter 27, "The PANEL Procedure."

ODS Table Names

PROC TSCSREG assigns a name to each table it creates. You can use these names to reference the table when you use the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in the following table.

Table 40.2 ODS Tables Produced in PROC TSCSREG

ODS Table Name	Description	Option
ODS Tables Created by the MODEL Statement		
ModelDescription	Model description	default
FitStatistics	Fit statistics	default
FixedEffectsTest	*F* test for no fixed effects	FIXONE, FIXTWO, RANONE, RANTWO
ParameterEstimates	Parameter estimates	default
CovB	Covariance of parameter estimates	COVB
CorrB	Correlations of parameter estimates	CORRB
VarianceComponents	Variance component estimates	FULLER, DASILVA, M=, RANONE, RANTWO
RandomEffectsTest	Hausman test for random effects	FULLER, DASILVA, M=, RANONE, RANTWO
AR1Estimates	First order autoregressive parameter estimates	PARKS, RHO
EstimatedPhiMatrix	Estimated phi matrix	PARKS
EstimatedAutocovariances	Estimates of autocovariances	DASILVA, M=
ODS Tables Created by the TEST Statement		
TestResults	Test results	

Examples: The TSCSREG Procedure

For examples of analysis of panel data, see Chapter 27, "The PANEL Procedure."

References

Greene, W. H. (1990). *Econometric Analysis*. New York: Macmillan.

Chapter 41
The UCM Procedure

Contents

Overview: UCM Procedure **2744**
Getting Started: UCM Procedure **2745**
A Seasonal Series with Linear Trend 2745
Syntax: UCM Procedure **2753**
Functional Summary 2753
PROC UCM Statement 2756
AUTOREG Statement 2759
BLOCKSEASON Statement 2760
BY Statement 2762
CYCLE Statement 2762
DEPLAG Statement 2763
ESTIMATE Statement 2764
FORECAST Statement 2767
ID Statement 2769
IRREGULAR Statement 2769
LEVEL Statement 2772
MODEL Statement 2774
NLOPTIONS Statement 2774
OUTLIER Statement 2774
PERFORMANCE Statement 2775
RANDOMREG Statement 2775
SEASON Statement 2776
SLOPE Statement 2778
SPLINEREG Statement 2779
SPLINESEASON Statement 2781
Details: UCM Procedure **2782**
An Introduction to Unobserved Component Models 2782
The UCMs as State Space Models 2788
Outlier Detection 2797
Missing Values 2797
Parameter Estimation 2797
Bootstrap Prediction Intervals (Experimental) 2799
Computational Issues 2799
Displayed Output 2800
Statistical Graphics 2801
ODS Table Names 2811

ODS Graph Names 2814
OUTFOR= Data Set 2818
OUTEST= Data Set 2819
Statistics of Fit 2820
Examples: UCM Procedure **2821**
Example 41.1: The Airline Series Revisited 2821
Example 41.2: Variable Star Data 2826
Example 41.3: Modeling Long Seasonal Patterns 2829
Example 41.4: Modeling Time-Varying Regression Effects 2833
Example 41.5: Trend Removal Using the Hodrick-Prescott Filter 2839
Example 41.6: Using Splines to Incorporate Nonlinear Effects 2841
Example 41.7: Detection of Level Shift 2845
Example 41.8: ARIMA Modeling 2849
References **2852**

Overview: UCM Procedure

The UCM procedure analyzes and forecasts equally spaced univariate time series data by using an unobserved components model (UCM). The UCMs are also called *structural models* in the time series literature. A UCM decomposes the response series into components such as trend, seasonals, cycles, and the regression effects due to predictor series. The components in the model are supposed to capture the salient features of the series that are useful in explaining and predicting its behavior. Harvey (1989) is a good reference for time series modeling that uses the UCMs. Harvey calls the components in a UCM the "stylized facts" about the series under consideration. Traditionally, the ARIMA models and, to some limited extent, the exponential smoothing models have been the main tools in the analysis of this type of time series data. It is fair to say that the UCMs capture the versatility of the ARIMA models while possessing the interpretability of the smoothing models. A thorough discussion of the correspondence between the ARIMA models and the UCMs, and the relative merits of UCM and ARIMA modeling, is given in Harvey (1989). The UCMs are also very similar to another set of models, called the *dynamic models*, that are popular in the Bayesian time series literature (West and Harrison 1999). In SAS/ETS, you can use PROC SSM for multivariate (and more general univariate) UCMs (see Chapter 34, "The SSM Procedure"), PROC ARIMA for ARIMA modeling (see Chapter 8, "The ARIMA Procedure"), PROC ESM for exponential smoothing modeling (see Chapter 15, "The ESM Procedure"), and the Time Series Forecasting System for a point-and-click interface to ARIMA and exponential smoothing modeling.

You can use the UCM procedure to fit a wide range of UCMs that can incorporate complex trend, seasonal, and cyclical patterns and can include multiple predictors. It provides a variety of diagnostic tools to assess the fitted model and to suggest the possible extensions or modifications. The components in the UCM provide a succinct description of the underlying mechanism governing the series. You can print, save, or plot the estimates of these component series. Along with the standard forecast and residual plots, the study of these component plots is an essential part of time series analysis using the UCMs. Once a suitable UCM is found for the series under consideration, it can be used for a variety of purposes. For example, it can be used for the following:

- forecasting the values of the response series and the component series in the model
- obtaining a model-based seasonal decomposition of the series
- obtaining a "denoised" version and interpolating the missing values of the response series in the historical period
- obtaining the full sample or "smoothed" estimates of the component series in the model

Getting Started: UCM Procedure

The analysis of time series using the UCMs involves recognizing the salient features present in the series and modeling them suitably. The UCM procedure provides a variety of models for estimating and forecasting the commonly observed features in time series. These models are discussed in detail later in the section "An Introduction to Unobserved Component Models" on page 2782. First the procedure is illustrated using an example.

A Seasonal Series with Linear Trend

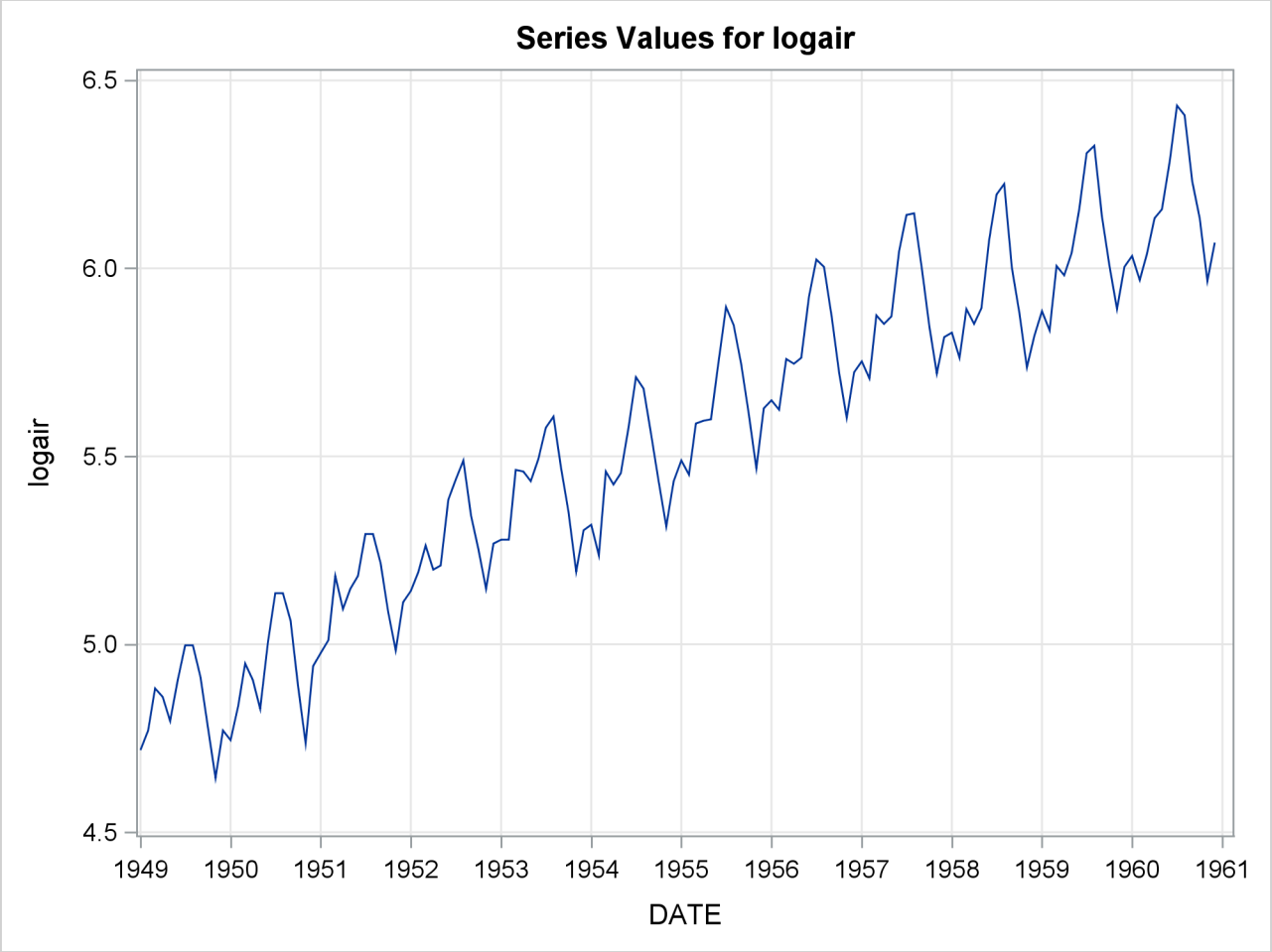
The airline passenger series, given as Series G in Box and Jenkins (1976), is often used in time series literature as an example of a nonstationary seasonal time series. This series is a monthly series consisting of the number of airline passengers who traveled during the years 1949 to 1960. Its main features are a steady rise in the number of passengers from year to year and the seasonal variation in the numbers during any given year. It also exhibits an increase in variability around the trend. A log transformation is used to stabilize this variability. The following DATA step prepares the log-transformed passenger series analyzed in this example:

```
data seriesG;
   set sashelp.air;
   logair = log( air );
run;
```

The following statements produce a time series plot of the series by using the TIMESERIES procedure (see Chapter 39, "The TIMESERIES Procedure"). The trend and seasonal features of the series are apparent in the plot in Figure 41.1.

```
proc timeseries data=seriesG plot=series;
   id date interval=month;
   var logair;
run;
```

Figure 41.1 Series Plot of Log-Transformed Airline Passenger Series



In this example this series is modeled using an unobserved component model called the basic structural model (BSM). The BSM models a time series as a sum of three stochastic components: a trend component μ_t, a seasonal component γ_t, and random error ϵ_t. Formally, a BSM for a response series y_t can be described as

$$y_t = \mu_t + \gamma_t + \epsilon_t$$

Each of the stochastic components in the model is modeled separately. The random error ϵ_t, also called the *irregular component*, is modeled simply as a sequence of independent, identically distributed (i.i.d.) zero-mean Gaussian random variables. The trend and the seasonal components can be modeled in a few different ways. The model for trend used here is called a *locally linear time trend*. This trend model can be written as follows:

$$\begin{aligned} \mu_t &= \mu_{t-1} + \beta_{t-1} + \eta_t, \quad & \eta_t &\sim i.i.d.\ N(0, \sigma_\eta^2) \\ \beta_t &= \beta_{t-1} + \xi_t, & \xi_t &\sim i.i.d.\ N(0, \sigma_\xi^2) \end{aligned}$$

These equations specify a trend where the level μ_t as well as the slope β_t is allowed to vary over time. This variation in slope and level is governed by the variances of the disturbance terms η_t and ξ_t in their respective equations. Some interesting special cases of this model arise when you manipulate these disturbance variances. For example, if the variance of ξ_t is zero, the slope will be constant (equal to β_0); if the variance of η_t is also zero, μ_t will be a deterministic trend given by the line $\mu_0 + \beta_0 t$. The seasonal model used in this example is called a trigonometric seasonal. The stochastic equations governing a trigonometric seasonal are explained later (see the section "Modeling Seasons" on page 2784). However, it is interesting to note here that this seasonal model reduces to the familiar regression with deterministic seasonal dummies if the variance of the disturbance terms in its equations is equal to zero. The following statements specify a BSM with these three components:

```
proc ucm data=seriesG;
   id date interval=month;
   model logair;
   irregular;
   level;
   slope;
   season length=12 type=trig print=smooth;
   estimate;
   forecast lead=24 print=decomp;
run;
```

The PROC UCM statement signifies the start of the UCM procedure, and the input data set, seriesG, containing the dependent series is specified there. The optional ID statement is used to specify a date, datetime, or time identification variable, date in this example, to label the observations. The INTERVAL=MONTH option in the ID statement indicates that the measurements were collected on a monthly basis. The model specification begins with the MODEL statement, where the response series is specified (logair in this case). After this the components in the model are specified using separate statements that enable you to control their individual properties. The irregular component ϵ_t is specified using the IRREGULAR statement and the trend component μ_t is specified using the LEVEL and SLOPE statements. The seasonal component γ_t is specified using the SEASON statement. The specifics of the seasonal characteristics such as the season length, its stochastic evolution properties, etc., are specified using the options in the SEASON statement. The seasonal component used in this example has a season length of 12, corresponding to the monthly seasonality, and is of the *trigonometric* type. Different types of seasonals are explained later (see the section "Modeling Seasons" on page 2784).

The parameters of this model are the variances of the disturbance terms in the evolution equations of μ_t, β_t, and γ_t and the variance of the irregular component ϵ_t. These parameters are estimated by maximizing the likelihood of the data. The ESTIMATE statement options can be used to specify the span of data used in parameter estimation and to display and save the results of the estimation step and the model diagnostics. You can use the estimated model to obtain the forecasts of the series as well as the components. The options in the individual component statements can be used to display the component forecasts—for example, PRINT=SMOOTH option in the SEASON statement requests the displaying of smoothed forecasts of the seasonal component γ_t. The series forecasts and forecasts of the sum of components can be requested using the FORECAST statement. The option PRINT=DECOMP in the FORECAST statement requests the printing of the smoothed trend μ_t and the trend plus seasonal component $(\mu_t + \gamma_t)$.

The parameter estimates for this model are displayed in Figure 41.2.

Figure 41.2 BSM for the Logair Series

The UCM Procedure

Final Estimates of the Free Parameters

Component	Parameter	Estimate	Approx Std Error	t Value	Approx Pr > |t|
Irregular	Error Variance	0.00023436	0.0001079	2.17	0.0298
Level	Error Variance	0.00029828	0.0001057	2.82	0.0048
Slope	Error Variance	8.47916E-13	6.2271E-10	0.00	0.9989
Season	Error Variance	0.00000356	1.32347E-6	2.69	0.0072

The estimates suggest that except for the slope component, the disturbance variances of all the components are significant—that is, all these components are *stochastic*. The slope component, however, appears to be deterministic because its error variance is quite insignificant. It might then be useful to check if the slope component can be dropped from the model—that is, if $\beta_0 = 0$. This can be checked by examining the significance analysis table of the components given in Figure 41.3.

Figure 41.3 Component Significance Analysis for the Logair Series

Significance Analysis of Components (Based on the Final State)

Component	DF	Chi-Square	Pr > ChiSq
Irregular	1	0.08	0.7747
Level	1	117867	<.0001
Slope	1	43.78	<.0001
Season	11	507.75	<.0001

This table provides the significance of the components in the model at the end of the estimation span. If a component is deterministic, this analysis is equivalent to checking whether the corresponding regression effect is significant. However, if a component is stochastic, then this analysis pertains only to the portion of the series near the end of the estimation span. In this example the slope appears quite significant and should be retained in the model, possibly as a deterministic component. Note that, on the basis of this table, the irregular component's contribution appears insignificant toward the end of the estimation span; however, since it is a stochastic component, it cannot be dropped from the model on the basis of this analysis alone. The slope component can be made deterministic by holding the value of its error variance fixed at zero. This is done by modifying the SLOPE statement as follows:

```
slope variance=0 noest;
```

After a tentative model is fit, its adequacy can be checked by examining different goodness-of-fit measures and other diagnostic tests and plots that are based on the model residuals. Once the model appears satisfactory, it can be used for forecasting. An interesting feature of the UCM procedure is that, apart from the series forecasts, you can request the forecasts of the individual components in the model. The plots of component forecasts can be useful in understanding their contributions to the series. The following statements illustrate some of these features:

```
proc ucm data=seriesG;
   id date interval = month;
   model logair;
   irregular;
   level plot=smooth;
   slope variance=0 noest;
   season length=12 type=trig
       plot=smooth;
   estimate;
   forecast lead=24 plot=decomp;
run;
```

The table given in Figure 41.4 shows the goodness-of-fit statistics that are computed by using the one-step-ahead prediction errors (see the section "Statistics of Fit" on page 2820). These measures indicate a good agreement between the model and the data. Additional diagnostic measures are also printed by default but are not shown here.

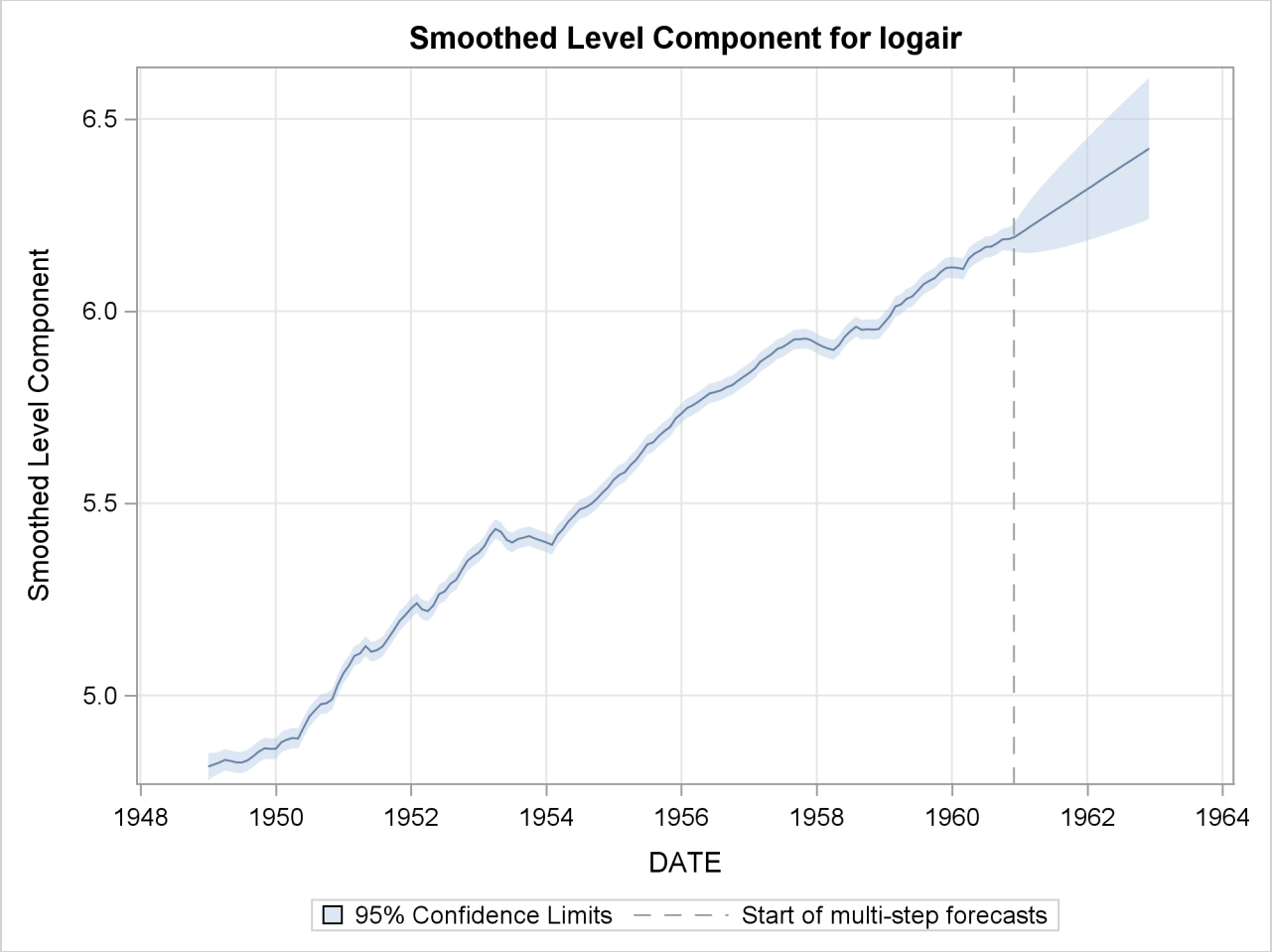
Figure 41.4 Fit Statistics for the Logair Series

The UCM Procedure

Fit Statistics Based on Residuals	
Mean Squared Error	0.00147
Root Mean Squared Error	0.03830
Mean Absolute Percentage Error	0.54132
Maximum Percent Error	2.19097
R-Square	0.99061
Adjusted R-Square	0.99046
Random Walk R-Square	0.87288
Amemiya's Adjusted R-Square	0.99017
Number of non-missing residuals used for computing the fit statistics = 131	

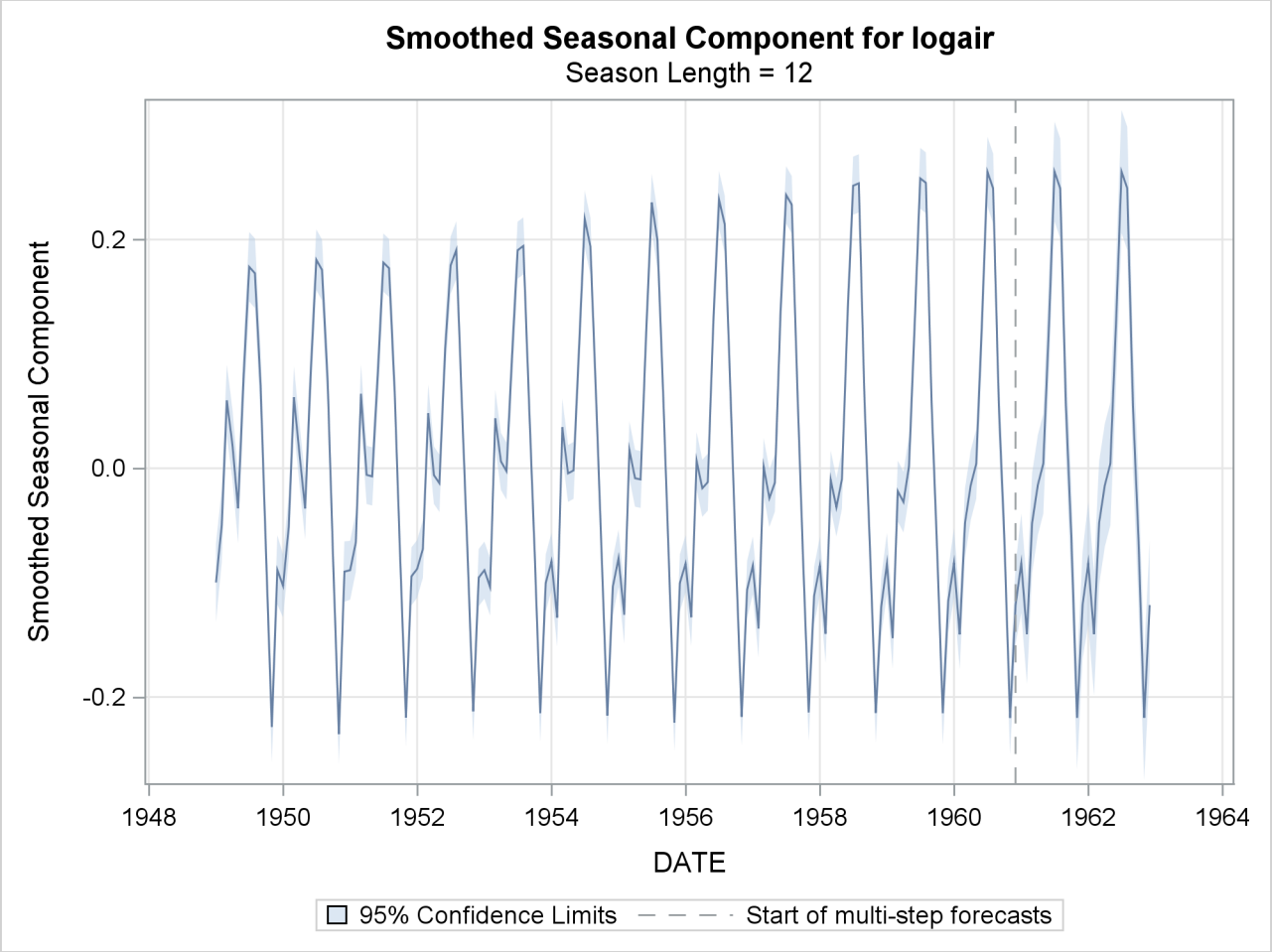
The first plot, shown in Figure 41.5, is produced by the PLOT=SMOOTH option in the LEVEL statement, it shows the smoothed level of the series.

Figure 41.5 Smoothed Trend in the Logair Series



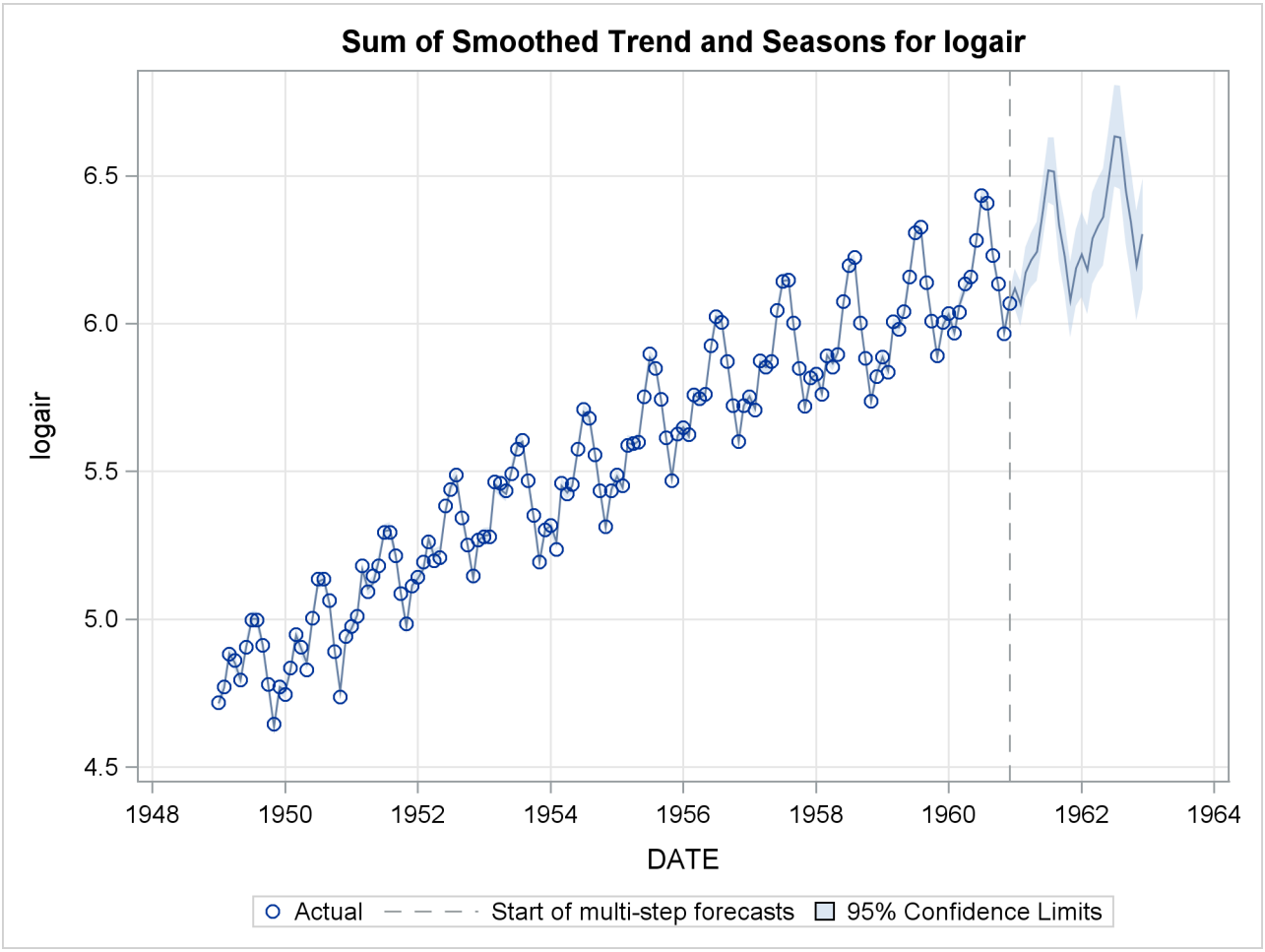
The second plot (Figure 41.6), produced by the PLOT=SMOOTH option in the SEASON statement, shows the smoothed seasonal component by itself.

Figure 41.6 Smoothed Seasonal in the Logair Series



The plot of the sum of the trend and seasonal component, produced by the PLOT=DECOMP option in the FORECAST statement, is shown in Figure 41.7. You can see that, at least visually, the model seems to fit the data well. In all these decomposition plots the component estimates are extrapolated for two years in the future based on the LEAD=24 option specified in the FORECAST statement.

Figure 41.7 Smoothed Trend plus Seasonal in the Logair Series



Syntax: UCM Procedure

The UCM procedure uses the following statements:

```
PROC UCM < options > ;
    AUTOREG < options > ;
    BLOCKSEASON options ;
    BY variables ;
    CYCLE < options > ;
    DEPLAG options ;
    ESTIMATE < options > ;
    FORECAST < options > ;
    ID variable options ;
    IRREGULAR < options > ;
    LEVEL < options > ;
    MODEL dependent variable < = regressors > ;
    NLOPTIONS options ;
    PERFORMANCE options ;
    OUTLIER options ;
    RANDOMREG regressors < / options > ;
    SEASON options ;
    SLOPE < options > ;
    SPLINEREG regressor < options > ;
    SPLINESEASON options ;
```

The PROC UCM and MODEL statements are required. In addition, the model must contain at least one component with nonzero disturbance variance.

Functional Summary

The statements and options controlling the UCM procedure are summarized in the following table. Most commonly needed scenarios are listed; see the individual statements for additional details. You can use the PRINT= and PLOT= options in the individual component statements for printing and plotting the corresponding component forecasts.

Table 41.1 Functional Summary

Description	Statement	Option
Data Set Options		
specify the input data set	PROC UCM	DATA=
write parameter estimates to an output data set	ESTIMATE	OUTEST=
write series and component forecasts to an output data set	FORECAST	OUTFOR=

Table 41.1 *continued*

Description	Statement	Option
Model Specification		
specify the dependent variable and simple predictors	MODEL	
specify predictors with time-varying coefficients	RANDOMREG	
specify a nonlinear predictor	SPLINEREG	
specify the irregular component	IRREGULAR	
specify the random walk trend	LEVEL	
specify the locally linear trend	LEVEL and SLOPE	
specify a cycle component	CYCLE	
specify a dummy seasonal component	SEASON	TYPE=DUMMY
specify a trigonometric seasonal component	SEASON	TYPE=TRIG
drop some harmonics from a trigonometric seasonal component	SEASON	DROPH=
specify a list of harmonics to keep in a trigonometric seasonal component	SEASON	KEEPH=
specify a spline-season component	SPLINESEASON	
specify a block-season component	BLOCKSEASON	
specify an autoreg component	AUTOREG	
specify the lags of the dependent variable	DEPLAG	
Controlling the Likelihood Optimization Process		
request optimization of the profile likelihood	ESTIMATE	PROFILE
request optimization of the usual likelihood	ESTIMATE	NOPROFILE
specify the optimization technique	NLOPTIONS	TECH=
limit the number of iterations	NLOPTIONS	MAXITER=
Outlier Detection		
turn on the search for additive outliers		Default
turn on the search for level shifts	LEVEL	CHECKBREAK
specify the significance level for outlier tests	OUTLIER	ALPHA=
limit the number of outliers	OUTLIER	MAXNUM=
limit the number of outliers to a percentage of the series length	OUTLIER	MAXPCT=
Controlling the Series Span		
exclude some initial observations from analysis during the parameter estimation	ESTIMATE	SKIPFIRST=
exclude some observations at the end from analysis during the parameter estimation	ESTIMATE	BACK=
exclude some initial observations from analysis during forecasting	FORECAST	SKIPFIRST=
exclude some observations at the end from analysis during forecasting	FORECAST	BACK=

Table 41.1 *continued*

Description	Statement	Option
Graphical Residual Analysis		
get a panel of plots consisting of residual autocorrelation plots and residual normality plots	ESTIMATE	PLOT=PANEL
get the residual CUSUM plot	ESTIMATE	PLOT=CUSUM
get the residual cumulative sum of squares plot	ESTIMATE	PLOT=CUSUMSQ
get a plot of *p*-values for the portmanteau white noise test	ESTIMATE	PLOT=WN
get a time series plot of residuals with overlaid LOESS smoother	ESTIMATE	PLOT=LOESS
Series Decomposition and Forecasting		
specify the number of periods to forecast in the future	FORECAST	LEAD=
specify the significance level of the forecast confidence interval	FORECAST	ALPHA=
request printing of smoothed series decomposition	FORECAST	PRINT=DECOMP
request printing of one-step-ahead and multistep-ahead forecasts	FORECAST	PRINT=FORECASTS
request plotting of smoothed series decomposition	FORECAST	PLOT=DECOMP
request plotting of one-step-ahead and multistep-ahead forecasts	FORECAST	PLOT=FORECASTS
request bootstrap standard errors	FORECAST	BOOTSTRAP
BY Groups		
specify BY-group processing	BY	
Global Printing and Plotting Options		
turn off all the printing for the procedure	PROC UCM	NOPRINT
turn on all the printing options for the procedure	PROC UCM	PRINTALL
turn off all the plotting for the procedure	PROC UCM	PLOTS=NONE
turn on all the plotting options for the procedure	PROC UCM	PLOTS=ALL
turn on a variety of plotting options for the procedure	PROC UCM	PLOTS=
ID		
specify a variable that provides the time index for the series values	ID	

PROC UCM Statement

PROC UCM < *options* > **;**

The PROC UCM statement is required. The following options can be used in the PROC UCM statement:

DATA=*SAS-data-set*
specifies the name of the SAS data set containing the time series. If the DATA= option is not specified in the PROC UCM statement, the most recently created SAS data set is used.

NOPRINT
turns off all the printing for the procedure. The subsequent print options in the procedure are ignored.

PLOTS< (*global-plot-options***) > < = ***plot-request* < *(options)* > **>**

PLOTS< (*global-plot-options***) > < = (***plot-request* < *(options)* > **< ...** *plot-request* < *(options)* > **>) >**
controls the plots produced with ODS Graphics. When you specify only one plot request, you can omit the parentheses around the plot request.

Here are some examples:

```
plots=none
plots=all
plots=residuals(acf loess)
plots(noclm)=(smooth(decomp) residual(panel loess))
```

For general information about ODS Graphics, see Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*).

```
proc ucm;
   model y = x;
   irregular;
   level;
run;

proc ucm plots=all;
   model y = x;
   irregular;
   level;
run;
```

The first PROC UCM step does not specify the PLOTS= option, so the default plot that displays the series forecasts in the forecast region is produced. The PLOTS=ALL option in the second PROC UCM step produces all the plots that are appropriate for the specified model.

In addition to the PLOTS= option in the PROC UCM statement, you can request plots by using the PLOT= option in other statements of the UCM procedure. This way of requesting plots provides finer control over the plot production. If you do not specify any specific plot request, then PROC UCM produces the plot of series forecasts in the forecast horizon by default.

Global Plot Options:
The *global-plot-options* apply to all relevant plots generated by the UCM procedure. The following *global-plot-option* is supported:

NOCLM
suppresses the confidence limits in all the component and forecast plots.

Specific Plot Options:
The following list describes the specific plots and their options:

ALL
produces all plots appropriate for the particular analysis.

NONE
suppresses all plots.

FILTER (< *filter-plot-options* **>)**
produces time series plots of the filtered component estimates. The following *filter-plot-options* are available:

ALL
produces all the filtered component estimate plots appropriate for the particular analysis.

LEVEL
produces a time series plot of the filtered level component estimate, provided the model contains the level component.

SLOPE
produces a time series plot of the filtered slope component estimate, provided the model contains the slope component.

CYCLE
produces time series plots of the filtered cycle component estimates for all cycle components in the model, if there are any.

SEASON
produces time series plots of the filtered season component estimates for all seasonal components in the model, if there are any.

DECOMP
produces time series plots of the filtered estimates of the series decomposition.

RESIDUAL (< *residual-plot-options* **>)**
produces the residuals plots. The following *residual-plot-options* are available:

ALL
produces all the residual diagnostics plots appropriate for the particular analysis.

ACF
produces the autocorrelation plot of residuals.

CUSUM
produces the plot of cumulative residuals against time.

CUSUMSQ
produces the plot of cumulative squared residuals against time.

HISTOGRAM
produces the histogram of residuals.

LOESS
produces a scatter plot of residuals against time, which has an overlaid loess-fit.

PACF
produces the partial-autocorrelation plot of residuals.

PANEL
produces a summary panel of the residual diagnostics consisting of the following:

- histogram of residuals
- normal quantile plot of residuals
- the residual-autocorrelation-plot
- the residual-partial-autocorrelation-plot

QQ
produces a normal quantile plot of residuals.

RESIDUAL
produces a needle plot of residuals against time.

WN
produces the plot of Ljung-Box white-noise test *p*-values at different lags (in log scale).

SMOOTH (< *smooth-plot-options* >)
produces time series plots of the smoothed component estimates. The following *smooth-plot-options* are available:

ALL
produces all the smoothed component estimate plots appropriate for the particular analysis.

LEVEL
produces time series plot of the smoothed level component estimate, provided the model contains the level component.

SLOPE
produces time series plot of the smoothed slope component estimate, provided the model contains the slope component.

CYCLE
produces time series plots of the smoothed cycle component estimates for all cycle components in the model, if there are any.

SEASON
produces time series plots of the smoothed season component estimates for all season components in the model, if there are any.

DECOMP
produces time series plots of the smoothed estimates of the series decomposition.

PRINTALL
turns on all the printing options for the procedure. The subsequent NOPRINT options in the procedure are ignored.

AUTOREG Statement

AUTOREG < *options* > **;**

The AUTOREG statement specifies an autoregressive component in the model. An autoregressive component is a special case of cycle that corresponds to the frequency of zero or π. It is modeled separately for easier interpretation. A stochastic equation for an autoregressive component r_t can be written as follows:

$$r_t = \rho r_{t-1} + v_t, \quad v_t \sim i.i.d.\ N(0, \sigma_v^2)$$

The damping factor ρ can take any value in the interval (–1, 1), including –1 but excluding 1. If $\rho = 1$, the autoregressive component cannot be distinguished from the random walk level component. If $\rho = -1$, the autoregressive component corresponds to a seasonal component with a season length of 2, or a nonstationary cycle with period 2. If $|\rho| < 1$, then the autoregressive component is stationary. The following example illustrates the AUTOREG statement. This statement includes an autoregressive component in the model. The damping factor ρ and the disturbance variance σ_v^2 are estimated from the data.

```
autoreg;
```

NOEST=RHO
NOEST=VARIANCE
NOEST=(RHO VARIANCE)
fixes the values of ρ and σ_v^2 to those specified in the RHO= and VARIANCE= options.

PLOT=FILTER
PLOT=SMOOTH
PLOT=(< FILTER > < SMOOTH >)
requests plotting of the filtered or smoothed estimate of the autoreg component.

PRINT=FILTER

PRINT=SMOOTH

PRINT=(< FILTER > < SMOOTH >)

requests printing of the filtered or smoothed estimate of the autoreg component.

RHO=*value*

specifies an initial value for the damping factor ρ during the parameter estimation process. The value of ρ must be in the interval (–1, 1), including –1 but excluding 1.

VARIANCE=*value*

specifies an initial value for the disturbance variance σ_v^2 during the parameter estimation process. Any nonnegative value, including zero, is an acceptable starting value.

BLOCKSEASON Statement

BLOCKSEASON *NBLOCKS = integer BLOCKSIZE = integer < options >* **;**

The BLOCKSEASON or BLOCKSEASONAL statement is used to specify a seasonal component γ_t that has a special block structure. The seasonal γ_t is called a *block seasonal* of block size m and number of blocks k if its season length, s, can be factored as $s = m * k$ and its seasonal effects have a block form—that is, the first m seasonal effects are all equal to some number τ_1, the next m effects are all equal to some number τ_2, and so on.

This type of seasonal structure can be appropriate in some cases; for example, consider a series that is recorded on an hourly basis. Further assume that, in this particular case, the hour-of-the-day effect and the day-of-the-week effect are additive. In this situation the hour-of-the-week seasonality, having a season length of 168, can be modeled as a sum of two components. The hour-of-the-day effect is modeled using a simple seasonal of season length 24, while the day-of-the-week is modeled as a block seasonal component that has the days of the week as blocks. This day-of-the-week block seasonal component has seven blocks, each of size 24.

A block seasonal specification requires, at the minimum, the block size m and the number of blocks in the seasonal k. These are specified using the BLOCKSIZE= and NBLOCKS= option, respectively. In addition, you might need to specify the position of the first observation of the series by using the OFFSET= option if it is not at the beginning of one of the blocks. In the example just considered, this corresponds to a situation where the first series measurement is not at the start of the day. Suppose that the first measurement of the series corresponds to the hour between 6:00 and 7:00 a.m., which is the seventh hour within that day or at the seventh position within that block. This is specified as OFFSET=7.

The other options in this statement are very similar to the options in the SEASON statement; for example, a block seasonal can also be of one of the two types, DUMMY and TRIG. There can be more than one block seasonal component in the model, each specified using a separate BLOCKSEASON statement. No two block seasonals in the model can have the same NBLOCKS= and BLOCKSIZE= specifications. The following example illustrates the use of the BLOCKSEASON statement to specify the additive, hour-of-the-week seasonal model:

```
season length=24 type=trig;
blockseason nblocks=7 blocksize=24;
```

BLOCKSIZE=*integer*

specifies the block size, *m*. This is a required option in this statement. The block size can be any integer larger than or equal to two. Typical examples of block sizes are 24, corresponding to the hours of the day when a day is being used as a block in hourly data, or 60, corresponding to the minutes in an hour when an hour is being used as a block in data recorded by minutes, etc.

NBLOCKS=*integer*

specifies the number of blocks, *k*. This is a required option in this statement. The number of blocks can be any integer greater than or equal to two.

NOEST

fixes the value of the disturbance variance parameter to the value specified in the VARIANCE= option.

OFFSET=*integer*

specifies the position of the first measurement within the block, if the first measurement is not at the start of a block. The OFFSET= value must be between one and the block size. The default value is one. The first measurement refers to the start of the estimation span and the forecast span. If these spans differ, their starting measurements must be separated by an integer multiple of the block size.

PLOT=FILTER

PLOT=SMOOTH

PLOT=F_ANNUAL

PLOT=S_ANNUAL

PLOT=(< plot request > . . . < plot request >)

requests plots of the season component. When you specify only one plot request, you can omit the parentheses around the plot request. You can use the FILTER and SMOOTH options to plot the filtered and smoothed estimates of the season component γ_t. You can use the F_ANNUAL and S_ANNUAL options to get the plots of "annual" variation in the filtered and smoothed estimates of γ_t. The annual plots are useful to see the change in the contribution of a particular month over the span of years. Here "month" and "year" are generic terms that change appropriately with the interval type being used to label the observations and the season length. For example, for monthly data with a season length of 12, the usual meaning applies, while for daily data with a season length of 7, the days of the week serve as months and the weeks serve as years. The first period in each block is plotted over the years.

PRINT=FILTER

PRINT=SMOOTH

PRINT=(< FILTER > < SMOOTH >)

requests the printing of the filtered or smoothed estimate of the block seasonal component γ_t.

TYPE=DUMMY | TRIG

specifies the type of the block seasonal component. The default type is DUMMY.

VARIANCE=*value*

specifies an initial value for the disturbance variance, σ_ω^2, in the γ_t equation at the start of the parameter estimation process. Any nonnegative value, including zero, is an acceptable starting value.

BY Statement

BY *variables* **;**

A BY statement can be used in the UCM procedure to process a data set in groups of observations defined by the BY variables. The model specified using the MODEL and other component statements is applied to all the groups defined by the BY variables. When a BY statement appears, the procedure expects the input data set to be sorted in order of the BY variables. The variables are one or more variables in the input data set.

CYCLE Statement

CYCLE < *options* > **;**

The CYCLE statement is used to specify a cycle component, ψ_t, in the model. The stochastic equation governing a cycle component of period p and damping factor ρ is as follows

$$\begin{bmatrix} \psi_t \\ \psi_t^* \end{bmatrix} = \rho \begin{bmatrix} \cos\lambda & \sin\lambda \\ -\sin\lambda & \cos\lambda \end{bmatrix} \begin{bmatrix} \psi_{t-1} \\ \psi_{t-1}^* \end{bmatrix} + \begin{bmatrix} \nu_t \\ \nu_t^* \end{bmatrix}$$

where ν_t and ν_t^* are independent, zero-mean, Gaussian disturbances with variance σ_ν^2 and $\lambda = 2 * \pi / p$ is the angular frequency of the cycle. Any p strictly greater than two is an admissible value for the period, and the damping factor ρ can be any value in the interval (0, 1), including one but excluding zero. The cycles with frequency zero and π, which correspond to the periods equal to infinity and two, respectively, can be specified using the AUTOREG statement. The values of ρ less than one give rise to a stationary cycle, while $\rho = 1$ gives rise to a nonstationary cycle. As a default, values of ρ, p, and σ_ν^2 are estimated from the data. However, if necessary, you can fix the values of some or all of these parameters.

There can be multiple cycles in a model, each specified using a separate CYCLE statement. The examples that follow illustrate the use of the CYCLE statement.

The following statements request including two cycles in the model. The parameters of each of these cycles are estimated from the data.

```
cycle;
cycle;
```

The following statement requests inclusion of a nonstationary cycle in the model. The cycle period p and the disturbance variance σ_ν^2 are estimated from the data.

```
cycle rho=1 noest=rho;
```

In the following statement a nonstationary cycle with a fixed period of 12 is specified. Moreover, a starting value is supplied for σ_ν^2.

```
cycle period=12 rho=1 variance=4 noest=(rho period);
```

NOEST=PERIOD

NOEST=RHO

NOEST=VARIANCE

NOEST=(< RHO > < PERIOD > < VARIANCE >)
fixes the values of the component parameters to those specified in the RHO=, PERIOD=, and VARIANCE= options. This option enables you to fix any combination of parameter values.

PERIOD=*value*
specifies an initial value for the cycle period during the parameter estimation process. Period value must be strictly greater than 2.

PLOT=FILTER

PLOT=SMOOTH

PLOT=(< FILTER > < SMOOTH >)
requests plotting of the filtered or smoothed estimate of the cycle component.

PRINT=FILTER

PRINT=SMOOTH

PRINT=(< FILTER > < SMOOTH >)
requests the printing of a filtered or smoothed estimate of the cycle component ψ_t.

RHO=*value*
specifies an initial value for the damping factor in this component during the parameter estimation process. Any value in the interval (0, 1), including one but excluding zero, is an acceptable initial value for the damping factor.

VARIANCE=*value*
specifies an initial value for the disturbance variance parameter, σ_v^2, to be used during the parameter estimation process. Any nonnegative value, including zero, is an acceptable starting value.

DEPLAG Statement

DEPLAG *LAGS = order* < *PHI = value . . .* > < *NOEST* > **;**

The DEPLAG statement is used to specify the lags of the dependent variable to be included as predictors in the model. The following examples illustrate the use of DEPLAG statement.

If the dependent series is denoted by y_t, the following statement specifies the inclusion of $\phi_1 y_{t-1} + \phi_2 y_{t-2}$ in the model. The parameters ϕ_1 and ϕ_2 are estimated from the data.

```
deplag lags=2;
```

The following statement requests including $\phi_1 y_{t-1} + \phi_2 y_{t-4} - \phi_1\phi_2 y_{t-5}$ in the model. The values of ϕ_1 and ϕ_2 are fixed at 0.8 and –1.2.

```
deplag lags=(1)(4) phi=0.8 -1.2 noest;
```

The dependent lag parameters are not constrained to lie in any particular region. In particular, this implies that a UCM that contains only an irregular component and dependent lags, resulting in a traditional autoregressive model, is not constrained to be a stationary model. In the DEPLAG statement, if an initial value is supplied for any one of the parameters, the initial values must also be supplied for all other parameters.

LAGS=*order*

LAGS=(*lag, . . . , lag* **) . . . (***lag, . . . , lag* **)**

is a required option in this statement. LAGS=($l_1, l_2, \ldots, l_k$) defines a model with specified lags of the dependent variable included as predictors. LAGS=*order* is equivalent to LAGS=(1, 2, . . . , *order*).

A concatenation of parenthesized lists specifies a factored model. For example, LAGS=(1)(12) specifies that the lag values, 1, 12, and 13, corresponding to the following polynomial in the backward shift operator, be included in the model

$$(1 - \phi_{1,1}B)(1 - \phi_{2,1}B^{12})$$

Note that, in this case, the coefficient of the thirteenth lag is constrained to be the product of the coefficients of the first and twelfth lags.

NOEST

fixes the values of the parameters to those specified in PHI= option.

PHI=*value* **. . .**

lists starting values for the coefficients of the lagged dependent variable. The order of the values listed corresponds with the order of the lags specified in the LAGS= option.

ESTIMATE Statement

ESTIMATE < *options* > **;**

The ESTIMATE statement is an optional statement used to control the overall model-fitting environment. Using this statement, you can control the span of observations used to fit the model by using the SKIPFIRST= and BACK= options. This can be useful in model diagnostics. You can request a variety of goodness-of-fit statistics and other model diagnostic information including different residual diagnostic plots. Note that the ESTIMATE statement is not used to control the nonlinear optimization process itself. That is done using the NLOPTIONS statement, where you can control the number of iterations, choose between the different optimization techniques, and so on. You can save the estimated parameters and other related information in a data set by using the OUTEST= option. You can request the optimization of the profile likelihood, the likelihood obtained by concentrating out a disturbance variance, for parameter estimation by using the PROFILE option. The following example illustrates the use of this statement:

```
estimate skipfirst=12 back=24;
```

This statement requests that the initial 12 measurements and the last 24 measurements be excluded during the model-fitting process. The actual observation span used to fit the model is decided as follows: Suppose that n_0 and n_1 are the observation numbers of the first and the last nonmissing values of the response variable, respectively. As a result of SKIPFIRST=12 and BACK=24, the measurements between observation numbers $n_0 + 12$ and $n_1 - 24$ form the estimation span. Of course, the model fitting might not take place if there are insufficient data in the resulting span. The model fitting does not take place if there are regressors in the model that have missing values in the estimation span.

BACK=*integer*

SKIPLAST=*integer*

indicates that some ending part of the data needs to be ignored during the parameter estimation. This can be useful when you want to study the forecasting performance of a model on the observed data. BACK=10 results in skipping the last 10 measurements of the response series during the parameter estimation. The default is BACK=0.

EXTRADIFFUSE=*k*

enables continuation of the diffuse filtering iterations for k additional iterations beyond the first instance where the initialization of the diffuse state would have otherwise taken place. If the specified k is larger than the sample size, the diffuse iterations continue until the end of the sample. Note that one-step-ahead residuals are produced only after the diffuse state is initialized. Delaying the initialization leads to a reduction in the number of one-step-ahead residuals available for computing the residual diagnostic measures. This option is useful when you want to ignore the first few one-step-ahead residuals that often have large variance.

NOPROFILE

requests that the usual likelihood be optimized for parameter estimation. For more information, see the section "Parameter Estimation by Profile Likelihood Optimization" on page 2798.

OUTEST=*SAS-data-set*

specifies an output data set for the estimated parameters.

In the ESTIMATE statement, the PLOT= option is used to obtain different residual diagnostic plots. The different possibilities are as follows:

PLOT=ACF

PLOT=MODEL

PLOT=LOESS

PLOT=HISTOGRAM

PLOT=PACF

PLOT=PANEL

PLOT=QQ

PLOT=RESIDUAL

PLOT=WN

PLOT=(< plot request > . . . < plot request >)

requests different residual diagnostic plots. The different options are as follows:

ACF

produces the residual-autocorrelation plot.

CUSUM

produces the plot of cumulative residuals against time.

CUSUMSQ

produces the plot of cumulative squared residuals against time.

MODEL
produces the plot of one-step-ahead forecasts in the estimation span.

HISTOGRAM
produces the histogram of residuals.

LOESS
produces a scatter plot of residuals against time, which has an overlaid loess-fit.

PACF
produces the residual-partial-autocorrelation plot.

PANEL
produces a summary panel of the residual diagnostics consisting of

- histogram of residuals
- normal quantile plot of residuals
- the residual-autocorrelation-plot
- the residual-partial-autocorrelation-plot

QQ
produces a normal quantile plot of residuals.

RESIDUAL
produces a needle plot of residuals against time.

WN
produces a plot of *p*-values, in log-scale, at different lags for the Ljung-Box portmanteau white noise test statistics.

PRINT=NONE
suppresses all the printed output related to the model fitting, such as the parameter estimates, the goodness-of-fit statistics, and so on.

PROFILE
requests that the profile likelihood, obtained by concentrating out one of the disturbance variances from the likelihood, be optimized for parameter estimation. By default, the profile likelihood is not optimized if any of the disturbance variance parameters is held fixed to a nonzero value. For more information see the section "Parameter Estimation by Profile Likelihood Optimization" on page 2798.

SKIPFIRST=*integer*
indicates that some early part of the data needs to be ignored during the parameter estimation. This can be useful if there is a reason to believe that the model being estimated is not appropriate for this portion of the data. SKIPFIRST=10 results in skipping the first 10 measurements of the response series during the parameter estimation. The default is SKIPFIRST=0.

FORECAST Statement

FORECAST < *options* > **;**

The FORECAST statement is an optional statement that is used to specify the overall forecasting environment for the specified model. It can be used to specify the span of observations, the historical period, to use to compute the forecasts of the future observations. This is done using the SKIPFIRST= and BACK= options. The number of periods to forecast beyond the historical period, and the significance level of the forecast confidence interval, is specified using the LEAD= and ALPHA= options. You can request one-step-ahead series and component forecasts by using the PRINT= option. You can save the series forecasts, and the model-based decomposition of the series, in a data set by using the OUTFOR= option. You can use the BOOTSTRAP option to request the computation of bootstrap prediction standard errors and the associated confidence intervals. The following example illustrates the use of this statement:

```
forecast skipfirst=12 back=24 lead=30;
```

This statement requests that the initial 12 and the last 24 response values be excluded during the forecast computations. The forecast horizon, specified using the LEAD= option, is 30 periods; that is, multistep forecasting begins at the end of the historical period and continues for 30 periods. The actual observation span used to compute the multistep forecasting is decided as follows: Suppose that n_0 and n_1 are the observation numbers of the first and the last nonmissing values of the response variable, respectively. As a result of SKIPFIRST=12 and BACK=24, the historical period, or the forecast span, begins at $n_0 + 12$ and ends at $n_1 - 24$. Multistep forecasts are produced for the next 30 periods—that is, for the observation numbers $n_1 - 23$ to $n_1 + 6$. Of course, the forecast computations can fail if the model has regressor variables that have missing values in the forecast span. If the regressors contain missing values in the forecast horizon—that is, between the observations $n_1 - 23$ and $n_1 + 6$—the forecast horizon is reduced accordingly.

ALPHA=*value*

specifies the significance level of the forecast confidence intervals; for example, ALPHA=0.05, which is the default, results in a 95% confidence interval.

BACK=*integer*

SKIPLAST=*integer*

specifies the holdout sample for the evaluation of the forecasting performance of the model. For example, BACK=10 results in treating the last 10 observed values of the response series as unobserved. A post-sample-prediction-analysis table is produced for comparing the predicted values with the actual values in the holdout period. The default is BACK=0.

BOOTSTRAP(NREP=*integer* **<SEED=***integer* **>)** (Experimental)

enables the computation of bootstrap prediction standard errors based on the specified number of replications (NREP). The value of NREP must be at least 2. Optionally, you can specify the random number seed that is associated with the first replication by using the SEED= option. The seeds for the subsequent replications are assigned sequentially. The default seed value that is associated with the first replication is 123. The BOOTSTRAP option has no effect if the number of parameters to be estimated is zero (that is, all the model parameters are known). Note that this option is computationally expensive. The computational cost of NREP replications is comparable to the cost of estimating parameters NREP times.

EXTRADIFFUSE=*k*

enables continuation of the diffuse filtering iterations for *k* additional iterations beyond the first instance where the initialization of the diffuse state would have otherwise taken place. If the specified *k* is larger than the sample size, the diffuse iterations continue until the end of the sample. Note that one-step-ahead forecasts are produced only after the diffuse state is initialized. Delaying the initialization leads to reduction in the number of one-step-ahead forecasts. This option is useful when you want to ignore the first few one-step-ahead forecasts that often have large variance.

LEAD=*integer*

specifies the number of periods to forecast beyond the historical period defined by the SKIPFIRST= and BACK= options; for example, LEAD=10 results in the forecasting of 10 future values of the response series. The default is LEAD=12.

OUTFOR=*SAS-data-set*

specifies an output data set for the forecasts. The output data set contains the ID variable (if specified), the response and predictor series, the one-step-ahead and out-of-sample response series forecasts, the forecast confidence intervals, the smoothed values of the response series, and the smoothed forecasts produced as a result of the model-based decomposition of the series.

PLOT=DECOMP

PLOT=DECOMPVAR

PLOT=FDECOMP

PLOT=FDECOMPVAR

PLOT=FORECASTS

PLOT=TREND

PLOT=(< plot request > . . . < plot request >)

requests forecast and model decomposition plots. The FORECASTS option provides the plot of the series forecasts, the TREND and DECOMP options provide the plots of the smoothed trend and other decompositions, the DECOMPVAR option can be used to plot the variance of these components, and the FDECOMP and FDECOMPVAR options provide the same plots for the filtered decomposition estimates and their variances.

PRINT=DECOMP

PRINT=FDECOMP

PRINT=FORECASTS

PRINT=NONE

PRINT=(< print request > . . . < print request >)

controls the printing of the series forecasts and the printing of smoothed model decomposition estimates. By default, the series forecasts are printed only for the forecast horizon specified by the LEAD= option; that is, the one-step-ahead predicted values are not printed. You can request forecasts for the entire forecast span by specifying the PRINT=FORECASTS option. Using PRINT=DECOMP, you can get smoothed estimates of the following effects: trend, trend plus regression, trend plus regression plus cycle, and sum of all components except the irregular. If some of these effects are absent in the model, then they are ignored. Similarly you can get filtered estimates of these effects by using PRINT=FDECOMP. You can use PRINT=NONE to suppress the printing of all the forecast output.

SKIPFIRST=*integer*
indicates that some early part of the data needs to be ignored during the forecasting calculations. This can be useful if there is a reason to believe that the model being used for forecasting is not appropriate for this portion of the data. SKIPFIRST=10 results in skipping the first 10 measurements of the response series during the forecast calculations. The default is SKIPFIRST=0.

ID Statement

ID *variable INTERVAL*=value < *ALIGN*=value > **;**

The ID statement names a numeric variable that identifies observations in the input and output data sets. The ID variable's values are assumed to be SAS date, time, or datetime values. In addition, the ID statement specifies the frequency associated with the time series. The ID statement options also specify how the observations are aligned to form the time series. If the ID statement is specified, the INTERVAL= option must also be specified. If the ID statement is not specified, the observation number, with respect to the BY group, is used as the time ID. The values of the ID variable are extrapolated for the forecast observations based on the values of the INTERVAL= option.

ALIGN=*value*
controls the alignment of SAS dates used to identify output observations. The ALIGN= option has the following possible values: BEGINNING | BEG | B, MIDDLE | MID | M, and ENDING | END | E. The default is BEGINNING. The ALIGN= option is used to align the ID variable with the beginning, middle, or end of the time ID interval specified by the INTERVAL= option.

INTERVAL=*value*
specifies the time interval between observations. This option is required in the ID statement. INTERVAL=*value* is used in conjunction with the ID variable to check that the input data are in order and have no gaps. The INTERVAL= option is also used to extrapolate the ID values past the end of the input data. For a complete discussion of the intervals supported, see Chapter 5, "Date Intervals, Formats, and Functions."

IRREGULAR Statement

IRREGULAR < *options* > **;**

The IRREGULAR statement includes an irregular component in the model. There can be at most one IRREGULAR statement in the model specification. The irregular component corresponds to the overall random error ϵ_t in the model. By default the irregular component is modeled as white noise—that is, as a sequence of independent, identically distributed, zero-mean, Gaussian random variables. However, you can also model it as an autoregressive moving average (ARMA) process. The options for specifying an ARMA model for the irregular component are given in a separate subsection: "ARMA Specification" on page 2770.

The options in this statement enable you to specify the model for the irregular component and to output its estimates. Two examples of the IRREGULAR statement are given next. In the first example the statement is in its simplest form, resulting in the inclusion of an irregular component that is white noise with unknown variance:

```
irregular;
```

The following statement provides a starting value for the white noise variance σ_ϵ^2 to be used in the nonlinear parameter estimation process. It also requests the printing of smoothed estimates of ϵ_t. The smoothed irregulars are useful in model diagnostics.

```
irregular variance=4 print=smooth;
```

NOEST
fixes the value of σ_ϵ^2 to the value specified in the VARIANCE= option. Also see the NOEST= option in the subsection "ARMA Specification" on page 2770.

PLOT=FILTER

PLOT=SMOOTH

PLOT=(< FILTER > < SMOOTH >)
requests plotting of the filtered or smoothed estimate of the irregular component.

PRINT=FILTER

PRINT=SMOOTH

PRINT=(< FILTER > < SMOOTH >)
requests printing of the filtered or smoothed estimate of the irregular component.

VARIANCE=*value*
specifies an initial value for σ_ϵ^2 during the parameter estimation process. Any nonnegative value, including zero, is an acceptable starting value.

ARMA Specification

This section details the options for specifying an ARMA model for the irregular component. The specification of ARMA models requires some notation, which is explained first.

Let B denote the backshift operator—that is, for any sequence ϵ_t, $B\epsilon_t = \epsilon_{t-1}$. The higher powers of B represent larger shifts (for example, $B^3\epsilon_t = \epsilon_{t-3}$). A random sequence ϵ_t follows a zero-mean ARMA$(p,q)\times(P,Q)_s$ model with nonseasonal autoregressive order p, seasonal autoregressive order P, nonseasonal moving average order q, and seasonal moving average order Q, if it satisfies the following difference equation specified in terms of the polynomials in the backshift operator where a_t is a white noise sequence and s is the season length:

$$\phi(B)\Phi(B^s)\epsilon_t = \theta(B)\Theta(B^s)a_t$$

The polynomials ϕ, Φ, θ, and Θ are of orders p, P, q, and Q, respectively, which can be any nonnegative integers. The season length s must be a positive integer. For example, ϵ_t satisfies an ARMA(1,1) model (that is, $p = 1, q = 1$, $P = 0$, and $Q = 0$) if

$$\epsilon_t = \phi_1\epsilon_{t-1} + a_t - \theta_1 a_{t-1}$$

for some coefficients ϕ_1 and θ_1 and a white noise sequence a_t. Similarly ϵ_t satisfies an ARMA$(1,1)\times(1,1)_{12}$ model if

$$\epsilon_t = \phi_1\epsilon_{t-1} + \Phi_1\epsilon_{t-12} - \phi_1\Phi_1\epsilon_{t-13} + a_t - \theta_1 a_{t-1} - \Theta_1 a_{t-12} + \theta_1\Theta_1 a_{t-13}$$

for some coefficients ϕ_1, Φ_1, θ_1, and Θ_1 and a white noise sequence a_t. The ARMA process is stationary and invertible if the defining polynomials ϕ, Φ, θ, and Θ have all their roots outside the unit circle—that is, their absolute values are strictly larger than 1.0. It is assumed that the ARMA model specified for the irregular component is stationary and invertible—that is, the coefficients of the polynomials ϕ, Φ, θ, and Θ are constrained so that the stationarity and invertibility conditions are satisfied. The unknown coefficients of these polynomials become part of the model parameter vector that is estimated using the data.

The notation for a closely related class of models, autoregressive integrated moving average (ARIMA) models, is also given here. A random sequence y_t is said to follow an ARIMA$(p,d,q)\times(P,D,Q)_s$ model if, for some nonnegative integers d and D, the differenced series $\epsilon_t = (1-B)^d(1-B^s)^D y_t$ follows an ARMA$(p,q)\times(P,Q)_s$ model. The integers d and D are called nonseasonal and seasonal differencing orders, respectively. You can specify ARIMA models by using the DEPLAG statement for specifying the differencing orders and by using the IRREGULAR statement for the ARMA specification. See Example 41.8 for an example of ARIMA$(0,1,1)\times(0,1,1)_{12}$ model specification. Brockwell and Davis (1991) can be consulted for additional information about ARIMA models.

You can use options of the IRREGULAR statement to specify the desired ARMA model and to request printed and graphical output. A few examples of the IRREGULAR statement are given next.

The following statement specifies an irregular component that is modeled as an ARMA(1,1) process. It also requests plotting its smoothed estimate.

```
irregular p=1 q=1 plot=smooth;
```

The following statement specifies an ARMA$(1,1)\times(1,1)_{12}$ model. It also fixes the coefficient of the first-order seasonal moving average polynomial to 0.1. The other coefficients and the white noise variance are estimated using the data.

```
irregular p=1 sp=1 q=1 sq=1 s=12 sma=0.1 noest=(sma);
```

AR=$\phi_1\ \phi_2\ \ldots\phi_p$
lists the starting values of the coefficients of the nonseasonal autoregressive polynomial

$$\phi(B) = 1 - \phi_1 B - \ldots - \phi_p B^p$$

where the order p is specified in the P= option. The coefficients ϕ_i must define a stationary autoregressive polynomial.

MA=$\theta_1\ \theta_2\ \ldots\theta_q$
lists the starting values of the coefficients of the nonseasonal moving average polynomial

$$\theta(B) = 1 - \theta_1 B - \ldots - \theta_q B^q$$

where the order q is specified in the Q= option. The coefficients θ_i must define an invertible moving average polynomial.

NOEST=(<VARIANCE> <AR> <SAR> <MA> <SMA>)
fixes the values of the ARMA parameters and the value of the white noise variance to those specified in the AR=, SAR=, MA=, SMA=, or VARIANCE= options.

P=*integer*
specifies the order of the nonseasonal autoregressive polynomial. The order can be any nonnegative integer; the default value is 0. In practice the order is a small integer such as 1, 2, or 3.

Q=*integer*
specifies the order of the nonseasonal moving average polynomial. The order can be any nonnegative integer; the default value is 0. In practice the order is a small integer such as 1, 2, or 3.

S=*integer*
specifies the season length used during the specification of the seasonal autoregressive or seasonal moving average polynomial. The season length can be any positive integer; for example, S=4 might be an appropriate value for a quarterly series. The default value is S=1.

SAR=$\Phi_1\ \Phi_2\ \ldots\ \Phi_P$
lists the starting values of the coefficients of the seasonal autoregressive polynomial

$$\Phi(B^s) = 1 - \Phi_1 B^s - \ldots - \Phi_P B^{sP}$$

where the order P is specified in the SP= option and the season length s is specified in the S= option. The coefficients Φ_i must define a stationary autoregressive polynomial.

SMA=$\Theta_1\ \Theta_2\ \ldots\ \Theta_Q$
lists the starting values of the coefficients of the seasonal moving average polynomial

$$\Theta(B^s) = 1 - \Theta_1 B^s - \ldots - \Theta_Q B^{sQ}$$

where the order Q is specified in the SQ= option and the season length s is specified in the S= option. The coefficients Θ_i must define an invertible moving average polynomial.

SP=*integer*
specifies the order of the seasonal autoregressive polynomial. The order can be any nonnegative integer; the default value is 0. In practice the order is a small integer such as 1 or 2.

SQ=*integer*
specifies the order of the seasonal moving average polynomial. The order can be any nonnegative integer; the default value is 0. In practice the order is a small integer such as 1 or 2.

LEVEL Statement

LEVEL < *options* > **;**

The LEVEL statement is used to include a level component in the model. The level component, either by itself or together with a slope component (see the SLOPE statement), forms the trend component, μ_t, of the model. If the slope component is absent, the resulting trend is a random walk (RW) specified by the following equations:

$$\mu_t = \mu_{t-1} + \eta_t, \quad \eta_t \sim i.i.d.\ N(0, \sigma_\eta^2)$$

If the slope component is present, signified by the presence of a SLOPE statement, a locally linear trend (LLT) is obtained. The equations of LLT are as follows:

$$\begin{aligned} \mu_t &= \mu_{t-1} + \beta_{t-1} + \eta_t, \quad \eta_t \sim i.i.d.\ N(0, \sigma_\eta^2) \\ \beta_t &= \beta_{t-1} + \xi_t, \quad \xi_t \sim i.i.d.\ N(0, \sigma_\xi^2) \end{aligned}$$

In either case, the options in the LEVEL statement are used to specify the value of σ_η^2 and to request forecasts of μ_t. The SLOPE statement is used for similar purposes in the case of slope β_t. The following examples illustrate the use of the LEVEL statement. Assuming that a SLOPE statement is not added subsequently, a simple random walk trend is specified by the following statement:

```
level;
```

The following statements specify a locally linear trend with value of σ_η^2 fixed at 4. It also requests printing of filtered values of μ_t. The value of σ_ξ^2, the disturbance variance in the slope equation, is estimated from the data.

```
level variance=4 noest print=filter;
slope;
```

CHECKBREAK
turns on the checking of breaks in the level component.

NOEST
fixes the value of σ_η^2 to the value specified in the VARIANCE= option.

PLOT=FILTER

PLOT=SMOOTH

PLOT=(< FILTER > < SMOOTH >)
requests plotting of the filtered or smoothed estimate of the level component.

PRINT=FILTER

PRINT=SMOOTH

PRINT=(< FILTER > < SMOOTH >)
requests printing of the filtered or smoothed estimate of the level component.

VARIANCE=*value*
specifies an initial value for σ_η^2, the disturbance variance in the μ_t equation at the start of the parameter estimation process. Any nonnegative value, including zero, is an acceptable starting value.

MODEL Statement

MODEL *dependent* < = *regressors* > **;**

The MODEL statement specifies the response variable and, optionally, the predictor or regressor variables for the UCM model. This is a required statement in the UCM procedure. The predictors specified in the MODEL statement are assumed to have a linear and time-invariant relationship with the response. The predictors that have time-varying regression coefficients are specified separately in the RANDOMREG statement. Similarly, the predictors that have a nonlinear effect on the response variable are specified separately in the SPLINEREG statement. Only one MODEL statement can be specified.

NLOPTIONS Statement

NLOPTIONS < *options* > **;**

PROC UCM uses the nonlinear optimization (NLO) subsystem to perform the nonlinear optimization of the likelihood function during the estimation of model parameters. You can use the NLOPTIONS statement to control different aspects of this optimization process. For most problems the default settings of the optimization process are adequate. However, in some cases it might be useful to change the optimization technique or to change the maximum number of iterations. This can be done by using the TECH= and MAXITER= options in the NLOPTIONS statement as follows:

```
nloptions tech=dbldog maxiter=200;
```

This sets the maximum number of iterations to 200 and changes the optimization technique to DBLDOG rather than the default technique, TRUREG, used in PROC UCM. A discussion of the full range of options that can be used with the NLOPTIONS statement is given in Chapter 7, "Nonlinear Optimization Methods." In PROC UCM all these options are available except the options related to the printing of the optimization history. In this version of PROC UCM all the printed output from the NLO subsystem is suppressed.

OUTLIER Statement

OUTLIER < *options* > **;**

The OUTLIER statement enables you to control the reporting of the additive outliers (AO) and level shifts (LS) in the response series. The AOs are searched by default. You can turn on the search for LSs by using the CHECKBREAK option in the LEVEL statement.

ALPHA=*significance-level*
: specifies the significance level for reporting the outliers. The default is 0.05.

MAXNUM=*number*
: limits the number of outliers to search. The default is MAXNUM=5.

MAXPCT=*number*
is similar to the MAXNUM= option. In the MAXPCT= option you can limit the number of outliers to search for according to a percentage of the series length. The default is MAXPCT=1. When both of these options are specified, the minimum of the two search numbers is used.

PRINT=SHORT | DETAIL
enables you to control the printed output of the outlier search. The PRINT=SHORT option, which is the default, produces an outlier summary table containing the most significant outliers, either AO or LS, discovered in the outlier search. The PRINT=DETAIL option produces, in addition to the outlier summary table, separate tables containing the AO and LS structural break chi-square statistics computed at each time point in the estimation span.

PERFORMANCE Statement

PERFORMANCE *options* **;**

The PERFORMANCE statement defines performance parameters for distributed and multithreaded computing and passes variables that describe the distributed computing environment. In the UCM procedure, this statement is applicable only if you specify the BOOTSTRAP option in the FORECAST statement. In addition, the number of nodes that you specify in the NODES= option in the PERFORMANCE statement must be strictly smaller than the number of bootstrap replications that you specify in the BOOTSTRAP option. The following statements illustrate how you can use this statement to perform bootstrap computations that use 10 nodes on a grid named *hpa.sas.com*:

```
proc ucm data=seriesG;
   id date interval=month;
   model logair;
   irregular;
   level;
   forecast lead=24 bootstrap(nrep=50 seed=1234);
   performance nodes=10 host="hpa.sas.com";
run;
```

For more information about the PERFORMANCE statement, see the section "PERFORMANCE Statement" on page 87 of Chapter 3, "Shared Concepts and Topics."

RANDOMREG Statement

RANDOMREG *regressors* < / *options* > **;**

The RANDOMREG statement is used to specify regressors with time-varying regression coefficients. Each regression coefficient—say, β_t— is assumed to evolve as a random walk:

$$\beta_t = \beta_{t-1} + \eta_t, \quad \eta_t \sim i.i.d.\ N(0, \sigma^2)$$

Of course, if the random walk disturbance variance σ^2 is zero, then the regression coefficient is not time varying, and it reduces to the standard regression setting. There can be multiple RANDOMREG statements, and each statement can contain one or more regressors. The regressors in a given RANDOMREG statement form a group that is assumed to share the same disturbance variance parameter. The random walks associated

with different regressors are assumed to be independent. For an example of using this statement see Example 41.4. See the section "Reporting Parameter Estimates for Random Regressors" on page 2795 for additional information about the way parameter estimates are reported for this type of regressors.

NOEST
fixes the value of σ^2 to the value specified in the VARIANCE= option.

PLOT=FILTER
PLOT=SMOOTH
PLOT=(< FILTER > < SMOOTH >)
requests plotting of filtered or smoothed estimate of the time-varying regression coefficient.

PRINT=FILTER
PRINT=SMOOTH
PRINT=(< FILTER > < SMOOTH >)
requests printing of the filtered or smoothed estimate of the time-varying regression coefficient.

VARIANCE=*value*
specifies an initial value for σ^2 during the parameter estimation process. Any nonnegative value, including zero, is an acceptable starting value.

SEASON Statement

SEASON *LENGTH = integer < options >* **;**

The SEASON or SEASONAL statement is used to specify a seasonal component, γ_t, in the model. A seasonal component can be one of the two types, DUMMY or TRIG. A DUMMY seasonal with season length s satisfies the following stochastic equation:

$$\sum_{i=0}^{s-1} \gamma_{t-i} = \omega_t, \qquad \omega_t \sim i.i.d.\ N(0, \sigma_\omega^2)$$

The equations for a TRIG (short for trigonometric) seasonal component are as follows

$$\gamma_t = \sum_{j=1}^{[s/2]} \gamma_{j,t}$$

where $[s/2]$ equals $s/2$ if s is even and $(s-1)/2$ if it is odd. The sinusoids, also called *harmonics*, $\gamma_{j,t}$ have frequencies $\lambda_j = 2\pi j/s$ and are specified by the matrix equation

$$\begin{bmatrix} \gamma_{j,t} \\ \gamma^*_{j,t} \end{bmatrix} = \begin{bmatrix} \cos\lambda_j & \sin\lambda_j \\ -\sin\lambda_j & \cos\lambda_j \end{bmatrix} \begin{bmatrix} \gamma_{j,t-1} \\ \gamma^*_{j,t-1} \end{bmatrix} + \begin{bmatrix} \omega_{j,t} \\ \omega^*_{j,t} \end{bmatrix}$$

where the disturbances $\omega_{j,t}$ and $\omega^*_{j,t}$ are assumed to be independent and, for fixed j, $\omega_{j,t}$ and $\omega^*_{j,t} \sim N(0, \sigma_\omega^2)$. If s is even, then the equation for $\gamma^*_{s/2,t}$ is not needed and $\gamma_{s/2,t}$ is given by

$$\gamma_{s/2,t} = -\gamma_{s/2,t-1} + \omega_{s/2,t}$$

In the TRIG seasonal case, the option KEEPH= or DROPH= can be used to obtain *subset trigonometric* seasonals that contain only a subset of the full set of harmonics $\gamma_{j,t}$, $j = 1, 2, \ldots, [s/2]$. This is particularly useful when the season length s is large and the seasonal pattern is relatively smooth.

Note that whether the seasonal type is DUMMY or TRIG, there is only one parameter, the disturbance variance σ_ω^2, in the seasonal model.

There can be more than one seasonal component in the model, necessarily with different season lengths if the seasons are full. You can have multiple *subset* season components with the same season length, if you need to use separate disturbance variances for different sets of harmonics. Each seasonal component is specified using a separate SEASON statement. A model with multiple seasonal components can easily become quite complex and might need a large amount of data and computing resources for its estimation and forecasting. The examples that follow illustrate the use of SEASON statement.

The following statement specifies a DUMMY type (default) seasonal component with a season length of four, corresponding to the quarterly seasonality. The disturbance variance σ_ω^2 is estimated from the data.

```
season length=4;
```

The following statement specifies a trigonometric seasonal with monthly seasonality. It also provides a starting value for σ_ω^2.

```
season length=12 type=trig variance=4;
```

DROPHARMONICS|DROPH=*number-list* | *n* **TO** *m* **BY** *p*
enables you to drop some harmonics $\gamma_{j,t}$ from the full set of harmonics used to obtain a trigonometric seasonal. The drop list can include any integer between 1 and $[s/2]$, s being the season length. For example, the following specification results in a specification of a trigonometric seasonal with a season length 12 that consists of only the first four harmonics $\gamma_{j,t}$, $j = 1, 2, 3, 4$:

```
season length=12 type=trig DROPH=5 6;
```

The last two *high* frequency harmonics are dropped. The DROPH= option cannot be used with the KEEPH= option.

KEEPHARMONICS|KEEPH=*number-list* | *n* **TO** *m* **BY** *p*
enables you to keep only the harmonics $\gamma_{j,t}$ listed in the option to obtain a trigonometric seasonal. The keep list can include any integer between 1 and $[s/2]$, s being the season length. For example, the following specification results in a specification of a trigonometric seasonal with a season length of 12 that consists of all the six harmonics $\gamma_{j,t}$, $j = 1, \ldots 6$:

```
season length=12 type=trig KEEPH=1 to 3;
season length=12 type=trig KEEPH=4 to 6;
```

However, these six harmonics are grouped into two groups, each having its own disturbance variance parameter. The DROPH= option cannot be used with the KEEPH= option.

LENGTH=*integer*
specifies the season length, s. This is a required option in this statement. The season length can be any integer greater than or equal to 2. Typical examples of season lengths are 12, corresponding to the monthly seasonality, or 4, corresponding to the quarterly seasonality.

NOEST
fixes the value of the disturbance variance parameter to the value specified in the VARIANCE= option.

PLOT=FILTER

PLOT=SMOOTH

PLOT=F_ANNUAL

PLOT=S_ANNUAL

PLOT=(<plot request> . . . <plot request>)
requests plots of the season component. When you specify only one plot request, you can omit the parentheses around the plot request. You can use the FILTER and SMOOTH options to plot the filtered and smoothed estimates of the season component γ_t. You can use the F_ANNUAL and S_ANNUAL options to get the plots of "annual" variation in the filtered and smoothed estimates of γ_t. The annual plots are useful to see the change in the contribution of a particular month over the span of years. Here "month" and "year" are generic terms that change appropriately with the interval type being used to label the observations and the season length. For example, for monthly data with a season length of 12, the usual meaning applies, while for daily data with a season length of 7, the days of the week serve as months and the weeks serve as years.

PRINT=HARMONICS
requests printing of the summary of harmonics present in the seasonal component. This option is valid only for the trigonometric seasonal component.

PRINT=FILTER

PRINT=SMOOTH

PRINT=(< print request > . . . < print request >)
requests printing of the filtered or smoothed estimate of the seasonal component γ_t.

TYPE=DUMMY | TRIG
specifies the type of the seasonal component. The default type is DUMMY.

VARIANCE=*value*
specifies an initial value for the disturbance variance, σ_ω^2, in the γ_t equation at the start of the parameter estimation process. Any nonnegative value, including zero, is an acceptable starting value.

SLOPE Statement

SLOPE < *options* > **;**

The SLOPE statement is used to include a slope component in the model. The slope component cannot be used without the level component (see the LEVEL statement). The level and slope specifications jointly

define the trend component of the model. A SLOPE statement without the accompanying LEVEL statement is ignored. The equations of the trend, defined jointly by the level μ_t and slope β_t, are as follows:

$$\begin{aligned} \mu_t &= \mu_{t-1} + \beta_{t-1} + \eta_t, \quad \eta_t \sim i.i.d.\ N(0, \sigma_\eta^2) \\ \beta_t &= \beta_{t-1} + \xi_t, \quad \xi_t \sim i.i.d.\ N(0, \sigma_\xi^2) \end{aligned}$$

The SLOPE statement is used to specify the value of the disturbance variance, σ_ξ^2, in the slope equation, and to request forecasts of β_t. The following examples illustrate this statement:

```
level;
slope;
```

The preceding statements fit a model with a locally linear trend. The disturbance variances σ_η^2 and σ_ξ^2 are estimated from the data. You can request a locally linear trend with fixed slope by using the following statements:

```
level;
slope variance=0 noest;
```

NOEST
fixes the value of the disturbance variance, σ_ξ^2, to the value specified in the VARIANCE= option.

PLOT=FILTER

PLOT=SMOOTH

PLOT=(< FILTER > < SMOOTH >)
requests plotting of the filtered or smoothed estimate of the slope component.

PRINT=FILTER

PRINT=SMOOTH

PRINT=(< FILTER > < SMOOTH >)
requests printing of the filtered or smoothed estimate of the slope component β_t.

VARIANCE=*value*
specifies an initial value for the disturbance variance, σ_ξ^2, in the β_t equation at the start of the parameter estimation process. Any nonnegative value, including zero, is an acceptable starting value.

SPLINEREG Statement

SPLINEREG *regressor* < *options* > **;**

The SPLINEREG statement is used to specify a regressor that has a nonlinear relationship with the dependent series that can be approximated by a given B-spline. If the specified spline has degree d and is based on n internal knots, then it is known that it can be written as a linear combination of $(n + d + 1)$ regressors that are derived from the original regressor. The span of these $(n + d + 1)$ derived regressors includes constant; therefore, to avoid multicollinearity with the level component, one of these regressors is dropped. Specifying the SPLINEREG statement is equivalent to specifying a RANDOMREG statement with these

derived regressors. There can be multiple SPLINEREG statements. You must specify at least one interior knot, either using the NKNOTS= option or the KNOTS= option. For more information about splines, see Chapter 117, "The TRANSREG Procedure" (*SAS/STAT User's Guide*). For an example of using this statement, see Example 41.6. See the section "Reporting Parameter Estimates for Random Regressors" on page 2795 for additional information about the way parameter estimates are reported for this type of regressors.

DEGREE=*integer*

specifies the degree of the spline. It can be any integer larger than or equal to zero. The default value is 3. The polynomial degree should be a small integer, usually 0, 1, 2, or 3. Larger values are rarely useful. If you have any doubt as to what degree to specify, use the default.

KNOTS=*number-list* | *n* **TO** *m* **BY** *p*

specifies the interior knots or break points. The values in the knot list must be nondecreasing and must lie between the minimum and the maximum of the spline regressor values in the input data set. The first time you specify a value in the knot list, it indicates a discontinuity in the nth (from DEGREE=n) derivative of the transformation function at the value of the knot. The second mention of a value indicates a discontinuity in the $(n-1)$th derivative of the transformation function at the value of the knot. Knots can be repeated any number of times for decreasing smoothness at the break points, but the values in the knot list can never decrease.

You cannot use the KNOTS= option with the NKNOTS= option. You should keep the number of knots small.

NKNOTS=*m*

creates m knots, the first at the $100/(m+1)$ percentile, the second at the $200/(m+1)$ percentile, and so on. Knots are always placed at data values; there is no interpolation. For example, if NKNOTS=3, knots are placed at the 25th percentile, the median, and the 75th percentile. The value specified for the NKNOTS= option must be ≥ 1. You cannot use the NKNOTS=option with the KNOTS= option.

NOTE: Specifying knots by using the NKNOTS= option can result in different sets of knots in the estimation and forecast stages if the distributions of regressor values in the estimation and forecast spans differ. The estimation span is based on the BACK= and SKIPFIRST= options in the ESTIMATE statement, and the forecast span is based on the BACK= and SKIPFIRST= options in the FORECAST statement.

NOEST

fixes the value of the regression coefficient random walk disturbance variance to the value specified in the VARIANCE= option.

PLOT=FILTER

PLOT=SMOOTH

PLOT=(< FILTER > < SMOOTH >)

requests plotting of filtered or smoothed estimate of the time-varying regression coefficient.

PRINT=FILTER

PRINT=SMOOTH

PRINT=(< FILTER > < SMOOTH >)

requests printing of filtered or smoothed estimate of the time-varying regression coefficient.

VARIANCE=*value*

specifies an initial value for the regression coefficient random walk disturbance variance during the parameter estimation process. Any nonnegative value, including zero, is an acceptable starting value.

SPLINESEASON Statement

SPLINESEASON *LENGTH = integer KNOTS= integer*$_1$ *integer*$_2$ ... *< options >* **;**

The SPLINESEASON statement is used to specify a seasonal pattern that is to be approximated by a given B-spline. If the specified spline has degree d and is based on n internal knots, then it can be written as a linear combination of $(n + d)$ regressors that are derived from the seasonal dummy regressors. The SPLINESEASON specification is equivalent to specifying a RANDOMREG specification with these derived regressors. Such approximation is useful only if the season length is relatively large, at least larger than $(n+d)$. For additional information about splines, see Chapter 117, "The TRANSREG Procedure" (*SAS/STAT User's Guide*). For an example of using this statement, see Example 41.3.

DEGREE=*integer*

specifies the degree of the spline. It can be any integer greater than or equal to zero. The default value is 3.

KNOTS=*integer*$_1$ *integer*$_2$ **...**

lists the *internal* knots. This list of values must be a nondecreasing sequence of integers within the range of 2 to $(s - 1)$, where s is the season length specified in the LENGTH= option. This is a required option in this statement.

LENGTH=*integer*

specifies the season length, *s*. This is a required option in this statement. The length can be any integer greater than or equal to three.

NOEST

fixes the value of the regression coefficient random walk disturbance variance to the value specified in the VARIANCE= option.

OFFSET=*integer*

specifies the position of the first measurement within the season, if the first measurement is not at the start of the season. The OFFSET= value must be between one and the season length. The default value is one. The first measurement refers to the start of the estimation span and the forecast span. If these spans differ, their starting measurements must be separated by an integer multiple of the season length.

PLOT=FILTER

PLOT=SMOOTH

PLOT=(< FILTER > < SMOOTH >)

requests plots of the season component. When you specify only one plot request, you can omit the parentheses around the plot request. You can use the FILTER and SMOOTH options to plot the filtered and smoothed estimates of the season component.

PRINT=FILTER

PRINT=SMOOTH

PRINT=(< FILTER > < SMOOTH >)
requests the printing of the filtered or smoothed estimate of the spline season component.

RKNOTS=(*knot, . . . , knot* **) . . . (***knot, . . . , knot* **)**
specifies a grouping of knots such that the knots within the same group have identical seasonal values. The knots specified in this option must already be present in the list specified by the KNOTS= option. The knot groups must be non-overlapping and without any repeated knots.

VARIANCE=*value*
specifies an initial value for the regression coefficient random walk disturbance variance during the parameter estimation process. Any nonnegative value, including zero, is an acceptable starting value.

Details: UCM Procedure

An Introduction to Unobserved Component Models

A UCM decomposes the response series into components such as trend, seasons, cycles, and the regression effects due to predictor series. The following model shows a possible scenario:

$$\begin{aligned} y_t &= \mu_t + \gamma_t + \psi_t + \sum_{j=1}^{m} \beta_j x_{jt} + \epsilon_t \\ \epsilon_t &\sim i.i.d.\ N(0, \sigma_\epsilon^2) \end{aligned}$$

The terms μ_t, γ_t, and ψ_t represent the trend, seasonal, and cyclical components, respectively. In fact the model can contain multiple seasons and cycles, and the seasons can be of different types. For simplicity of discussion the preceding model contains only one of each of these components. The regression term, $\sum_{j=1}^{m} \beta_j x_{jt}$, includes contribution of regression variables with *fixed* regression coefficients. A model can also contain regression variables that have *time varying* regression coefficients or that have a nonlinear relationship with the dependent series (see "Incorporating Predictors of Different Kinds" on page 2794). The disturbance term ϵ_t, also called the *irregular* component, is usually assumed to be Gaussian white noise. In some cases it is useful to model the irregular component as a stationary ARMA process. See the section "Modeling the Irregular Component" on page 2786 for additional information.

By controlling the presence or absence of various terms and by choosing the proper flavor of the included terms, the UCMs can generate a rich variety of time series patterns. A UCM can be applied to variables after transforming them by transforms such as *log* and *difference*.

The components μ_t, γ_t, and ψ_t model structurally different aspects of the time series. For example, the trend μ_t models the natural tendency of the series in the absence of any other perturbing effects such as seasonality, cyclical components, and the effects of exogenous variables, while the seasonal component γ_t models the correction to the level due to the seasonal effects. These components are assumed to be statistically independent of each other and independent of the irregular component. All of the component

models can be thought of as stochastic generalizations of the relevant deterministic patterns in time. This way the deterministic cases emerge as special cases of the stochastic models. The different models available for these unobserved components are discussed next.

Modeling the Trend

As mentioned earlier, the trend in a series can be loosely defined as the natural tendency of the series in the absence of any other perturbing effects. The UCM procedure offers two ways to model the trend component μ_t. The first model, called the random walk (RW) model, implies that the trend remains roughly constant throughout the life of the series without any persistent upward or downward drift. In the second model the trend is modeled as a locally linear time trend (LLT). The RW model can be described as

$$\mu_t = \mu_{t-1} + \eta_t, \quad \eta_t \sim i.i.d.\ N(0, \sigma_\eta^2)$$

Note that if $\sigma_\eta^2 = 0$, then the model becomes $\mu_t = constant$. In the LLT model the trend is locally linear, consisting of both the *level* and *slope*. The LLT model is

$$\begin{aligned} \mu_t &= \mu_{t-1} + \beta_{t-1} + \eta_t, & \eta_t &\sim i.i.d.\ N(0, \sigma_\eta^2) \\ \beta_t &= \beta_{t-1} + \xi_t, & \xi_t &\sim i.i.d.\ N(0, \sigma_\xi^2) \end{aligned}$$

The disturbances η_t and ξ_t are assumed to be independent. There are some interesting special cases of this model obtained by setting one or both of the disturbance variances σ_η^2 and σ_ξ^2 equal to zero. If σ_ξ^2 is set equal to zero, then you get a linear trend model with fixed slope. If σ_η^2 is set to zero, then the resulting model usually has a smoother trend. If both the variances are set to zero, then the resulting model is the deterministic linear time trend: $\mu_t = \mu_0 + \beta_0 t$.

You can incorporate these trend patterns in your model by using the LEVEL and SLOPE statements.

Modeling a Cycle

A deterministic cycle ψ_t with frequency λ, $0 < \lambda < \pi$, can be written as

$$\psi_t = \alpha \cos(\lambda t) + \beta \sin(\lambda t)$$

If the argument t is measured on a continuous scale, then ψ_t is a periodic function with period $2\pi/\lambda$, amplitude $\gamma = (\alpha^2 + \beta^2)^{1/2}$, and phase $\phi = \tan^{-1}(\beta/\alpha)$. Equivalently, the cycle can be written in terms of the amplitude and phase as

$$\psi_t = \gamma \cos(\lambda t - \phi)$$

Note that when ψ_t is measured only at the integer values, it is not exactly periodic, unless $\lambda = (2\pi j)/k$ for some integers j and k. The cycles in their pure form are not used very often in practice. However, they are very useful as building blocks for more complex periodic patterns. It is well known that the periodic pattern of any complexity can be written as a sum of pure cycles of different frequencies and amplitudes. In time series situations it is useful to generalize this simple cyclical pattern to a stochastic cycle that has a fixed period but time-varying amplitude and phase. The stochastic cycle considered here is motivated by the following recursive formula for computing ψ_t:

$$\begin{bmatrix} \psi_t \\ \psi_t^* \end{bmatrix} = \begin{bmatrix} \cos\lambda & \sin\lambda \\ -\sin\lambda & \cos\lambda \end{bmatrix} \begin{bmatrix} \psi_{t-1} \\ \psi_{t-1}^* \end{bmatrix}$$

starting with $\psi_0 = \alpha$ and $\psi_0^* = \beta$. Note that ψ_t and ψ_t^* satisfy the relation

$$\psi_t^2 + \psi_t^{*2} = \alpha^2 + \beta^2 \quad \text{for all } t$$

A stochastic generalization of the cycle ψ_t can be obtained by adding random noise to this recursion and by introducing a damping factor, ρ, for additional modeling flexibility. This model can be described as follows

$$\begin{bmatrix} \psi_t \\ \psi_t^* \end{bmatrix} = \rho \begin{bmatrix} \cos\lambda & \sin\lambda \\ -\sin\lambda & \cos\lambda \end{bmatrix} \begin{bmatrix} \psi_{t-1} \\ \psi_{t-1}^* \end{bmatrix} + \begin{bmatrix} \nu_t \\ \nu_t^* \end{bmatrix}$$

where $0 \leq \rho \leq 1$, and the disturbances ν_t and ν_t^* are independent $N(0, \sigma_\nu^2)$ variables. The resulting stochastic cycle has a fixed period but time-varying amplitude and phase. The stationarity properties of the random sequence ψ_t depend on the damping factor ρ. If $\rho < 1$, ψ_t has a stationary distribution with mean zero and variance $\sigma_\nu^2/(1-\rho^2)$. If $\rho = 1$, ψ_t is nonstationary.

You can incorporate a cycle in a UCM by specifying a CYCLE statement. You can include multiple cycles in the model by using separate CYCLE statements for each included cycle.

As mentioned before, the cycles are very useful as building blocks for constructing more complex periodic patterns. Periodic patterns of almost any complexity can be created by superimposing cycles of different periods and amplitudes. In particular, the seasonal patterns, general periodic patterns with integer periods, can be constructed as sums of cycles. This important topic of modeling the seasonal components is considered next.

Modeling Seasons

The seasonal fluctuations are a common source of variation in time series data. These fluctuations arise because of the regular changes in seasons or some other periodic events. The seasonal effects are regarded as corrections to the general trend of the series due to the seasonal variations, and these effects sum to zero when summed over the full season cycle. Therefore the seasonal component γ_t is modeled as a stochastic periodic pattern of an integer period s such that the sum $\sum_{i=0}^{s-1} \gamma_{t-i}$ is always zero in the mean. The period s is called the season length. Two different models for the seasonal component are considered here. The first model is called the *dummy* variable form of the seasonal component. It is described by the equation

$$\sum_{i=0}^{s-1} \gamma_{t-i} = \omega_t, \qquad \omega_t \sim i.i.d.\ N(0, \sigma_\omega^2)$$

The other model is called the *trigonometric* form of the seasonal component. In this case γ_t is modeled as a sum of cycles of different frequencies. This model is given as follows

$$\gamma_t = \sum_{j=1}^{[s/2]} \gamma_{j,t}$$

where $[s/2]$ equals $s/2$ if s is even and $(s-1)/2$ if it is odd. The cycles $\gamma_{j,t}$ have frequencies $\lambda_j = 2\pi j/s$ and are specified by the matrix equation

$$\begin{bmatrix} \gamma_{j,t} \\ \gamma_{j,t}^* \end{bmatrix} = \begin{bmatrix} \cos\lambda_j & \sin\lambda_j \\ -\sin\lambda_j & \cos\lambda_j \end{bmatrix} \begin{bmatrix} \gamma_{j,t-1} \\ \gamma_{j,t-1}^* \end{bmatrix} + \begin{bmatrix} \omega_{j,t} \\ \omega_{j,t}^* \end{bmatrix}$$

where the disturbances $\omega_{j,t}$ and $\omega^*_{j,t}$ are assumed to be independent and, for fixed j, $\omega_{j,t}$ and $\omega^*_{j,t} \sim N(0, \sigma^2_\omega)$. If s is even, then the equation for $\gamma^*_{s/2,t}$ is not needed and $\gamma_{s/2,t}$ is given by

$$\gamma_{s/2,t} = -\gamma_{s/2,t-1} + \omega_{s/2,t}$$

The cycles $\gamma_{j,t}$ are called *harmonics*. If the seasonal component is deterministic, the decomposition of the seasonal effects into these harmonics is identical to its Fourier decomposition. In this case the sum of squares of the seasonal factors equals the sum of squares of the amplitudes of these harmonics. In many practical situations, the contribution of the high-frequency harmonics is negligible and can be ignored, giving rise to a simpler description of the seasonal. In the case of stochastic seasonals, the situation might not be so transparent; however, similar considerations still apply. Note that if the disturbance variance $\sigma^2_\omega = 0$, then both the dummy and the trigonometric forms of seasonal components reduce to constant seasonal effects. That is, the seasonal component reduces to a deterministic function that is completely determined by its first $s - 1$ values.

In the UCM procedure you can specify a seasonal component in a variety of ways, the SEASON statement being the simplest of these. The dummy and the trigonometric seasonal components discussed so far can be considered as *saturated* seasonal components that put no restrictions on the $s - 1$ seasonal values. In some cases a more parsimonious representation of the seasonal might be more appropriate. This is particularly useful for seasonal components with large season lengths. In the UCM procedure you can obtain parsimonious representations of the seasonal components by one of the following ways:

- Use a *subset* trigonometric seasonal component obtained by deleting a few of the $[s/2]$ harmonics used in its sum. For example, a slightly smoother seasonal component of length 12, corresponding to the monthly seasonality, can be obtained by deleting the highest-frequency harmonic of period 2. That is, such a seasonal component will be a sum of five stochastic cycles that have periods 12, 6, 4, 3, and 2.4. You can specify such subset seasonal components by using the KEEPH= or DROPH= option in the SEASON statement.

- Approximate the seasonal pattern by a suitable spline approximation. You can do this by using the SPLINESEASON statement.

- A *block-seasonal* pattern is a seasonal pattern where the pattern is divided into a few blocks of equal length such that the season values within a block are the same—for example, a monthly seasonal pattern that has only four different values, one for each quarter. In some situations a long seasonal pattern can be approximated by the sum of block season and a simple season, the length of the simple season being equal to the block length of the block season. You can obtain such approximation by using a combination of BLOCKSEASON and SEASON statements.

- Consider a seasonal component of a large season length as a sum of two or more seasonal components that are each of much smaller season lengths. This can be done by specifying more than one SEASON statements.

Note that the preceding techniques of obtaining parsimonious seasonal components can also enable you to specify seasonal components that are more *general* than the simple saturated seasonal components. For example, you can specify a saturated trigonometric seasonal component that has some of its harmonics evolving according to one disturbance variance parameter while the others evolve with another disturbance variance parameter.

Modeling an Autoregression

An autoregression of order one can be thought of as a special case of a cycle when the frequency λ is either 0 or π. Modeling this special case separately helps interpretation and parameter estimation. The autoregression component r_t is modeled as follows

$$r_t = \rho r_{t-1} + v_t, \quad v_t \sim i.i.d.\ N(0, \sigma_v^2)$$

where $-1 \leq \rho < 1$. An autoregression can also provide an alternative to the IRREGULAR component when the model errors show some autocorrelation. You can incorporate an autoregression in your model by using the AUTOREG statement.

Modeling Regression Effects

A predictor variable can affect the response variable in a variety of ways. The UCM procedure enables you to model several different types of predictor-response relationships:

- The predictor-response relationship is *linear*, and the regression coefficient does not change with time. This is the simplest kind of relationship and such predictors are specified in the MODEL statement.
- The predictor-response relationship is *linear*, but the regression coefficient does change with time. Such predictors are specified in the RANDOMREG statement. Here the regression coefficient is assumed to evolve as a random walk.
- The predictor-response relationship is *nonlinear* and the relationship can change with time. This type of relationship can be approximated by an appropriate time-varying spline. Such predictors are specified in the SPLINEREG statement.

A response variable can depend on its own past values—that is, lagged dependent values. Such a relationship can be specified in the DEPLAG statement.

Modeling the Irregular Component

The components—such as trend, seasonal and regression effects, and nonstationary cycles—are used to capture the structural dynamics of a response series. In contrast, the stationary cycles and the autoregression are used to capture the transient aspects of the response series that are important for its short-range prediction but have little impact on its long-term forecasts. The irregular component represents the residual variation remaining in the response series that is modeled using an appropriate selection of structural and transient effects. In most cases, the irregular component can be assumed to be simply Gaussian white noise. In some other cases, however, the residual variation can be more complicated. In such situations, it might be necessary to model the irregular component as a stationary ARMA process. Moreover, you can use the ARMA irregular component together with the dependent lag specification (see the DEPLAG statement) to specify an ARIMA$(p,d,q)\times(P,D,Q)_s$ model for the response series. See the IRREGULAR statement for the explanation of the ARIMA notation. See Example 41.8 for an example of modeling a series by using an ARIMA$(0,1,1)\times(0,1,1)_{12}$ model.

The Model Parameters

The parameter vector in a UCM consists of the variances of the disturbance terms of the unobserved components, the damping coefficients and frequencies in the cycles, the damping coefficient in the autoregression, and the regression coefficients in the regression terms. These parameters are estimated by maximizing the likelihood. It is possible to restrict the values of the model parameters to user-specified values.

Model Specification

A UCM is specified by describing the components in the model. For example, consider the model

$$y_t = \mu_t + \gamma_t + \epsilon_t$$

consisting of the irregular, level, slope, and seasonal components. This model is called the basic structural model (BSM) by Harvey (1989). The syntax for a BSM with monthly seasonality of trigonometric type is as follows:

```
model y;
irregular;
level;
slope;
season length=12 type=trig;
```

Similarly the following syntax specifies a BSM with a response variable y, a regressor x, and dummy-type monthly seasonality:

```
model y = x;
irregular;
level;
slope variance=0 noest;
season length=12 type=dummy;
```

Moreover, the disturbance variance of the slope component is restricted to zero, giving rise to a local linear trend with fixed slope.

A model can contain multiple cycle and seasonal components. In such cases the model syntax contains a separate statement for each of these multiple cycle or seasonal components; for example, the syntax for a model containing irregular and level components along with two cycle components could be as follows:

```
model y = x;
irregular;
level;
cycle;
cycle;
```

The UCMs as State Space Models

The UCMs considered in PROC UCM can be thought of as special cases of more general models, called (linear) Gaussian state space models (GSSM). A GSSM can be described as follows:

$$\begin{aligned} y_t &= Z_t \alpha_t \\ \alpha_{t+1} &= T_t \alpha_t + \zeta_{t+1}, \quad \zeta_t \sim \mathrm{N}(0, Q_t) \\ \alpha_1 &\sim \mathrm{N}(0, P) \end{aligned}$$

The first equation, called the *observation equation*, relates the response series y_t to a state vector α_t that is usually unobserved. The second equation, called the *state equation*, describes the evolution of the state vector in time. The system matrices Z_t and T_t are of appropriate dimensions and are known, except possibly for some unknown elements that become part of the parameter vector of the model. The noise series ζ_t consists of independent, zero-mean, Gaussian vectors with covariance matrices Q_t. For most of the UCMs considered here, the system matrices Z_t and T_t, and the noise covariances Q_t, are time invariant—that is, they do not depend on time. In a few cases, however, some or all of them can depend on time. The initial state vector α_1 is assumed to be independent of the noise series, and its covariance matrix P can be partially diffuse. A random vector has a partially diffuse covariance matrix if it can be partitioned such that one part of the vector has a properly defined probability distribution, while the covariance matrix of the other part is infinite—that is, you have no prior information about this part of the vector. The covariance of the initial state α_1 is assumed to have the following form:

$$P = P_* + \kappa P_\infty$$

where P_* and P_∞ are nonnegative definite, symmetric matrices and κ is a constant that is assumed to be close to ∞. In the case of UCMs considered here, P_∞ is always a diagonal matrix that consists of zeros and ones, and, if a particular diagonal element of P_∞ is one, then the corresponding row and column in P_* are zero.

The state space formulation of a UCM has many computational advantages. In this formulation there are convenient algorithms for estimating and forecasting the unobserved states $\{\alpha_t\}$ by using the observed series $\{y_t\}$. These algorithms also yield the in-sample and out-of-sample forecasts and the likelihood of $\{y_t\}$. The state space representation of a UCM does not need to be unique. In the representation used here, the unobserved components in the UCM often appear as elements of the state vector. This makes the elements of the state interpretable and, more important, the sample estimates and forecasts of these unobserved components are easily obtained. For additional information about the computational aspects of the state space modeling, see Durbin and Koopman (2001). Next, some notation is developed to describe the essential quantities computed during the analysis of the state space models.

Let $\{y_t, t = 1, \ldots, n\}$ be the observed sample from a series that satisfies a state space model. Next, for $1 \le t \le n$, let the one-step-ahead forecasts of the series, the states, and their variances be defined as follows, using the usual notation to denote the conditional expectation and conditional variance:

$$\begin{aligned} \hat{\alpha}_t &= \mathrm{E}(\alpha_t | y_1, y_2, \ldots, y_{t-1}) \\ \Gamma_t &= \mathrm{Var}(\alpha_t | y_1, y_2, \ldots, y_{t-1}) \\ \hat{y}_t &= \mathrm{E}(y_t | y_1, y_2, \ldots, y_{t-1}) \\ F_t &= \mathrm{Var}(y_t | y_1, y_2, \ldots, y_{t-1}) \end{aligned}$$

These are also called the *filtered* estimates of the series and the states. Similarly, for $t \geq 1$, let the following denote the full-sample estimates of the series and the state values at time t:

$$\begin{aligned}
\tilde{\alpha}_t &= \mathrm{E}(\alpha_t | y_1, y_2, \ldots, y_n) \\
\Delta_t &= \mathrm{Var}(\alpha_t | y_1, y_2, \ldots, y_n) \\
\tilde{y}_t &= \mathrm{E}(y_t | y_1, y_2, \ldots, y_n) \\
G_t &= \mathrm{Var}(y_t | y_1, y_2, \ldots, y_n)
\end{aligned}$$

If the time t is in the historical period— that is, if $1 \leq t \leq n$— then the full-sample estimates are called the *smoothed* estimates, and if t lies in the future then they are called out-of-sample forecasts. Note that if $1 \leq t \leq n$, then $\tilde{y}_t = y_t$ and $G_t = 0$, unless y_t is missing.

All the filtered and smoothed estimates ($\hat{\alpha}_t, \tilde{\alpha}_t, \ldots, G_t$, and so on) are computed by using the Kalman filtering and smoothing (KFS) algorithm, which is an iterative process. If the initial state is diffuse, as is often the case for the UCMs, its treatment requires modification of the traditional KFS, which is called the diffuse KFS (DKFS). The details of DKFS implemented in the UCM procedure can be found in De Jong and Chu-Chun-Lin (2003). Additional information on the state space models can be found in Durbin and Koopman (2001). The likelihood formulas described in this section are taken from the latter reference.

In the case of diffuse initial condition, the effect of the improper prior distribution of α_1 manifests itself in the first few filtering iterations. During these initial filtering iterations the distribution of the filtered quantities remains diffuse; that is, during these iterations the one-step-ahead series and state forecast variances F_t and Γ_t have the following form:

$$\begin{aligned}
F_t &= F_{*t} + \kappa F_{\infty t} \\
\Gamma_t &= \Gamma_{*t} + \kappa \Gamma_{\infty t}
\end{aligned}$$

The actual number of iterations—say, I— affected by this improper prior depends on the nature of the vectors Z_t, the number of nonzero diagonal elements of P_∞, and the pattern of missing values in the dependent series. After I iterations, $\Gamma_{\infty t}$ and $F_{\infty t}$ become zero and the one-step-ahead series and state forecasts have proper distributions. These first I iterations constitute the *initialization* phase of the DKFS algorithm. The post-initialization phase of the DKFS and the traditional KFS is the same. In the state space modeling literature the pre-initialization and post-initialization phases are some times called *pre-collapse* and *post-collapse* phases of the diffuse Kalman filtering. In certain missing value patterns it is possible for I to exceed the sample size; that is, the sample information can be insufficient to create a proper prior for the filtering process. In these cases, parameter estimation and forecasting is done on the basis of this improper prior, and some or all of the series and component forecasts can have infinite variances (or zero precision). The forecasts that have infinite variance are set to missing. The same situation can occur if the specified model contains components that are essentially multicollinear. In these situations no residual analysis is possible; in particular, no residuals-based goodness-of-fit statistics are produced.

The log likelihood of the sample (L_∞), which takes account of this diffuse initialization step, is computed by using the one-step-ahead series forecasts as follows

$$L_\infty(y_1, \ldots, y_n) = -\frac{(n-d)}{2} \log 2\pi - \frac{1}{2} \sum_{t=1}^{I} w_t - \frac{1}{2} \sum_{t=I+1}^{n} (\log F_t + \frac{v_t^2}{F_t})$$

where d is the number of diffuse elements in the initial state α_1, $v_t = y_t - Z_t\hat{\alpha}_t$ are the one-step-ahead residuals, and

$$\begin{aligned} w_t &= \log F_{\infty t} && \text{if } F_{\infty t} > 0 \\ &= \log F_{*t} + \frac{v_t^2}{F_{*t}} && \text{if } F_{\infty t} = 0 \end{aligned}$$

If y_t is missing at some time t, then the corresponding summand in the log likelihood expression is deleted, and the constant term is adjusted suitably. Moreover, if the initialization step does not complete—that is, if I exceeds the sample size— then the value of d is reduced to the number of diffuse states that are successfully initialized.

The portion of the log likelihood that corresponds to the post-initialization period is called the nondiffuse log likelihood (L_0). The nondiffuse log likelihood is given by

$$L_0(y_1, \ldots, y_n) = -\frac{1}{2} \sum_{t=I+1}^{n} (\log F_t + \frac{v_t^2}{F_t})$$

In the case of UCMs considered in PROC UCM, it often happens that the diffuse part of the likelihood, $\sum_{t=1}^{I} w_t$, does not depend on the model parameters, and in these cases the maximization of nondiffuse and diffuse likelihoods is equivalent. However, in some cases, such as when the model consists of dependent lags, the diffuse part does depend on the model parameters. In these cases the maximization of the diffuse and nondiffuse likelihood can produce different parameter estimates.

In some situations it is convenient to reparameterize the nondiffuse initial state covariance P_* as $\sigma^2 P_*$ and the state noise covariance Q_t as $\sigma^2 Q_t$ for some common scalar parameter σ^2. In this case the preceding log-likelihood expression, up to a constant, can be written as

$$L_\infty(y_1, \ldots, y_n) = -\frac{1}{2} \sum_{t=1}^{I} w_t - \frac{1}{2} \sum_{t=I+1}^{n} \log F_t - \frac{1}{2\sigma^2} \sum_{t=I+1}^{n} \frac{v_t^2}{F_t} - \frac{(n-d)}{2} \log \sigma^2$$

Solving analytically for the optimum, the maximum likelihood estimate of σ^2 can be shown to be

$$\hat{\sigma}^2 = \frac{1}{(n-d)} \sum_{t=I+1}^{n} \frac{v_t^2}{F_t}$$

When this expression of σ^2 is substituted back into the likelihood formula, an expression called the *profile likelihood* ($L_{profile}$) of the data is obtained:

$$-2L_{profile}(y_1, \ldots, y_n) = \sum_{t=1}^{I} w_t + \sum_{t=I+1}^{n} \log F_t + (n-d) \log(\sum_{t=I+1}^{n} \frac{v_t^2}{F_t})$$

In some situations the parameter estimation is done by optimizing the profile likelihood (see the section "Parameter Estimation by Profile Likelihood Optimization" on page 2798 and the PROFILE option in the ESTIMATE statement).

In the remainder of this section the state space formulation of UCMs is further explained by using some particular UCMs as examples. The examples show that the state space formulation of the UCMs depends

on the components in the model in a simple fashion; for example, the system matrix T is usually a block diagonal matrix with blocks that correspond to the components in the model. The only exception to this pattern is the UCMs that consist of the lags of dependent variable. This case is considered at the end of the section.

In what follows, $Diag\,[a, b, \ldots]$ denotes a diagonal matrix with diagonal entries $[a, b, \ldots]$, and the transpose of a matrix T is denoted as $T^{'}$.

Locally Linear Trend Model

Recall that the dynamics of the locally linear trend model are

$$\begin{aligned} y_t &= \mu_t + \epsilon_t \\ \mu_t &= \mu_{t-1} + \beta_{t-1} + \eta_t \\ \beta_t &= \beta_{t-1} + \xi_t \end{aligned}$$

Here y_t is the response series and ϵ_t, η_t, and ξ_t are independent, zero-mean Gaussian disturbance sequences with variances $\sigma_\epsilon^2, \sigma_\eta^2$, and σ_ξ^2, respectively. This model can be formulated as a state space model where the state vector $\alpha_t = [\ \epsilon_t\ \mu_t\ \beta_t\ ]^{'}$ and the state noise $\zeta_t = [\ \epsilon_t\ \eta_t\ \xi_t\ ]^{'}$. Note that the elements of the state vector are precisely the unobserved components in the model. The system matrices T and Z and the noise covariance Q corresponding to this choice of state and state noise vectors can be seen to be time invariant and are given by

$$Z = [\ 1\ 1\ 0\ ],\quad T = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } Q = Diag\left[\sigma_\epsilon^2, \sigma_\eta^2, \sigma_\xi^2\right]$$

The distribution of the initial state vector α_1 is diffuse, with $P_* = Diag\left[\sigma_\epsilon^2, 0, 0\right]$ and $P_\infty = Diag\,[0, 1, 1]$. The parameter vector θ consists of all the disturbance variances—that is, $\theta = (\sigma_\epsilon^2, \sigma_\eta^2, \sigma_\xi^2)$.

Basic Structural Model

The basic structural model (BSM) is obtained by adding a seasonal component, γ_t, to the local level model. In order to economize on the space, the state space formulation of a BSM with a relatively short season length, season length = 4 (quarterly seasonality), is considered here. The pattern for longer season lengths such as 12 (monthly) and 52 (weekly) is easy to see.

Let us first consider the dummy form of seasonality. In this case the state and state noise vectors are $\alpha_t = \left[\ \epsilon_t\ \mu_t\ \beta_t\ \gamma_{1,t}\ \gamma_{2,t}\ \gamma_{3,t}\ \right]^{'}$ and $\zeta_t = [\ \epsilon_t\ \eta_t\ \xi_t\ \omega_t\ 0\ 0\ ]^{'}$, respectively. The first three elements of the state vector are the irregular, level, and slope components, respectively. The remaining elements, $\gamma_{i,t}$, are lagged versions of the seasonal component γ_t. $\gamma_{1,t}$ corresponds to lag zero—that is, the same as γ_t, $\gamma_{2,t}$ to lag 1 and $\gamma_{3,t}$ to lag 2. The system matrices are

$$Z = [\ 1\ 1\ 0\ 1\ 0\ 0\ ],\quad T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & -1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

and $Q = Diag\left[\sigma_\epsilon^2, \sigma_\eta^2, \sigma_\xi^2, \sigma_\omega^2, 0, 0\right]$. The distribution of the initial state vector α_1 is diffuse, with $P_* = Diag\left[\sigma_\epsilon^2, 0, 0, 0, 0, 0\right]$ and $P_\infty = Diag\left[0, 1, 1, 1, 1, 1\right]$.

In the case of the trigonometric type of seasonality, $\alpha_t = \left[\begin{array}{cccccc} \epsilon_t & \mu_t & \beta_t & \gamma_{1,t} & \gamma_{1,t}^* & \gamma_{2,t} \end{array}\right]'$ and $\zeta_t = \left[\begin{array}{cccccc} \epsilon_t & \eta_t & \xi_t & \omega_{1,t} & \omega_{1,t}^* & \omega_{2,t} \end{array}\right]'$. The disturbance sequences, $\omega_{j,t}, 1 \leq j \leq 2$, and $\omega_{1,t}^*$, are independent, zero-mean, Gaussian sequences with variance σ_ω^2. The system matrices are

$$Z = \left[\begin{array}{cccccc} 1 & 1 & 0 & 1 & 0 & 1 \end{array}\right], \quad T = \left[\begin{array}{cccccc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos\lambda_1 & \sin\lambda_1 & 0 \\ 0 & 0 & 0 & -\sin\lambda_1 & \cos\lambda_1 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cos\lambda_2 \end{array}\right]$$

and $Q = Diag\left[\sigma_\epsilon^2, \sigma_\eta^2, \sigma_\xi^2, \sigma_\omega^2, \sigma_\omega^2, \sigma_\omega^2\right]$. Here $\lambda_j = (2\pi j)/4$. The distribution of the initial state vector α_1 is diffuse, with $P_* = Diag\left[\sigma_\epsilon^2, 0, 0, 0, 0, 0\right]$ and $P_\infty = Diag\left[0, 1, 1, 1, 1, 1\right]$. The parameter vector in both the cases is $\theta = (\sigma_\epsilon^2, \sigma_\eta^2, \sigma_\xi^2, \sigma_\omega^2)$.

Seasons with Blocked Seasonal Values

Block seasonals are special seasonal components that impose a special block structure on the seasonal effects. Let us consider a BSM with monthly seasonality that has a quarterly block structure—that is, months within the same quarter are assumed to have identical effects except for some random perturbation. Such a seasonal component is a block seasonal with block size m equal to 3 and the number of blocks k equal to 4. The state space structure for such a model with dummy-type seasonality is as follows: The state and state noise vectors are $\alpha_t = \left[\begin{array}{cccccc} \epsilon_t & \mu_t & \beta_t & \gamma_{1,t} & \gamma_{2,t} & \gamma_{3,t} \end{array}\right]'$ and $\zeta_t = \left[\begin{array}{cccccc} \epsilon_t & \eta_t & \xi_t & \omega_t & 0 & 0 \end{array}\right]'$, respectively. The first three elements of the state vector are the irregular, level, and slope components, respectively. The remaining elements, $\gamma_{i,t}$, are lagged versions of the seasonal component γ_t. $\gamma_{1,t}$ corresponds to lag zero—that is, the same as γ_t, $\gamma_{2,t}$ to lag m and $\gamma_{3,t}$ to lag $2m$. All the system matrices are time invariant, except the matrix T. They can be seen to be $Z = \left[\begin{array}{cccccc} 1 & 1 & 0 & 1 & 0 & 0 \end{array}\right]$, $Q = Diag\left[\sigma_\epsilon^2, \sigma_\eta^2, \sigma_\xi^2, \sigma_\omega^2, 0, 0\right]$, and

$$T_t = \left[\begin{array}{cccccc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & -1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right]$$

when t is a multiple of the block size m, and

$$T_t = \left[\begin{array}{cccccc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

otherwise. Note that when t is not a multiple of m, the portion of the T_t matrix corresponding to the seasonal is identity. The distribution of the initial state vector α_1 is diffuse, with $P_* = Diag\left[\sigma_\epsilon^2, 0, 0, 0, 0, 0\right]$ and $P_\infty = Diag\left[0, 1, 1, 1, 1, 1\right]$.

Similarly in the case of the trigonometric form of seasonality, $\alpha_t = \left[\begin{array}{cccccc} \epsilon_t & \mu_t & \beta_t & \gamma_{1,t} & \gamma_{1,t}^* & \gamma_{2,t} \end{array}\right]'$ and $\zeta_t = \left[\begin{array}{cccccc} \epsilon_t & \eta_t & \xi_t & \omega_{1,t} & \omega_{1,t}^* & \omega_{2,t} \end{array}\right]'$. The disturbance sequences, $\omega_{j,t}, 1 \leq j \leq 2$, and $\omega_{1,t}^*$, are independent, zero-mean, Gaussian sequences with variance σ_ω^2. $Z = [\,1\,1\,0\,1\,0\,1\,]$, $Q = Diag\left[\sigma_\epsilon^2, \sigma_\eta^2, \sigma_\xi^2, \sigma_\omega^2, \sigma_\omega^2, \sigma_\omega^2\right]$, and

$$T_t = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos\lambda_1 & \sin\lambda_1 & 0 \\ 0 & 0 & 0 & -\sin\lambda_1 & \cos\lambda_1 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cos\lambda_2 \end{bmatrix}$$

when t is a multiple of the block size m, and

$$T_t = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

otherwise. As before, when t is not a multiple of m, the portion of the T_t matrix corresponding to the seasonal is identity. Here $\lambda_j = (2\pi j)/4$. The distribution of the initial state vector α_1 is diffuse, with $P_* = Diag\left[\sigma_\epsilon^2, 0, 0, 0, 0, 0\right]$ and $P_\infty = Diag\left[0, 1, 1, 1, 1, 1\right]$. The parameter vector in both the cases is $\theta = (\sigma_\epsilon^2, \sigma_\eta^2, \sigma_\xi^2, \sigma_\omega^2)$.

Cycles and Autoregression

The preceding examples have illustrated how to build a state space model corresponding to a UCM that includes components such as irregular, trend, and seasonal. There you can see that the state vector and the system matrices have a simple block structure with blocks corresponding to the components in the model. Therefore, here only a simple model consisting of a single cycle and an irregular component is considered. The state space form for more complex UCMs consisting of multiple cycles and other components can be easily deduced from this example.

Recall that a stochastic cycle ψ_t with frequency λ, $0 < \lambda < \pi$, and damping coefficient ρ can be modeled as

$$\begin{bmatrix} \psi_t \\ \psi_t^* \end{bmatrix} = \rho \begin{bmatrix} \cos\lambda & \sin\lambda \\ -\sin\lambda & \cos\lambda \end{bmatrix} \begin{bmatrix} \psi_{t-1} \\ \psi_{t-1}^* \end{bmatrix} + \begin{bmatrix} \nu_t \\ \nu_t^* \end{bmatrix}$$

where ν_t and ν_t^* are independent, zero-mean, Gaussian disturbances with variance σ_ν^2. In what follows, a state space form for a model consisting of such a stochastic cycle and an irregular component is given.

The state vector $\alpha_t = \left[\begin{array}{ccc} \epsilon_t & \psi_t & \psi_t^* \end{array}\right]'$, and the state noise vector $\zeta_t = \left[\begin{array}{ccc} \epsilon_t & \nu_t & \nu_t^* \end{array}\right]'$. The system matrices are

$$Z = [\,1\,1\,0\,] \quad T = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \rho\cos\lambda & \rho\sin\lambda \\ 0 & -\rho\sin\lambda & \rho\cos\lambda \end{bmatrix} \quad Q = Diag\left[\sigma_\epsilon^2, \sigma_\nu^2, \sigma_\nu^2\right]$$

The distribution of the initial state vector α_1 is proper, with $P_* = Diag\left[\sigma_\epsilon^2, \sigma_\psi^2, \sigma_\psi^2\right]$, where $\sigma_\psi^2 = \sigma_\nu^2(1-\rho^2)^{-1}$. The parameter vector $\theta = (\sigma_\epsilon^2, \rho, \lambda, \sigma_\nu^2)$.

An autoregression r_t can be considered as a special case of cycle with frequency λ equal to 0 or π. In this case the equation for ψ_t^* is not needed. Therefore, for a UCM consisting of an autoregressive component and an irregular component, the state space model simplifies to the following form.

The state vector $\alpha_t = [\ \epsilon_t\ r_t\ ]'$, and the state noise vector $\zeta_t = [\ \epsilon_t\ \nu_t\ ]'$. The system matrices are

$$Z = [\ 1\ 1\ ],\quad T = \begin{bmatrix} 0 & 0 \\ 0 & \rho \end{bmatrix} \text{ and } Q = Diag\left[\sigma_\epsilon^2, \sigma_\nu^2\right]$$

The distribution of the initial state vector α_1 is proper, with $P_* = Diag\left[\sigma_\epsilon^2, \sigma_r^2\right]$, where $\sigma_r^2 = \sigma_\nu^2(1-\rho^2)^{-1}$. The parameter vector $\theta = (\sigma_\epsilon^2, \rho, \sigma_\nu^2)$.

Incorporating Predictors of Different Kinds

In the UCM procedure, predictors can be incorporated in a UCM in a variety of ways: simple time-invariant linear predictors are specified in the MODEL statement, predictors with time-varying coefficients can be specified in the RANDOMREG statement, and predictors that have a nonlinear relationship with the response variable can be specified in the SPLINEREG statement. As with earlier examples, how to obtain a state space form of a UCM consisting of such variety of predictors is illustrated using a simple special case. Consider a random walk trend model with predictors x, u_1, u_2, and v. Let us assume that x is a simple regressor specified in the MODEL statement, u_1 and u_2 are random regressors with time-varying regression coefficients that are specified in the same RANDOMREG statement, and v is a nonlinear regressor specified on a SPLINEREG statement. Let us further assume that the spline associated with v has degree one and is based on two internal knots. As explained in the section "SPLINEREG Statement" on page 2779, using v is equivalent to using $(nknots + degree) = (2+1) = 3$ derived (random) regressors: say, s_1, s_2, s_3. In all there are $(1+2+3) = 6$ regressors, the first one being a simple regressor and the others being time-varying coefficient regressors. The time-varying regressors are in two groups, the first consisting of u_1 and u_2 and the other consisting of s_1, s_2, and s_3. The dynamics of this model are as follows:

$$\begin{aligned}
y_t &= \mu_t + \beta x_t + \kappa_{1t}u_{1t} + \kappa_{2t}u_{2t} + \sum_{i=1}^{3}\gamma_{it}s_{it} + \epsilon_t \\
\mu_t &= \mu_{t-1} + \eta_t \\
\kappa_{1t} &= \kappa_{1(t-1)} + \xi_{1t} \\
\kappa_{2t} &= \kappa_{2(t-1)} + \xi_{2t} \\
\gamma_{1t} &= \gamma_{1(t-1)} + \zeta_{1t} \\
\gamma_{2t} &= \gamma_{2(t-1)} + \zeta_{2t} \\
\gamma_{3t} &= \gamma_{3(t-1)} + \zeta_{3t}
\end{aligned}$$

All the disturbances $\epsilon_t, \eta_t, \xi_{1t}, \xi_{2t}, \zeta_{1t}, \zeta_{2t}$, and ζ_{3t} are independent, zero-mean, Gaussian variables, where ξ_{1t}, ξ_{2t} share a common variance parameter σ_ξ^2 and $\zeta_{1t}, \zeta_{2t}, \zeta_{3t}$ share a common variance σ_ζ^2. These dynamics can be captured in the state space form by taking state $\alpha_t = [\ \epsilon_t\ \mu_t\ \beta\ \kappa_{1t}\ \kappa_{2t}\ \gamma_{1t}\ \gamma_{2t}\ \gamma_{3t}\ ]'$, state

disturbance $\zeta_t = [\ \epsilon_t\ \eta_t\ 0\ \xi_{1t}\ \xi_{2t}\ \zeta_{1t}\ \zeta_{2t}\ \zeta_{3t}\ ]^{'}$, and the system matrices

$$\begin{aligned} Z_t &= [\ 1\ 1\ x_t\ u_{1t}\ u_{2t}\ s_{1t}\ s_{2t}\ s_{3t}\ ] \\ T &= Diag\,[0,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1] \\ Q &= Diag\left[\sigma_\epsilon^2,\ \sigma_\eta^2,\ 0,\ \sigma_\xi^2,\ \sigma_\xi^2,\ \sigma_\zeta^2,\ \sigma_\zeta^2,\ \sigma_\zeta^2\right] \end{aligned}$$

Note that the regression coefficients are elements of the state vector and that the system vector Z_t is not time invariant. The distribution of the initial state vector α_1 is diffuse, with $P_* = Diag\left[\sigma_\epsilon^2, 0, 0, 0, 0, 0, 0, 0\right]$ and $P_\infty = Diag\,[0, 1, 1, 1, 1, 1, 1, 1]$. The parameters of this model are the disturbance variances, σ_ϵ^2, σ_η^2, σ_ξ^2, and σ_ζ^2, which get estimated by maximizing the likelihood. The regression coefficients, time-invariant β and time-varying $\kappa_{1t}, \kappa_{2t}, \gamma_{1t}, \gamma_{2t}$ and γ_{3t}, get implicitly estimated during the state estimation (smoothing).

Reporting Parameter Estimates for Random Regressors

If the random walk disturbance variance associated with a random regressor is held fixed at zero, then its coefficient is no longer time-varying. In the UCM procedure the random regressor parameter estimates are reported differently if the random walk disturbance variance associated with a random regressor is held fixed at zero. The following points explain how the parameter estimates are reported in the parameter estimates table and in the OUTEST= data set.

- If the random walk disturbance variance associated with a random regressor is not held fixed, then its estimate is reported in the parameter estimates table and in the OUTEST= data set.
- If more that one random regressor is specified in a RANDOMREG statement, then the first regressor in the list is used as a representative of the list while reporting the corresponding common variance parameter estimate.
- If the random walk disturbance variance is held fixed at zero, then the parameter estimates table and the OUTEST= data set contain the corresponding regression parameter estimate rather than the variance parameter estimate.
- Similar considerations apply in the case of the derived random regressors associated with a spline-regressor.

Forecasting with Predictor Variables

If regression effects are included in the model (via a MODEL statement or RANDOMREG and SPLINEREG statements) and the FORECAST statement is used to compute multistep forecasts, then future values of the predictor variables must be included in the DATA= data set for the forecast horizon that is defined by the BACK= and LEAD= options. For more information about how the forecast horizon is defined, see the FORECAST statement.

ARMA Irregular Component

The state space form for the irregular component that follows an ARMA$(p,q)\times(P,Q)_s$ model is described in this section. The notation for ARMA models is explained in the IRREGULAR statement. A number of alternate state space forms are possible in this case; the one given here is based on Jones (1980). With slight abuse of notation, let $p = p + sP$ denote the effective autoregressive order and $q = q + sQ$ denote the effective moving average order of the model. Similarly, let ϕ be the effective autoregressive polynomial

and θ be the effective moving average polynomial in the backshift operator with coefficients $\phi_1, \ldots, \phi_p$ and $\theta_1, \ldots, \theta_q$, obtained by multiplying the respective nonseasonal and seasonal factors. Then, a random sequence ϵ_t that follows an ARMA$(p,q)\times(P,Q)_s$ model with a white noise sequence a_t has a state space form with state vector of size $m = \max(p, q+1)$. The system matrices, which are time invariant, are as follows: $Z = [1\ 0\ \ldots\ 0]$. The state transition matrix T, in a blocked form, is given by

$$T = \begin{bmatrix} 0 & I_{m-1} \\ \phi_m & \ldots \quad \phi_1 \end{bmatrix}$$

where $\phi_i = 0$ if $i > p$ and I_{m-1} is an $(m-1)$ dimensional identity matrix. The covariance of the state disturbance matrix $Q = \sigma^2 \psi \psi'$ where σ^2 is the variance of the white noise sequence a_t and the vector $\psi = [\psi_0 \ldots \psi_{m-1}]'$ contains the first m values of the impulse response function—that is, the first m coefficients in the expansion of the ratio θ/ϕ. Since ϵ_t is a stationary sequence, the initial state is nondiffuse and $P_\infty = 0$. The description of P_*, the covariance matrix of the initial state, is a little involved; the details are given in Jones (1980).

Models with Dependent Lags

The state space form of a UCM consisting of the lags of the dependent variable is quite different from the state space forms considered so far. Let us consider an example to illustrate this situation. Consider a model that has random walk trend, two simple time-invariant regressors, and that also includes a few—say, k—lags of the dependent variable. That is,

$$\begin{aligned} y_t &= \sum_{i=1}^{k} \phi_i y_{t-i} + \mu_t + \beta_1 x_{1t} + \beta_2 x_{2t} + \epsilon_t \\ \mu_t &= \mu_{t-1} + \eta_t \end{aligned}$$

The state space form of this augmented model can be described in terms of the state space form of a model that has random walk trend with two simple time-invariant regressors. A superscript dagger (†) has been added to distinguish the augmented model state space entities from the corresponding entities of the state space form of the random walk with predictors model. With this notation, the state vector of the augmented model $\alpha_t^\dagger = \begin{bmatrix} \alpha_t' & y_t & y_{t-1} & \ldots & y_{t-k+1} \end{bmatrix}'$ and the new state noise vector $\zeta_t^\dagger = \begin{bmatrix} \zeta_t' & u_t & 0 \ldots & 0 \end{bmatrix}'$, where u_t is the matrix product $Z_t \zeta_t$. Note that the length of the new state vector is $k + \text{length}(\alpha_t) = k + 4$. The new system matrices, in block form, are

$$Z_t^\dagger = [\,0\,0\,0\,0\,1\ \ldots\ 0\,], \quad T_t^\dagger = \begin{bmatrix} T_t & 0 & \ldots & 0 \\ Z_{t+1} T_t & \phi_1 & \ldots & \phi_k \\ 0 & I_{k-1,k-1} & & 0 \end{bmatrix}$$

where $I_{k-1,k-1}$ is the $k-1$ dimensional identity matrix and

$$Q_t^\dagger = \begin{bmatrix} Q_t & Q_t Z_t' & 0 \\ Z_t Q_t & Z_t Q_t Z_t' & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Note that the T and Q matrices of the random walk with predictors model are time invariant, and in the expressions above their time indices are kept because they illustrate the pattern for more general models. The initial state vector is diffuse, with

$$P_*^\dagger = \begin{bmatrix} P_* & 0 \\ 0 & 0 \end{bmatrix}, \quad P_\infty^\dagger = \begin{bmatrix} P_\infty & 0 \\ 0 & I_{k,k} \end{bmatrix}$$

The parameters of this model are the disturbance variances σ_ϵ^2 and σ_η^2, the lag coefficients $\phi_1, \phi_2, \ldots, \phi_k$, and the regression coefficients β_1 and β_2. As before, the regression coefficients get estimated during the state smoothing, and the other parameters are estimated by maximizing the likelihood.

Outlier Detection

In time series analysis it is often useful to detect changes over time in the characteristics of the response series. In the UCM procedure you can search for two types of changes, additive outliers (AO) and level shifts (LS). An additive outlier is an unusual value in the series, the cause of which might be a data recording error or a temporary shock to the series generation process. A level shift represents a permanent shift, either up or down, in the level of the series. You can control different aspects of the outlier search, such as the significance level of the reported outliers, by choosing different options in the OUTLIER statement. The search for AOs is done by default, whereas the CHECKBREAK option in the LEVEL statement must be used to turn on the search for LSs.

The outlier detection process implemented in the UCM procedure is based on De Jong and Penzer (1998). In this approach the fitted model is taken to be the *null* model, and the series values and level shifts that are not adequately accounted for by the null model are flagged as outliers. The unusualness of a response series value at a particular time point t_0, with respect to the fitted model, can be judged by estimating its value based on the rest of the data (that is, the series obtained by *deleting* the series value at t_0) and comparing the estimated value to the observed value. If the difference between the estimated and observed values is statistically significant, then such value can be regarded as an AO. Note that this difference between the estimated and observed values is also the regression coefficient of a *dummy* regressor that takes the value 1.0 at t_0 and is 0.0 elsewhere, assuming such a regressor is added to the null model. In this way the series value at t_0 is regarded as AO if the regression coefficient of this dummy regressor is significant. Similarly, you can say that a level shift has occurred at a time point t_0 if the regression coefficient of a regressor, which is 0.0 before t_0 and 1.0 at t_0 and thereafter, is statistically significant. De Jong and Penzer (1998) provide an efficient way to compute such AO and LS regression coefficients and their standard errors at all time points in the series. The outlier summary table, which is produced by default, simply lists the most statistically significant candidates among these.

Missing Values

Embedded missing values in the dependent variable usually cause no problems in UCM modeling. However, no missing values are allowed in the predictor variables. Certain patterns of missing values in the dependent variable can lead to failure of the initialization step of the diffuse Kalman filtering for some models. For example, if in a monthly series all values are missing for a certain month—say, May—then a BSM with monthly seasonality leads to such a situation. However, in this case the initialization step can complete successfully for a nonseasonal model such as local linear model.

Parameter Estimation

The parameter vector in a UCM consists of the variances of the disturbance terms of the unobserved components, the damping coefficients and frequencies in the cycles, the damping coefficient in the autoregression,

the lag coefficients of the dependent lags, and the regression coefficients in the regression terms. The regression coefficients are always part of the state vector and are estimated by state smoothing. The remaining parameters are estimated by maximizing either the full diffuse likelihood or the nondiffuse likelihood. The decision to use the full diffuse likelihood or the nondiffuse likelihood depends on the presence or absence of the dependent lag coefficients in the parameter vector. If the parameter vector does not contain any dependent lag coefficients, then the full diffuse likelihood is used. If, on the other hand, the parameter vector does contain some dependent lag coefficients, then the parameters are estimated by maximizing the nondiffuse likelihood. The optimization of the full diffuse likelihood is often unstable when the parameter vector contains dependent lag coefficients. In this sense, when the parameter vector contains dependent lag coefficients, the parameter estimates are not true maximum likelihood estimates.

The optimization of the likelihood, either full or nondiffuse, is carried out using one of several nonlinear optimization algorithms. The user can control many aspects of the optimization process by using the NLOPTIONS statement and by providing the starting values of the parameters while specifying the corresponding components. However, in most cases the default settings work quite well. The optimization process is not guaranteed to converge to a maximum likelihood estimate. In most cases the difficulties in parameter estimation are associated with the specification of a model that is not appropriate for the series being modeled.

Parameter Estimation by Profile Likelihood Optimization

If a disturbance variance, such as the disturbance variance of the irregular component, is a part of the UCM and is a free parameter, then it can be profiled out of the likelihood. This means solving analytically for its optimum and plugging this expression back into the likelihood formula, giving rise to the so-called *profile* likelihood. The expression of the profile likelihood and the MLE of the profiled variance are given earlier in the section "The UCMs as State Space Models" on page 2788, where the computation of the likelihood of the state space model is also discussed.

In some situations the optimization of the profile likelihood can be more efficient because the number of parameters to optimize is reduced by one; however, for a variety of reasons such gains might not always be observed. Moreover, in theory the estimates obtained by optimizing the profile likelihood and the usual likelihood should be the same, but in practice this might not hold because of numerical rounding and other conditions.

In the UCM procedure, by default the usual likelihood is optimized if any of the disturbance variance parameters is held fixed to a nonzero value by using the NOEST option in the corresponding component statement. In other cases the decision whether to optimize the profile likelihood or the usual likelihood is based on several factors that are difficult to document. You can choose which likelihood to optimize during parameter estimation by specifying the PROFILE option for the profile likelihood optimization or the NOPROFILE option for the usual likelihood optimization. In the presence of the PROFILE option, the disturbance variance to profile is checked in a specific order, so that if the irregular component disturbance variance is free then it is always chosen. The situation in other cases is more complicated.

Profiling in the Presence of Fixed Variance Parameters

Note that when the parameter estimation is done by optimizing the profile likelihood, the interpretation of the variance parameters that are held fixed to nonzero values changes. In the presence of the PROFILE option, the disturbance variances that are held at a fixed value by using the NOEST option in their respective component statements are interpreted as being restricted to be that fixed multiple of the profiled variance rather than being fixed at that nominal value. That is, implicitly, the parameter estimation is done under the restriction of holding the disturbance variance *ratio* fixed at a given value rather than the disturbance variance

itself. See Example 41.5 for an example of this type of restriction to obtain a UC model that is equivalent to the famous Hodrick-Prescott filter.

t values

The *t* values reported in the table of parameter estimates are approximations whose accuracy depends on the validity of the model, the nature of the model, and the length of the observed series. The distributional properties of the maximum likelihood estimates of general unobserved components models have not been explored fully; therefore the probability values that correspond to a *t* distribution should be interpreted carefully, as they can be misleading. This is particularly true if the parameters in question are close to the boundary of the parameter space. The two sources by Harvey (1989, 2001) are good references for information about this topic. For some parameters, such as, the cycle period, the reported *t* values are uninformative because comparison of the estimated parameter with zero is never needed. In such cases the *t* values and the corresponding probability values should be ignored.

Bootstrap Prediction Intervals (Experimental)

By default, the UCM procedure computes the standard errors of the series and component forecasts (both the filtered and smoothed estimates) by assuming that the estimated parameters are in fact the true parameters. Rodriguez and Ruiz (2010) describe a bootstrap-based procedure to compute the standard errors of the series and component forecasts that takes into account the uncertainty of parameter estimation. As an experimental feature in this release, you can request the computation of standard errors based on this bootstrap-based procedure by specifying the BOOTSTRAP option in the FORECAST statement. Subsequently, the confidence intervals for the series and component forecasts are based on these bootstrap standard errors. The algorithm that PROC UCM uses closely follows the first procedure described in Section 3 of Rodriguez and Ruiz (2010). Note that this bootstrap algorithm is computationally expensive. The computational burden increases with the number of bootstrap replications and is comparable to the computational burden of fitting the specified model as many times as the number of replications. Fortunately, these replications can be executed in parallel, and the UCM procedure can use multiple cores and multiple grid nodes (if they are available) to complete these calculations faster. For a single machine with multiple cores, the procedure automatically detects and uses all the cores. If a grid environment with multiple machines is available (with the appropriate SAS license), you must use the PERFORMANCE statement to supply the necessary information to the UCM procedure.

Computational Issues

Convergence Problems

As explained in the section "Parameter Estimation" on page 2797, the model parameters are estimated by nonlinear optimization of the likelihood. This process is not guaranteed to succeed. For some data sets, the optimization algorithm can fail to converge. Nonconvergence can result from a number of causes, including flat or ridged likelihood surfaces and ill-conditioned data. It is also possible for the algorithm to converge to a point that is not the global optimum of the likelihood.

If you experience convergence problems, the following points might be helpful:

- Data that are extremely large or extremely small can adversely affect results because of the internal tolerances used during the filtering steps of the likelihood calculation. Rescaling the data can improve stability.
- Examine your model for redundancies in the included components and regressors. If some of the included components or regressors are nearly collinear to each other, then the optimization process can become unstable.
- Experimenting with different options offered by the NLOPTIONS statement can help.
- Lack of convergence can indicate model misspecification or a violation of the normality assumption.

Computer Resource Requirements

The computing resources required for the UCM procedure depend on several factors. The memory requirement for the procedure is largely dependent on the number of observations to be processed and the size of the state vector underlying the specified model. If n denotes the sample size and m denotes the size of the state vector, the memory requirement for the smoothing stage of the Kalman filter is of the order of $6 \times 8 \times n \times m^2$ bytes, ignoring the lower-order terms. If the smoothed component estimates are not needed then the memory requirement is of the order of $6 \times 8 \times (m^2 + n)$ bytes. Besides m and n, the computing time for the parameter estimation depends on the type of components included in the model. For example, the parameter estimation is usually faster if the model parameter vector consists only of disturbance variances, because in this case there is an efficient way to compute the likelihood gradient.

Displayed Output

The default printed output produced by the UCM procedure is described in the following list:

- brief information about the input data set, including the data set name and label, and the name of the ID variable specified in the ID statement
- summary statistics for the data in the estimation and forecast spans, including the names of the variables in the model, their categorization as dependent or predictor, the index of the beginning and ending observations in the spans, the total number of observations and the number of missing observations, the smallest and largest measurements, and the mean and standard deviation
- information about the model parameters at the start of the model-fitting stage, including the fixed parameters in the model and the initial estimates of the free parameters in the model
- convergence status of the likelihood optimization process if any parameter estimation is done
- estimates of the free parameters at the end of the model fitting-stage, including the parameter estimates, their approximate standard errors, t statistics, and the approximate p-value
- the likelihood-based goodness-of-fit statistics, including the full likelihood, the portion of the likelihood corresponding to the diffuse initialization, the sum of squares of residuals normalized by their standard errors, and the information criteria: AIC, AICC, HQIC, BIC, and CAIC

- the fit statistics that are based on the raw residuals (observed minus predicted), including the mean squared error (MSE), the root mean squared error (RMSE), the mean absolute percentage error (MAPE), the maximum percentage error (MAXPE), the R square, the adjusted R square, the random walk R square, and Amemiya's R square
- the significance analysis of the components included in the model that is based on the estimation span
- brief information about the components included in the model
- additive outliers in the series, if any are detected
- the multistep series forecasts
- post-sample-prediction analysis table that compares the multistep forecasts with the observed series values, if the BACK= option is used in the FORECAST statement

Statistical Graphics

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*).

Before you create graphs, ODS Graphics must be enabled (for example, with the ODS GRAPHICS ON statement). For more information about enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" in that chapter.

The overall appearance of graphs is controlled by ODS styles. Styles and other aspects of using ODS Graphics are discussed in the section "A Primer on ODS Statistical Graphics" in that chapter.

This section provides information about the basic ODS statistical graphics produced by the UCM procedure.

You can obtain most plots relevant to the specified model by using the global PLOTS= option in the PROC UCM statement. The plot of series forecasts in the forecast horizon is produced by default. You can further control the production of individual plots by using the PLOT= options in the different statements.

The main types of plots available are as follows:

- Time series plots of the component estimates, either filtered or smoothed, can be requested by using the PLOT= option in the respective component statements. For example, the use of PLOT=SMOOTH option in a CYCLE statement produces a plot of smoothed estimate of that cycle.
- Residual plots for model diagnostics can be obtained by using the PLOT= option in the ESTIMATE statement.
- Plots of series forecasts and model decompositions can be obtained by using the PLOT= option in the FORECAST statement.

The following example is a simple illustration of the available plot options.

Analysis of Sunspot Data: Illustration of ODS Graphics

In this example a well-known series, Wolfer's sunspot data (Anderson 1971), is considered. The data consist of yearly sunspot numbers recorded from 1749 to 1924. These sunspot numbers are known to have a cyclical pattern with a period of about eleven years. The following DATA step creates the input data set:

```
data sunspot;
   input year wolfer @@;
   year = mdy(1,1, year);
   format year year4.;
datalines;
1749  809 1750  834 1751  477 1752  478 1753  307 1754  122 1755   96
1756  102 1757  324 1758  476 1759  540 1760  629 1761  859 1762  612
1763  451 1764  364 1765  209 1766  114 1767  378 1768  698 1769 1061

   ... more lines ...
```

The following statements specify a UCM that includes a cycle component and a random walk trend component:

```
proc ucm data=sunspot;
   id year interval=year;
   model wolfer;
   irregular;
   level ;
   cycle plot=(filter smooth);
   estimate back=24 plot=(loess panel cusum wn);
   forecast back=24 lead=24 plot=(forecasts decomp);
run;
```

The following subsections explain the graphics produced by the above statements.

Component Plots

The plots in Figure 41.8 and Figure 41.9, produced by specifying PLOT=(FILTER SMOOTH) in the CYCLE statement, show the filtered and smoothed estimates, respectively, of the cycle component in the model. Note that the smoothed estimate appears smoother than the filtered estimate. This is always true because the filtered estimate of a component at time t is based on the observations prior to time t—that is, it uses measurements from the first observation up to the $(t-1)$th observation. On the other hand, the corresponding smoothed estimate uses all the available observations—that is, all the measurements from the first observation to the last. This makes the smoothed estimate of the component more precise than the filtered estimate for the time points within historical period. In the forecast horizon, both filtered and smoothed estimates are identical, being based on the same set of observations.

Figure 41.8 Sunspots Series: Filtered Cycle

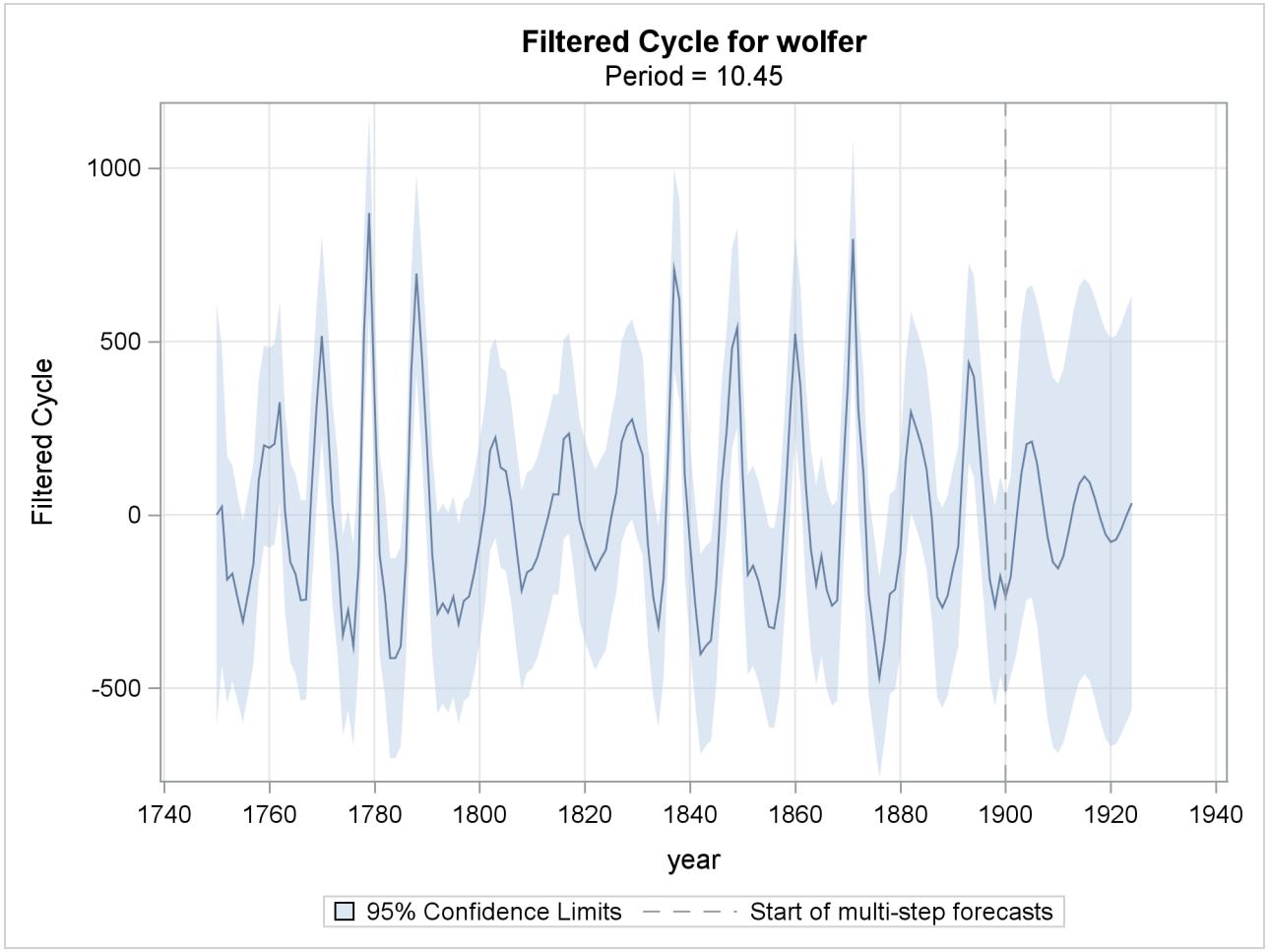
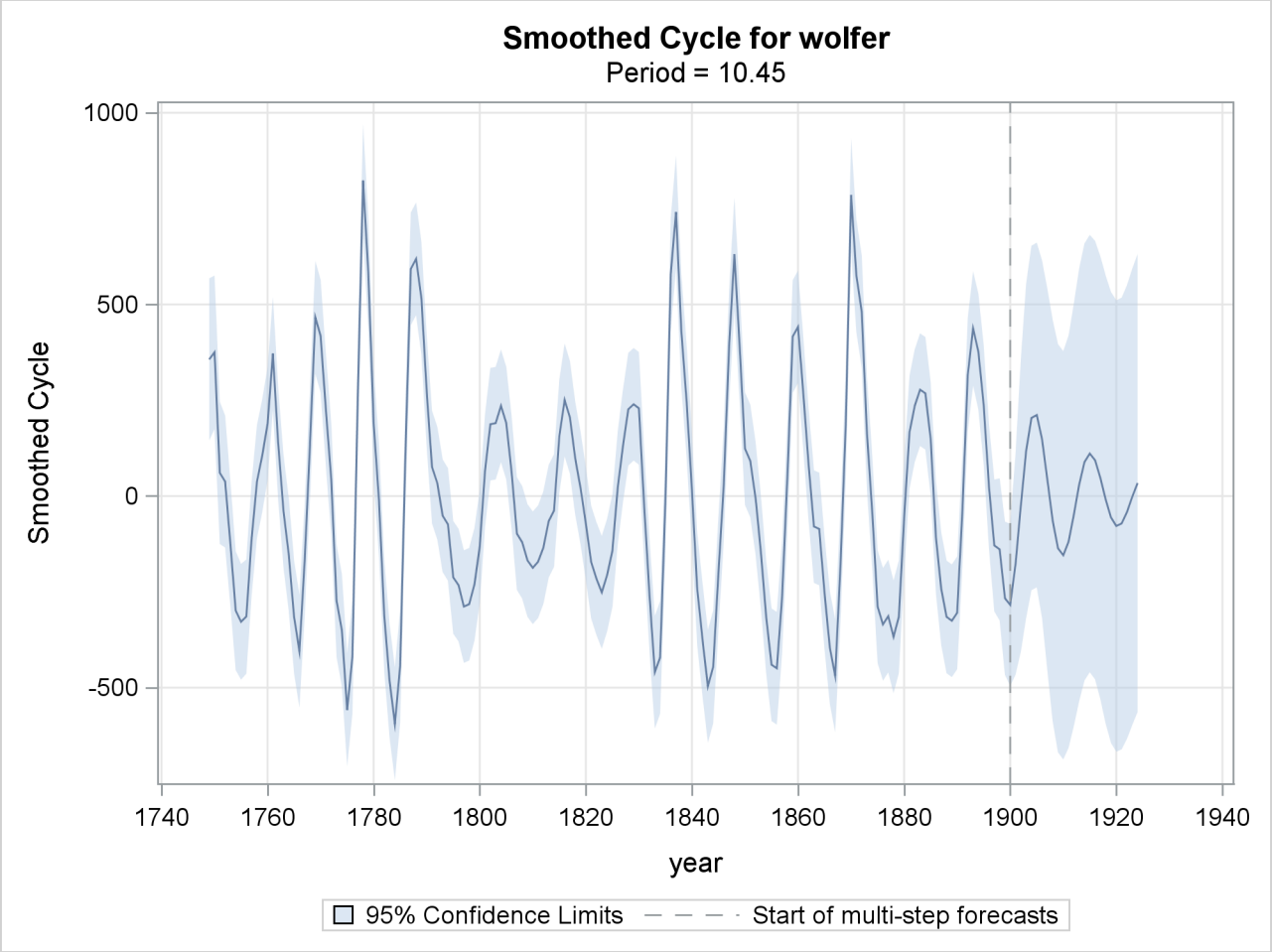


Figure 41.9 Sunspots Series: Smoothed Cycle

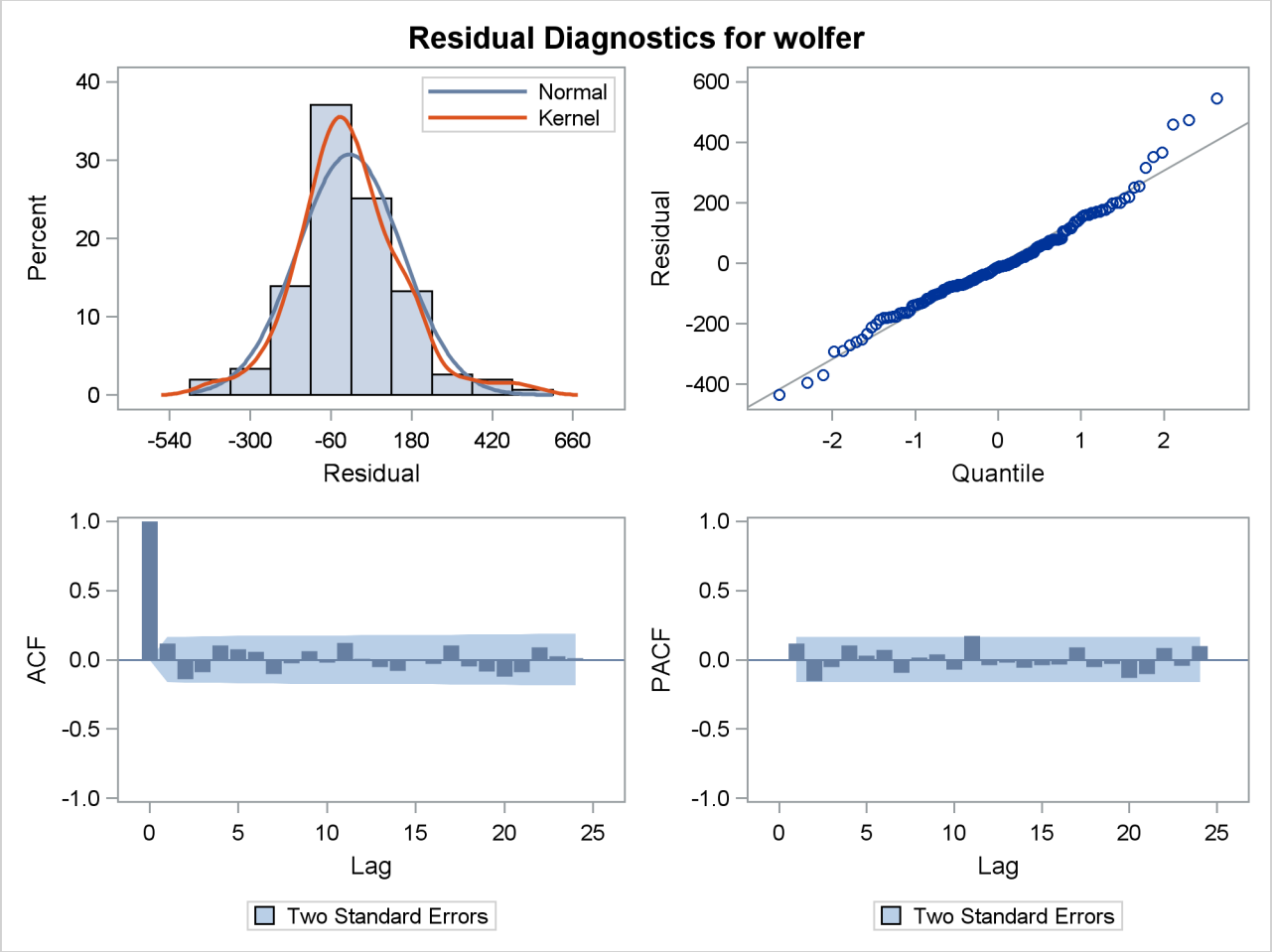


Residual Diagnostics

If the fitted model is appropriate for the given data, then the corresponding one-step-ahead residuals should be approximately *white*—that is, uncorrelated—and approximately normal. Moreover, the residuals should not display any discernible pattern. You can detect departures from these conditions graphically. Different residual diagnostic plots can be requested by using the PLOT= option in the ESTIMATE statement.

The normality can be checked by examining the histogram and the normal quantile plot of residuals. The whiteness can be checked by examining the ACF and PACF plots that show the sample autocorrelation and sample partial-autocorrelation at different lags. The diagnostic panel shown in Figure 41.10, produced by specifying PLOT=PANEL, contains these four plots.

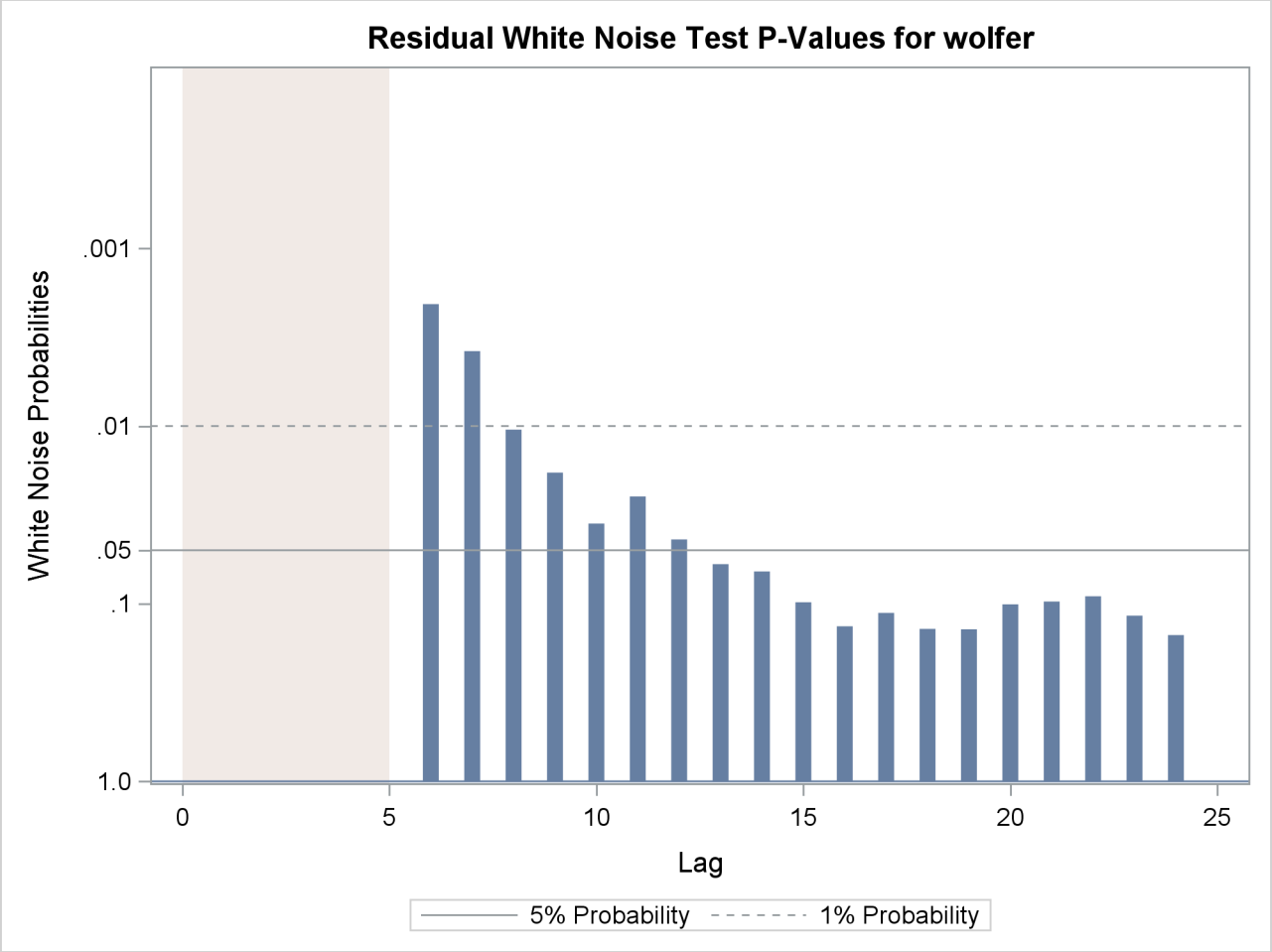
Figure 41.10 Sunspots Series: Residual Diagnostics



The residual histogram and Q-Q plot show no serious violation of normality. The histogram appears reasonably symmetric and follows the overlaid normal density curve reasonably closely. Similarly in the Q-Q plot the residuals follow the reference line fairly closely. The ACF and PACF plots also do not exhibit any violation of the whiteness assumption; the correlations at all nonzero lags seem to be insignificant.

The residual whiteness can also be formally tested by using the Ljung-Box portmanteau test. The plot in Figure 41.11, produced by specifying PLOT=WN, shows the p-values of the Ljung-Box test statistics at different lags. In these plots the p-values for the first few lags, equal to the number of estimated parameters in the model, are not shown because they are always missing. This portion of the plot is shaded blue to indicate this fact. In the case of this model, five parameters are estimated so the p-values for the first five lags are not shown. The p-values are displayed on a log scale in such a way that higher bars imply more extreme test statistics. In this plot some early p-values appear extreme. However, these p-values are based on large sample theory, which suggests that these statistics should be examined for lags larger than the square root of sample size. In this example it means that the p-values for the first $\sqrt{154} \approx 12$ lags can be ignored. With this consideration, the plot shows no violation of whiteness since the p-values after the 12th lag do not appear extreme.

Figure 41.11 Sunspots Series: Ljung-Box Portmanteau Test



The plot in Figure 41.12, produced by specifying PLOT=LOESS, shows the residuals plotted against time with an overlaid LOESS curve. This plot is useful for checking whether any discernible pattern remains in the residuals. Here again, no significant pattern appears to be present.

Figure 41.12 Sunspots Series: Residual Loess Plot



The plot in Figure 41.13, produced by specifying PLOT=CUSUM, shows the cumulative residuals plotted against time. This plot is useful for checking structural breaks. Here, there appears to be no evidence of structural break since the cumulative residuals remain within the confidence band throughout the sample period. Similarly you can request a plot of the squared cumulative residuals by specifying PLOT=CUSUMSQ.

Figure 41.13 Sunspots Series: CUSUM Plot



Brockwell and Davis (1991) can be consulted for additional information on diagnosing residuals. For more information about CUSUM and CUSUMSQ plots, you can consult Harvey (1989).

Forecast and Series Decomposition Plots

You can use the PLOT= option in the FORECAST statement to obtain the series forecast plot and the series decomposition plots. The series decomposition plots show the result of successively adding different components in the model starting with the trend component. The IRREGULAR component is left out of this process. The following two plots, produced by specifying PLOT=DECOMP, show the results of successive component addition for this example. The first plot, shown in Figure 41.14, shows the smoothed trend component and the second plot, shown in Figure 41.15, shows the sum of smoothed trend and cycle.

Figure 41.14 Sunspots Series: Smoothed Trend

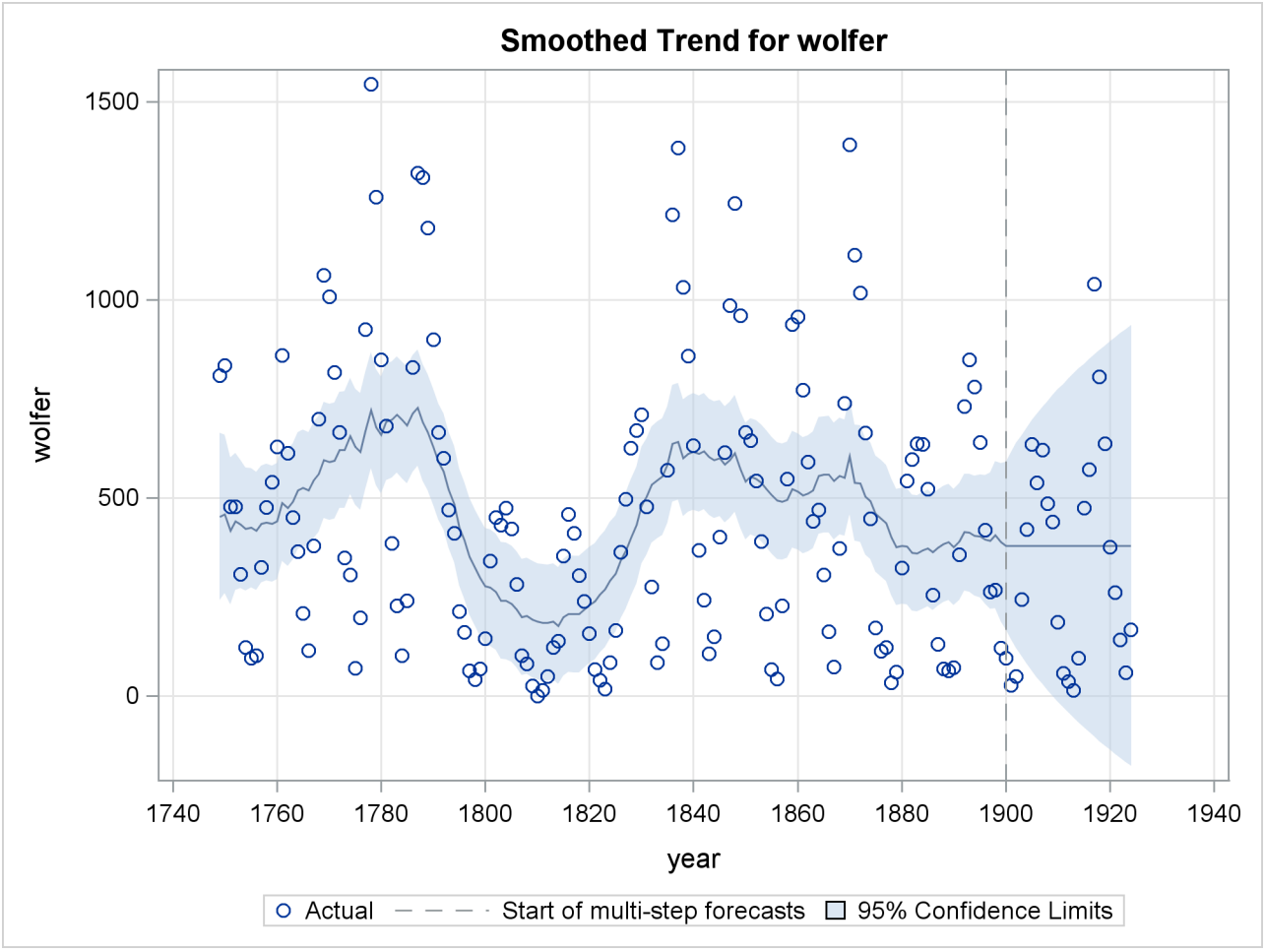
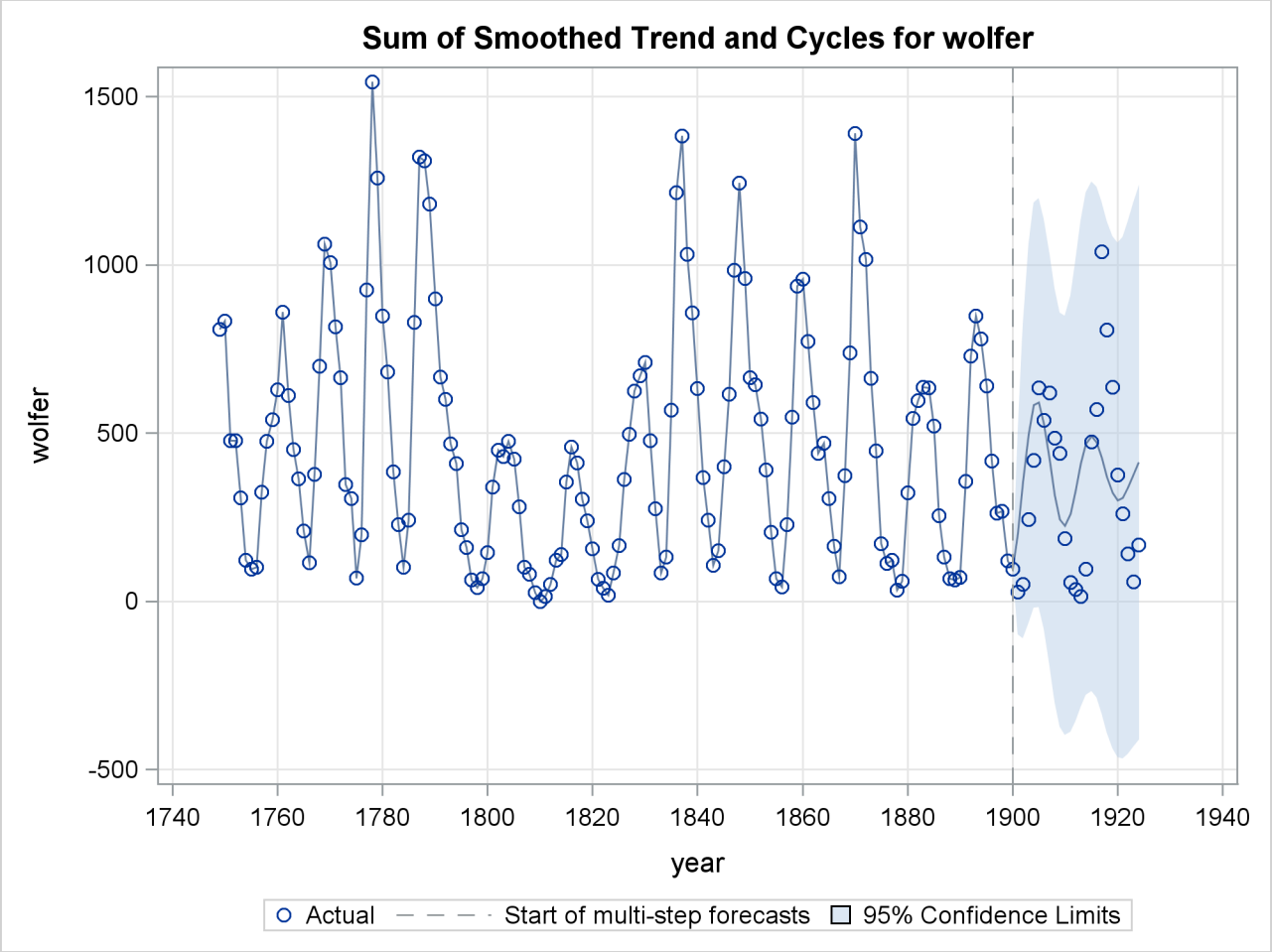
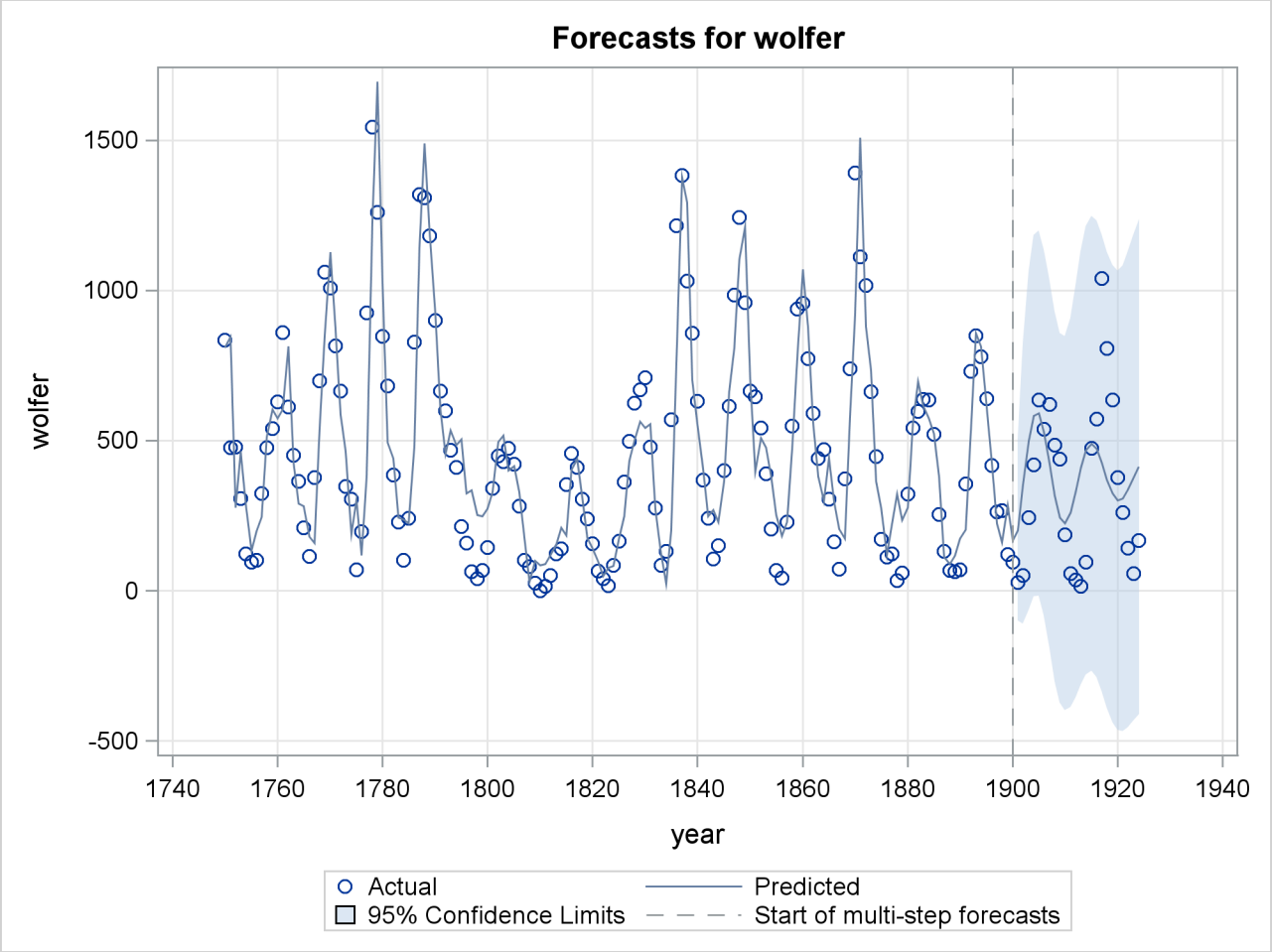


Figure 41.15 Sunspots Series: Smoothed Trend plus Cycle



Finally, Figure 41.16 shows the forecast plot.

Figure 41.16 Sunspots Series: Series Forecasts



ODS Table Names

The UCM procedure assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in Table 41.2.

Table 41.2 ODS Tables Produced by PROC UCM

ODS Table Name	Description	Statement	Option
Tables Summarizing the Estimation and Forecast Spans			
EstimationSpan	Estimation span summary information		default
ForecastSpan	Forecast span summary information		default

Table 41.2 *continued*

ODS Table Name	Description	Statement	Option
Tables Related to Model Parameters			
ConvergenceStatus	Convergence status of the estimation process		default
FixedParameters	Fixed parameters in the model		default
InitialParameters	Initial estimates of the free parameters		default
ParameterEstimates	Final estimates of the free parameters		default
Tables Related to Model Information and Diagnostics			
BlockSeasonDescription	Information about the block seasonals in the model		default
ComponentSignificance	Significance analysis of the components in the model		default
CycleDescription	Information about the cycles in the model		default
FitStatistics	Fit statistics based on the one-step-ahead predictions		default
FitSummary	Likelihood-based fit statistics		default
OutlierSummary	Summary table of the detected outliers		default
SeasonDescription	Information about the seasonals in the model		default
SeasonHarmonics	Summary of harmonics in a trigonometric seasonal component	SEASON	PRINT=HARMONICS
SplineSeasonDescription	Information about the spline-seasonals in the model		default
TrendInformation	Summary information of the level and slope components		default
Tables Related to Filtered Component Estimates			
FilteredAutoReg	Filtered estimate of an autoreg component	AUTOREG	PRINT=FILTER
FilteredBlockSeason	Filtered estimate of a block seasonal component	BLOCKSEASON	PRINT=FILTER
FilteredCycle	Filtered estimate of a cycle component	CYCLE	PRINT=FILTER
FilteredIrregular	Filtered estimate of the irregular component	IRREGULAR	PRINT=FILTER
FilteredLevel	Filtered estimate of the level component	LEVEL	PRINT=FILTER

Table 41.2 *continued*

ODS Table Name	Description	Statement	Option
FilteredRandomReg	Filtered estimate of the time-varying random-regression coefficient	RANDOMREG	PRINT=FILTER
FilteredSeason	Filtered estimate of a seasonal component	SEASON	PRINT=FILTER
FilteredSlope	Filtered estimate of the slope component	SLOPE	PRINT=FILTER
FilteredSplineReg	Filtered estimate of the time-varying spline-regression coefficient	SPLINEREG	PRINT=FILTER
FilteredSplineSeason	Filtered estimate of a spline-seasonal component	SPLINESEASON	PRINT=FILTER
Tables Related to Smoothed Component Estimates			
SmoothedAutoReg	Smoothed estimate of an autoreg component	AUTOREG	PRINT=SMOOTH
SmoothedBlockSeason	Smoothed estimate of a block seasonal component	BLOCKSEASON	PRINT=SMOOTH
SmoothedCycle	Smoothed estimate of the cycle component	CYCLE	PRINT=SMOOTH
SmoothedIrregular	Smoothed estimate of the irregular component	IRREGULAR	PRINT=SMOOTH
SmoothedLevel	Smoothed estimate of the level component	LEVEL	PRINT=SMOOTH
SmoothedRandomReg	Smoothed estimate of the time-varying random-regression coefficient	RANDOMREG	PRINT=SMOOTH
SmoothedSeason	Smoothed estimate of a seasonal component	SEASON	PRINT=SMOOTH
SmoothedSlope	Smoothed estimate of the slope component	SLOPE	PRINT=SMOOTH
SmoothedSplineReg	Smoothed estimate of the time-varying spline-regression coefficient	SPLINEREG	PRINT=SMOOTH
SmoothedSplineSeason	Smoothed estimate of a spline-seasonal component	SPLINESEASON	PRINT=SMOOTH
Tables Related to Series Decomposition and Forecasting			
FilteredAllExceptIrreg	Filtered estimate of sum of all components except the irregular component	FORECAST	PRINT=FDECOMP
FilteredTrend	Filtered estimate of trend	FORECAST	PRINT= FDECOMP

Table 41.2 *continued*

ODS Table Name	Description	Statement	Option
FilteredTrendReg	Filtered estimate of trend plus regression	FORECAST	PRINT=FDECOMP
FilteredTrendRegCyc	Filtered estimate of trend plus regression plus cycles and autoreg	FORECAST	PRINT=FDECOMP
Forecasts	Dependent series forecasts		default
PostSamplePrediction	Forecasting performance in the holdout period	FORECAST	BACK=
SmoothedAllExceptIrreg	Smoothed estimate of sum of all components except the irregular component	FORECAST	PRINT=DECOMP
SmoothedTrend	Smoothed estimate of trend	FORECAST	PRINT= DECOMP
SmoothedTrendReg	Smoothed estimate of trend plus regression	FORECAST	PRINT=DECOMP
SmoothedTrendRegCyc	Smoothed estimate of trend plus regression plus cycles and autoreg	FORECAST	PRINT=DECOMP

NOTE: The tables are related to a single series within a BY group. In the case of models that contain multiple cycles, seasonal components, or block seasonal components, the corresponding component estimate tables are sequentially numbered. For example, if a model contains two cycles and a seasonal component and the PRINT=SMOOTH option is used for each of them, the ODS tables containing the smoothed estimates will be named SmoothedCycle1, SmoothedCycle2, and SmoothedSeason. Note that the seasonal table is not numbered because there is only one seasonal component. There are some exceptions to this numbering rule: the tables, FilteredRandomReg, SmoothedRandomReg, FilteredSplineReg, and SmoothedSplineReg, are always numbered starting with zero.

ODS Graph Names

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*).

Before you create graphs, ODS Graphics must be enabled (for example, with the ODS GRAPHICS ON statement). For more information about enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" in that chapter.

The overall appearance of graphs is controlled by ODS styles. Styles and other aspects of using ODS Graphics are discussed in the section "A Primer on ODS Statistical Graphics" in that chapter.

You can reference every graph produced through ODS Graphics with a name. The names of the graphs that PROC UCM generates are listed in Table 41.3, along with the required statements and options.

Table 41.3 ODS Graphics Produced by PROC UCM

ODS Graph Name	Description	Statement	Option
Plots Related to Residual Analysis			
ErrorACFPlot	Prediction error autocorrelation plot	ESTIMATE	PLOT=ACF
ErrorPACFPlot	Prediction error partial-autocorrelation plot	ESTIMATE	PLOT=PACF
ErrorHistogram	Prediction error histogram	ESTIMATE	PLOT=NORMAL
ErrorQQPlot	Prediction error normal quantile plot	ESTIMATE	PLOT=QQ
ErrorPlot	Plot of prediction errors	ESTIMATE	PLOT=RESIDUAL
ErrorWhiteNoiseLogProbPlot	Plot of p-values at different lags for the Ljung-Box portmanteau white noise test statistics	ESTIMATE	PLOT=WN
CUSUMPlot	Plot of cumulative residuals	ESTIMATE	PLOT=CUSUM
CUSUMSQPlot	Plot of cumulative squared residuals	ESTIMATE	PLOT=CUSUMSQ
ModelPlot	Plot of one-step-ahead forecasts in the estimation span	ESTIMATE	PLOT=MODEL
PanelResidualPlot	Panel of residual diagnostic plots	ESTIMATE	PLOT=PANEL
ResidualLoessPlot	Time series plot of residuals with superimposed LOESS smoother	ESTIMATE	PLOT=LOESS
Plots Related to Filtered Component Estimates			
FilteredAutoregPlot	Plot of filtered autoreg component	AUTOREG	PLOT=FILTER
FilteredBlockSeasonPlot	Plot of filtered block season component	BLOCKSEASON	PLOT=FILTER
FilteredCyclePlot	Plot of filtered cycle component	CYCLE	PLOT=FILTER
FilteredIrregularPlot	Plot of filtered irregular component	IRREGULAR	PLOT=FILTER
FilteredLevelPlot	Plot of filtered level component	LEVEL	PLOT=FILTER
FilteredRandomRegPlot	Plot of filtered time-varying regression coefficient	RANDOMREG	PLOT=FILTER
FilteredSeasonPlot	Plot of filtered season component	SEASON	PLOT=FILTER
FilteredSlopePlot	Plot of filtered slope component	SLOPE	PLOT=FILTER
FilteredSplineRegPlot	Plot of filtered time-varying regression coefficient	SPLINEREG	PLOT=FILTER

Table 41.3 *continued*

ODS Graph Name	Description	Statement	Option
FilteredSplineSeasonPlot	Plot of filtered spline-season component	SPLINESEASON	PLOT=FILTER
AnnualSeasonPlot	Plot of annual variation in the filtered season component	SEASON	PLOT=F_ANNUAL
Plots Related to Smoothed Component Estimates			
SmoothedAutoregPlot	Plot of smoothed autoreg component	AUTOREG	PLOT=SMOOTH
SmoothedBlockSeasonPlot	Plot of smoothed block season component	BLOCKSEASON	PLOT=SMOOTH
SmoothedCyclePlot	Plot of smoothed cycle component	CYCLE	PLOT=SMOOTH
SmoothedIrregularPlot	Plot of smoothed irregular component	IRREGULAR	PLOT=SMOOTH
SmoothedLevelPlot	Plot of smoothed level component	LEVEL	PLOT=SMOOTH
SmoothedRandomRegPlot	Plot of smoothed time-varying regression coefficient	RANDOMREG	PLOT=SMOOTH
SmoothedSeasonPlot	Plot of smoothed season component	SEASON	PLOT=SMOOTH
SmoothedSlopePlot	Plot of smoothed slope component	SLOPE	PLOT=SMOOTH
SmoothedSplineRegPlot	Plot of smoothed time-varying regression coefficient	SPLINEREG	PLOT=SMOOTH
SmoothedSplineSeasonPlot	Plot of smoothed spline-season component	SPLINESEASON	PLOT=SMOOTH
AnnualSeasonPlot	Plot of annual variation in the smoothed season component	SEASON	PLOT=S_ANNUAL
Plots Related to Series Decomposition and Forecasting			
ForecastsOnlyPlot	Series forecasts beyond the historical period	FORECAST	DEFAULT
ForecastsPlot	One-step-ahead as well as multistep-ahead forecasts	FORECAST	PLOT=FORECASTS
FilteredAllExceptIrregPlot	Plot of sum of all filtered components except the irregular component	FORECAST	PLOT= FDECOMP
FilteredTrendPlot	Plot of filtered trend	FORECAST	PLOT= FDECOMP

Table 41.3 *continued*

ODS Graph Name	Description	Statement	Option
FilteredTrendRegCycPlot	Plot of sum of filtered trend, cycles, and regression effects	FORECAST	PLOT= FDECOMP
FilteredTrendRegPlot	Plot of filtered trend plus regression effects	FORECAST	PLOT= FDECOMP
SmoothedAllExceptIrregPlot	Plot of sum of all smoothed components except the irregular component	FORECAST	PLOT= DECOMP
SmoothedTrendPlot	Plot of smoothed trend	FORECAST	PLOT= TREND
SmoothedTrendRegPlot	Plot of smoothed trend plus regression effects	FORECAST	PLOT= DECOMP
SmoothedTrendRegCycPlot	Plot of sum of smoothed trend, cycles, and regression effects	FORECAST	PLOT= DECOMP
FilteredAllExceptIrregVarPlot	Plot of standard error of sum of all filtered components except the irregular	FORECAST	PLOT= FDECOMPVAR
FilteredTrendVarPlot	Plot of standard error of filtered trend	FORECAST	PLOT= FDECOMPVAR
FilteredTrendRegVarPlot	Plot of standard error of filtered trend plus regression effects	FORECAST	PLOT= FDECOMPVAR
FilteredTrendRegCycVarPlot	Plot of standard error of filtered trend, cycles, and regression effects	FORECAST	PLOT= FDECOMPVAR
SmoothedAllExceptIrregVarPlot	Plot of standard error of sum of all smoothed components except the irregular	FORECAST	PLOT= DECOMPVAR
SmoothedTrendVarPlot	Plot of standard error of smoothed trend	FORECAST	PLOT= DECOMPVAR
SmoothedTrendRegVarPlot	Plot of standard error of smoothed trend plus regression effects	FORECAST	PLOT= DECOMPVAR
SmoothedTrendRegCycVarPlot	Plot of standard error of smoothed trend, cycles, and regression effects	FORECAST	PLOT= DECOMPVAR

OUTFOR= Data Set

You can use the OUTFOR= option in the FORECAST statement to store the series and component forecasts produced by the procedure. This data set contains the following columns:

- the BY variables
- the ID variable. If an ID variable is not specified, then a numerical variable, _ID_, is created that contains the observation numbers from the input data set.
- the dependent series and the predictor series
- FORECAST, a numerical variable containing the one-step-ahead predicted values and the multistep forecasts
- RESIDUAL, a numerical variable containing the difference between the actual and forecast values
- STD, a numerical variable containing the standard error of prediction
- LCL and UCL, numerical variables containing the lower and upper forecast confidence limits
- S_SERIES and VS_SERIES, numerical variables containing the smoothed values of the dependent series and their variances
- S_IRREG and VS_IRREG, numerical variables containing the smoothed values of the irregular component and their variances. These variables are present only if the model has an irregular component.
- F_LEVEL, VF_LEVEL, S_LEVEL, and VS_LEVEL, numerical variables containing the filtered and smoothed values of the level component and the respective variances. These variables are present only if the model has a level component.
- F_SLOPE, VF_SLOPE, S_SLOPE, and VS_SLOPE, numerical variables containing the filtered and smoothed values of the slope component and the respective variances. These variables are present only if the model has a slope component.
- F_AUTOREG, VF_AUTOREG, S_AUTOREG, and VS_AUTOREG, numerical variables containing the filtered and smoothed values of the autoreg component and the respective variances. These variables are present only if the model has an autoreg component.
- F_CYCLE, VF_CYCLE, S_CYCLE, and VS_CYCLE, numerical variables containing the filtered and smoothed values of the cycle component and the respective variances. If there are multiple cycles in the model, these variables are sequentially numbered as F_CYCLE1, F_CYCLE2, etc. These variables are present only if the model has at least one cycle component.
- F_SEASON, VF_SEASON, S_SEASON, and VS_SEASON, numerical variables containing the filtered and smoothed values of the season component and the respective variances. If there are multiple seasons in the model, these variables are sequentially numbered as F_SEASON1, F_SEASON2, etc. These variables are present only if the model has at least one season component.
- F_BLKSEAS, VF_BLKSEAS, S_BLKSEAS, and VS_BLKSEAS, numerical variables containing the filtered and smoothed values of the blockseason component and the respective variances. If there are multiple block seasons in the model, these variables are sequentially numbered as F_BLKSEAS1, F_BLKSEAS2, etc.

- F_SPLSEAS, VF_SPLSEAS, S_SPLSEAS, and VS_SPLSEAS, numerical variables containing the filtered and smoothed values of the splineseason component and the respective variances. If there are multiple spline seasons in the model, these variables are sequentially numbered as F_SPLSEAS1, F_SPLSEAS2, etc. These variables are present only if the model has at least one splineseason component.
- Filtered and smoothed estimates, and their variances, of the time-varying regression coefficients of the variables specified in the RANDOMREG and SPLINEREG statements. A variable is not included if its coefficient is time-invariant, that is, if the associated disturbance variance is zero.
- S_TREG and VS_TREG, numerical variables containing the smoothed values of level plus regression component and their variances. These variables are present only if the model has at least one predictor variable or has dependent lags.
- S_TREGCYC and VS_TREGCYC, numerical variables containing the smoothed values of level plus regression plus cycle component and their variances. These variables are present only if the model has at least one cycle or an autoreg component.
- S_NOIRREG and VS_NOIRREG, numerical variables containing the smoothed values of the sum of all components except the irregular component and their variances. These variables are present only if the model has at least one seasonal or block seasonal component.

OUTEST= Data Set

You can use the OUTEST= option in the ESTIMATE statement to store the model parameters and the related estimation details. This data set contains the following columns:

- the BY variables
- COMPONENT, a character variable containing the name of the component corresponding to the parameter being described
- PARAMETER, a character variable containing the parameter name
- TYPE, a character variable indicating whether the parameter value was fixed by the user or estimated
- _STATUS_, a character variable indicating whether the parameter estimation process converged or failed or there was an error of some other kind
- ESTIMATE, a numerical variable containing the parameter estimate
- STD, a numerical variable containing the standard error of the parameter estimate. This has a missing value if the parameter value is fixed.
- TVALUE, a numerical variable containing the t-statistic. This has a missing value if the parameter value is fixed.
- PVALUE, a numerical variable containing the p-value. This has a missing value if the parameter value is fixed.

Statistics of Fit

This section explains the goodness-of-fit statistics reported to measure how well the specified model fits the data.

First the various statistics of fit that are computed using the prediction errors, $y_t - \hat{y}_t$, are considered. In these formulas, *n* is the number of nonmissing prediction errors and *k* is the number of fitted parameters in the model. Moreover, the sum of squared errors, $SSE = \sum (y_t - \hat{y}_t)^2$, and the total sum of squares for the series corrected for the mean, $SST = \sum (y_t - \overline{y})^2$, where $\overline{y}$ is the series mean, and the sums are over all the nonmissing prediction errors.

Mean Squared Error
The mean squared prediction error, $MSE = \frac{1}{n} SSE$

Root Mean Squared Error
The root mean square error, RMSE = $\sqrt{MSE}$

Mean Absolute Percent Error
The mean absolute percent prediction error, MAPE = $\frac{100}{n} \sum_{t=1}^{n} |(y_t - \hat{y}_t)/y_t|$.
The summation ignores observations where $y_t = 0$.

R-square
The R-square statistic, $R^2 = 1 - SSE/SST$.
If the model fits the series badly, the model error sum of squares, *SSE*, might be larger than *SST* and the R-square statistic will be negative.

Adjusted R-square
The adjusted R-square statistic, $1 - (\frac{n-1}{n-k})(1 - R^2)$

Amemiya's Adjusted R-square
Amemiya's adjusted R-square, $1 - (\frac{n+k}{n-k})(1 - R^2)$

Random Walk R-square
The random walk R-square statistic (Harvey's R-square statistic that uses the random walk model for comparison), $1 - (\frac{n-1}{n})SSE/RWSSE$, where $RWSSE = \sum_{t=2}^{n} (y_t - y_{t-1} - \mu)^2$, and $\mu = \frac{1}{n-1} \sum_{t=2}^{n} (y_t - y_{t-1})$

Maximum Percent Error
The largest percent prediction error, $100 \max((y_t - \hat{y}_t)/y_t)$. In this computation the observations where $y_t = 0$ are ignored.

The likelihood-based fit statistics are reported separately (see the section "The UCMs as State Space Models" on page 2788). They include the full log likelihood (L_∞), the diffuse part of the log likelihood, the normalized residual sum of squares, and several information criteria: AIC, AICC, HQIC, BIC, and CAIC. Let *q* denote the number of estimated parameters, *n* be the number of nonmissing measurements in the estimation span, and *d* be the number of diffuse elements in the initial state vector that are successfully initialized during the Kalman filtering process. Moreover, let $n^* = (n - d)$. The reported information criteria, all in smaller-is-better form, are described in Table 41.4:

Table 41.4 Information Criteria

Criterion	Formula	Reference
AIC	$-2L_\infty + 2q$	Akaike (1974)
AICC	$-2L_\infty + 2qn^*/(n^* - q - 1)$	Hurvich and Tsai (1989)
		Burnham and Anderson (1998)
HQIC	$-2L_\infty + 2q \log\log(n^*)$	Hannan and Quinn (1979)
BIC	$-2L_\infty + q \log(n^*)$	Schwarz (1978)
CAIC	$-2L_\infty + q(\log(n^*) + 1)$	Bozdogan (1987)

Examples: UCM Procedure

Example 41.1: The Airline Series Revisited

The series in this example, the monthly airline passenger series, has already been discussed earlier; see the section "A Seasonal Series with Linear Trend" on page 2745. Recall that the series consists of monthly numbers of international airline travelers (from January 1949 to December 1960). Here additional output features of the UCM procedure are illustrated, such as how to use the ESTIMATE and FORECAST statements to limit the span of the data used in parameter estimation and forecasting. The following statements fit a BSM to the logarithm of the airline passenger numbers. The disturbance variance for the slope component is held fixed at value 0; that is, the trend is locally linear with constant slope. In order to evaluate the performance of the fitted model on observed data, some of the observed data are withheld during parameter estimation and forecast computations. The observations in the last two years, years 1959 and 1960, are not used in parameter estimation, while the observations in the last year, year 1960, are not used in the forecasting computations. This is done using the BACK= option in the ESTIMATE and FORECAST statements. In addition, a panel of residual diagnostic plots is obtained using the PLOT=PANEL option in the ESTIMATE statement.

```
data seriesG;
   set sashelp.air;
   logair = log(air);
run;

proc ucm data = seriesG;
   id date interval = month;
   model logair;
   irregular;
   level;
   slope var = 0 noest;
   season length = 12 type=trig;
   estimate back=24 plot=panel;
   forecast back=12 lead=24 print=forecasts;
run;
```

The following tables display the summary of data used in estimation and forecasting (Output 41.1.1 and Output 41.1.2). These tables provide simple summary statistics for the estimation and forecast spans; they include useful information such as the beginning and ending dates of the span, the number of nonmissing values, etc.

Output 41.1.1 Observation Span Used in Parameter Estimation (partial output)

Variable	Type	First	Last	Nobs	Mean
logair	Dependent	JAN1949	DEC1958	120	5.43035

Output 41.1.2 Observation Span Used in Forecasting (partial output)

Variable	Type	First	Last	Nobs	Mean
logair	Dependent	JAN1949	DEC1959	132	5.48654

The following tables display the fixed parameters in the model, the preliminary estimates of the free parameters, and the final estimates of the free parameters (Output 41.1.3, Output 41.1.4, and Output 41.1.5).

Output 41.1.3 Fixed Parameters in the Model

The UCM Procedure

Fixed Parameters in the Model		
Component	**Parameter**	**Value**
Slope	**Error Variance**	0

Output 41.1.4 Starting Values for the Parameters to Be Estimated

Preliminary Estimates of the Free Parameters		
Component	**Parameter**	**Estimate**
Irregular	**Error Variance**	6.64120
Level	**Error Variance**	2.49045
Season	**Error Variance**	1.26676

Output 41.1.5 Maximum Likelihood Estimates of the Free Parameters

Final Estimates of the Free Parameters					
Component	**Parameter**	**Estimate**	**Approx Std Error**	**t Value**	**Approx Pr > |t|**
Irregular	**Error Variance**	0.00018686	0.0001212	1.54	0.1233
Level	**Error Variance**	0.00040314	0.0001566	2.57	0.0100
Season	**Error Variance**	0.00000350	1.66319E-6	2.10	0.0354

Two types of goodness-of-fit statistics are reported after a model is fit to the series (see Output 41.1.6 and Output 41.1.7). The first type is the likelihood-based goodness-of-fit statistics, which include the full likelihood of the data, the diffuse portion of the likelihood (see the section "Details: UCM Procedure" on page 2782), and the information criteria. The second type of statistics is based on the raw residuals, residual = observed – predicted. If the model is nonstationary, then one-step-ahead predictions are not available for some initial observations, and the number of values used in computing these fit statistics will be different from those used in computing the likelihood-based test statistics.

Output 41.1.6 Likelihood-Based Fit Statistics for the Airline Data

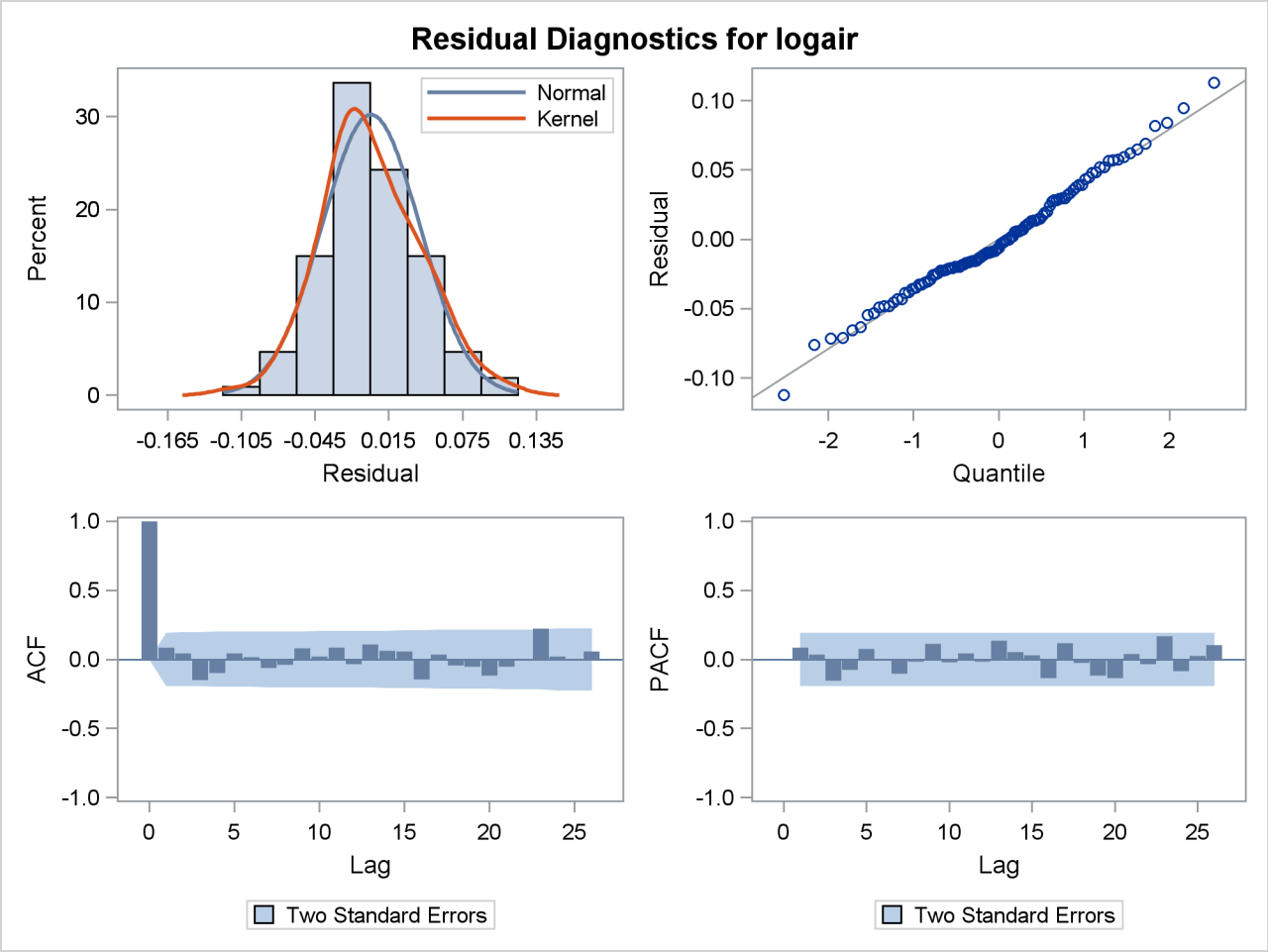
Likelihood Based Fit Statistics	
Statistic	**Value**
Full Log Likelihood	180.63
Diffuse Part of Log Likelihood	-13.93
Non-Missing Observations Used	120
Estimated Parameters	3
Initialized Diffuse State Elements	13
Normalized Residual Sum of Squares	107
AIC (smaller is better)	-355.3
BIC (smaller is better)	-347.2
AICC (smaller is better)	-355
HQIC (smaller is better)	-352
CAIC (smaller is better)	-344.2

Output 41.1.7 Residuals-Based Fit Statistics for the Airline Data

Fit Statistics Based on Residuals	
Mean Squared Error	0.00156
Root Mean Squared Error	0.03944
Mean Absolute Percentage Error	0.57677
Maximum Percent Error	2.19396
R-Square	0.98705
Adjusted R-Square	0.98680
Random Walk R-Square	0.86370
Amemiya's Adjusted R-Square	0.98630
Number of non-missing residuals used for computing the fit statistics = 107	

The diagnostic plots based on the one-step-ahead residuals are shown in Output 41.1.8. The residual histogram and the Q-Q plot show no reasons to question the approximate normality of the residual distribution. The remaining plots check for the *whiteness* of the residuals. The sample correlation plots, the autocorrelation function (ACF) and the partial autocorrelation function (PACF), also do not show any significant violations of the whiteness of the residuals. Therefore, on the whole, the model seems to fit the data well.

Output 41.1.8 Residual Diagnostics for the Airline Series Using a BSM



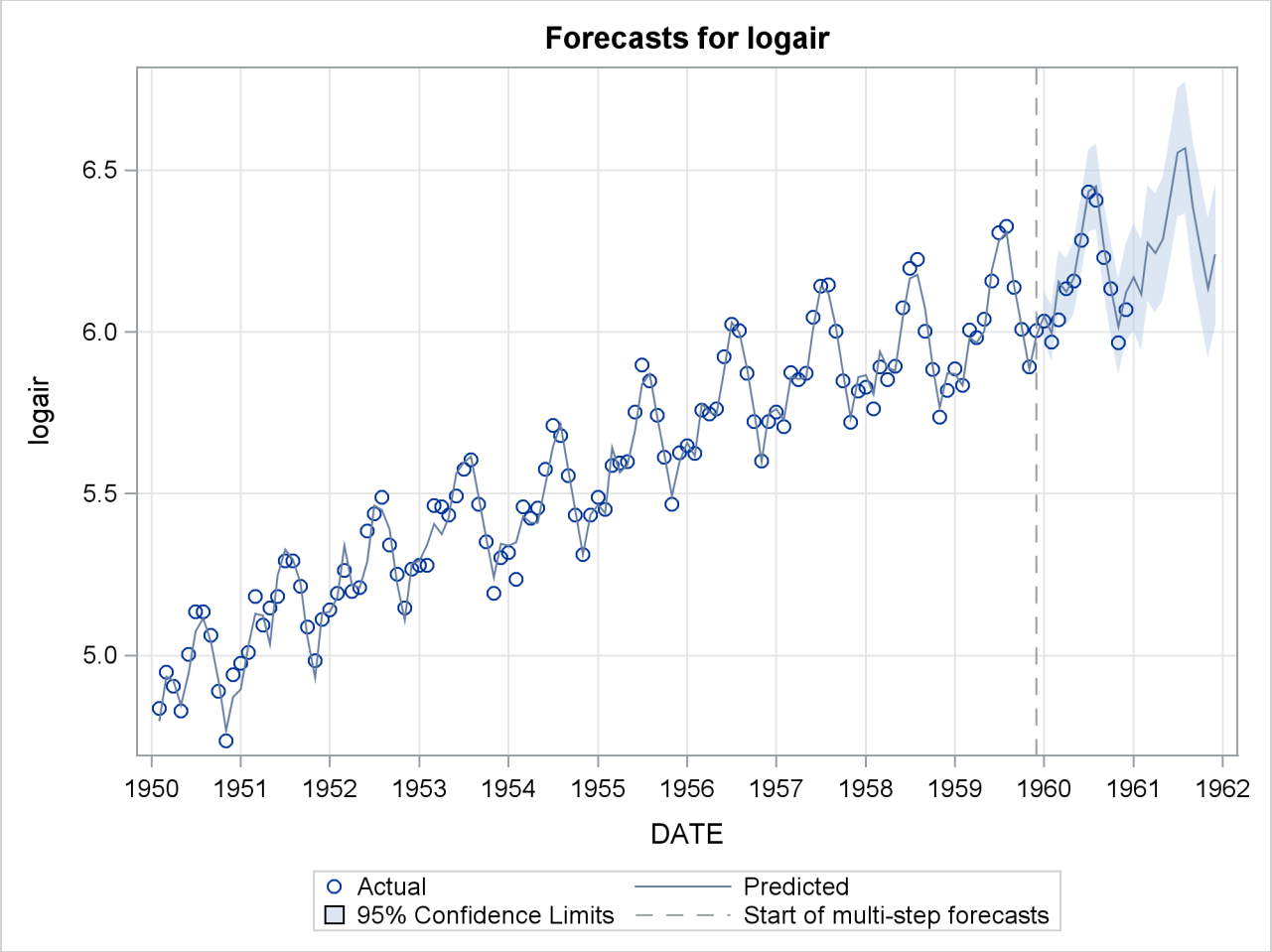
The forecasts are given in Output 41.1.9. In order to save the space, the upper and lower confidence limit columns are dropped from the output, and only the rows corresponding to the year 1960 are shown. Recall that the actual measurements in the years 1959 and 1960 were withheld during the parameter estimation, and the ones in 1960 were not used in the forecast computations.

Output 41.1.9 Forecasts for the Airline Data

Obs	date	Forecast	StdErr	logair	Residual
133	JAN60	6.050	0.038	6.033	-0.017
134	FEB60	5.996	0.044	5.969	-0.027
135	MAR60	6.156	0.049	6.038	-0.118
136	APR60	6.124	0.053	6.133	0.010
137	MAY60	6.168	0.058	6.157	-0.011
138	JUN60	6.303	0.061	6.282	-0.021
139	JUL60	6.435	0.065	6.433	-0.002
140	AUG60	6.450	0.068	6.407	-0.043
141	SEP60	6.265	0.071	6.230	-0.035
142	OCT60	6.138	0.073	6.133	-0.005
143	NOV60	6.015	0.075	5.966	-0.049
144	DEC60	6.121	0.077	6.068	-0.053

The figure Output 41.1.10 shows the forecast plot. The forecasts in the year 1960 show that the model predictions were quite good.

Output 41.1.10 Forecast Plot of the Airline Series Using a BSM



Example 41.2: Variable Star Data

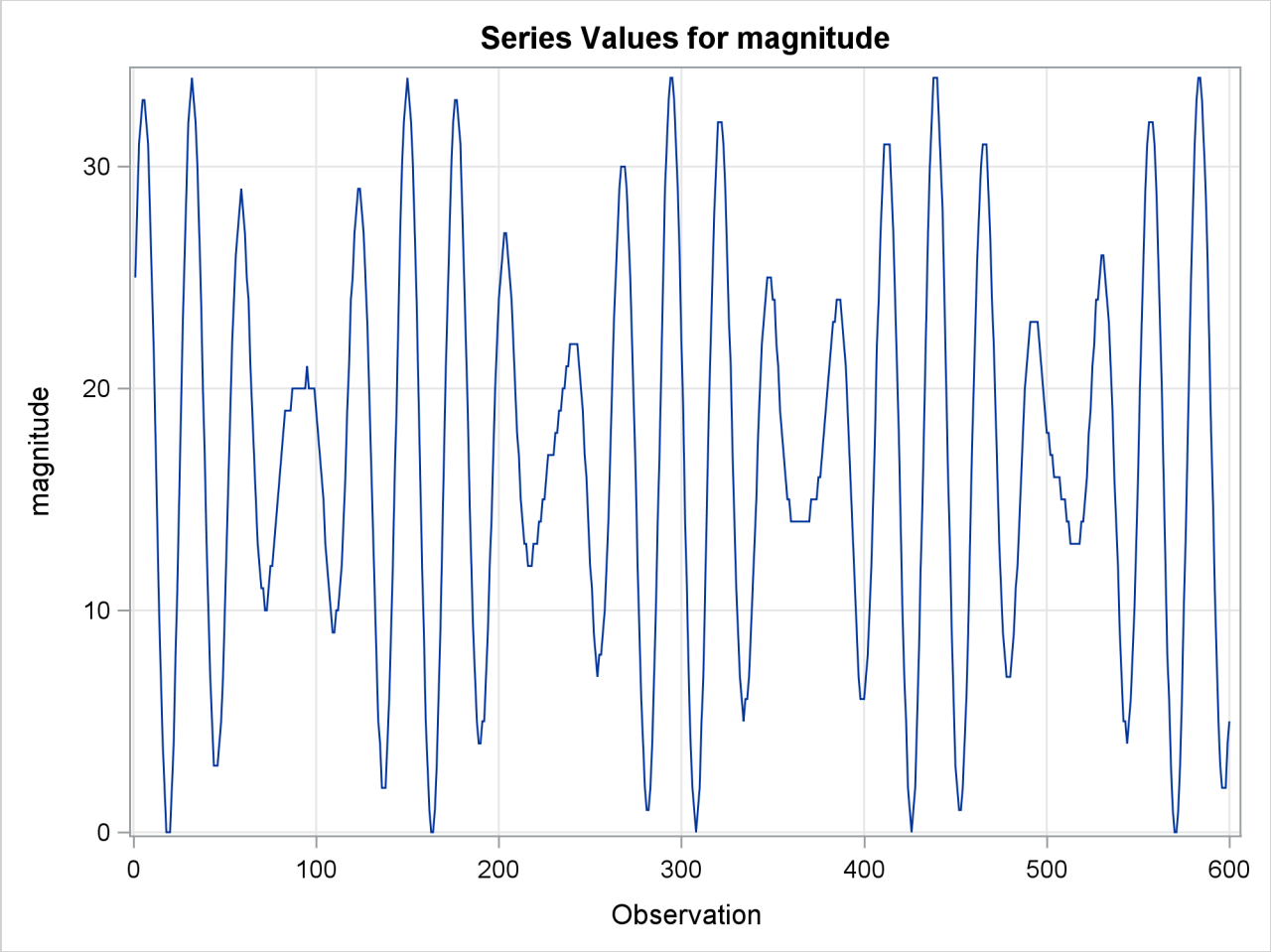
The series in this example is studied in detail in Bloomfield (2000). This series consists of brightness measurements (magnitude) of a variable star taken at midnight for 600 consecutive days. The data can be downloaded from a time series archive maintained by the University of York, England (http://www.york.ac.uk/depts/maths/data/ts/welcome.htm (series number 26)). The following DATA step statements read the data in a SAS data set.

```
data star;
   input magnitude @@;
   day = _n_;
datalines;
  25  28  31  32  33  33  32  31  28   25   22   18
  14  10   7   4   2   0   0   0   2    4    8   11
  15  19  23  26  29  32  33  34  33   32   30   27
  24  20  17  13  10   7   5   3   3    3    4    5
   7  10  13  16  19  22  24  26  27   28   29   28
  27  25  24  21  19  17  15  13  12   11   11   10
  10  11  12  12  13  14  15  16  17   18   19   19

   ... more lines ...
```

The following statements use the TIMESERIES procedure to get a timeseries plot of the series (see Output 41.2.1).

```
proc timeseries data=star plot=series;
   var magnitude;
run;
```

Output 41.2.1 Plot of Star Brightness on Successive Days



The plot clearly shows the cyclic nature of the series. Bloomfield shows that the series is very well explained by a model that includes two deterministic cycles that have periods 29.0003 and 24.0001 days, a constant term, and a simple error term. He also mentions the difficulty involved in estimating the periods from the data (Bloomfield 2000, Chapter 3). In his case the cycle periods are estimated by least squares, and the sum of squares surface has multiple local optima and ridges. The following statements show how to use the UCM procedure to fit this two-cycle model to the series. The constant term in the model is specified by holding the variance parameter of the level component to zero.

```
proc ucm data=star;
   model magnitude;
   irregular;
   level var=0 noest;
   cycle;
   cycle;
   estimate;
run;
```

The final parameter estimates and the goodness-of-fit statistics are shown in Output 41.2.2 and Output 41.2.3, respectively. The model fit appears to be good.

Output 41.2.2 Two-Cycle Model: Parameter Estimates

The UCM Procedure

Final Estimates of the Free Parameters

Component	Parameter	Estimate	Approx Std Error	t Value	Approx Pr > |t|
Irregular	Error Variance	0.09257	0.0053845	17.19	<.0001
Cycle_1	Damping Factor	1.00000	1.81175E-7	5519514	<.0001
Cycle_1	Period	29.00036	0.0022709	12770.4	<.0001
Cycle_1	Error Variance	0.00000882	5.27213E-6	1.67	0.0944
Cycle_2	Damping Factor	1.00000	2.11939E-7	4718334	<.0001
Cycle_2	Period	24.00011	0.0019128	12547.2	<.0001
Cycle_2	Error Variance	0.00000535	3.56374E-6	1.50	0.1330

Output 41.2.3 Two-Cycle Model: Goodness of Fit

Fit Statistics Based on Residuals	
Mean Squared Error	0.12072
Root Mean Squared Error	0.34745
Mean Absolute Percentage Error	2.65141
Maximum Percent Error	36.38991
R-Square	0.99850
Adjusted R-Square	0.99849
Random Walk R-Square	0.97281
Amemiya's Adjusted R-Square	0.99847
Number of non-missing residuals used for computing the fit statistics = 599	

A summary of the cycles in the model is given in Output 41.2.4.

Output 41.2.4 Two-Cycle Model: Summary

Name	Type	period	Rho	ErrorVar
Cycle_1	Stationary	29.00036	1.00000	0.00000882
Cycle_2	Stationary	24.00011	1.00000	0.00000535

Note that the estimated periods are the same as in Bloomfield's model, the damping factors are nearly equal to 1.0, and the disturbance variances are very close to zero, implying persistent deterministic cycles. In fact, this model is identical to Bloomfield's model.

Example 41.3: Modeling Long Seasonal Patterns

This example illustrates some of the techniques you can use to model long seasonal patterns in a series. If the seasonal pattern is of moderate length and the underlying dynamics are simple, then it is easily modeled by using the basic settings of the SEASON statement and these additional techniques are not needed. However, if the seasonal pattern has a long season length and/or has a complex stochastic dynamics, then the techniques discussed here can be useful. You can obtain parsimonious models for a long seasonal pattern by using an appropriate subset of trigonometric harmonics, or by using a suitable spline function, or by using a block-season pattern in combination with a seasonal component of much smaller length. You can also vary the disturbance variances of the subcomponents that combine to form the seasonal component.

The time series used in this example consists of number of calls received per shift at a call center. Each shift is six hours long, and the first shift of the day begins at midnight, resulting in four shifts per day. The observations are available from December 15, 1999, to April 30, 2000. This series is seasonal with season length 28, which is moderate, and in fact there is no particular need to use pattern approximation techniques in this case. However, it is adequate for demonstration purposes. The plan of this example is as follows. First an initial model with a full seasonal component is created. This model is used as a baseline for comparing alternate models created by the techniques that are being illustrated. In practice any candidate model is first checked for adequacy by using various diagnostic procedures. In this illustration the main focus is on the different ways a long seasonal pattern can be modeled and no model diagnostics are done for the models being entertained. The alternate models are compared by using the sum of absolute prediction errors in the holdout region.

The following DATA step statements create the input data set used in this example.

```
data callCenter;
   input calls @@;
   label calls= "Number of Calls Received in a 6 Hour Shift";
   start = '15dec99:00:00'dt;
   datetime = INTNX( 'dthour6', start, _n_-1 );
   format datetime datetime10.;
datalines;
  18    122    244    128     19    113    230    119     17    112
 219     93     14     73    139     53     11     32     74     56
  15    137    289    153     20    125    227    106     16    101
 201     92     14     94    187     69     11     59     94     21

   ... more lines ...
```

Initial exploration of the series clearly indicates that the series does not show any significant trend, and time of day and day of the week have a significant influence on the number of calls received. These considerations suggest a simple random walk trend model along with a seasonal component of season length 28, the total number of shifts in a week. The following statements specify this model. Note the PRINT=HARMONICS option in the SEASON statement, which produces a table that lists the full set of harmonics contributing to the seasonal along with the significance of their contribution. This table will be useful later in choosing a

subset trigonometric model. The BACK=28 and the LEAD=28 specifications in the FORECAST statement create a holdout region of 28 observations. The sum of absolute prediction errors (SAE) in this holdout region are used to compare the different models.

```
proc ucm data=callCenter;
   id datetime interval=dthour6;
   model calls;
   irregular;
   level;
   season length=28 type=trig
      print=(harmonics);
   estimate back=28;
   forecast back=28 lead=28;
run;
```

The forecasting performance of this model in the holdout region is shown in Output 41.3.1. The sum of absolute prediction errors SAE = 516.22, which appears in the last row of the holdout analysis table.

Output 41.3.1 Predictions in the Holdout Region: Baseline Model

Obs	datetime	Actual	Forecast	Error	SAE
525	24APR00:00	12	-4.004	16.004	16.004
526	24APR00:06	136	110.825	25.175	41.179
527	24APR00:12	295	262.820	32.180	73.360
528	24APR00:18	172	145.127	26.873	100.232
529	25APR00:00	20	2.188	17.812	118.044
530	25APR00:06	127	105.442	21.558	139.602
531	25APR00:12	236	217.043	18.957	158.559
532	25APR00:18	125	114.313	10.687	169.246
533	26APR00:00	16	2.855	13.145	182.391
534	26APR00:06	108	95.202	12.798	195.189
535	26APR00:12	207	194.184	12.816	208.005
536	26APR00:18	112	97.687	14.313	222.317
537	27APR00:00	15	1.270	13.730	236.047
538	27APR00:06	98	85.875	12.125	248.172
539	27APR00:12	200	184.891	15.109	263.281
540	27APR00:18	113	93.113	19.887	283.168
541	28APR00:00	15	-1.120	16.120	299.288
542	28APR00:06	104	84.983	19.017	318.305
543	28APR00:12	205	177.940	27.060	345.365
544	28APR00:18	89	64.292	24.708	370.073
545	29APR00:00	12	-6.020	18.020	388.093
546	29APR00:06	68	46.286	21.714	409.807
547	29APR00:12	116	100.339	15.661	425.468
548	29APR00:18	54	34.700	19.300	444.768
549	30APR00:00	10	-6.209	16.209	460.978
550	30APR00:06	30	12.167	17.833	478.811
551	30APR00:12	66	49.524	16.476	495.287
552	30APR00:18	61	40.071	20.929	516.216

*

Now that a baseline model is created, the exploration for alternate models can begin. The review of the harmonic table in Output 41.3.2 shows that all but the last three harmonics are significant, and deleting any of them to form a subset trigonometric seasonal component will lead to a poorer model. The last three harmonics, 12th, 13th and 14th, with periods of 2.333, 2.15 and 2.0, respectively, do appear to be possible choices for deletion. Note that the disturbance variance of the seasonal component is not very insignificant (see Output 41.3.3); therefore the seasonal component is stochastic and the preceding logic, which is based on the final state estimate, provides only a rough guideline.

Output 41.3.2 Harmonic Analysis of the Season: Initial Model

The UCM Procedure

Harmonic Analysis of Trigonometric Seasons (Based on the Final State)

Name	Season Length	Harmonic	Period	Chi-Square	DF	Pr > ChiSq
Season	28	1	28.00000	234.19	2	<.0001
Season	28	2	14.00000	264.19	2	<.0001
Season	28	3	9.33333	95.65	2	<.0001
Season	28	4	7.00000	105.64	2	<.0001
Season	28	5	5.60000	146.74	2	<.0001
Season	28	6	4.66667	121.93	2	<.0001
Season	28	7	4.00000	4299.12	2	<.0001
Season	28	8	3.50000	150.79	2	<.0001
Season	28	9	3.11111	89.68	2	<.0001
Season	28	10	2.80000	8.95	2	0.0114
Season	28	11	2.54545	6.14	2	0.0464
Season	28	12	2.33333	2.20	2	0.3325
Season	28	13	2.15385	3.40	2	0.1828
Season	28	14	2.00000	2.33	1	0.1272

Output 41.3.3 Parameter Estimates: Initial Model

Final Estimates of the Free Parameters

Component	Parameter	Estimate	Approx Std Error	t Value	Approx Pr > |t|
Irregular	Error Variance	92.14591	13.10986	7.03	<.0001
Level	Error Variance	44.83595	10.65465	4.21	<.0001
Season	Error Variance	0.01250	0.0065153	1.92	0.0551

The following statements fit a subset trigonometric model formed by dropping the last three harmonics by specifying the DROPH= option in the SEASON statement:

```
proc ucm data=callCenter;
   id datetime interval=dthour6;
   model calls;
   irregular;
   level;
   season length=28 type=trig droph=12 13 14;
   estimate back=28;
   forecast back=28 lead=28;
run;
```

The last row of the holdout region prediction analysis table for the preceding model is shown in Output 41.3.4. It shows that the subset trigonometric model has better prediction performance in the holdout region than the full trigonometric model, its SAE = 471.53 compared to the SAE = 516.22 for the full model.

Output 41.3.4 SAE for the Subset Trigonometric Model

Obs	datetime	Actual	Forecast	Error	SAE
552	30APR00:18	61	40.836	20.164	471.534

The following statements illustrate a spline approximation to this seasonal component. In the spline specification the knot placement is quite important, and usually some experimentation is needed. In the following model the knots are placed at the beginning and the middle of each day. Note that the knots at the beginning and end of the season, 1 and 28 in this case, should not be listed in the knot list because knots are always placed there anyway.

```
proc ucm data=callCenter;
   id datetime interval=dthour6;
   model calls;
   irregular;
   level;
   splineseason length=28
      knots=3 5 7 9 11 13 15 17 19 21 23 25 27
      degree=3;
   estimate back=28;
   forecast back=28 lead=28;
run;
```

The spline season model takes about half the time to fit that the baseline model takes. The last row of the holdout region prediction analysis table for this model is shown in Output 41.3.5, which shows that the spline season model performs even better than the previous two models in the holdout region, its SAE = 313.79 compared to SAE = 471.53 for the previous model.

Output 41.3.5 SAE for the Spline Season Model

Obs	datetime	Actual	Forecast	Error	SAE
552	30APR00:18	61	23.350	37.650	313.792

The following statements illustrate yet another way to approximate a long seasonal component. Here a combination of BLOCKSEASON and SEASON statements results in a seasonal component that is a sum of two seasonal patterns: one seasonal pattern is simply a regular season with season length 4 that captures the *within-day* seasonal pattern, and the other seasonal pattern is a block seasonal pattern that remains constant during the day but varies from day to day within a week. Note the use of NLOPTIONS statement to change the optimization technique during the parameter estimation to DBLDOG, which in this case performs better than the default technique, TRUREG.

```
proc ucm data=callCenter;
   id datetime interval=dthour6;
   model calls;
   irregular;
   level;
   season length=4 type=trig;
   blockseason nblocks=7 blocksize=4
      type=trig;
   estimate back=28;
   forecast back=28 lead=28;
   nloptions tech=dbldog;
run;
```

This model also takes about half the time to fit that the baseline model takes. The last row of the holdout region prediction analysis table for this model is shown in Output 41.3.6, which shows that the block season model does slightly better than the baseline model but not as good as the other two models, its SAE = 508.52 compared to the SAE = 516.22 of the baseline model.

Output 41.3.6 SAE for the Block Season Model

Obs	datetime	Actual	Forecast	Error	SAE
552	30APR00:18	61	39.339	21.661	508.522

This example showed a few different ways to model a long seasonal pattern. It showed that parsimonious models for long seasonal patterns can be useful, and in some cases even more effective than the full model. Moreover, for very long seasonal patterns the high memory requirements and long computing times might make full models impractical.

Example 41.4: Modeling Time-Varying Regression Effects

In April 1979 the Albuquerque Police Department began a special enforcement program aimed at reducing the number of DWI (driving while intoxicated) accidents. The program was administered by a squad of police officers, who used breath alcohol testing (BAT) devices and a van that houses a BAT device (Batmobile). These data were collected by the Division of Governmental Research of the University of New Mexico, under a contract with the National Highway Traffic Safety Administration of the U.S. Department of Transportation, to evaluate the Batmobile program. The first 29 observations are for a control period, and the next 23 observations are for the experimental (Batmobile) period. The data, freely available at http://lib.stat.cmu.edu/DASL/Datafiles/batdat.html, consist of two variables: ACC, which represents injuries and fatalities from Wednesday to Saturday nighttime accidents, and FUEL, which represents fuel consumption (millions of gallons) in Albuquerque. The variables are measured quarterly starting from the first quarter of 1972 up to the last quarter of 1984, covering the span of 13 years. The following DATA step statements create the input data set.

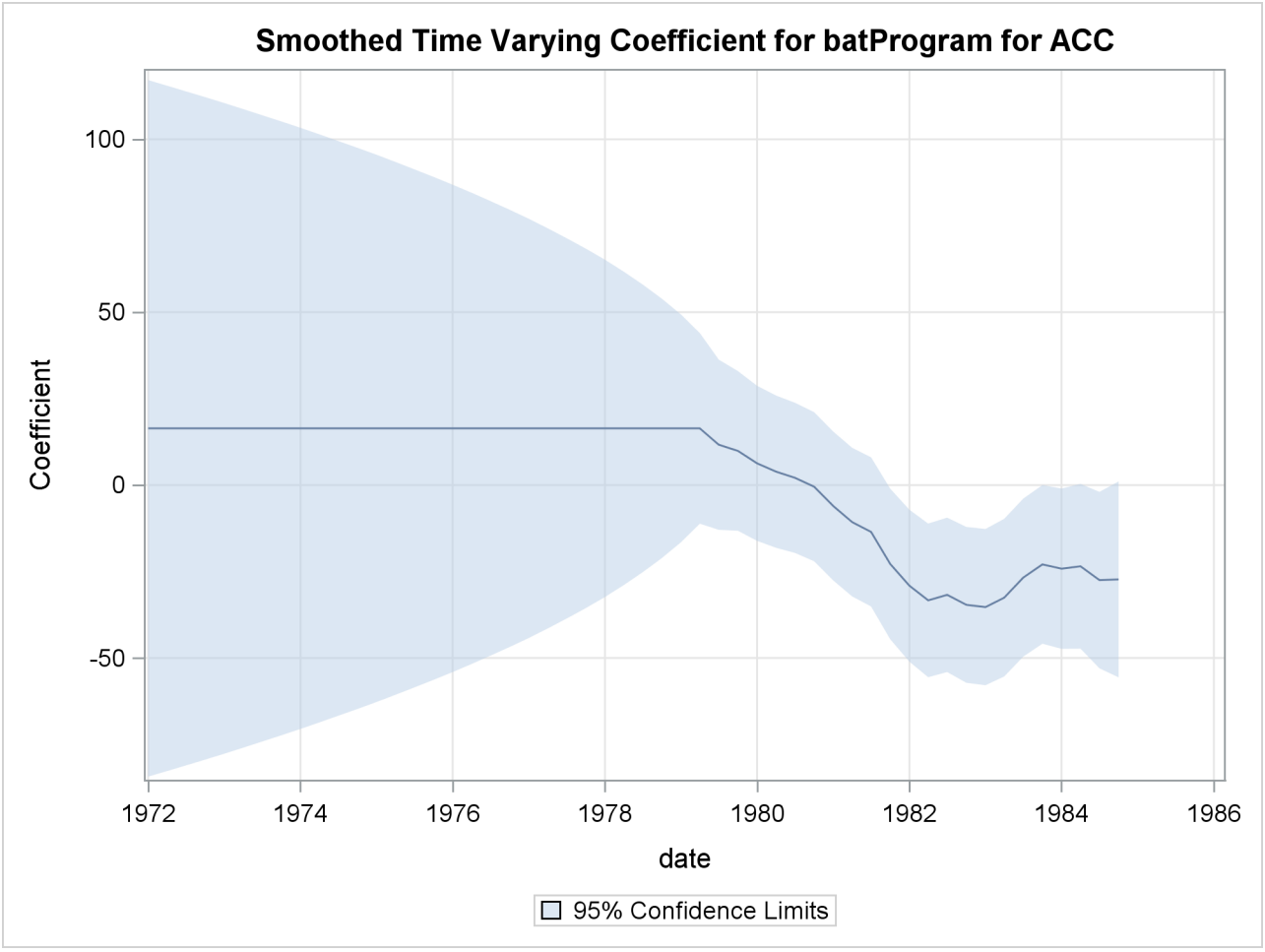
```
data bat;
   input ACC FUEL @@;
   batProgram = 0;
   if _n_ > 29 then batProgram = 1;
   date = INTNX( 'qtr', '1jan1972'd, _n_- 1 );
   format date qtr8.;
datalines;
192    32.592    238    37.250    232    40.032
246    35.852    185    38.226    274    38.711
266    43.139    196    40.434    170    35.898
234    37.111    272    38.944    234    37.717
210    37.861    280    42.524    246    43.965
248    41.976    269    42.918    326    49.789
342    48.454    257    45.056    280    49.385
290    42.524    356    51.224    295    48.562
279    48.167    330    51.362    354    54.646
331    53.398    291    50.584    377    51.320
327    50.810    301    46.272    269    48.664
314    48.122    318    47.483    288    44.732
242    46.143    268    44.129    327    46.258
253    48.230    215    46.459    263    50.686
319    49.681    263    51.029    206    47.236
286    51.717    323    51.824    306    49.380
230    47.961    304    46.039    311    55.683
292    52.263
;
```

There are a number of ways to study these data and the question of the effectiveness of the BAT program. One possibility is to study the *before-after* difference in the injuries and fatalities per million gallons of fuel consumed, by regressing ACC on the FUEL and the dummy variable BATPROGRAM, which is zero before the program began and one while the program is in place. However, it is possible that the effect of the Batmobiles might well be cumulative, because as awareness of the program becomes dispersed, its effectiveness as a deterrent to driving while intoxicated increases. This suggests that the regression coefficient of the BATPROGRAM variable might be *time varying*. The following program fits a model that incorporates these considerations. A seasonal component is included in the model since it is easy to see that the data show strong quarterly seasonality.

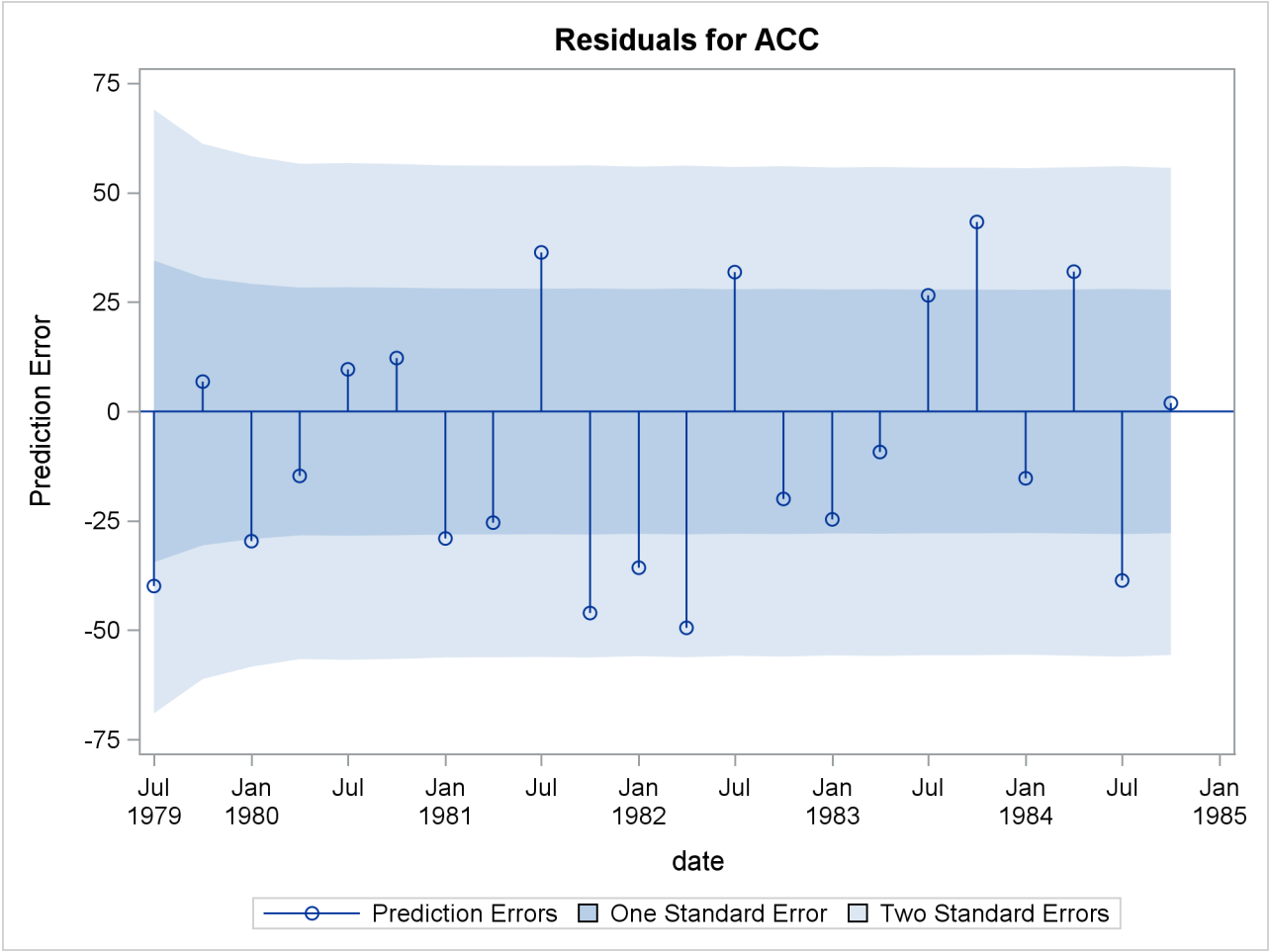
```
proc ucm data=bat;
   model acc = fuel;
   id date interval=qtr;
   irregular;
   level var=0 noest;
   randomreg batProgram / plot=smooth;
   season length=4 var=0 noest plot=smooth;
   estimate plot=(panel residual);
   forecast plot=forecasts lead=0;
run;
```

The model seems to fit the data adequately. No data are withheld for model validation because the series is relatively short. The plot of the time-varying coefficient of BATPROGRAM is shown in Output 41.4.1. As expected, it shows that the effectiveness of the program increases as awareness of the program becomes dispersed. The effectiveness eventually seems to level off. The residual diagnostic plots are shown in Output 41.4.2 and Output 41.4.3, the forecast plot is in Output 41.4.4, the goodness-of-fit statistics are in Output 41.4.5, and the parameter estimates are in Output 41.4.6.

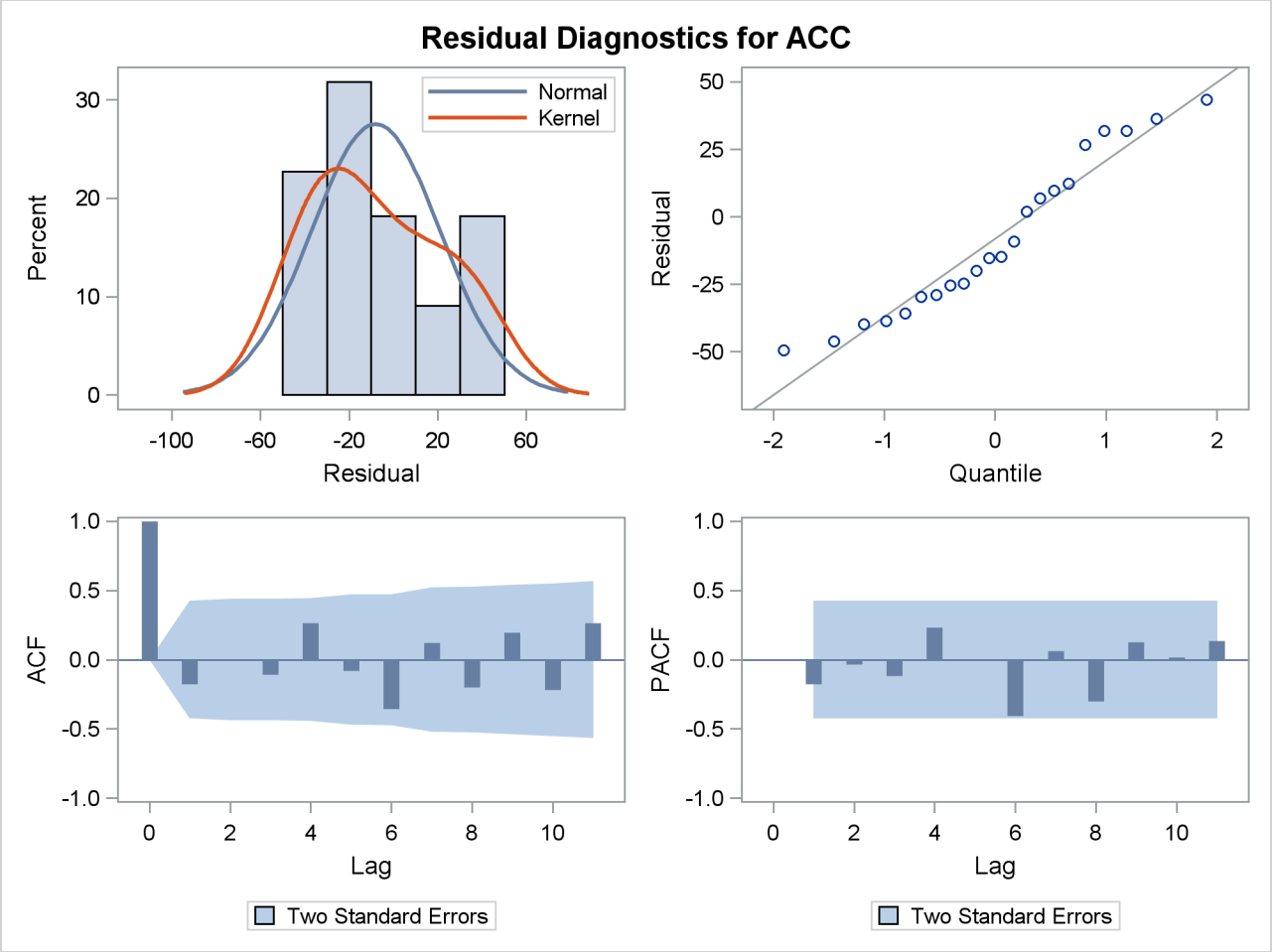
Output 41.4.1 Time-Varying Regression Coefficient of BATPROGRAM



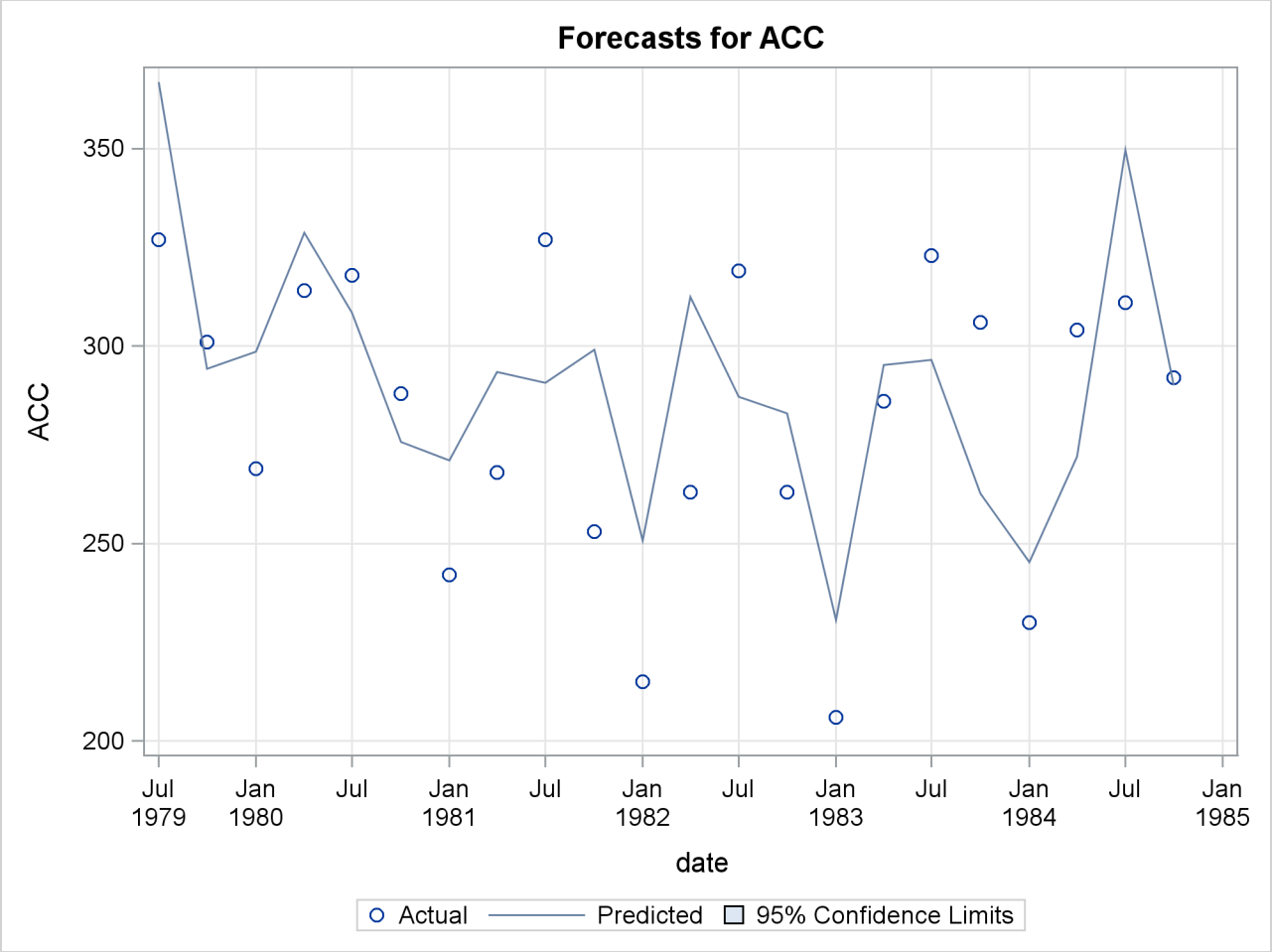
Output 41.4.2 Residuals for the Time-Varying Regression Model



Output 41.4.3 Residual Diagnostics for the Time-Varying Regression Model



Output 41.4.4 One-Step-Ahead Forecasts for the Time-Varying Regression Model



Output 41.4.5 Model Fit for the Time-Varying Regression Model

Fit Statistics Based on Residuals	
Mean Squared Error	866.75562
Root Mean Squared Error	29.44071
Mean Absolute Percentage Error	9.50326
Maximum Percent Error	14.15368
R-Square	0.32646
Adjusted R-Square	0.29278
Random Walk R-Square	0.63010
Amemiya's Adjusted R-Square	0.19175
Number of non-missing residuals used for computing the fit statistics = 22	

Output 41.4.6 Parameter Estimates for the Time-Varying Regression Model

Final Estimates of the Free Parameters

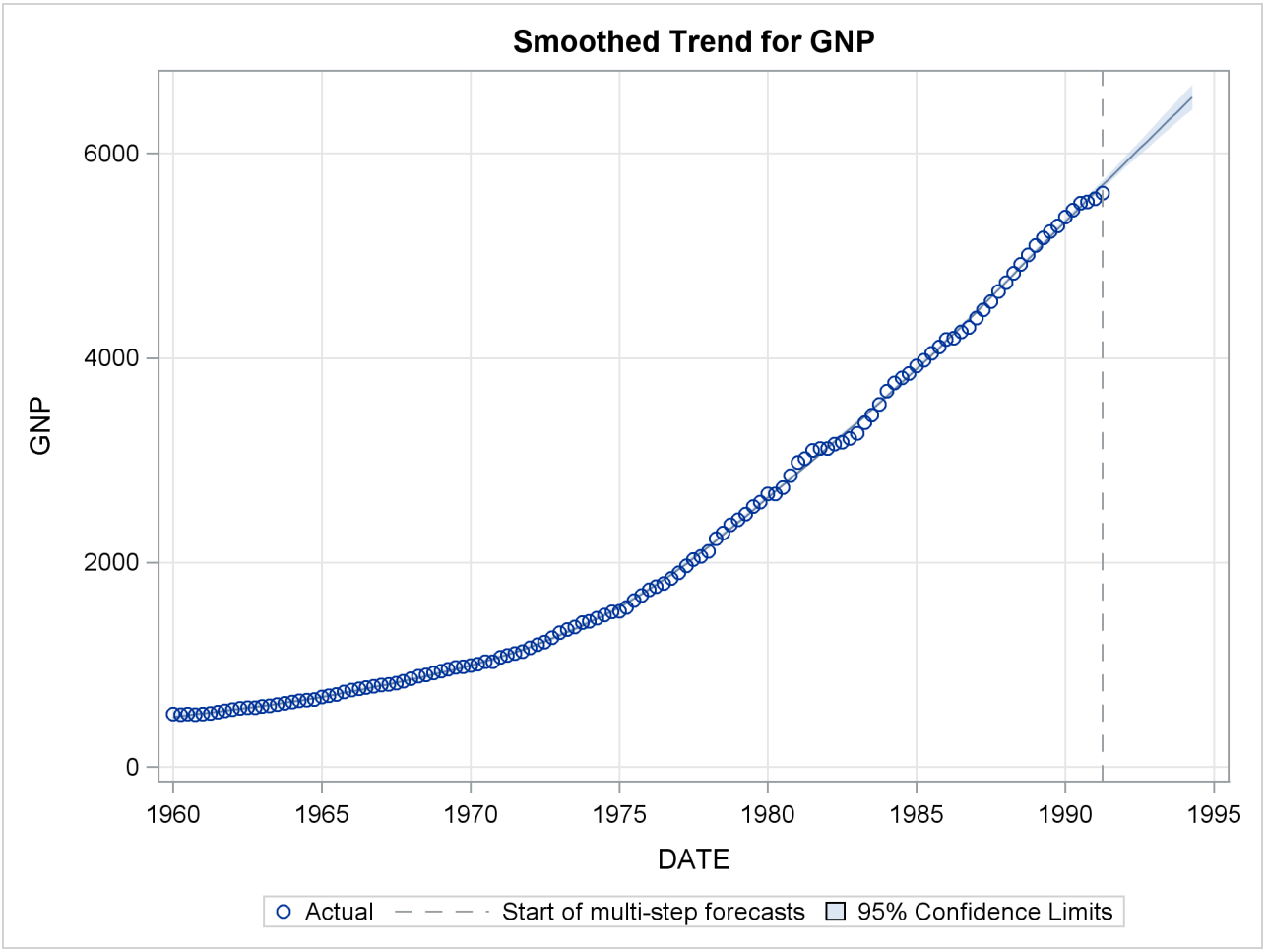
Component	Parameter	Estimate	Approx Std Error	t Value	Approx Pr > |t|
Irregular	Error Variance	480.92258	109.21980	4.40	<.0001
FUEL	Coefficient	6.23279	0.67533	9.23	<.0001
batProgram	Error Variance	84.22334	79.88166	1.05	0.2917

Example 41.5: Trend Removal Using the Hodrick-Prescott Filter

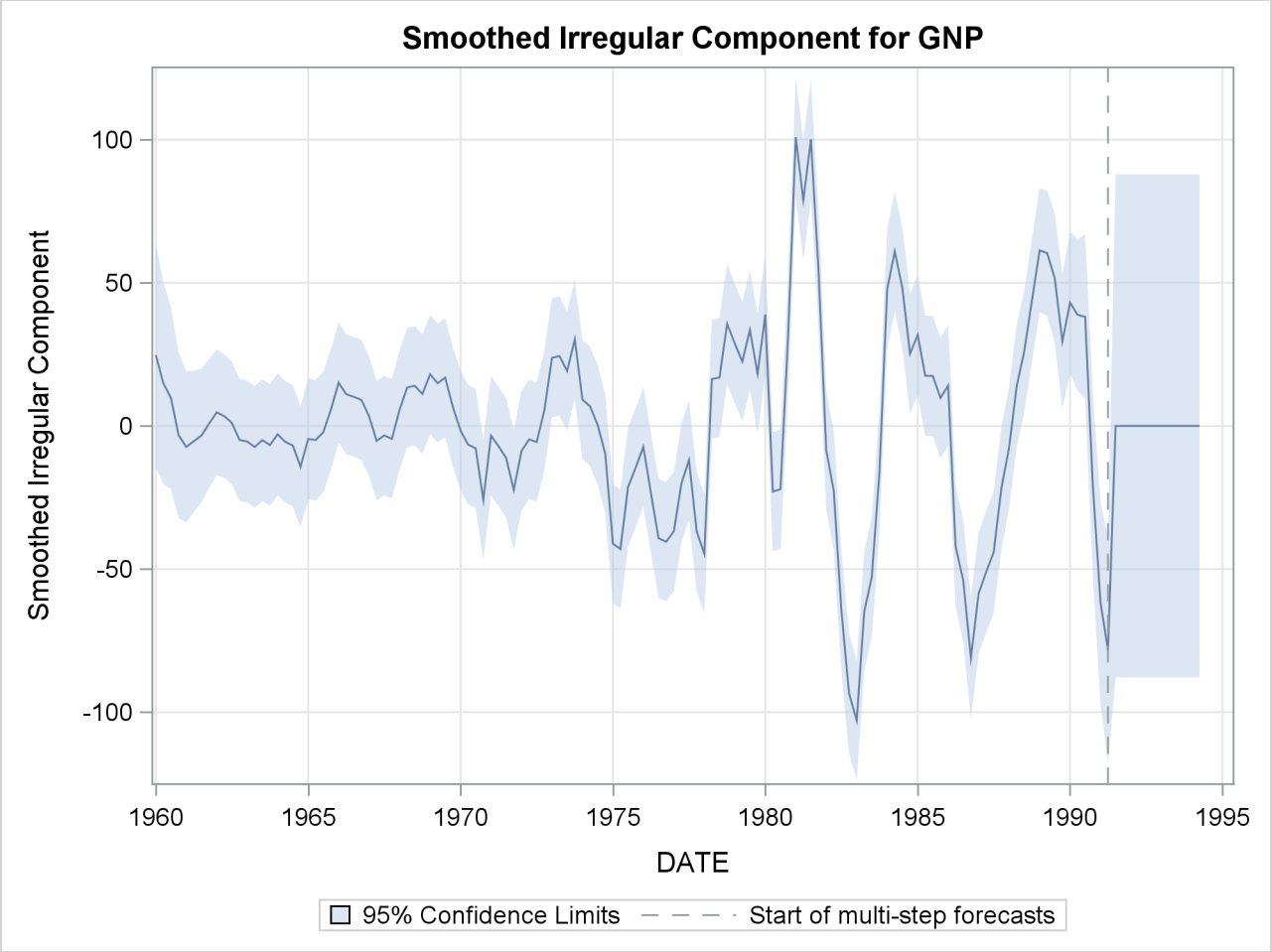
Hodrick-Prescott filter (Hodrick and Prescott 1997) is a popular tool in macroeconomics for fitting smooth trend to time series. It is well known that the trend computation according to this filter is equivalent to fitting the local linear trend plus irregular model with the level disturbance variance restricted to zero and the slope disturbance variance restricted to be a suitable multiple of the irregular component variance. The multiple used depends on the frequency of the series; for example, for quarterly series the commonly recommended multiple is $1/1600 = 0.000625$. For other intervals there is no consensus, but a frequently suggested value for monthly series is $1/14400$ and the value for an annual series can range from $1/400 = 0.0025$ to $1/7 = 0.15$. The data set considered in this example consists of quarterly GNP values for the United States from 1960 to 1991. In the UCM procedure statements that follow, the presence of the PROFILE option in the ESTIMATE statement implies that the restriction that the disturbance variance of the slope component be fixed at 0.000625 is interpreted differently: it implies that the disturbance variance of the slope component be restricted to be 0.000625 *times* the estimated irregular component variance, as needed for the Hodrick-Prescott filter. The plot of the fitted trend is shown in Output 41.5.1, and the plot of the smoothed irregular component, which corresponds to the detrended series, is given in Output 41.5.2. The detrended series can be further analyzed for business cycles.

```
proc ucm data=sashelp.gnp;
   id date interval=qtr;
   model gnp;
   irregular plot=smooth;
   level var=0 noest plot=smooth;
   slope var=0.000625 noest;
   estimate PROFILE;
   forecast plot=(decomp);
run;
```

Output 41.5.1 Smoothed Trend for the GNP Series as per the Hodrick-Prescott Filter



Output 41.5.2 Detrended GNP Series



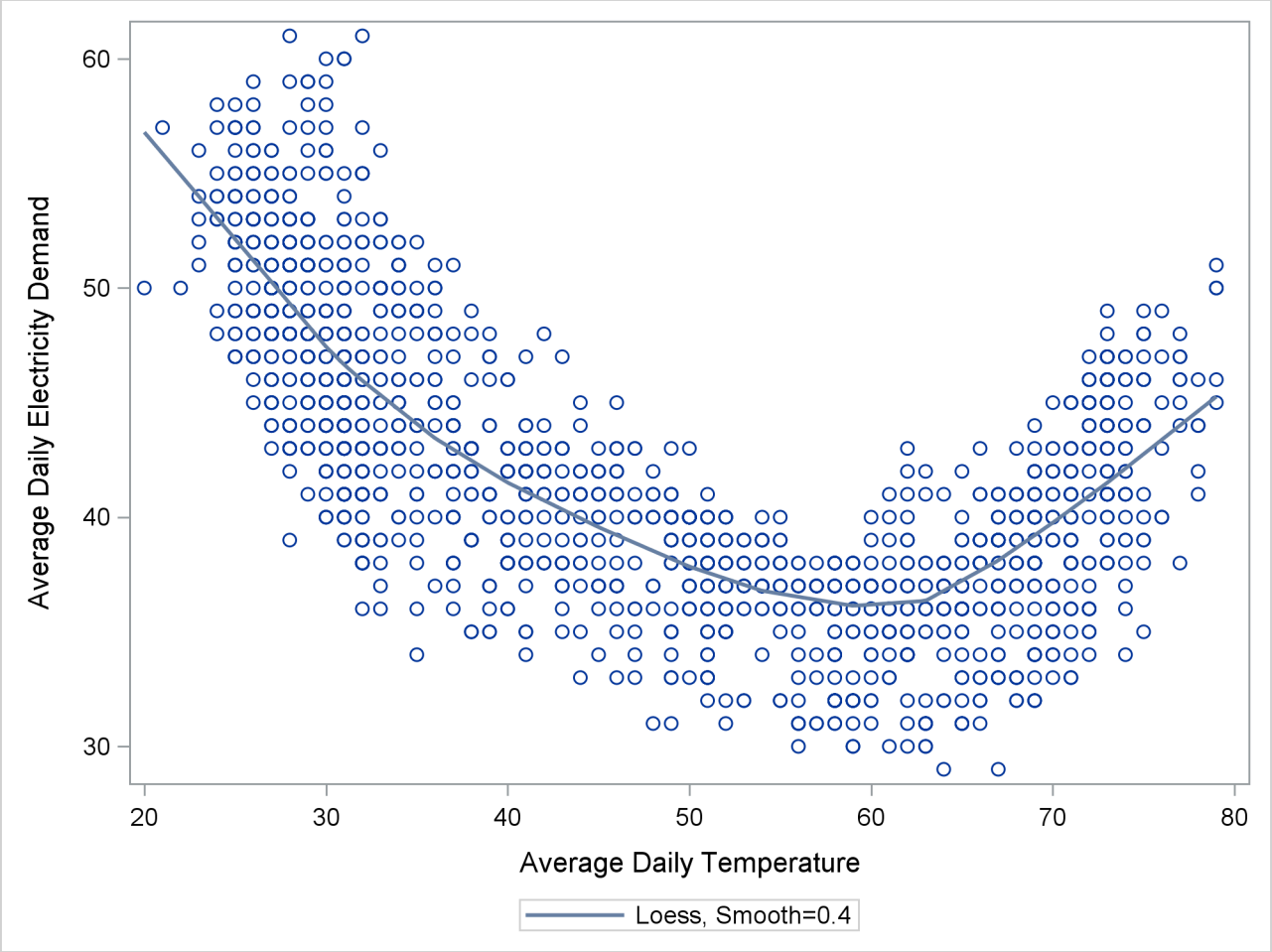
Example 41.6: Using Splines to Incorporate Nonlinear Effects

The data in this example are created to mirror the electricity demand and temperature data recorded at a utility company in the midwest region of the United States. The data set (not shown), utility, has three variables: load, temp, and date. The load column contains the daily electricity demand, the temp column has the average daily temperature readings, and the date column records the observation date.

The following statements produce a plot, shown in Output 41.6.1, of electricity load versus temperature. Clearly the relationship is smooth but nonlinear: the load generally increases when the temperatures are away from the comfortable sixties.

```
proc sgplot data=utility;
    loess x=temp y=load / smooth=0.4;
run;
```

Output 41.6.1 Load versus Temperature Plot



The time series plot of the load (not shown) also shows that, apart from a day-of-the-week seasonal effect, there are no additional easily identifiable patterns in the series. The series has no apparent upward or downward trend. The following statements fit a UCM to the series that takes into account these observations. The particular choice of the model is a result of a little modeling exercise that compared a small number of competing models. The chosen model is adequate but by no means the best possible. The temperature effect is modeled by a deterministic three-degree spline with knots at 30, 40, 50, 60, and 75. The knot locations and the degree were chosen by visual inspection of the plot (Output 41.6.1). An autoreg component is used in place of the simple irregular component, which improved the residual analysis. The last 60 days of data are withheld for out-of-sample forecast evaluation (note the BACK= option in both the ESTIMATE and FORECAST statements). The OUTLIER statement is used to increase the number of outliers reported to 10. Since no CHECKBREAK option is used in the LEVEL statement, only the additive outliers are searched. In this example the use of the EXTRADIFFUSE= option in the ESTIMATE and FORECAST statements is useful for discarding some early one-step-ahead forecasts and residuals with large variance.

```
proc ucm data=utility;
   id date interval=day;
   model load;
   autoreg;
   level plot=smooth;
```

```
   splinereg temp knots=30 40 50 65 75 degree=3
      variance=0 noest;
   season length=7 var=0 noest;
   estimate plot=panel back=60
      extradiffuse=50;
   outlier maxnum=10;
   forecast back=60 lead=60
      extradiffuse=50;
run;
```

The parameter estimates are given in Output 41.6.2, and the residual goodness-of-fit statistics are shown in Output 41.6.3. The residual diagnostic plots are shown in Output 41.6.4. The ACF and PACF plots appear satisfactory, but the normality plots, particularly the Q-Q plot, show possible violations. It appears that, at least in part, this nonNormal behavior of the residuals might be attributable to the outliers in the series. The outlier summary table, Output 41.6.5, shows the most likely outlying observations. Notice that most of these outliers are holidays, like July 4th, when the electricity load is lower than usual for that day of the week.

Output 41.6.2 Electricity Load: Parameter Estimates

The UCM Procedure

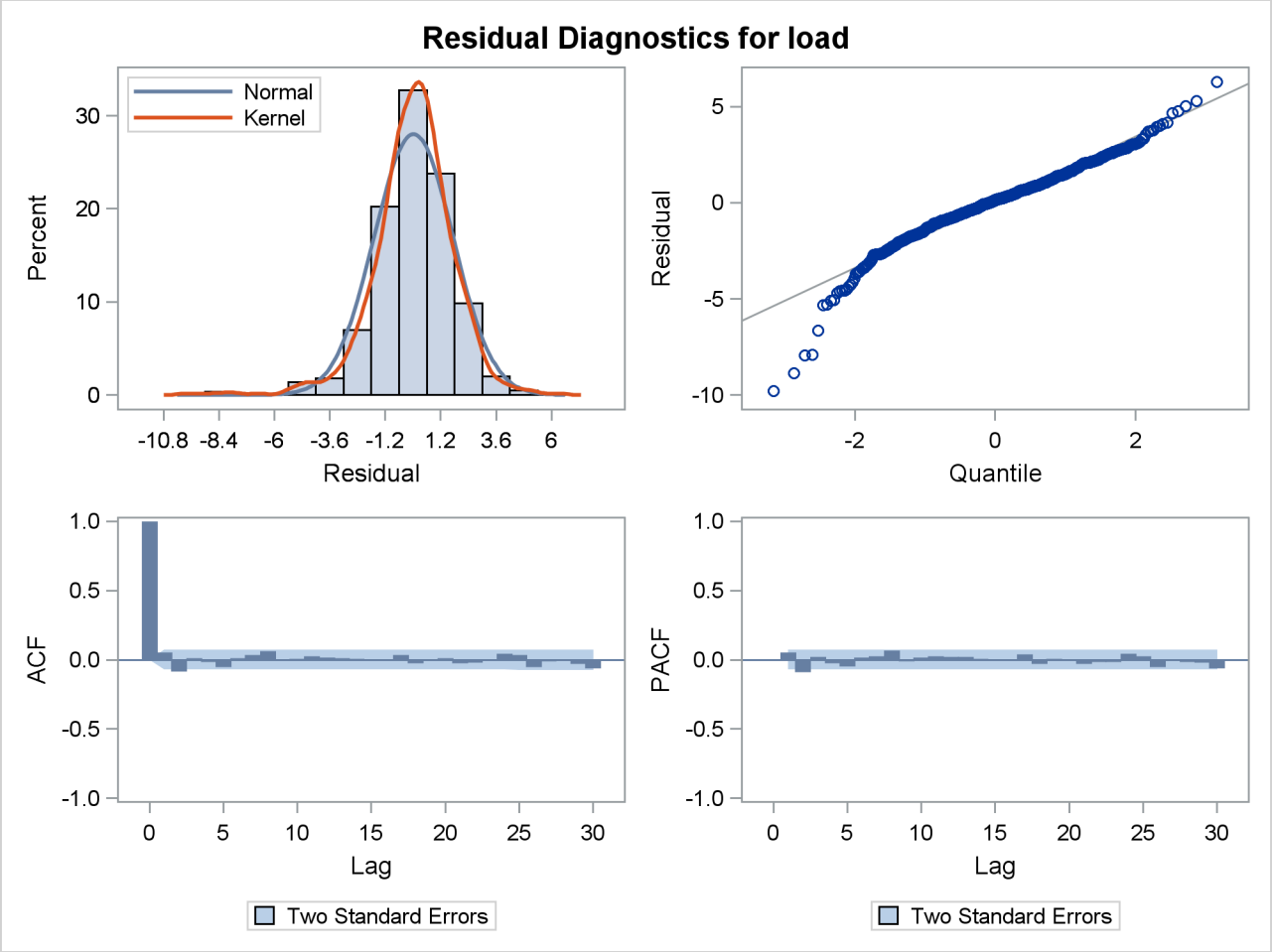
Final Estimates of the Free Parameters

Component	Parameter	Estimate	Approx Std Error	t Value	Approx Pr > |t|
Level	Error Variance	0.21185	0.05025	4.22	<.0001
AutoReg	Damping Factor	0.57522	0.03466	16.60	<.0001
AutoReg	Error Variance	2.21057	0.20478	10.79	<.0001
temp	Spline Coefficient_1	4.72502	1.93997	2.44	0.0149
temp	Spline Coefficient_2	2.19116	1.71243	1.28	0.2007
temp	Spline Coefficient_3	-7.14492	1.56805	-4.56	<.0001
temp	Spline Coefficient_4	-11.39950	1.45098	-7.86	<.0001
temp	Spline Coefficient_5	-16.38055	1.36977	-11.96	<.0001
temp	Spline Coefficient_6	-18.76075	1.28898	-14.55	<.0001
temp	Spline Coefficient_7	-8.04628	1.09017	-7.38	<.0001
temp	Spline Coefficient_8	-2.30525	1.25102	-1.84	0.0654

Output 41.6.3 Electricity Load: goodness-of-fit

Fit Statistics Based on Residuals	
Mean Squared Error	2.90945
Root Mean Squared Error	1.70571
Mean Absolute Percentage Error	2.92586
Maximum Percent Error	14.96281
R-Square	0.92739
Adjusted R-Square	0.92721
Random Walk R-Square	0.69618
Amemiya's Adjusted R-Square	0.92684
Number of non-missing residuals used for computing the fit statistics = 791	

Output 41.6.4 Electricity Load: Residual Diagnostics

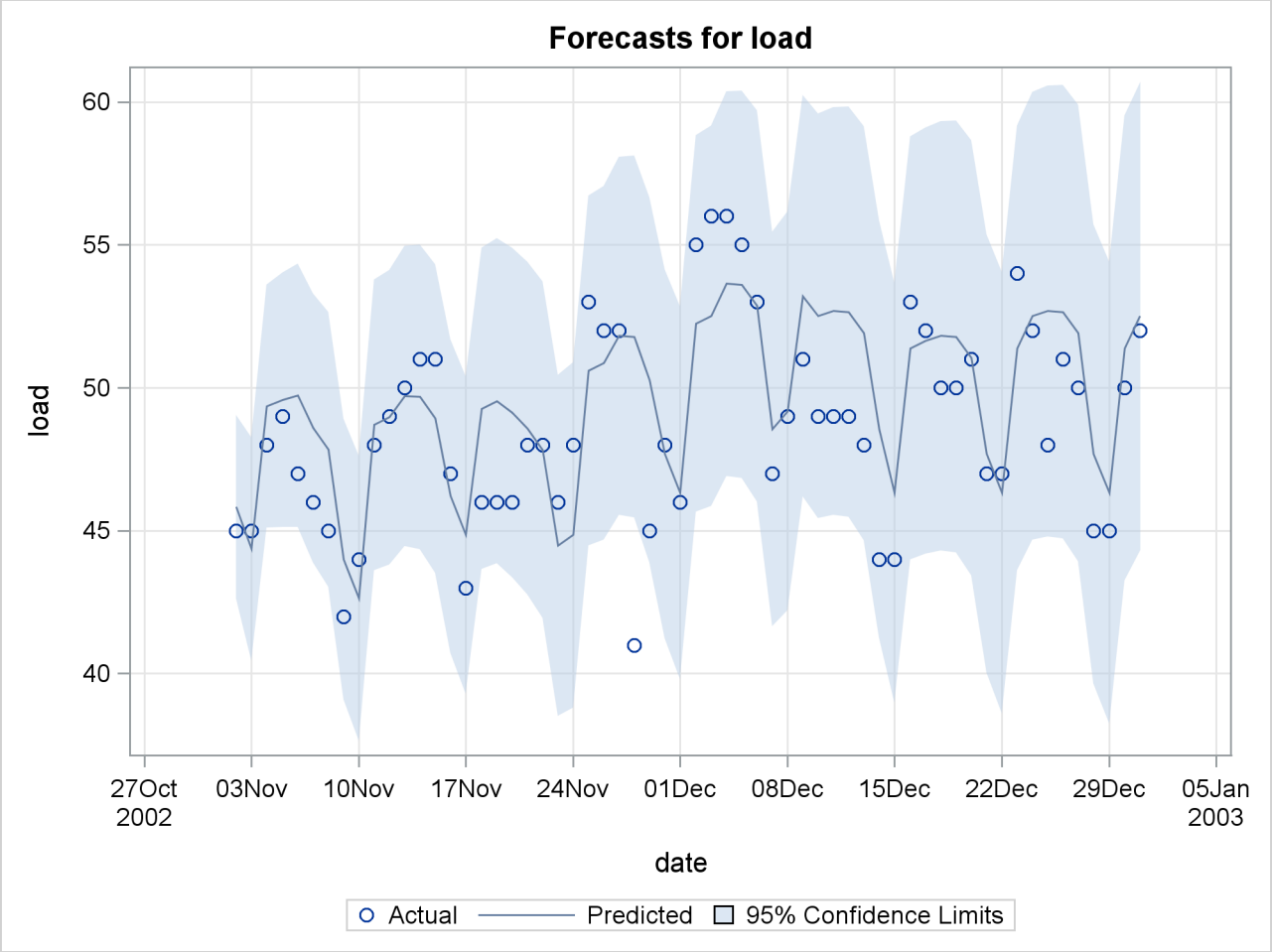


Output 41.6.5 Additive Outliers in the Electricity Load Series

Obs	Time	Estimate	StdErr	ChiSq	DF	ProbChiSq
1281	04JUL2002	-7.99908	1.3417486	35.54	1	<.0001
916	04JUL2001	-6.55778	1.338431	24.01	1	<.0001
329	25NOV1999	-5.85047	1.3379735	19.12	1	<.0001
977	03SEP2001	-5.67254	1.3389138	17.95	1	<.0001
1341	02SEP2002	-5.49631	1.337843	16.88	1	<.0001
693	23NOV2000	-5.27968	1.3374368	15.58	1	<.0001
915	03JUL2001	5.06557	1.3375273	14.34	1	0.0002
1057	22NOV2001	-5.01550	1.3386184	14.04	1	0.0002
551	04JUL2000	-4.89965	1.3381557	13.41	1	0.0003
879	28MAY2001	-4.76135	1.3375349	12.67	1	0.0004

The plot of the load forecasts for the withheld data is shown in Output 41.6.6.

Output 41.6.6 Electricity Load: Forecast Evaluation of the Withheld Data



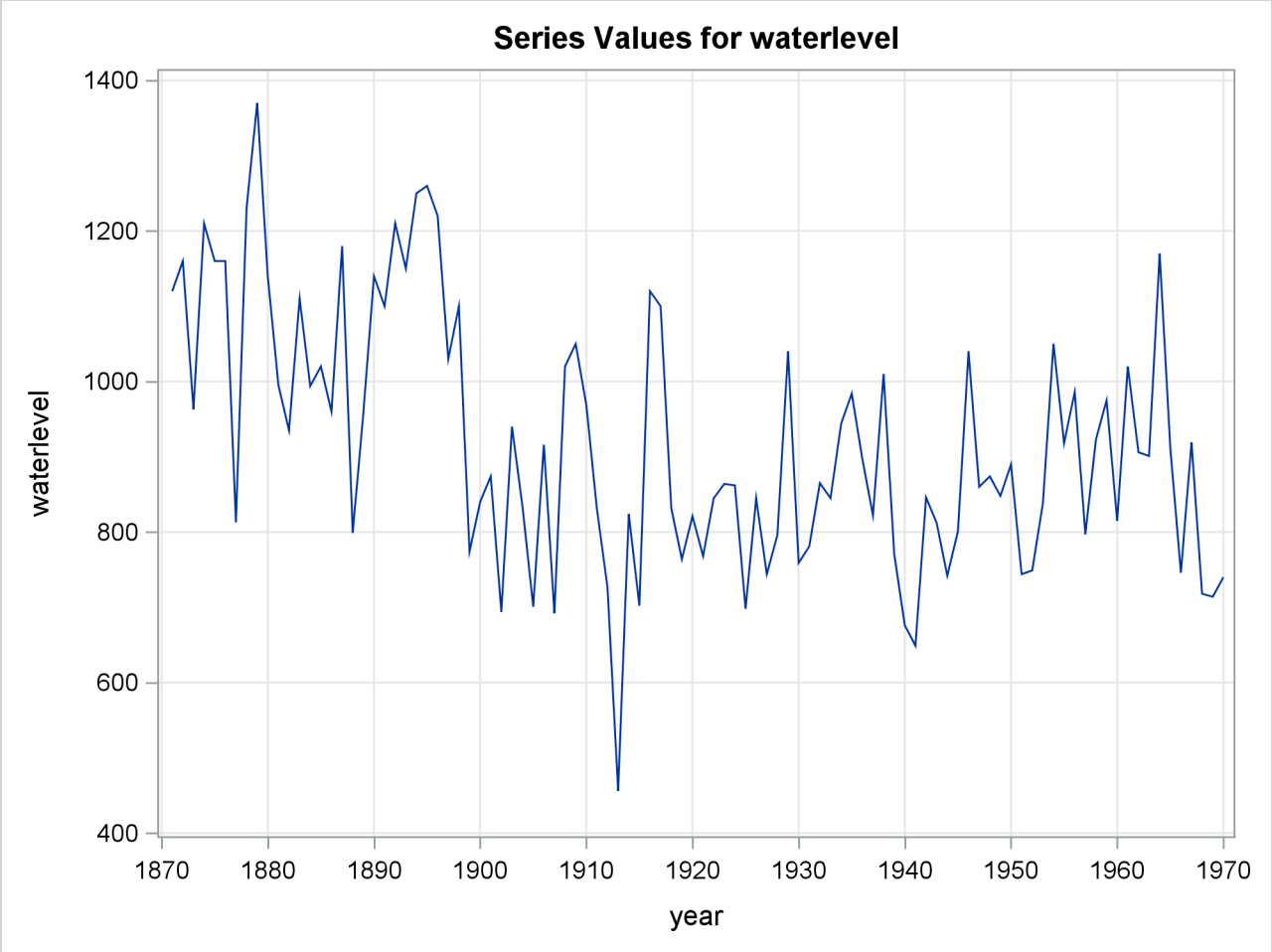
Example 41.7: Detection of Level Shift

The series in this example consists of the yearly water level readings of the Nile River recorded at Aswan, Egypt (Cobb 1978; De Jong and Penzer 1998). The readings are from the years 1871 to 1970. The series does not show any apparent trend or any other distinctive patterns; however, there is a shift in the water level starting at the year 1899. This shift could be attributed to the start of construction of a dam near Aswan in that year. A time series plot of this series is given in Output 41.7.1. The following DATA step statements create the input data set.

```
data nile;
   input waterlevel @@;
   year = intnx( 'year', '1jan1871'd, _n_-1 );
   format year year4.;
datalines;
   1120  1160  963  1210  1160  1160  813  1230   1370  1140
   995   935   1110 994   1020  960   1180 799    958   1140
   1100  1210  1150 1250  1260  1220  1030 1100   774   840
   874   694   940  833   701   916   692  1020   1050  969
   831   726   456  824   702   1120  1100 832    764   821
```

```
   768   845   864  862   698   845   744  796    1040  759
   781   865   845  944   984   897   822  1010   771   676
   649   846   812  742   801   1040  860  874    848   890
   744   749   838  1050  918   986   797  923    975   815
   1020  906   901  1170  912   746   919  718    714   740
;

proc timeseries data=nile plot=series;
   id year interval=year;
   var waterlevel;
run;
```

Output 41.7.1 Nile Water Level

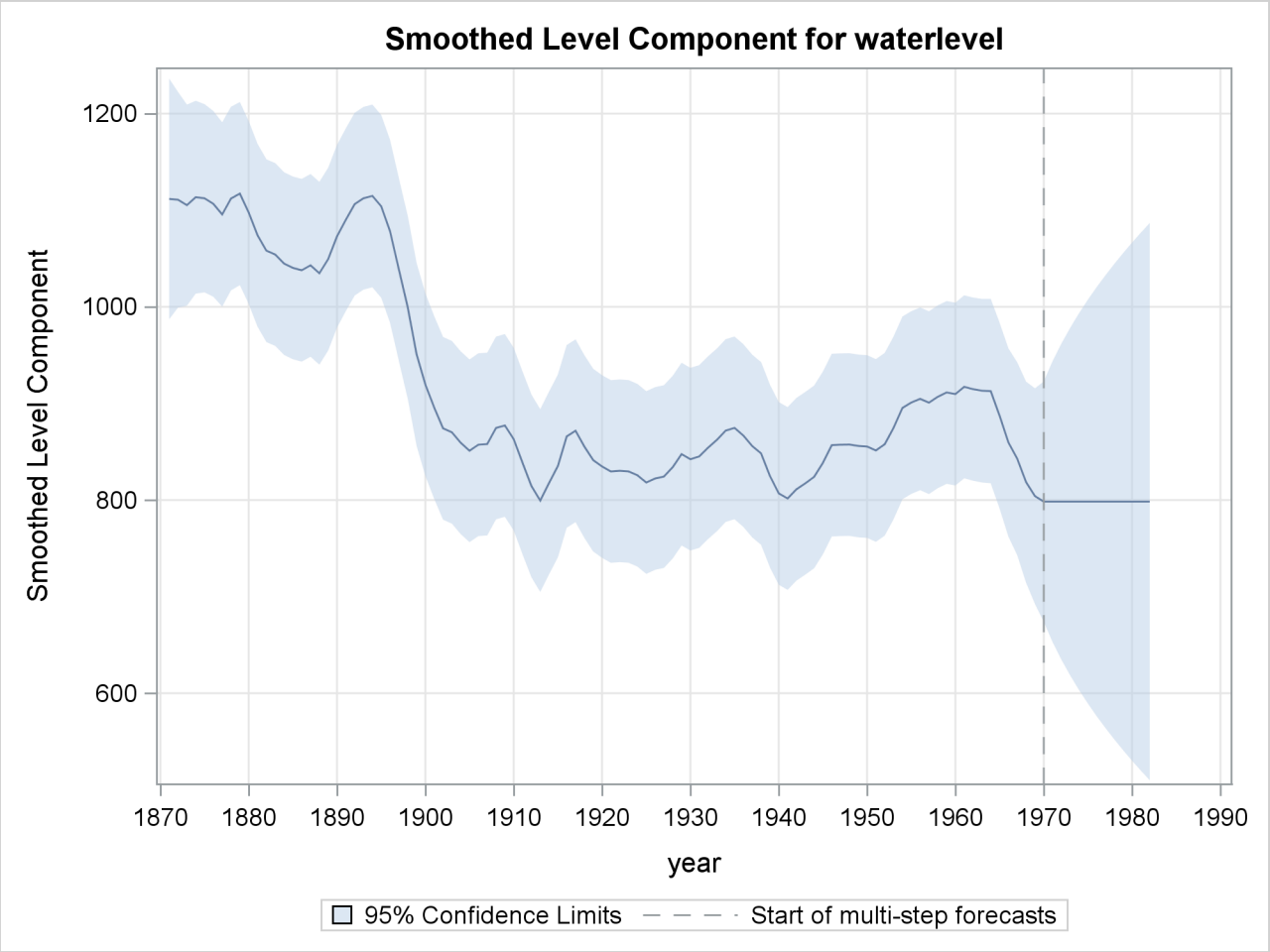


In this situation it is known that a shift in the water level occurred within the span of the series, and its effect can be easily taken into account by including an appropriate indicator variable as a regressor. However, in many situation such prior information is not available, and it is useful to detect such a shift in a data analytic fashion. You can check for breaks in the level by using the CHECKBREAK option in the LEVEL statement. The following statements fit a simple locally constant level plus error model to the series:

```
proc ucm data=nile;
   id year interval=year;
   model waterlevel;
   irregular;
   level plot=smooth checkbreak;
   estimate;
   forecast plot=decomp;
run;
```

The plot in Output 41.7.2 shows a noticeable drop in the smoothed water level around 1899.

Output 41.7.2 Smoothed Trend without the Shift of 1899



The "Outlier Summary" table in Output 41.7.3 shows the most likely types of breaks and their locations within the series span. The shift of 1899 is easily detected.

Output 41.7.3 Detection of Structural Breaks in the Nile River Level

Outlier Summary

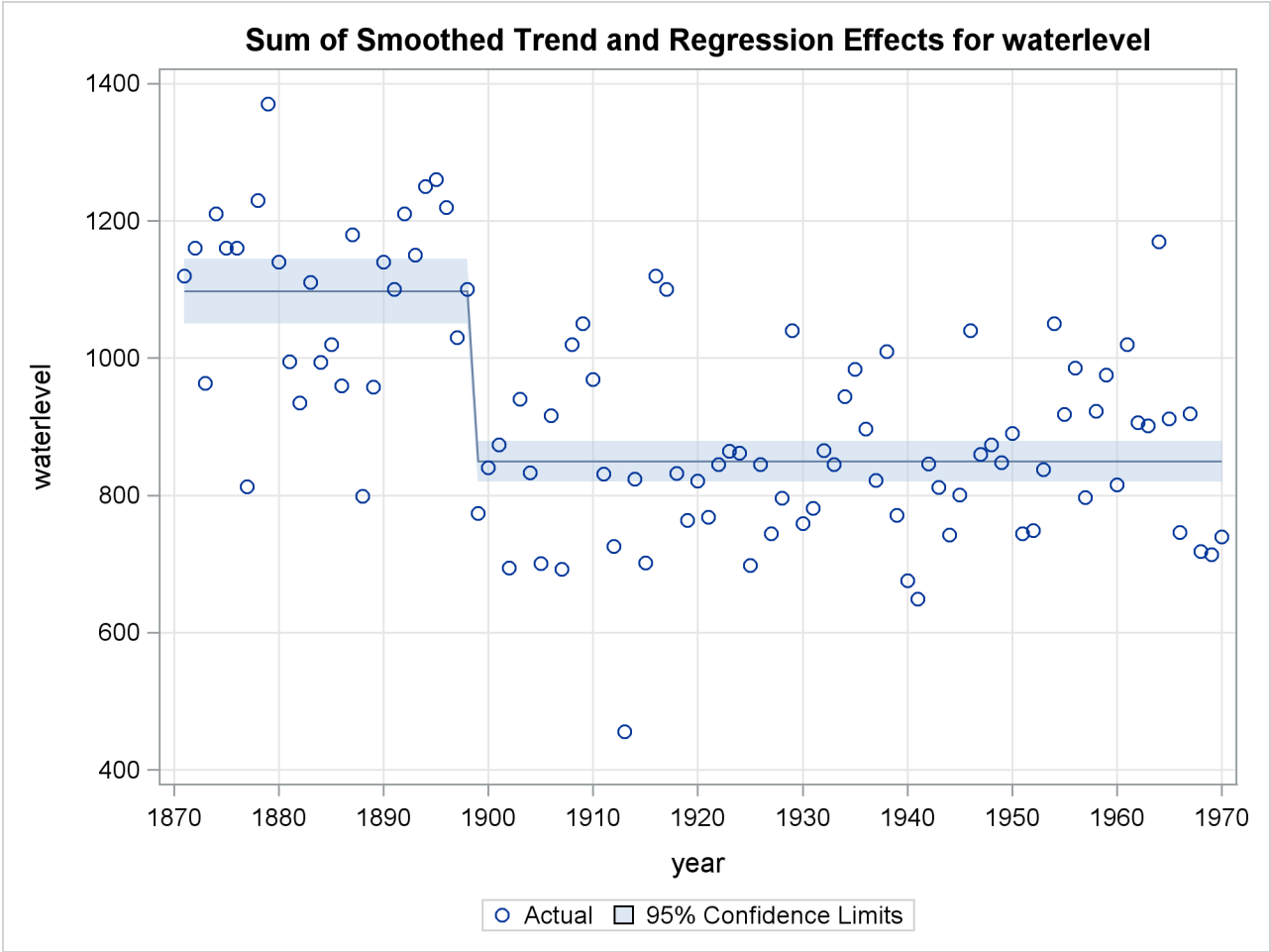
Obs	year	Break Type	Estimate	Standard Error	Chi-Square	DF	Pr > ChiSq
29	**1899**	**Level**	-315.73791	97.639753	10.46	1	0.0012

The following statements specify a UCM that models the level of the river as a locally constant series with a shift in the year 1899, represented by a dummy regressor (SHIFT1899):

```
data nile;
   set nile;
   shift1899 = ( year >= '1jan1899'd );
run;

proc ucm data=nile;
   id year interval=year;
   model waterlevel = shift1899;
   irregular;
   level;
   estimate;
   forecast plot=decomp;
run;
```

The plot in Output 41.7.4 shows the smoothed trend, including the correction due to the shift in the year 1899. Notice the simplicity in the shape of the smoothed curve after the incorporation of the shift information.

Output 41.7.4 Smoothed Trend plus Shift of 1899



Example 41.8: ARIMA Modeling

This example shows how you can use the UCM procedure for ARIMA modeling. The parameter estimates and predictions for ARIMA models obtained by using PROC UCM will be close to those obtained by using PROC ARIMA (in the presence of the ML option in its ESTIMATE statement) if the model is stationary or if the model is nonstationary and there are no missing values in the data. For more information about the ARIMA procedure, see Chapter 8, "The ARIMA Procedure." However, if there are missing values in the data and the model is nonstationary, then the UCM and ARIMA procedures can produce significantly different parameter estimates and predictions. An article by Kohn and Ansley (1986) suggests a statistically sound method of estimation, prediction, and interpolation for nonstationary ARIMA models with missing data. This method is based on an algorithm that is equivalent to the Kalman filtering and smoothing algorithm used in the UCM procedure. The results of an illustrative example in their article are reproduced here using the UCM procedure. In this example an ARIMA$(0,1,1)\times(0,1,1)_{12}$ model is applied to the logarithm of the air series in the sashelp.air data set. Four different missing value patterns are considered to highlight different aspects of the problem:

- *Data1.* The full data set of 144 observations.
- *Data2.* The set of 78 observations that omit January through November in each of the last 6 years.
- *Data3.* The data set with the 5 observations July 1949, June, July, and August 1957, and July 1960 missing.
- *Data4.* The data set with all July observations missing and June and August 1957 also missing.

The following DATA steps create these data sets:

```
data Data1;
   set sashelp.air;
   logair = log(air);
run;

data Data2;
   set data1;
   if year(date) >= 1955 and month(date) < 12 then logair = .;
run;

data Data3;
   set data1;
   if (year(date) = 1949 and month(date) = 7) then logair = .;
   if ( year(date) = 1957 and
        (month(date) = 6 or month(date) = 7 or month(date) = 8))
        then logair = .;
   if (year(date) = 1960 and month(date) = 7) then logair = .;
run;

data Data4;
   set data1;
   if month(date) = 7 then logair = .;
   if year(date) = 1957 and (month(date) = 6 or month(date) = 8)
      then logair = .;
run;
```

The following statements specify the ARIMA$(0, 1, 1) \times (0, 1, 1)_{12}$ model for the logair series in the first data set (Data1):

```
proc ucm data=Data1;
   id date interval=month;
   model logair;
   irregular q=1 sq=1 s=12;
   deplag lags=(1)(12) phi=1 1 noest;
   estimate outest=est1;
   forecast outfor=for1;
run;
```

Note that the moving average part of the model is specified by using the Q=, SQ=, and S= options in the IRREGULAR statement and the differencing operator, $(1 - B)(1 - B^{12})$, is specified by using the DEPLAG statement. The model does not contain an intercept term; therefore no LEVEL statement is needed. The parameter estimates are saved in a data set EST1 by using the OUTEST= option in the ESTIMATE statement and the forecasts and the component estimates are saved in a data set FOR1 by using the OUTFOR= option in the FORECAST statement. The same analysis is performed on the other three data sets, but is not shown here.

Output 41.8.1 resembles Table 1 in Kohn and Ansley (1986). This table is generated by merging the parameter estimates from the four analyses. Only the moving average parameter estimates and their standard errors are reported. The columns EST1 and STD1 correspond to the estimates for Data1. The parameter estimates and their standard errors for other three data sets are similarly named. Note that the parameter estimates closely match the parameter estimates in the article. However, their standard errors differ slightly. This difference could be the result of different ways of computing the Hessian at the optimum. The white noise error variance estimates are not reported here, but they agree quite closely with those in the article.

Output 41.8.1 Data Sets 1–4: Parameter Estimates and Standard Errors

PARAMETER	est1	std1	est2	std2	est3	std3	est4	std4
MA_1	0.402	0.090	0.457	0.121	0.408	0.092	0.431	0.091
SMA_1	0.557	0.073	0.758	0.236	0.566	0.075	0.573	0.074

Output 41.8.2 resembles Table 2 in Kohn and Ansley (1986). It contains forecasts and their standard errors for the four data sets. The numbers are very close to those in the article.

Output 41.8.2 Data Sets 1–4: Forecasts and Standard Errors

DATE	for1	std1	for2	std2	for3	std3	for4	std4
JAN61	6.110	0.037	6.084	0.052	6.110	0.037	6.111	0.037
FEB61	6.054	0.043	6.091	0.058	6.054	0.043	6.055	0.043
MAR61	6.172	0.048	6.247	0.063	6.173	0.048	6.174	0.048
APR61	6.199	0.053	6.205	0.068	6.199	0.053	6.200	0.052
MAY61	6.233	0.057	6.199	0.072	6.232	0.058	6.233	0.056
JUN61	6.369	0.061	6.308	0.076	6.367	0.062	6.368	0.060
JUL61	6.507	0.065	6.409	0.079	6.497	0.067	.	.
AUG61	6.503	0.069	6.414	0.082	6.503	0.069	6.503	0.067
SEP61	6.325	0.072	6.299	0.085	6.325	0.072	6.326	0.071
OCT61	6.209	0.075	6.174	0.087	6.209	0.076	6.209	0.074
NOV61	6.063	0.079	6.043	0.089	6.064	0.079	6.064	0.077
DEC61	6.168	0.082	6.174	0.086	6.168	0.082	6.169	0.080

Output 41.8.3 is based on Data2. It resembles Table 3 in Kohn and Ansley (1986). The columns S_SERIES and VS_SERIES in the OUTFOR= data set contain the interpolated values of logair and their variances. The estimate column in Output 41.8.3 reports interpolated values (which are the same as S_SERIES), and the std column reports their standard errors (which are computed as square root of VS_SERIES) for January–November 1957. The actual logair values for these months, which are missing in Data2, are also provided for comparison. The numbers are very close to those in the article.

Output 41.8.3 Data Set 2: Interpolated Values and Standard Errors

DATE	logair	estimate	std
JAN57	5.753	5.733	0.045
FEB57	5.707	5.738	0.049
MAR57	5.875	5.893	0.052
APR57	5.852	5.850	0.054
MAY57	5.872	5.843	0.055
JUN57	6.045	5.951	0.055
JUL57	6.142	6.051	0.055
AUG57	6.146	6.055	0.054
SEP57	6.001	5.938	0.052
OCT57	5.849	5.812	0.049
NOV57	5.720	5.680	0.045

Output 41.8.4 resembles Table 4 in Kohn and Ansley (1986). These numbers are based on Data3, and they also are very close to those in the article.

Output 41.8.4 Data Set 3: Interpolated Values and Standard Errors

DATE	logair	estimate	std
JUL49	4.997	5.013	0.031
JUN57	6.045	6.024	0.030
JUL57	6.142	6.147	0.031
AUG57	6.146	6.148	0.030
JUL60	6.433	6.409	0.031

Output 41.8.5 resembles Table 5 in Kohn and Ansley (1986). As before, the numbers are very close to those in the article.

Output 41.8.5 Data Set 4: Interpolated Values and Standard Errors

DATE	logair	estimate	std
JUN57	6.045	6.023	0.030
AUG57	6.146	6.147	0.030

The similarity between the outputs in this example and the results shown in Kohn and Ansley (1986) demonstrate that PROC UCM can be effectively used for nonstationary ARIMA models with missing data.

References

Akaike, H. (1974). "A New Look at the Statistical Model Identification." *IEEE Transactions on Automatic Control* AC-19:716–723.

Anderson, T. W. (1971). *The Statistical Analysis of Time Series*. New York: John Wiley & Sons.

Bloomfield, P. (2000). *Fourier Analysis of Time Series*. 2nd ed. New York: John Wiley & Sons.

Box, G. E. P., and Jenkins, G. M. (1976). *Time Series Analysis: Forecasting and Control*. Rev. ed. San Francisco: Holden-Day.

Bozdogan, H. (1987). "Model Selection and Akaike's Information Criterion (AIC): The General Theory and Its Analytical Extensions." *Psychometrika* 52:345–370.

Brockwell, P. J., and Davis, R. A. (1991). *Time Series: Theory and Methods*. 2nd ed. New York: Springer-Verlag.

Burnham, K. P., and Anderson, D. R. (1998). *Model Selection and Inference: A Practical Information-Theoretic Approach*. New York: Springer-Verlag.

Cobb, G. W. (1978). "The Problem of the Nile: Conditional Solution to a Change Point Problem." *Biometrika* 65:243–251.

De Jong, P., and Chu-Chun-Lin, S. (2003). "Smoothing with an Unknown Initial Condition." *Journal of Time Series Analysis* 24:141–148.

De Jong, P., and Penzer, J. (1998). "Diagnosing Shocks in Time Series." *Journal of the American Statistical Association* 93:796–806.

Durbin, J., and Koopman, S. J. (2001). *Time Series Analysis by State Space Methods*. Oxford: Oxford University Press.

Hannan, E. J., and Quinn, B. G. (1979). "The Determination of the Order of an Autoregression." *Journal of the Royal Statistical Society, Series B* 41:190–195.

Harvey, A. C. (1989). *Forecasting, Structural Time Series Models, and the Kalman Filter*. Cambridge: Cambridge University Press.

Harvey, A. C. (2001). "Testing in Unobserved Components Models." *Journal of Forecasting* 20:1–19.

Hodrick, R. J., and Prescott, E. C. (1997). "Postwar U.S. Business Cycles: An Empirical Investigation." *Journal of Money, Credit, and Banking* 29:1–16.

Hurvich, C. M., and Tsai, C.-L. (1989). "Regression and Time Series Model Selection in Small Samples." *Biometrika* 76:297–307.

Jones, R. H. (1980). "Maximum Likelihood Fitting of ARMA Models to Time Series with Missing Observations." *Technometrics* 22:389–396.

Kohn, R., and Ansley, C. F. (1986). "Estimation, Prediction, and Interpolation for ARIMA Models with Missing Data." *Journal of the American Statistical Association* 81:751–761.

Rodriguez, A., and Ruiz, E. (2010). "Bootstrap Prediction Mean Squared Errors of Unobserved States Based on the Kalman Filter with the Estimated Parameters." Working Paper 10-03 (01), Statistics and Econometric Series, Departamento de Estadística, Universidad Carlos de Madrid.

Schwarz, G. (1978). "Estimating the Dimension of a Model." *Annals of Statistics* 6:461–464.

West, M., and Harrison, J. (1999). *Bayesian Forecasting and Dynamic Models*. 2nd ed. New York: Springer-Verlag.

Chapter 42
The VARMAX Procedure

Contents

Overview: VARMAX Procedure **2856**
Getting Started: VARMAX Procedure **2858**
Vector Autoregressive Model 2858
Bayesian Vector Autoregressive Model 2865
Vector Error Correction Model 2866
Bayesian Vector Error Correction Model 2872
Vector Autoregressive Model with Exogenous Variables 2873
Parameter Estimation and Testing on Restrictions 2876
Causality Testing 2878
Multivariate GARCH Models 2879
Syntax: VARMAX Procedure **2890**
Functional Summary 2890
PROC VARMAX Statement 2893
BOUND Statement 2896
BY Statement 2897
CAUSAL Statement 2897
COINTEG Statement 2898
GARCH Statement 2902
ID Statement 2903
INITIAL Statement 2904
MODEL Statement 2906
NLOPTIONS Statement 2921
OUTPUT Statement 2921
RESTRICT Statement 2922
TEST Statement 2934
Details: VARMAX Procedure **2936**
Missing Values 2936
VARMAX Model 2936
Dynamic Simultaneous Equations Modeling 2940
Impulse Response Function 2943
Forecasting 2953
Tentative Order Selection 2958
VAR and VARX Modeling 2963
Seasonal Dummies and Time Trends 2969
Bayesian VAR and VARX Modeling 2970
VARMA and VARMAX Modeling 2972

Model Diagnostic Checks . . . 2981
Cointegration . . . 2983
Vector Error Correction Modeling . . . 2986
I(2) Model . . . 3002
Vector Error Correction Model in ARMA Form . . . 3005
Multivariate GARCH Modeling . . . 3006
Output Data Sets . . . 3017
OUT= Data Set . . . 3017
OUTEST= Data Set . . . 3019
OUTHT= Data Set . . . 3021
OUTSTAT= Data Set . . . 3023
Printed Output . . . 3025
ODS Table Names . . . 3026
ODS Graphics . . . 3031
Computational Issues . . . 3032
Examples: VARMAX Procedure . . . 3033
Example 42.1: Analysis of United States Economic Variables . . . 3033
Example 42.2: Analysis of German Economic Variables . . . 3044
Example 42.3: Analysis of Restricted Cointegrated Systems . . . 3055
Example 42.4: Analysis of Euro Foreign Exchange Reference Rates . . . 3063
Example 42.5: Numerous Examples . . . 3075
Example 42.6: Illustration of ODS Graphics . . . 3078
References . . . 3081

Overview: VARMAX Procedure

Given a multivariate time series, the VARMAX procedure estimates the model parameters and generates forecasts that are associated with vector autoregressive moving average processes with exogenous regressors (VARMAX) models. Often, economic or financial variables are not only contemporaneously correlated with each other, but also correlated with each other's past values. You can use the VARMAX procedure to model these types of time relationships. In many economic and financial applications, the variables of interest (dependent, response, or endogenous variables) are influenced by variables external to the system under consideration (independent, input, predictor, regressor, or exogenous variables). The VARMAX procedure enables you to model the dynamic relationships both among the dependent variables and between the dependent and independent variables.

A VARMAX model is defined in terms of the orders of the autoregressive or moving average processes (or both). When you use the VARMAX procedure, these orders can be specified by options or they can be automatically determined according to the information criteria. The VARMAX procedure supports the following information criteria: Akaike's information criterion (AIC), the corrected AIC (AICC), the Hannan-Quinn criterion (HQC), the final prediction error (FPE), and the Schwarz Bayesian criterion (SBC), which is also known as the Bayesian information criterion (BIC). For the definitions and usages of the information

criteria, see the section "The Minimum Information Criterion (MINIC) Method" on page 2962 for more details.

If you do not want to use automatic order selection, the VARMAX procedure provides the following autoregressive order identification aids: partial cross-correlations, partial autoregressive coefficients, partial canonical correlations, and Yule-Walker estimates.

For situations where the stationarity of the time series is in question, the VARMAX procedure provides the following tests to aid in determining the presence of unit roots and cointegration: Dickey-Fuller tests, the Stock-Watson common trends test for the possibility of cointegration among nonstationary vector processes of integrated order one, and Johansen cointegration tests for nonstationary vector processes of integrated order one and order two.

For stationary vector times series or nonstationary series that are made stationary by appropriate differencing or cointegration, the VARMAX procedure provides the vector autoregressive and moving average (VARMA) model and the vector error correction model (VECM). The vector error correction model can be in both autoregressive (AR) and autoregressive and moving average (ARMA) forms.

To cope with the problem of high dimensionality in the parameters of the VAR model and the VECM, the VARMAX procedure provides both the Bayesian vector autoregressive (BVAR) model and the Bayesian vector error correction model (BVECM). Bayesian models are used when prior information about the model parameters is available.

The VARMAX procedure also allows independent (exogenous) variables and their distributed lags to influence dependent (endogenous) variables in various models. These models are identified by an X suffix added to the original model name; for example, VARMAX, VECMX, BVARX, and BVECMX.

Correlations in the second moments of the vector time series might exist; this is called conditional heteroscedasticity. The VARMAX procedure supports three forms of multivariate generalized autoregressive conditional heteroscedasticity (GARCH) models to model the conditional heteroscedasticity: the Baba-Engle-Kroner-Kraft (BEKK) GARCH model, the constant conditional correlation (CCC) GARCH model, and the dynamic conditional correlation (DCC) GARCH model. For CCC and DCC GARCH models, five subforms of univariate GARCH models are supported: the GARCH model, the exponential GARCH (EGARCH) model, the quadratic GARCH (QGARCH) model, the threshold GARCH (TGARCH) model, and the power GARCH (PGARCH) model.

You can use the VARMAX-GARCH model or the VEC-ARMAX-GARCH model to simultaneously model both the first and second moments of the time series.

Forecasting is one of the main objectives of multivariate time series analysis. After successfully fitting the VARMAX, BVARX, VECMX, BVECMX, and multivariate GARCH models, the VARMAX procedure computes predicted values and conditional heteroscedasticity based on the parameter estimates and the past values of the vector time series. Out-of-sample multistep-ahead forecasts are also supported.

The following model parameter estimation methods are supported:

- the least squares (LS) method, which can be applied to VARX models
- the maximum likelihood (ML) method, which can be applied to all types of models
- the conditional maximum likelihood (CML) method, which can be applied to VARMAX models

When you use the ML or CML method, you can start your optimization with the default or with different initial parameter values.

The VARMAX procedure supports the estimation of the restricted model when you impose linear constraints on the parameters of interest. The VARMAX procedure also supports various hypothesis tests of long-run effects and adjustment coefficients by using the likelihood ratio test that is based on Johansen cointegration analysis. It also supports the likelihood ratio test of weak exogeneity for each variable. In fact, because the VARMAX procedure outputs log-likelihood values for all models, you can always use the likelihood ratio test to check any linear hypothesis on parameters that are estimated in the models by estimating the restricted and unrestricted models separately. The VARMAX procedure also supports another alternative test, the Wald test.

After fitting the model parameters, the VARMAX procedure uses the following tests to provide model checks and residual analysis: Durbin-Watson (DW) statistics, the F test for autoregressive conditional heteroscedastic (ARCH) disturbance, the F test for AR disturbance, the Jarque-Bera normality test, and the Portmanteau test.

The VARMAX procedure supports several modeling features, including seasonal deterministic terms, linear and quadratic time trends, subset models, multiple regression with distributed lags, the dead-start model (which does not have present values of the exogenous variables), and so on.

The VARMAX procedure provides a Granger causality test to determine the Granger-causal relationships between two distinct groups of variables. It also provides the following: the infinite order AR representation, the impulse response function (also called infinite order MA representation), the decomposition of the predicted error covariances, roots of the characteristic functions for both the AR and MA parts to evaluate the proximity of the roots to the unit circle, and contemporaneous relationships among the components of the vector time series.

Getting Started: VARMAX Procedure

This section provides several examples of the types of models that the VARMAX procedure supports.

Vector Autoregressive Model

Let $\mathbf{y}_t = (y_{1t}, \ldots, y_{kt})'$, $t = 1, 2, \ldots$, denote a k-dimensional time series vector of random variables of interest. The pth-order VAR process is written as

$$\mathbf{y}_t = \boldsymbol{\delta} + \Phi_1 \mathbf{y}_{t-1} + \cdots + \Phi_p \mathbf{y}_{t-p} + \boldsymbol{\epsilon}_t$$

where $\boldsymbol{\epsilon}_t = (\epsilon_{1t}, \ldots, \epsilon_{kt})'$ is a vector white noise process such that $\mathrm{E}(\boldsymbol{\epsilon}_t) = 0$, $\mathrm{E}(\boldsymbol{\epsilon}_t \boldsymbol{\epsilon}_t') = \Sigma$, and $\mathrm{E}(\boldsymbol{\epsilon}_t \boldsymbol{\epsilon}_s') = 0$ for $t \neq s$; $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_k)'$ is a constant vector; and Φ_i is a $k \times k$ matrix.

Analyzing and modeling the series jointly enables you to understand the dynamic relationships over time among the series and to improve the accuracy of forecasts for individual series by using the additional information available from the related series and their forecasts.

Consider the first-order stationary bivariate vector autoregressive model:

$$\mathbf{y}_t = \begin{pmatrix} 1.2 & -0.5 \\ 0.6 & 0.3 \end{pmatrix} \mathbf{y}_{t-1} + \boldsymbol{\epsilon}_t, \quad \text{with } \Sigma = \begin{pmatrix} 1.0 & 0.5 \\ 0.5 & 1.25 \end{pmatrix}$$

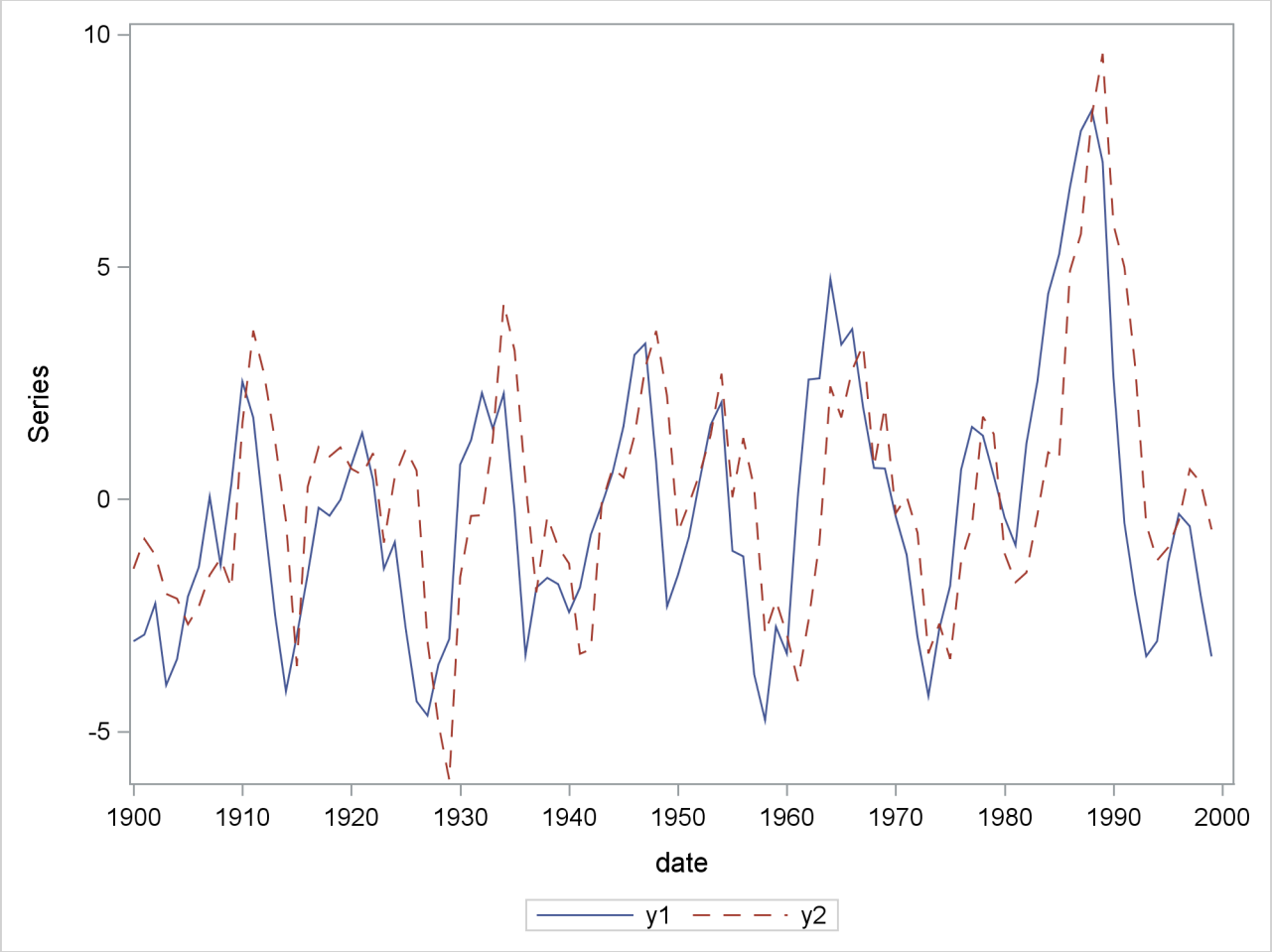
The following IML procedure statements simulate a bivariate vector time series from this model to provide test data for the VARMAX procedure:

```
proc iml;
   sig = {1.0  0.5, 0.5 1.25};
   phi = {1.2 -0.5, 0.6 0.3};
   /* simulate the vector time series */
   call varmasim(y,phi) sigma = sig n = 100 seed = 34657;
   cn = {'y1' 'y2'};
   create simul1 from y[colname=cn];
   append from y;
quit;
```

The following statements plot the simulated vector time series $\mathbf{y}_t$, which is shown in Figure 42.1:

```
data simul1;
   set simul1;
   date = intnx( 'year', '01jan1900'd, _n_-1 );
   format date year4.;
run;

proc sgplot data=simul1;
   series x=date y=y1 / lineattrs=(pattern=solid);
   series x=date y=y2 / lineattrs=(pattern=dash);
   yaxis label="Series";
run;
```

Figure 42.1 Plot of the Generated Data Process



The following statements fit a VAR(1) model to the simulated data:

```
/*--- Vector Autoregressive Model ---*/

proc varmax data=simul1;
   id date interval=year;
   model y1 y2 / p=1 noint lagmax=3
                 print=(estimates diagnose);
   output out=for lead=5;
run;
```

First, you specify the input data set in the PROC VARMAX statement. Then, you use the MODEL statement to designate the dependent variables, y_1 and y_2. To estimate a zero-mean VAR model, you specify the order of the autoregressive model in the P= option and the NOINT option. The MODEL statement fits the model to the data and prints parameter estimates and their significance. The PRINT=ESTIMATES option prints the matrix form of parameter estimates, and the PRINT=DIAGNOSE option prints various diagnostic tests. The LAGMAX=3 option prints the output for the residual diagnostic checks.

To output the forecasts to a data set, you specify the OUT= option in the OUTPUT statement. If you want to forecast five steps ahead, you use the LEAD=5 option. The ID statement specifies the yearly interval between observations and provides the Time column in the forecast output.

The VARMAX procedure output is shown in Figure 42.2 through Figure 42.10. The VARMAX procedure first displays descriptive statistics, as shown in Figure 42.2. The Type column indicates that the variables are dependent variables. The N column indicates the number of nonmissing observations.

Figure 42.2 Descriptive Statistics

The VARMAX Procedure

Number of Observations	100
Number of Pairwise Missing	0

Simple Summary Statistics

Variable	Type	N	Mean	Standard Deviation	Min	Max
y1	**Dependent**	100	-0.21653	2.78210	-4.75826	8.37032
y2	**Dependent**	100	0.16905	2.58184	-6.04718	9.58487

Figure 42.3 shows the model type and the estimation method that is used to fit the model to the simulated data. It also shows the AR coefficient matrix in terms of lag 1, the schematic representation, and the parameter estimates and their significance that can indicate how well the model fits the data.

The "AR" table shows the AR coefficient matrix. The "Schematic Representation" table schematically represents the parameter estimates and enables you to easily verify their significance in matrix form.

In the "Model Parameter Estimates" table, the first column shows the variable on the left side of the equation; the second column is the parameter name ARl_i_j, which indicates the (i, j) element of the lag l autoregressive coefficient; the next four columns provide the estimate, standard error, t value, and p-value for the parameter; and the last column is the regressor that corresponds to the displayed parameter.

Figure 42.3 Model Type and Parameter Estimates

The VARMAX Procedure

Type of Model	VAR(1)
Estimation Method	Least Squares Estimation

		AR	
Lag	**Variable**	**y1**	**y2**
1	**y1**	1.15977	-0.51058
	y2	0.54634	0.38499

Schematic Representation	
Variable/Lag	**AR1**
y1	+-
y2	++

+ is > 2*std error, - is < -2*std error, . is between, * is N/A

Model Parameter Estimates

Equation	**Parameter**	**Estimate**	**Standard Error**	**t Value**	**Pr > |t|**	**Variable**
y1	**AR1_1_1**	1.15977	0.05508	21.06	0.0001	y1(t-1)
	AR1_1_2	-0.51058	0.05898	-8.66	0.0001	y2(t-1)
y2	**AR1_2_1**	0.54634	0.05779	9.45	0.0001	y1(t-1)
	AR1_2_2	0.38499	0.06188	6.22	0.0001	y2(t-1)

The fitted VAR(1) model with estimated standard errors in parentheses is given as

$$\mathbf{y}_t = \begin{pmatrix} 1.160 & -0.511 \\ (0.055) & (0.059) \\ 0.546 & 0.385 \\ (0.058) & (0.062) \end{pmatrix} \mathbf{y}_{t-1} + \boldsymbol{\epsilon}_t$$

Clearly, all parameter estimates in the coefficient matrix Φ_1 are significant.

The model can also be written as two univariate regression equations:

$$\begin{aligned} y_{1t} &= 1.160\, y_{1,t-1} - 0.511\, y_{2,t-1} + \epsilon_{1t} \\ y_{2t} &= 0.546\, y_{1,t-1} + 0.385\, y_{2,t-1} + \epsilon_{2t} \end{aligned}$$

The table in Figure 42.4 shows the innovation covariance matrix estimates, the log likelihood, and the various information criteria results. The variable names in the table for the innovation covariance matrix estimates $\hat{\Sigma}$ are printed for convenience: $y1$ means the innovation for $y1$, and $y2$ means the innovation

for $y2$. The log likelihood for a VAR model that is estimated through least squares method is defined as $-T(\log(|\hat{\Sigma}_{\mathrm{ML}}|) + k)/2$, where $T(= 100 - 1 = 99)$ is the sample size except the presample being skipped because of the AR lag order, $k(= 2)$ is the number of dependent variables, and $\hat{\Sigma}_{\mathrm{ML}}$ is the maximum likelihood estimate (MLE) of innovation covariance matrix. The matrix $\hat{\Sigma}_{\mathrm{ML}}$ is computed from the reported least squares estimate of the innovation covariance matrix, $\hat{\Sigma}$, by adjusting the degrees of freedom. $\hat{\Sigma}_{\mathrm{ML}} = \frac{T-r_b}{T}\hat{\Sigma}$, where $r_b(= 2)$ is the number of parameters in each equation. You can use the information criteria to compare the fit of competing models to a set of data. The model that has a smaller value of the information criterion is preferred when it is compared to other models. For more information about how to calculate the information criteria, see the section "Multivariate Model Diagnostic Checks" on page 2981.

Figure 42.4 Innovation Covariance Estimates, Log Likelihood, and Information Criteria

Covariances of Innovations

Variable	y1	y2
y1	1.28875	0.39751
y2	0.39751	1.41839

Log-likelihood	-122.362

Information Criteria

AICC	259.9557
HQC	266.0748
AIC	258.7249
SBC	276.8908
FPEC	1.738092

Figure 42.5 shows the cross covariances of the residuals. The values of the lag 0 are slightly different from Figure 42.4 because of the different degrees of freedom.

Figure 42.5 Multivariate Diagnostic Checks

Cross Covariances of Residuals

Lag	Variable	y1	y2
0	y1	1.26271	0.38948
	y2	0.38948	1.38974
1	y1	0.03121	0.05675
	y2	-0.04646	-0.05398
2	y1	0.08134	0.10599
	y2	0.03482	-0.01549
3	y1	0.01644	0.11734
	y2	0.00609	0.11414

Figure 42.6 and Figure 42.7 show tests for white noise residuals that are based on the cross correlations of the residuals. The output shows that you cannot reject the null hypothesis that the residuals are uncorrelated.

Figure 42.6 Multivariate Diagnostic Checks, Continued

Cross Correlations of Residuals

Lag	Variable	y1	y2
0	y1	1.00000	0.29401
	y2	0.29401	1.00000
1	y1	0.02472	0.04284
	y2	-0.03507	-0.03884
2	y1	0.06442	0.08001
	y2	0.02628	-0.01115
3	y1	0.01302	0.08858
	y2	0.00460	0.08213

Schematic Representation of Cross Correlations of Residuals

Variable/Lag	0	1	2	3
y1	++	. .	. .	. .
y2	++	. .	. .	. .

+ is > 2*std error, - is < -2*std error, . is between

Figure 42.7 Multivariate Diagnostic Checks, Continued

Portmanteau Test for Cross Correlations of Residuals

Up To Lag	DF	Chi-Square	Pr > ChiSq
2	4	1.58	0.8124
3	8	2.78	0.9473

The VARMAX procedure provides diagnostic checks for the univariate form of the equations. The table in Figure 42.8 describes how well each univariate equation fits the data. From the two univariate regression equations shown in Figure 42.3, the values of R^2 in the second column of Figure 42.8 are 0.84 and 0.79. The standard deviations in the third column are the square roots of the diagonal elements of the covariance matrix from Figure 42.4. The F statistics in the fourth column test the null hypotheses $\phi_{11} = \phi_{12} = 0$ and $\phi_{21} = \phi_{22} = 0$, where ϕ_{ij} is the (i, j) element of the matrix Φ_1. The last column shows the p-values of the F statistics. The results show that each univariate model is significant.

Figure 42.8 Univariate Diagnostic Checks

Univariate Model ANOVA Diagnostics

Variable	R-Square	Standard Deviation	F Value	Pr > F
y1	0.8351	1.13523	491.25	<.0001
y2	0.7906	1.19096	366.29	<.0001

The check for white noise residuals in terms of the univariate equation is shown in Figure 42.9. This output contains information that indicates whether the residuals are correlated and heteroscedastic. In the first table, the second column contains the Durbin-Watson test statistics to test the null hypothesis that the residuals are

uncorrelated. The third and fourth columns show the Jarque-Bera normality test statistics and their *p*-values to test the null hypothesis that the residuals have normality. The last two columns show *F* statistics and their *p*-values for ARCH(1) disturbances to test the null hypothesis that the residuals have equal covariances. The second table includes *F* statistics and their *p*-values for AR(1), AR(1,2), AR(1,2,3) and AR(1,2,3,4) models of residuals to test the null hypothesis that the residuals are uncorrelated.

Figure 42.9 Univariate Diagnostic Checks, Continued

Univariate Model White Noise Diagnostics

		Normality		ARCH	
Variable	Durbin Watson	Chi-Square	Pr > ChiSq	F Value	Pr > F
y1	1.94534	3.56	0.1686	0.13	0.7199
y2	2.06276	5.42	0.0667	2.10	0.1503

Univariate Model AR Diagnostics

	AR1		AR2		AR3		AR4	
Variable	F Value	Pr > F	F Value	Pr > F	F Value	Pr > F	F Value	Pr > F
y1	0.02	0.8980	0.14	0.8662	0.09	0.9629	0.82	0.5164
y2	0.52	0.4709	0.41	0.6650	0.32	0.8136	0.32	0.8664

The table in Figure 42.10 shows forecasts, their prediction errors, and 95% confidence limits. For more information, see the section "Forecasting" on page 2953.

Figure 42.10 Forecasts

Forecasts

					95% Confidence Limits	
Variable	Obs	Time	Forecast	Standard Error		
y1	101	2000	-3.59212	1.13523	-5.81713	-1.36711
	102	2001	-3.09448	1.70915	-6.44435	0.25539
	103	2002	-2.17433	2.14472	-6.37792	2.02925
	104	2003	-1.11395	2.43166	-5.87992	3.65203
	105	2004	-0.14342	2.58740	-5.21463	4.92779
y2	101	2000	-2.09873	1.19096	-4.43298	0.23551
	102	2001	-2.77050	1.47666	-5.66469	0.12369
	103	2002	-2.75724	1.74212	-6.17173	0.65725
	104	2003	-2.24943	2.01925	-6.20709	1.70823
	105	2004	-1.47460	2.25169	-5.88782	2.93863

Bayesian Vector Autoregressive Model

The Bayesian vector autoregressive (BVAR) model avoids problems of collinearity and overparameterization that often occur with the use of VAR models. BVAR models avoid these problems by imposing priors on the AR parameters.

The following statements fit a BVAR(1) model to the simulated data:

```
/*--- Bayesian Vector Autoregressive Process ---*/

proc varmax data=simul1;
   model y1 y2 / p=1 noint
                 prior=(lambda=0.9 theta=0.1);
run;
```

The hyperparameters, LAMBDA=0.9 and THETA=0.1, in the PRIOR= option control the prior covariance. Part of the VARMAX procedure output is shown in Figure 42.11, whose parameter estimates are slightly different from those in Figure 42.3. By choosing the appropriate priors, you might be able to obtain more accurate forecasts by using a BVAR model instead of an unconstrained VAR model. For more information, see the section "Bayesian VAR and VARX Modeling" on page 2970.

Figure 42.11 Parameter Estimates for the BVAR(1) Model

The VARMAX Procedure

Type of Model	BVAR(1)
Estimation Method	Maximum Likelihood Estimation
Prior Lambda	0.9
Prior Theta	0.1

Model Parameter Estimates

Equation	Parameter	Estimate	Standard Error	t Value	Pr > |t|	Variable
y1	**AR1_1_1**	1.05623	0.04999	21.13	0.0001	y1(t-1)
	AR1_1_2	-0.34707	0.04807	-7.22	0.0001	y2(t-1)
y2	**AR1_2_1**	0.40068	0.04867	8.23	0.0001	y1(t-1)
	AR1_2_2	0.48728	0.05670	8.59	0.0001	y2(t-1)

Covariances of Innovations

Variable	y1	y2
y1	1.36278	0.45343
y2	0.45343	1.48077

Vector Error Correction Model

A vector error correction model (VECM) can lead to a better understanding of the nature of any nonstationarity among the different component series and can also improve longer-term forecasting compared to an unconstrained model.

The VECM(p) form with the cointegration rank, $r(\leq k)$, is written as

$$\Delta \mathbf{y}_t = \boldsymbol{\delta} + \Pi \mathbf{y}_{t-1} + \sum_{i=1}^{p-1} \Phi_i^* \Delta \mathbf{y}_{t-i} + \boldsymbol{\epsilon}_t$$

where Δ is the differencing operator, such that $\Delta \mathbf{y}_t = \mathbf{y}_t - \mathbf{y}_{t-1}$; $\Pi = \alpha\beta'$, where α and β are $k \times r$ matrices; and Φ_i^* is a $k \times k$ matrix.

The VECM(p) form has an equivalent VAR(p) representation as described in the section "Vector Autoregressive Model" on page 2858.

$$\mathbf{y}_t = \boldsymbol{\delta} + (I_k + \Pi + \Phi_1^*)\mathbf{y}_{t-1} + \sum_{i=2}^{p-1}(\Phi_i^* - \Phi_{i-1}^*)\mathbf{y}_{t-i} - \Phi_{p-1}^*\mathbf{y}_{t-p} + \boldsymbol{\epsilon}_t$$

where I_k is a $k \times k$ identity matrix.

An example of the second-order nonstationary vector autoregressive model is

$$\mathbf{y}_t = \begin{pmatrix} -0.2 & 0.1 \\ 0.5 & 0.2 \end{pmatrix}\mathbf{y}_{t-1} + \begin{pmatrix} 0.8 & 0.7 \\ -0.4 & 0.6 \end{pmatrix}\mathbf{y}_{t-2} + \boldsymbol{\epsilon}_t$$

with

$$\Sigma = \begin{pmatrix} 100 & 0 \\ 0 & 100 \end{pmatrix} \text{ and } \mathbf{y}_{-1} = \mathbf{y}_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

This process can be given the following VECM(2) representation with the cointegration rank one:

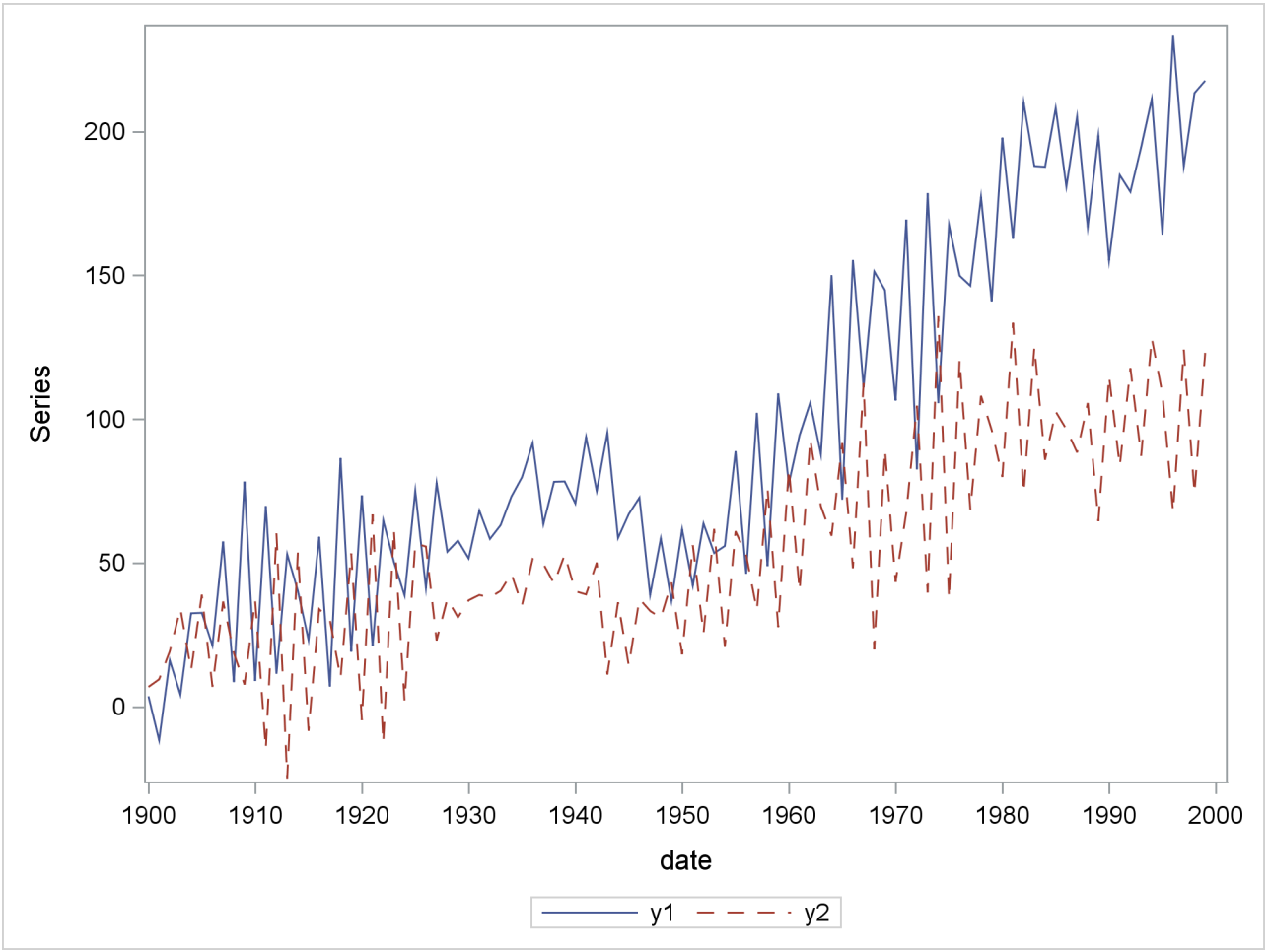
$$\Delta\mathbf{y}_t = \begin{pmatrix} -0.4 \\ 0.1 \end{pmatrix}(1, -2)\mathbf{y}_{t-1} - \begin{pmatrix} 0.8 & 0.7 \\ -0.4 & 0.6 \end{pmatrix}\Delta\mathbf{y}_{t-1} + \boldsymbol{\epsilon}_t$$

The following PROC IML statements generate simulated data for this VECM(2) form and the PROC SGPLOT statements plot the data, as shown in Figure 42.12:

```
proc iml;
   sig = 100*i(2);
   phi = {-0.2 0.1, 0.5 0.2, 0.8 0.7, -0.4 0.6};
   call varmasim(y,phi) sigma=sig n=100 initial=0
                        seed=45876;
   cn = {'y1' 'y2'};
   create simul2 from y[colname=cn];
   append from y;
quit;

data simul2;
```

```
   set simul2;
   date = intnx( 'year', '01jan1900'd, _n_-1 );
   format date year4. ;
run;

proc sgplot data=simul2;
   series x=date y=y1 / lineattrs=(pattern=solid);
   series x=date y=y2 / lineattrs=(pattern=dash);
   yaxis label="Series";
run;
```

Figure 42.12 Plot of Generated Data Process



Cointegration Testing

The following statements use the Johansen cointegration rank test. The COINTTEST=(JOHANSEN) option performs the Johansen trace test and is equivalent to specifying the COINTTEST option with no additional suboptions or specifying the COINTTEST=(JOHANSEN=(TYPE=TRACE)) option.

```
/*--- Cointegration Test ---*/

proc varmax data=simul2;
   model y1 y2 / p=2 noint dftest cointtest=(johansen);
run;
```

Figure 42.13 shows the output for Dickey-Fuller tests for the nonstationarity of each series and the Johansen cointegration rank test between series.

Figure 42.13 Dickey-Fuller Tests and Cointegration Rank Test

The VARMAX Procedure

Unit Root Test

Variable	Type	Rho	Pr < Rho	Tau	Pr < Tau
y1	Zero Mean	1.47	0.9628	1.65	0.9755
	Single Mean	-0.80	0.9016	-0.47	0.8916
	Trend	-10.88	0.3573	-2.20	0.4815
y2	Zero Mean	-0.05	0.6692	-0.03	0.6707
	Single Mean	-6.03	0.3358	-1.72	0.4204
	Trend	-50.49	0.0003	-4.92	0.0006

Cointegration Rank Test Using Trace

H0: Rank=r	H1: Rank>r	Eigenvalue	Trace	Pr > Trace	Drift in ECM	Drift in Process
0	0	0.5086	70.7279	<.0001	NOINT	Constant
1	1	0.0111	1.0921	0.3441		

In Dickey-Fuller tests, the second column specifies three types of models, which are zero mean, single mean, or trend. The third column (Rho) and the fifth column (Tau) are the test statistics that are used to test the null hypothesis that the series has a unit root. Other columns are their p-values. You can see that both series have unit roots. For a description of Dickey-Fuller tests, see the section "PROBDF Function for Dickey-Fuller Tests" on page 187 in Chapter 6, "SAS Macros and Functions."

In the "Cointegration Rank Test Using Trace" table, the last two columns explain the drift in the model or process. Because the NOINT option is specified, the model is

$$\Delta \mathbf{y}_t = \Pi \mathbf{y}_{t-1} + \Phi_1^* \Delta \mathbf{y}_{t-1} + \boldsymbol{\epsilon}_t$$

The column Drift in ECM indicates that there is no separate drift in the error correction model, and the column Drift in Process indicates that the process has a constant drift before differencing.

H0 is the null hypothesis, and H1 is the alternative hypothesis. The first row tests the cointegration rank $r = 0$ against $r > 0$, and the second row tests $r = 1$ against $r > 1$. The trace test statistics in the fourth column are computed by $-T \sum_{i=r+1}^{k} \log(1 - \lambda_i)$, where T is the available number of observations and λ_i is the eigenvalue in the third column. The p-values for these statistics are output in the fifth column. If you compare the p-value in each row to the significance level of interest (such as 5%), the null hypothesis that there is no cointegrated process (H0: $r = 0$) is rejected, whereas the null hypothesis that there is at most one cointegrated process (H0: $r = 1$) cannot be rejected.

The following statements fit a VECM(2) form to the simulated data:

```
/*--- Vector Error Correction Model ---*/

proc varmax data=simul2;
   model y1 y2 / p=2 noint lagmax=3
                 print=(iarr estimates);
   cointeg rank=1 normalize=y1;
run;
```

The results in Figure 42.13 indicate that the time series are cointegrated with rank = 1. So you might want to specify the RANK=1 option in the COINTEG statement. For normalizing the value of the cointegrated vector, you specify the normalized variable by using the NORMALIZE= option in the COINTEG statement. The COINTEG statement produces the estimates of the long-run parameter, $\boldsymbol{\beta}$, and the adjustment coefficient, $\boldsymbol{\alpha}$. The PRINT=(IARR) option provides the VAR(2) representation.

The VARMAX procedure output is shown in Figure 42.14 through Figure 42.17. In Figure 42.14, "1" indicates the first column of the $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ matrices. Because the cointegration rank is 1 in the bivariate system, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are two-dimensional vectors. The estimated cointegrating vector is $\hat{\boldsymbol{\beta}} = (1, -1.96)'$. Therefore, the long-run relationship between y_{1t} and y_{2t} is $y_{1t} = 1.96y_{2t}$. The first element of $\hat{\boldsymbol{\beta}}$ is 1 because y_1 is specified as the normalized variable. Asymptotically, $\boldsymbol{\alpha}$ follows a normal distribution, and the t values and p-values of its elements are shown in the "Alpha and Beta Parameter Estimates" table; however, because $\boldsymbol{\beta}$ follows a nonnormal distribution, the corresponding standard errors, t values, and p-values are missing. The Variable column shows the variables that correspond to the coefficients. For example, for the coefficient $\boldsymbol{\alpha}_{ij}$ (the ith element in the jth column of $\boldsymbol{\alpha}$), ALPHAi_j, the variable is the inner product of the transpose of the jth column of $\boldsymbol{\beta}$ (Beta[,j]$'$) and the vector of lag 1 dependent variables $\mathbf{y}_{t-1}$ (_DEP_(t–1)).

Figure 42.14 Parameter Estimates for the VECM(2) Form

The VARMAX Procedure

Type of Model	VECM(2)
Estimation Method	Maximum Likelihood Estimation
Cointegrated Rank	1

Beta	
Variable	**1**
y1	1.00000
y2	-1.95575

Alpha	
Variable	**1**
y1	-0.46680
y2	0.10667

Alpha and Beta Parameter Estimates

Equation	Parameter	Estimate	Standard Error	t Value	Pr > |t|	Variable
D_y1	**ALPHA1_1**	-0.46680	0.04786	-9.75	<.0001	Beta[,1]'*_DEP_(t-1)
	BETA1_1	1.00000				y1(t-1)
D_y2	**ALPHA2_1**	0.10667	0.05146	2.07	0.0409	Beta[,1]'*_DEP_(t-1)
	BETA2_1	-1.95575				y2(t-1)

Figure 42.15 shows the parameter estimates in terms of lag 1 coefficients, $\mathbf{y}_{t-1}$, and lag 1 first-differenced coefficients, $\Delta\mathbf{y}_{t-1}$, and their significance. "Alpha * Beta′" indicates the coefficients of $\mathbf{y}_{t-1}$ and is obtained by multiplying the Alpha and Beta estimates in Figure 42.14. The parameter AR1_i_j (which is shown in the "Model Parameter Estimates" table) corresponds to the elements in the "Alpha * Beta′" matrix. The parameter AR2_i_j corresponds to the elements in the differenced lagged AR coefficient matrix. The "D_" prefixed to a variable name in Figure 42.15 implies differencing.

Figure 42.15 Parameter Estimates for the VECM(2) Form, Continued

Parameter Alpha * Beta' Estimates

Variable	y1	y2
y1	-0.46680	0.91295
y2	0.10667	-0.20862

AR Coefficients of Differenced Lag

DIF Lag	Variable	y1	y2
1	y1	-0.74332	-0.74621
	y2	0.40493	-0.57157

Model Parameter Estimates

Equation	Parameter	Estimate	Standard Error	t Value	Pr > |t|	Variable
D_y1	AR1_1_1	-0.46680	0.04786	-9.75	<.0001	y1(t-1)
	AR1_1_2	0.91295	0.09359	9.75	<.0001	y2(t-1)
	AR2_1_1	-0.74332	0.04526	-16.42	<.0001	D_y1(t-1)
	AR2_1_2	-0.74621	0.04769	-15.65	<.0001	D_y2(t-1)
D_y2	AR1_2_1	0.10667	0.05146	2.07	0.0409	y1(t-1)
	AR1_2_2	-0.20862	0.10064	-2.07	0.0409	y2(t-1)
	AR2_2_1	0.40493	0.04867	8.32	<.0001	D_y1(t-1)
	AR2_2_2	-0.57157	0.05128	-11.15	<.0001	D_y2(t-1)

Figure 42.16 shows the parameter estimates of the innovations covariance matrix and their significance.

Figure 42.16 Parameter Estimates for the VECM(2) Form, Continued

Covariance Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Pr > |t|
COV1_1	94.75575	13.53654	7.00	<.0001
COV1_2	4.52684	10.30302	0.44	0.6614
COV2_2	109.57038	15.65291	7.00	<.0001

The fitted model is represented as

$$\Delta \mathbf{y}_t = \begin{pmatrix} -0.467 & 0.913 \\ (0.048) & (0.094) \\ 0.107 & -0.209 \\ (0.051) & (0.100) \end{pmatrix} \mathbf{y}_{t-1} + \begin{pmatrix} -0.743 & -0.746 \\ (0.045) & (0.048) \\ 0.405 & -0.572 \\ (0.049) & (0.051) \end{pmatrix} \Delta \mathbf{y}_{t-1} + \boldsymbol{\epsilon}_t$$

Figure 42.17 Change the VECM(2) Form to the VAR(2) Model

Infinite Order AR Representation			
Lag	Variable	y1	y2
1	y1	-0.21013	0.16674
	y2	0.51160	0.21980
2	y1	0.74332	0.74621
	y2	-0.40493	0.57157
3	y1	0.00000	0.00000
	y2	0.00000	0.00000

The PRINT=(IARR) option in the previous SAS statements prints the reparameterized coefficient estimates. Because LAGMAX=3 in those statements, the coefficient matrix of lag 3 is zero.

The VECM(2) form in Figure 42.17 can be rewritten as the following second-order vector autoregressive model:

$$\mathbf{y}_t = \begin{pmatrix} -0.210 & 0.167 \\ 0.512 & 0.220 \end{pmatrix} \mathbf{y}_{t-1} + \begin{pmatrix} 0.743 & 0.746 \\ -0.405 & 0.572 \end{pmatrix} \mathbf{y}_{t-2} + \boldsymbol{\epsilon}_t$$

Bayesian Vector Error Correction Model

Bayesian inference on a cointegrated system begins by using the priors of $\boldsymbol{\beta}$, which are obtained from the VECM(p) form. Bayesian vector error correction models can improve forecast accuracy for cointegrated processes.

To use a Bayesian vector error correction model, you specify both the PRIOR= option in the MODEL statement and the COINTEG statement. The following statements fit a BVECM(2) form to the simulated data:

```
/*--- Bayesian Vector Error Correction Model ---*/

proc varmax data=simul2;
   model y1 y2 / p=2 noint
                 prior=( lambda=0.5 theta=0.2 )
                 print=(estimates);
   cointeg rank=1 normalize=y1;
run;
```

The VARMAX procedure output in Figure 42.18 shows the model type fitted to the data, the estimates of the adjustment coefficient ($\boldsymbol{\alpha}$), the parameter estimates in terms of lag 1 coefficients ($\mathbf{y}_{t-1}$), and lag 1 first-differenced coefficients ($\Delta\mathbf{y}_{t-1}$).

Figure 42.18 Parameter Estimates for the BVECM(2) Form

The VARMAX Procedure

Type of Model	BVECM(2)
Estimation Method	Maximum Likelihood Estimation
Cointegrated Rank	1
Prior Lambda	0.5
Prior Theta	0.2

Alpha	
Variable	**1**
y1	-0.34392
y2	0.16659

Parameter Alpha * Beta' Estimates		
Variable	**y1**	**y2**
y1	-0.34392	0.67262
y2	0.16659	-0.32581

AR Coefficients of Differenced Lag			
DIF Lag	**Variable**	**y1**	**y2**
1	**y1**	-0.80070	-0.59320
	y2	0.33417	-0.53480

Vector Autoregressive Model with Exogenous Variables

A VAR process can be affected by other observable variables that are determined outside the system of interest. Such variables are called exogenous (independent) variables. Exogenous variables can be stochastic or nonstochastic. The process can also be affected by the lags of exogenous variables. A model used to describe this process is called a VARX(p,s) model.

The VARX(p,s) model is written as

$$\mathbf{y}_t = \boldsymbol{\delta} + \sum_{i=1}^{p} \Phi_i \mathbf{y}_{t-i} + \sum_{i=0}^{s} \Theta_i^* \mathbf{x}_{t-i} + \boldsymbol{\epsilon}_t$$

where $\mathbf{x}_t = (x_{1t}, \ldots, x_{rt})'$ is an r-dimensional time series vector and Θ_i^* is a $k \times r$ matrix.

For example, a VARX(1,0) model is

$$\mathbf{y}_t = \boldsymbol{\delta} + \Phi_1 \mathbf{y}_{t-1} + \Theta_0^* \mathbf{x}_t + \boldsymbol{\epsilon}_t$$

where $\mathbf{y}_t = (y_{1t}, y_{2t}, y_{3t})'$ and $\mathbf{x}_t = (x_{1t}, x_{2t})'$.

The following statements fit the VARX(1,0) model to the given data:

```
data grunfeld;
   input year y1 y2 y3 x1 x2 x3;
   label y1='Gross Investment GE'
         y2='Capital Stock Lagged GE'
         y3='Value of Outstanding Shares GE Lagged'
         x1='Gross Investment W'
         x2='Capital Stock Lagged W'
         x3='Value of Outstanding Shares Lagged W';
datalines;
1935  33.1 1170.6  97.8 12.93  191.5   1.8
1936  45.0 2015.8 104.4 25.90  516.0    .8
1937  77.2 2803.3 118.0 35.05  729.0   7.4
1938  44.6 2039.7 156.2 22.89  560.4  18.1

   ... more lines ...


/*--- Vector Autoregressive Process with Exogenous Variables ---*/

proc varmax data=grunfeld;
   model y1-y3 = x1 x2 / p=1 lagmax=5
                         printform=univariate
                         print=(impulsx=(all) estimates);
run;
```

The VARMAX procedure output is shown in Figure 42.19 through Figure 42.21.

Figure 42.19 shows the descriptive statistics for the dependent (endogenous) and independent (exogenous) variables with labels.

Figure 42.19 Descriptive Statistics for the VARX(1, 0) Model

The VARMAX Procedure

Number of Observations	20
Number of Pairwise Missing	0

Simple Summary Statistics

Variable	Type	N	Mean	Standard Deviation	Min	Max	Label
y1	**Dependent**	20	102.29000	48.58450	33.10000	189.60000	Gross Investment GE
y2	**Dependent**	20	1941.32500	413.84329	1170.60000	2803.30000	Capital Stock Lagged GE
y3	**Dependent**	20	400.16000	250.61885	97.80000	888.90000	Value of Outstanding Shares GE Lagged
x1	**Independent**	20	42.89150	19.11019	12.93000	90.08000	Gross Investment W
x2	**Independent**	20	670.91000	222.39193	191.50000	1193.50000	Capital Stock Lagged W

Figure 42.20 shows the parameter estimates for the constant, the lag zero coefficients of exogenous variables, and the lag one AR coefficients. From the schematic representation of parameter estimates, the significance of the parameter estimates can be easily verified. The symbol "C" means the constant and "XL0" means the lag zero coefficients of exogenous variables.

Figure 42.20 Parameter Estimates for the VARX(1, 0) Model

The VARMAX Procedure

Type of Model	VARX(1,0)
Estimation Method	Least Squares Estimation

Constant

Variable	Constant
y1	-12.01279
y2	702.08673
y3	-22.42110

XLag

Lag	Variable	x1	x2
0	**y1**	1.69281	-0.00859
	y2	-6.09850	2.57980
	y3	-0.02317	-0.01274

AR

Lag	Variable	y1	y2	y3
1	**y1**	0.23699	0.00763	0.02941
	y2	-2.46656	0.16379	-0.84090
	y3	0.95116	0.00224	0.93801

Schematic Representation

Variable/Lag	C	XL0	AR1
y1	.	+.	...
y2	+	.+	...
y3	-	..	+.+

+ is > 2*std error, - is < -2*std error, . is between, * is N/A

Figure 42.21 shows the parameter estimates and their significance.

Figure 42.21 Parameter Estimates for the VARX(1, 0) Model Continued

Model Parameter Estimates

Equation	Parameter	Estimate	Standard Error	t Value	Pr > |t|	Variable
y1	CONST1	-12.01279	27.47108	-0.44	0.6691	1
	XL0_1_1	1.69281	0.54395	3.11	0.0083	x1(t)
	XL0_1_2	-0.00859	0.05361	-0.16	0.8752	x2(t)
	AR1_1_1	0.23699	0.20668	1.15	0.2722	y1(t-1)
	AR1_1_2	0.00763	0.01627	0.47	0.6470	y2(t-1)
	AR1_1_3	0.02941	0.04852	0.61	0.5548	y3(t-1)
y2	CONST2	702.08673	256.48046	2.74	0.0169	1
	XL0_2_1	-6.09850	5.07849	-1.20	0.2512	x1(t)
	XL0_2_2	2.57980	0.50056	5.15	0.0002	x2(t)
	AR1_2_1	-2.46656	1.92967	-1.28	0.2235	y1(t-1)
	AR1_2_2	0.16379	0.15193	1.08	0.3006	y2(t-1)
	AR1_2_3	-0.84090	0.45304	-1.86	0.0862	y3(t-1)
y3	CONST3	-22.42110	10.31166	-2.17	0.0487	1
	XL0_3_1	-0.02317	0.20418	-0.11	0.9114	x1(t)
	XL0_3_2	-0.01274	0.02012	-0.63	0.5377	x2(t)
	AR1_3_1	0.95116	0.07758	12.26	0.0001	y1(t-1)
	AR1_3_2	0.00224	0.00611	0.37	0.7201	y2(t-1)
	AR1_3_3	0.93801	0.01821	51.50	0.0001	y3(t-1)

The fitted model is given as

$$\begin{pmatrix} y_{1t} \\ y_{2t} \\ y_{3t} \end{pmatrix} = \begin{pmatrix} -12.013 \\ (27.471) \\ 702.086 \\ (256.480) \\ -22.421 \\ (10.312) \end{pmatrix} + \begin{pmatrix} 1.693 & -0.009 \\ (0.544) & (0.054) \\ -6.099 & 2.580 \\ (5.078) & (0.501) \\ -0.023 & -0.013 \\ (0.204) & (0.020) \end{pmatrix} \begin{pmatrix} x_{1t} \\ x_{2t} \end{pmatrix}$$

$$+ \begin{pmatrix} 0.237 & 0.008 & 0.029 \\ (0.207) & (0.016) & (0.049) \\ -2.467 & 0.164 & -0.841 \\ (1.930) & (0.152) & (0.453) \\ 0.951 & 0.002 & 0.938 \\ (0.078) & (0.006) & (0.018) \end{pmatrix} \begin{pmatrix} y_{1,t-1} \\ y_{2,t-1} \\ y_{3,t-1} \end{pmatrix} + \begin{pmatrix} \epsilon_{1t} \\ \epsilon_{2t} \\ \epsilon_{3t} \end{pmatrix}$$

Parameter Estimation and Testing on Restrictions

In the previous example, the VARX(1,0) model is written as

$$\mathbf{y}_t = \boldsymbol{\delta} + \Theta_0^* \mathbf{x}_t + \Phi_1 \mathbf{y}_{t-1} + \boldsymbol{\epsilon}_t$$

with

$$\Theta_0^* = \begin{pmatrix} \theta_{11}^* & \theta_{12}^* \\ \theta_{21}^* & \theta_{22}^* \\ \theta_{31}^* & \theta_{32}^* \end{pmatrix} \quad \Phi_1 = \begin{pmatrix} \phi_{11} & \phi_{12} & \phi_{13} \\ \phi_{21} & \phi_{22} & \phi_{23} \\ \phi_{31} & \phi_{32} & \phi_{33} \end{pmatrix}$$

In Figure 42.21 of the preceding section, you can see several insignificant parameters. For example, the coefficients XL0_1_2, AR1_1_2, and AR1_3_2 are insignificant.

The following statements restrict the coefficients of $\theta_{12}^* = \phi_{12} = \phi_{32} = 0$ for the VARX(1,0) model.

```
/*--- Models with Restrictions and Tests ---*/

proc varmax data=grunfeld;
   model y1-y3 = x1 x2 / p=1 print=(estimates);
   restrict XL(0,1,2)=0, AR(1,1,2)=0, AR(1,3,2)=0;
run;
```

The output in Figure 42.22 shows that three parameters θ_{12}^*, ϕ_{12}, and ϕ_{32} are replaced by the restricted values, zeros, and their standard errors are also zeros to indicate that the parameters are fixed to these values.

Figure 42.22 Parameter Estimation with Restrictions

The VARMAX Procedure

Model Parameter Estimates

Equation	Parameter	Estimate	Standard Error	t Value	Pr > |t|	Variable
y1	CONST1	-2.16781	13.13755	-0.17	0.8715	1
	XL0_1_1	1.67592	0.40792	4.11	0.0012	x1(t)
	XL0_1_2	0.00000	0.00000			x2(t)
	AR1_1_1	0.27671	0.17606	1.57	0.1401	y1(t-1)
	AR1_1_2	0.00000	0.00000			y2(t-1)
	AR1_1_3	0.01747	0.03519	0.50	0.6279	y3(t-1)
y2	CONST2	768.14598	224.12735	3.43	0.0045	1
	XL0_2_1	-6.30880	4.85729	-1.30	0.2166	x1(t)
	XL0_2_2	2.65308	0.43840	6.05	0.0001	x2(t)
	AR1_2_1	-2.16968	1.83550	-1.18	0.2584	y1(t-1)
	AR1_2_2	0.10945	0.11751	0.93	0.3686	y2(t-1)
	AR1_2_3	-0.93053	0.41478	-2.24	0.0429	y3(t-1)
y3	CONST3	-19.88165	7.69575	-2.58	0.0227	1
	XL0_3_1	-0.03576	0.20079	-0.18	0.8614	x1(t)
	XL0_3_2	-0.00919	0.01747	-0.53	0.6076	x2(t)
	AR1_3_1	0.96398	0.06907	13.96	0.0001	y1(t-1)
	AR1_3_2	0.00000	0.00000			y2(t-1)
	AR1_3_3	0.93412	0.01473	63.41	0.0001	y3(t-1)

The output in Figure 42.23 shows the estimates of the Lagrangian parameters and their significance. Based on the p-values associated with the Lagrangian parameters, you cannot reject the null hypotheses $\theta_{12}^* = 0$, $\phi_{12} = 0$, and $\phi_{32} = 0$ with the 0.05 significance level.

Figure 42.23 RESTRICT Statement Results

Testing of the Restricted Parameters

Parameter	Estimate	Standard Error	t Value	Pr > |t|	Equation
Restrict0	1.74969	21.44026	0.08	0.9389	XL0_1_2 = 0
Restrict1	30.36254	70.74347	0.43	0.6899	AR1_1_2 = 0
Restrict2	55.42191	164.03075	0.34	0.7524	AR1_3_2 = 0

The TEST statement in the following example tests $\phi_{31} = 0$ and $\theta_{12}^{*} = \phi_{12} = \phi_{32} = 0$ for the VARX(1,0) model:

```
proc varmax data=grunfeld;
   model y1-y3 = x1 x2 / p=1;
   test AR(1,3,1)=0;
   test XL(0,1,2)=0, AR(1,1,2)=0, AR(1,3,2)=0;
run;
```

The output in Figure 42.24 shows that the first column in the output is the index corresponding to each TEST statement. You can reject the hypothesis test $\phi_{31} = 0$ at the 0.05 significance level, but you cannot reject the joint hypothesis test $\theta_{12}^{*} = \phi_{12} = \phi_{32} = 0$ at the 0.05 significance level.

Figure 42.24 TEST Statement Results

The VARMAX Procedure

Testing of the Parameters

Test	DF	Chi-Square	Pr > ChiSq
1	1	150.31	<.0001
2	3	0.34	0.9522

Causality Testing

The following statements use the CAUSAL statement to compute the Granger causality test for a VAR(1) model. For the Granger causality tests, the autoregressive order should be defined by the P= option in the MODEL statement. The variable groups are defined in the CAUSAL statement as well. Regardless of whether the variables specified in the GROUP1= and GROUP2= options are designated as dependent or exogenous (independent) variables in the MODEL statement, the CAUSAL statement fits the VAR(p) model by considering the variables in the two groups as dependent variables.

```
/*--- Causality Testing ---*/

proc varmax data=grunfeld;
   model y1-y3 = x1 x2 / p=1 noprint;
   causal group1=(x1) group2=(y1-y3);
   causal group1=(y3) group2=(y1 y2);
run;
```

The output in Figure 42.25 is associated with the CAUSAL statement. The first CAUSAL statement fits the VAR(1) model by using the variables $y1$, $y2$, $y3$, and $x1$. The second CAUSAL statement fits the VAR(1) model by using the variables $y1$, $y3$, and $y2$.

Figure 42.25 CAUSAL Statement Results

The VARMAX Procedure

Granger-Causality Wald Test			
Test	**DF**	**Chi-Square**	**Pr > ChiSq**
1	3	2.40	0.4946
2	2	262.88	<.0001

Test 1: Group 1 Variables: x1
Group 2 Variables: y1 y2 y3

Test 2: Group 1 Variables: y3
Group 2 Variables: y1 y2

The null hypothesis of the Granger causality test is that GROUP1 is influenced only by itself, and not by GROUP2.

The first column in the output is the index corresponding to each CAUSAL statement. The output shows that you cannot reject that $x1$ is influenced by itself and not by $(y1, y2, y3)$ at the 0.05 significance level for Test 1. You can reject that $y3$ is influenced by itself and not by $(y1, y2)$ for Test 2. See the section "VAR and VARX Modeling" on page 2963 for details.

Multivariate GARCH Models

Modeling and forecasting the volatility of time series has been the focus of many researchers and practitioners, especially in the fields of risk management, portfolio optimization, and asset pricing. One of the most powerful tools for volatility modeling is the autoregressive conditional heteroscedasticity (ARCH) model proposed by Engle (1982) and extended by Bollerslev (1986) to the generalized autoregressive conditional heteroscedasticity (GARCH) model. The VARMAX procedure supports three forms of multivariate GARCH models: BEKK, CCC, and DCC. This section shows some examples of how to specify, estimate, and compare various forms of multivariate GARCH models.

Data about two indices, the Dow Jones Industrial Average index and the Standard & Poor's 500 index, are obtained from Yahoo Finance and used in this section. The sample contains daily data from February 16, 2005, to February 13, 2015. The following statements input the daily prices and then generate the daily returns:

```
data indices;
   input date : MMDDYY10. DJIA SP500;
   logDJIA = log(DJIA); logSP500 = log(SP500);
   rDJIA = (logDJIA-lag(logDJIA))*100;
   rSP500 = (logSP500-lag(logSP500))*100;
datalines;
2/16/2005    10834.88    1210.34
2/17/2005    10754.26    1200.75
2/18/2005    10785.22    1201.59
```

```
   ... more lines ...

2/12/2015    17972.38    2088.48
2/13/2015    18019.35    2096.99
;
```

To model the volatility of bivariate returns, rDJIA and rSP500, you can start with the BEKK GARCH(1,1) model. The following equations describe the bivariate BEKK GARCH(1,1) model:

$$
\begin{aligned}
r_t &= H_t^{\frac{1}{2}} \epsilon_t \\
H_t &= C + A_1' r_{t-1} r_{t-1}' A_1 + G_1' H_{t-1} G_1 \\
&= \begin{bmatrix} c_{11} & c_{12} \\ c_{12} & c_{22} \end{bmatrix} + \begin{bmatrix} a_{11,1} & a_{12,1} \\ a_{21,1} & a_{22,1} \end{bmatrix}' \begin{bmatrix} r_{1,t-1} \\ r_{2,t-1} \end{bmatrix} \begin{bmatrix} r_{1,t-1} \\ r_{2,t-1} \end{bmatrix}' \begin{bmatrix} a_{11,1} & a_{12,1} \\ a_{21,1} & a_{22,1} \end{bmatrix} \\
&+ \begin{bmatrix} g_{11,1} & g_{12,1} \\ g_{21,1} & g_{22,1} \end{bmatrix}' \begin{bmatrix} h_{11,t-1} & h_{12,t-1} \\ h_{12,t-1} & h_{22,t-1} \end{bmatrix} \begin{bmatrix} g_{11,1} & g_{12,1} \\ g_{21,1} & g_{22,1} \end{bmatrix}
\end{aligned}
$$

In these equations, r_t is the vector of returns at time t, H_t is the conditional covariance matrix of r_t, $H_t^{\frac{1}{2}}$ denotes the square root of H_t such that the square of matrix $H_t^{\frac{1}{2}}$ is H_t, ϵ_t is the innovation at time t and follows an iid bivariate standard normal distribution, C is a 2×2 symmetric parameter matrix, A_1 is a 2×2 full parameter matrix for the first lag of the ARCH term, and G_1 is a 2×2 full parameter matrix for the first lag of the GARCH term. Hence, there are 11 parameters in total for a bivariate BEKK GARCH(1,1) model; that is, a vector $\left(c_{11}, c_{12}, c_{22}, a_{11,1}, a_{21,1}, a_{12,1}, a_{22,1}, g_{11,1}, g_{21,1}, g_{12,1}, g_{22,1}\right)'$.

You can use the FORM=BEKK option in the GARCH statement to specify the BEKK GARCH form, or you can omit this option because BEKK is the default value for the FORM= option. The Q= option in the GARCH statement specifies the lags of the ARCH terms, and the P= option in the GARCH statement specifies the lags of the GARCH terms. The forecasts of conditional covariance matrices are output to a SAS data set when you specify the OUTHT= option in the GARCH statement. The parameter estimates and their covariance matrix are output to a SAS data set when you specify the OUTEST= option together with the OUTCOV option in the PROC VARMAX statement.

The following statement specifies the BEKK GARCH(1,1) model:

```
/*--- BEKK ---*/

proc varmax data=indices outest=oebekk outcov;
   model rDJIA rSP500 / noint;
   garch p=1 q=1 form=bekk outht=ohbekk;
run;
```

Figure 42.26 shows the log likelihood and the information criteria. They are used later in the model comparison.

Figure 42.26 BEKK GARCH Log Likelihood and Information Criteria

The VARMAX Procedure

Log-likelihood	1360.976

Information Criteria	
AICC	-2699.85
HQC	-2676.68
AIC	-2699.95
SBC	-2635.82
FPEC	0.080617

Figure 42.27 shows the parameters estimates for the BEKK GARCH(1,1) model. For the constant term C, **GCHC***i_j*, $i, j = 1, 2$, correspond to parameters c_{ij}, respectively. Because C is symmetric, **GCHC2_1** is omitted. For the ARCH and GARCH terms, **ACH***l_i_j*, $l = 1, i, j = 1, 2$, correspond to $a_{ij,l}$, respectively, and **GCH***l_i_j*, $l = 1, i, j = 1, 2$, correspond to $g_{ij,l}$, respectively.

Figure 42.27 BEKK GARCH Parameter Estimates

GARCH Model Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Pr > |t|
GCHC1_1	0.19101	0.00000		
GCHC1_2	0.09343	0.00000		
GCHC2_2	0.00000	0.01807	0.00	1.0000
ACH1_1_1	0.27518	0.13503	2.04	0.0417
ACH1_2_1	0.20619	0.11122	1.85	0.0639
ACH1_1_2	0.24907	0.11982	2.08	0.0377
ACH1_2_2	0.23448	0.09739	2.41	0.0161
GCH1_1_1	0.11391	0.10984	1.04	0.2998
GCH1_2_1	0.64841	0.11363	5.71	0.0001
GCH1_1_2	0.75455	0.11263	6.70	0.0001
GCH1_2_2	0.20598	0.11520	1.79	0.0739

As shown in Figure 42.27, the standard errors of **GCHC1_1** and **GCHC1_2** are both zeros. It might be a sign that the numerical optimization for the BEKK GARCH model converges to a local minimum instead of the global minimum, which often happens for nonlinear optimization of complex models that have many parameters. A possible way to solve this problem is to try different initial values. To obtain reasonable initial values, the following statements fit a diagonal BEKK GARCH model (that is, a restricted BEKK GARCH model in which the ARCH and GARCH parameter matrices are diagonal):

```
/*--- Diagonal BEKK ---*/

proc varmax data=indices outest=oebekk outcov;
   model rDJIA rSP500 / noint;
   garch p=1 q=1 form=bekk;
   restrict ach(1,1,2), ach(1,2,1), gch(1,1,2), gch(1,2,1);
run;
```

The parameter estimates of the diagonal BEKK GARCH model are shown in Figure 42.28. As expected, the standard errors of the off-diagonal elements of the ARCH and GARCH parameter matrices (namely **ACH1_1_2**, **ACH1_2_1**, **GCH1_1_2**, and **GCH1_2_1**) are all zeros because they are restricted in the RESTRICT statement. All other parameters have valid standard errors.

Figure 42.28 Diagonal BEKK GARCH Parameter Estimates

The VARMAX Procedure

GARCH Model Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Pr > |t|
GCHC1_1	0.01407	0.00254	5.53	0.0001
GCHC1_2	0.01446	0.00262	5.51	0.0001
GCHC2_2	0.01598	0.00299	5.34	0.0001
ACH1_1_1	0.25702	0.01251	20.54	0.0001
ACH1_2_1	0.00000	0.00000		
ACH1_1_2	0.00000	0.00000		
ACH1_2_2	0.26061	0.01302	20.02	0.0001
GCH1_1_1	-0.95794	0.00413	-231.85	0.0001
GCH1_2_1	0.00000	0.00000		
GCH1_1_2	0.00000	0.00000		
GCH1_2_2	-0.95694	0.00443	-216.10	0.0001

Figure 42.29 shows the log likelihood and the information criteria. The log likelihood for the diagonal BEKK GARCH model is larger than that of the previous estimated BEKK GARCH model (which is shown in Figure 42.26). The larger value confirms that the previous BEKK GARCH model does not converge to the global minimium.

Figure 42.29 Diagonal BEKK GARCH Log Likelihood and Information Criteria

Log-likelihood	1520.235

Information Criteria	
AICC	-3026.43
HQC	-3011.66
AIC	-3026.47
SBC	-2985.66
FPEC	0.080617

The following statements reestimate the BEKK GARCH model whose initial values are parameter estimates of the diagonal BEKK GARCH model (which are shown in Figure 42.28):

```
/*--- BEKK with Initial Values ---*/

proc varmax data=indices outest=oebekk outcov;
   model rDJIA rSP500 / noint;
   garch p=1 q=1 form=bekk;
   initial gchc(1,1)=0.01407, gchc(1,2)=0.01446, gchc(2,2)=0.01598,
```

```
            ach(1,1,1)=0.25702, ach(1,2,2)=0.26061,
            gch(1,1,1)=-0.95794, gch(1,2,2)=-0.95694,
            ach(1,1,2), ach(1,2,1), gch(1,1,2), gch(1,2,1);
run;
```

The parameter estimates of the reestimated BEKK GARCH models are shown in Figure 42.30. The standard errors of all parameters are valid.

Figure 42.30 Reestimated BEKK GARCH Parameter Estimates

The VARMAX Procedure

GARCH Model Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Pr > |t|
GCHC1_1	0.01999	0.00394	5.07	0.0001
GCHC1_2	0.02043	0.00391	5.22	0.0001
GCHC2_2	0.02112	0.00408	5.18	0.0001
ACH1_1_1	0.07178	0.10153	0.71	0.4796
ACH1_2_1	0.22679	0.09285	2.44	0.0147
ACH1_1_2	-0.09556	0.11262	-0.85	0.3962
ACH1_2_2	0.41214	0.10167	4.05	0.0001
GCH1_1_1	-0.95018	0.03580	-26.55	0.0001
GCH1_2_1	0.01069	0.03266	0.33	0.7434
GCH1_1_2	0.03746	0.04018	0.93	0.3513
GCH1_2_2	-0.97038	0.03589	-27.04	0.0001

Figure 42.31 shows the log likelihood and information criteria of the reestimated BEKK GARCH model. As expected, the log likelihood of the reestimated BEKK GARCH model is larger than that of the diagonal BEKK GARCH model. Moreover, the reestimated BEKK GARCH model has a smaller SBC, compared to the SBC of the diagonal BEKK GARCH model (which is shown in Figure 42.29). The smaller SBC means that the BEKK GARCH model should be chosen instead of the diagonal BEKK GARCH model.

Figure 42.31 Reestimated BEKK GARCH Log Likelihood and Information Criteria

Log-likelihood	1542.362

Information Criteria	
AICC	-3062.62
HQC	-3039.45
AIC	-3062.72
SBC	-2998.59
FPEC	0.080617

The number of parameters for a BEKK GARCH model increases very quickly as the number of dependent variables increases. There are $(p+q)k^2 + k(k+1)/2$ parameters for a k-variate BEKK GARCH(p, q) model. For example, a 16-variate BEKK GARCH(1,1) model has 648 parameters to be estimated.

Compared with BEKK GARCH models, CCC GARCH models are much more parsimonious. In a CCC GARCH model, each series follows a GARCH process and their composition through the constant conditional

correlation matrix constructs the conditional covariance matrices. A bivariate CCC GARCH(1,1) has the form

$$
\begin{aligned}
r_t &= H_t^{\frac{1}{2}} \epsilon_t \\
H_t &= D_t S D_t \\
D_t &= \begin{bmatrix} \sqrt{h_{11,t}} & 0 \\ 0 & \sqrt{h_{22,t}} \end{bmatrix} \\
S &= \begin{bmatrix} 1 & s_{12} \\ s_{12} & 1 \end{bmatrix} \\
h_{11,t} &= c_{11} + a_{11,1} r_{1,t-1}^2 + g_{11,1} h_{11,t-1} \\
h_{22,t} &= c_{22} + a_{22,1} r_{2,t-1}^2 + g_{22,1} h_{22,t-1}
\end{aligned}
$$

where D_t is the diagonal matrix with conditional standard deviations and S is the time-invariant conditional correlation matrix. Hence, there are seven parameters to be estimated; that is, a vector $\left(s_{12}, c_{11}, c_{22}, a_{11,1}, a_{22,1}, g_{11,1}, g_{22,1}\right)'$. A k-variate CCC GARCH(p, q) model has $(p + q + 1)k + k(k-1)/2$ parameters. For example, a 16-variate CCC GARCH(1,1) model has 168 parameters to be estimated, about $1/4$ of the number that a BEKK GARCH(1,1) model has.

The following statements estimate a CCC GARCH(1,1) model:

```
/*--- CCC ---*/

proc varmax data=indices outest=oeccc outcov;
   model rDJIA rSP500 / noint;
   garch p=1 q=1 form=ccc outht=ohccc;
run;
```

Figure 42.32 shows the parameter estimates for the CCC GARCH(1,1) model. For the constant conditional correlation matrix S, `CCC1_2` corresponds to the parameter s_{12}.

Figure 42.32 CCC GARCH Parameter Estimates

The VARMAX Procedure

GARCH Model Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Pr > |t|
CCC1_2	0.97294	0.00109	890.75	0.0001
GCHC1_1	0.03713	0.00504	7.37	0.0001
GCHC2_2	0.04004	0.00536	7.47	0.0001
ACH1_1_1	0.06862	0.00737	9.31	0.0001
ACH1_2_2	0.06684	0.00690	9.68	0.0001
GCH1_1_1	0.88472	0.01183	74.81	0.0001
GCH1_2_2	0.88916	0.01099	80.92	0.0001

Figure 42.33 shows the log likelihood and the information criteria. Compared to the SBC for the BEKK GARCH model (shown in Figure 42.31), the SBC for the CCC GARCH model is much larger, which means the CCC GARCH model should not be preferred.

Figure 42.33 CCC GARCH Log Likelihood and Information Criteria

Log-likelihood	1474.578

Information Criteria	
AICC	-2935.11
HQC	-2920.34
AIC	-2935.16
SBC	-2894.34
FPEC	0.080617

The CCC GARCH model is not preferred over the BEKK GARCH model in this case because the basic assumption in the CCC GARCH model—that the conditional correlation matrix is time-invarying—might not hold. A DCC GARCH model relaxes this assumption and models the time-varying conditional correlation matrix in an ARMA form. A bivariate DCC GARCH(1,1) has the form

$$
\begin{aligned}
r_t &= H_t^{\frac{1}{2}} \boldsymbol{\epsilon}_t \\
H_t &= D_t S_t D_t \\
D_t &= \begin{bmatrix} \sqrt{h_{11,t}} & 0 \\ 0 & \sqrt{h_{22,t}} \end{bmatrix} \\
h_{11,t} &= c_{11} + a_{11,1} r_{1,t-1}^2 + g_{11,1} h_{11,t-1} \\
h_{22,t} &= c_{11} + a_{22,1} r_{2,t-1}^2 + g_{22,1} h_{22,t-1} \\
S_t &= \begin{bmatrix} 1 & s_{12,t} \\ s_{12,t} & 1 \end{bmatrix} \\
s_{12,t} &= \frac{q_{12,t}}{\sqrt{q_{11,t} q_{22,t}}} \\
q_{12,t} &= (1-\alpha-\beta) s_{12} + \alpha \frac{r_{1,t-1}}{\sqrt{h_{11,t-1}}} \frac{r_{2,t-1}}{\sqrt{h_{22,t-1}}} + \beta q_{12,t-1} \\
q_{11,t} &= (1-\alpha-\beta) + \alpha \frac{r_{1,t-1}^2}{h_{11,t-1}} + \beta q_{11,t-1} \\
q_{22,t} &= (1-\alpha-\beta) + \alpha \frac{r_{2,t-1}^2}{h_{22,t-1}} + \beta q_{22,t-1}
\end{aligned}
$$

where S_t is the time-varying conditional correlation matrix at time t. Compared to the CCC GARCH model, two more parameters, α and β, are added into the DCC GARCH model. There are nine parameters in total; that is, a vector $\left(\alpha, \beta, s_{12}, c_{11}, c_{22}, a_{11,1}, a_{22,1}, g_{11,1}, g_{22,1}\right)'$.

The following statements estimate a DCC GARCH model:

```
/*--- DCC ---*/

proc varmax data=indices outest=oedcc outcov;
   model rDJIA rSP500 / noint;
   garch p=1 q=1 form=dcc outht=ohdcc;
run;
```

Figure 42.34 shows the parameter estimates for the DCC GARCH(1,1) model. `DCCA` corresponds to the parameter α, `DCCB` corresponds to the parameter β, and `DCCS1_2` corresponds to the parameter s_{12}, the off-diagonal element in the unconditional correlation matrix. The standard errors of many parameter estimates are zeros, which might be a sign that the model does not converge to the global minimum.

Figure 42.34 DCC GARCH Parameter Estimates

The VARMAX Procedure

GARCH Model Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Pr > |t|
DCCA	0.01540	0.00000		
DCCB	0.00000	0.00000		
DCCS1_2	0.98743	0.00040	999.00	0.0001
GCHC1_1	1.28530	0.00000		
GCHC2_2	1.50117	0.00000		
ACH1_1_1	0.03378	0.00216	15.62	0.0001
ACH1_2_2	0.02694	0.00084	32.07	0.0001
GCH1_1_1	0.07596	0.00000		
GCH1_2_2	0.09939	0.00000		

Figure 42.35 shows the log likelihood and the information criteria.

Figure 42.35 DCC GARCH Log Likelihood and Information Criteria

Log-likelihood	700.3131

Information Criteria	
AICC	-1382.55
HQC	-1363.58
AIC	-1382.63
SBC	-1330.16
FPEC	0.080617

Because a CCC GARCH model can be regarded as a restricted DCC GARCH model in which α and β in the conditional correlation equations are restricted to zeros, it is expected that the log likelihood of the "unrestricted" DCC GARCH model should always be larger than (or at least equal to) the log likelihood of the

corresponding CCC GARCH model, even though DCC might have a larger information criterion and not be chosen. Hence, it is unexpected that the log likelihood of the DCC GARCH model (shown in Figure 42.35) is smaller than that of the CCC GARCH model (shown in Figure 42.33). This unexpected phenomenon confirms that the numerical optimization for the DCC GARCH model converges to a local minimum instead of the global minimum. Different initial values should be tried. In addition to some reasonable values for parameters DCCA and DCCB, the INITIAL statement specifies the initial values for the DCC GARCH model parameters in the following statements; these values are based on the corresponding parameter estimates of the CCC GARCH model (shown in Figure 42.32):

```
/*--- DCC with Initial Values ---*/

proc varmax data=indices outest=oedcc outcov;
   model rDJIA rSP500 / noint;
   garch p=1 q=1 form=dcc outht=ohdcc;
   initial DCCA=0.01, DCCB=0.98, DCCS(1,2)  = 0.97294,
           GCHC(1,1)  = 0.03713, GCHC(2,2)  = 0.04004,
           ACH(1,1,1) = 0.06862, ACH(1,2,2) = 0.06684,
           GCH(1,1,1) = 0.88472, GCH(1,2,2) = 0.88916;
run;
```

Figure 42.36 shows the parameter estimates for the reestimated DCC GARCH(1,1) model. All standard errors of parameter estimates are valid.

Figure 42.36 Reestimated DCC GARCH Parameter Estimates

The VARMAX Procedure

GARCH Model Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Pr > |t|
DCCA	0.03802	0.00634	6.00	0.0001
DCCB	0.93782	0.01084	86.49	0.0001
DCCS1_2	0.97401	0.00247	394.22	0.0001
GCHC1_1	0.02193	0.00370	5.93	0.0001
GCHC2_2	0.02395	0.00401	5.97	0.0001
ACH1_1_1	0.07842	0.00787	9.97	0.0001
ACH1_2_2	0.07758	0.00770	10.07	0.0001
GCH1_1_1	0.89540	0.01046	85.58	0.0001
GCH1_2_2	0.89738	0.01012	88.64	0.0001

As shown in Figure 42.37, the log likelihood of the DCC GARCH model increases dramatically. Compared to the SBC of the CCC GARCH model (shown in Figure 42.33), the SBC for the DCC GARCH model is much smaller, and the DCC GARCH model is chosen. However, compared to the SBC for the BEKK GARCH model (shown in Figure 42.31), the SBC for the DCC GARCH model is a little larger, The BEKK GARCH model should be chosen although it has two more parameters than the DCC GARCH model.

Figure 42.37 Reestimated DCC GARCH Log Likelihood and Information Criteria

Log-likelihood	1531.454

Information Criteria	
AICC	-3044.84
HQC	-3025.86
AIC	-3044.91
SBC	-2992.44
FPEC	0.080617

Compared to the BEKK GARCH model, in addition to parsimony, another advantage of the DCC (and also the CCC) GARCH model is that you can use subforms other than GARCH to model the conditional covariance of each series. For example, you can use the threshold GARCH (TGARCH) model for modeling the conditional covariances of rDJIA and rSP500. A bivariate DCC TGARCH(1,1) has the same form as the bivariate DCC GARCH(1,1) except that the conditional covariance equations are replaced by

$$
\begin{aligned}
h_{11,t} &= c_{11} + a_{11,1} r_{1,t-1}^2 + \mathbf{1}_{r_{1,t-1}<0} b_{11,1} r_{1,t-1}^2 + g_{11,1} h_{11,t-1} \\
h_{22,t} &= c_{11} + a_{22,1} r_{2,t-1}^2 + \mathbf{1}_{r_{2,t-1}<0} b_{22,1} r_{2,t-1}^2 + g_{22,1} h_{22,t-1}
\end{aligned}
$$

where the indicator function $\mathbf{1}_A$ is 1 if A is true and 0 otherwise. Compared to the DCC GARCH model, two more parameters, $b_{11,1}$ and $b_{22,1}$, are added to the DCC TGARCH model to catch the so-called leverage effect (that is, the positive and negative returns have different impacts on future volatility).

The following statements include the SUBFORM=TARCH option to fit a bivariate DCC TGACH(1,1) model with the same initial values that are used for the previous DCC GARCH(1,1) model:

```
proc varmax data=indices outest=oedcct outcov;
   model rDJIA rSP500 / noint;
   garch p=1 q=1 form=dcc outht=ohdcct subform=tgarch;
   initial DCCA=0.01, DCCB=0.98, DCCS(1,2)  = 0.97294,
           GCHC(1,1)  = 0.03713, GCHC(2,2)  = 0.04004,
           ACH(1,1,1) = 0.06862, ACH(1,2,2) = 0.06684,
           GCH(1,1,1) = 0.88472, GCH(1,2,2) = 0.88916;
   output out=odcct lead=0;
run;
```

Figure 42.38 shows the parameter estimates for the DCC TGARCH(1,1) model. **TACH1_1_1** and **TACH1_2_2** correspond to the parameters $b_{11,1}$ and $b_{22,1}$, respectively. They are significant, which means that the leverage effect exists.

Figure 42.38 DCC TGARCH Parameter Estimates

The VARMAX Procedure

GARCH Model Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Pr > |t|
DCCA	0.04302	0.00669	6.43	0.0001
DCCB	0.92807	0.01142	81.26	0.0001
DCCS1_2	0.97309	0.00248	392.01	0.0001
GCHC1_1	0.02068	0.00305	6.78	0.0001
GCHC2_2	0.02329	0.00346	6.73	0.0001
ACH1_1_1	0.00104	0.00684	0.15	0.8787
ACH1_2_2	0.00314	0.00698	0.45	0.6525
TACH1_1_1	0.11443	0.01207	9.48	0.0001
TACH1_2_2	0.10805	0.01166	9.27	0.0001
GCH1_1_1	0.91490	0.00956	95.68	0.0001
GCH1_2_2	0.91574	0.00964	95.03	0.0001

Figure 42.39 shows the log likelihood and the information criteria. The SBC for the DCC TGARCH model is smaller than the SBC for the BEKK GARCH model (which is shown in Figure 42.31). The smaller SBC means that the DCC TGARCH model is the final winner.

Figure 42.39 DCC TGARCH Log Likelihood and Information Criteria

Log-likelihood	1587.793

Information Criteria	
AICC	-3153.48
HQC	-3130.31
AIC	-3153.59
SBC	-3089.46
FPEC	0.080617

Other subforms of GARCH models—the exponential GARCH (EGARCH) model, the quadratic GARCH (QGARCH) model, and the power GARCH (PGARCH) model—are also supported for the CCC and DCC GARCH models. Furthermore, the multivariate GARCH models can be used together with VARMAX or vector error correction models. For more information and examples, see the sections "Multivariate GARCH Modeling" on page 3006 and "Example 42.4: Analysis of Euro Foreign Exchange Reference Rates" on page 3063.

Syntax: VARMAX Procedure

```
PROC VARMAX options ;
  BOUND restriction, . . . , restriction ;
  BY variables ;
  CAUSAL GROUP1=(variables) GROUP2=(variables) ;
  COINTEG RANK=number < options > ;
  GARCH options ;
  ID variable INTERVAL=value < ALIGN=value > ;
  INITIAL equation, . . . , equation ;
  MODEL dependents < = regressors > < , dependents < = regressors > . . . > < / options > ;
  NLOPTIONS options ;
  OUTPUT < options > ;
  RESTRICT restriction, . . . , restriction ;
  TEST restriction, . . . , restriction ;
```

Functional Summary

The statements and options available in the VARMAX procedure are summarized in Table 42.1.

Table 42.1 VARMAX Functional Summary

Description	Statement	Option
Data Set Options		
Specifies the input data set	VARMAX	DATA=
Writes parameter estimates to an output data set	VARMAX	OUTEST=
Includes covariances in the OUTEST= data set	VARMAX	OUTCOV
Writes the diagnostic checking tests for a model and the cointegration test results to an output data set	VARMAX	OUTSTAT=
Writes actuals, predictions, residuals, and confidence limits to an output data set	OUTPUT	OUT=
Writes the conditional covariance matrix to an output data set	GARCH	OUTHT=
BY Groups		
Specifies BY-group processing	BY	
ID Variable		
Specifies the identifying variable	ID	
Specifies the time interval between observations	ID	INTERVAL=
Controls the alignment of SAS date values	ID	ALIGN=
Options to Control the Optimization Process		
Specifies the optimization options	NLOPTIONS	

Table 42.1 *continued*

Description	Statement	Option
Printing Control Options		
Specifies how many lags to print results	MODEL	LAGMAX=
Suppresses the printed output	MODEL	NOPRINT
Requests all printing options	MODEL	PRINTALL
Requests the printing format	MODEL	PRINTFORM=
Controls plots produced through ODS GRAPHICS	VARMAX	PLOTS=
PRINT= Option		
Prints the correlation matrix of parameter estimates	MODEL	CORRB
Prints the cross-correlation matrices of independent variables	MODEL	CORRX
Prints the cross-correlation matrices of dependent variables	MODEL	CORRY
Prints the covariance matrices of prediction errors	MODEL	COVPE
Prints the cross-covariance matrices of the independent variables	MODEL	COVX
Prints the cross-covariance matrices of the dependent variables	MODEL	COVY
Prints the covariance matrix of parameter estimates	MODEL	COVB
Prints the decomposition of the prediction error covariance matrix	MODEL	DECOMPOSE
Prints the residual diagnostics	MODEL	DIAGNOSE
Prints the contemporaneous relationships among the components of the vector time series	MODEL	DYNAMIC
Prints the parameter estimates	MODEL	ESTIMATES
Prints the infinite order AR representation	MODEL	IARR
Prints the impulse response function	MODEL	IMPULSE=
Prints the impulse response function in the transfer function	MODEL	IMPULSX=
Prints the partial autoregressive coefficient matrices	MODEL	PARCOEF
Prints the partial canonical correlation matrices	MODEL	PCANCORR
Prints the partial correlation matrices	MODEL	PCORR
Prints the eigenvalues of the companion matrix	MODEL	ROOTS
Prints the Yule-Walker estimates	MODEL	YW
Model Estimation and Order Selection Options		
Specifies the initial parameter values for non-linear optimization when the model is estimated through the maximum likelihood method	INITIAL	
Centers the dependent variables	MODEL	CENTER
Specifies the degrees of differencing for the specified model variables	MODEL	DIF=
Specifies the degrees of differencing for all independent variables	MODEL	DIFX=

Table 42.1 *continued*

Description	Statement	Option
Specifies the degrees of differencing for all dependent variables	MODEL	DIFY=
Specifies the estimation method	MODEL	METHOD=
Selects the tentative order	MODEL	MINIC=
Suppresses the current values of independent variables	MODEL	NOCURRENTX
Suppresses the intercept parameters	MODEL	NOINT
Specifies the number of seasonal periods	MODEL	NSEASON=
Specifies the order of autoregressive polynomial	MODEL	P=
Specifies the Bayesian prior model	MODEL	PRIOR=
Specifies the order of moving-average polynomial	MODEL	Q=
Centers the seasonal dummies	MODEL	SCENTER
Specifies the degree of time trend polynomial	MODEL	TREND=
Specifies the denominator for error covariance matrix estimates	MODEL	VARDEF=
Specifies the lag order of independent variables	MODEL	XLAG=
GARCH-Related Options		
Specifies how to calculate the constant (unconditional) correlation matrix in the CCC (DCC) GARCH model	GARCH	CORRCONSTANT=
Specifies the type of the multivariate GARCH model	GARCH	FORM=
Specifies the order of the GARCH polynomial	GARCH	P=
Specifies the order of the ARCH polynomial	GARCH	Q=
Specifies the type of the univariate GARCH model for each innovation in the CCC or DCC GARCH model	GARCH	SUBFORM=
Cointegration-Related Options		
Specifies the restriction on the drift in the VECM	COINTEG	ECTREND
Prints the results from the weak exogeneity test of the long-run parameters	COINTEG	EXOGENEITY
Specifies the restriction on the cointegrated coefficient matrix	COINTEG	H=
Specifies the restriction on the adjustment coefficient matrix	COINTEG	J=
Specifies the nonlinear constraints that the adjustment coefficient matrix and the cointegrated coefficient matrix are both full rank	COINTEG	NLC
Specifies the variable name whose cointegrating vectors are normalized	COINTEG	NORMALIZE=
Specifies a cointegration rank	COINTEG	RANK=
Prints the Johansen cointegration rank test	MODEL	COINTTEST= (JOHANSEN=)

Table 42.1 *continued*

Description	Statement	Option
Prints the Stock-Watson common trends test	MODEL	COINTTEST=(SW=)
Prints the Dickey-Fuller unit root test	MODEL	DFTEST=
Specifies the vector error correction model (obsolete) [1]	MODEL	ECM=
Tests and Restrictions on Parameters		
Tests the Granger causality	CAUSAL	GROUP1= GROUP2=
Places and tests restrictions on parameter estimates	BOUND	
Places and tests restrictions on parameter estimates	RESTRICT	
Tests hypotheses on parameter estimates	TEST	
Forecasting Control Options		
Specifies the size of confidence limits for forecasting	OUTPUT	ALPHA=
Starts forecasting before end of the input data	OUTPUT	BACK=
Specifies how many periods to forecast	OUTPUT	LEAD=
Suppresses the printed forecasts	OUTPUT	NOPRINT

PROC VARMAX Statement

PROC VARMAX *options* **;**

The following options can be used in the PROC VARMAX statement:

DATA=*SAS-data-set*
specifies the input SAS data set. If the DATA= option is not specified, the PROC VARMAX statement uses the most recently created SAS data set.

OUTEST=*SAS-data-set*
writes the parameter estimates to the output data set.

COVOUT
OUTCOV
writes the covariance matrix for the parameter estimates to the OUTEST= data set. This option is valid only if the OUTEST= option is specified.

OUTSTAT=*SAS-data-set*
writes residual diagnostic results to an output data set. If the COINTTEST=(JOHANSEN) option is specified, the results of this option are also written to the output data set.

The following statements are the examples of these options in the PROC VARMAX statement:

[1] Starting with SAS/ETS 14.1, it is recommended that you use the COINTEG statement instead.

```
proc varmax data=one outest=est outcov outstat=stat;
   model y1-y3 / p=1;
run;
```

```
proc varmax data=one outest=est outstat=stat;
   model y1-y3 / p=1 cointtest=(johansen);
run;
```

PLOTS<(*global-plot-option***)> = ***plot-request-option* < *(options)* >

PLOTS<(*global-plot-option***)> = (** *plot-request-option* < *(options)* > **...** *plot-request-option* < *(options)* > **)**
controls the plots produced through ODS Graphics. When you specify only one plot, you can omit the parentheses around the plot request. Some examples follow:

```
plots=none
plots=all
plots(unpack)=residual(residual normal)
plots=(forecasts model)
```

For general information about ODS Graphics, see Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*).

```
proc varmax data=one plots=impulse(simple);
   model y1-y3 / p=1;
run;

proc varmax data=one plots=(model residual);
   model y1-y3 / p=1;
run;

proc varmax data=one plots=forecasts;
   model y1-y3 / p=1;
   output lead=12;
run;
```

The first VARMAX program produces the simple response impulse plots. The second VARMAX program produces the plots associated with the model and prediction errors. The plots associated with prediction errors are the ACF, PACF, IACF, distribution, white-noise, and Normal quantile plots and the prediction error plot. The third VARMAX program produces the FORECASTS and FORECASTSONLY plots.

The *global-plot-option* applies to the impulse and prediction error analysis plots generated by the VARMAX procedure. The following *global-plot-option* is available:

UNPACK displays each graph separately. (By default, some graphs can appear together in a single panel.)

The following *plot-request-options* are available:

ALL produces all plots appropriate for the particular analysis.

FORECASTS < (*forecasts-plot-options*) > produces plots of the forecasts. The forecasts-only plot that shows the multistep forecasts in the forecast region is produced by default. The following *forecasts-plot-options* are available:

- ALL produces the FORECASTSONLY and the FORECASTS plots. This is the default.
- FORECASTS produces a plot that shows the one-step-ahead as well as the multistep forecasts.
- FORECASTSONLY produces a plot that shows only the multistep forecasts.

IMPULSE < (*impulse-plot-options*) > produces the plots of impulse response function and the impulse response of the transfer function.

- ALL produces all impulse plots. This is the default.
- ACCUM produces the accumulated impulse plot.
- ORTH produces the orthogonalized impulse plot.
- SIMPLE produces the simple impulse plot.

MODEL produces plots of dependent variables listed in the MODEL statement and plots of the one-step-ahead predicted values for each dependent variables.

NONE suppresses all plots.

RESIDUAL < (*residual-plot-options*) > produces plots associated with the prediction errors obtained after modeling the data. The following *residual-plot-options* are available:

- ALL produces all plots associated with the analysis of the prediction errors. This is the default.
- RESIDUAL produces prediction error plot.
- DIAGNOSTICS produces a panel of plots useful in assessing the autocorrelations and white-noise of the prediction errors. The panel consists of the following:
  - the autocorrelation plot of the prediction errors
  - the partial autocorrelation plot of the prediction errors
  - the inverse autocorrelation plot of the prediction errors
  - the log scaled white noise plot of the prediction errors
- NORMAL produces a panel of plots useful in assessing normality of the prediction errors. The panel consists of the following:
  - distribution of the prediction errors with overlaid the normal curve
  - normal quantile plot of the prediction errors

Other Options

In addition, any of the following MODEL statement options can be specified in the PROC VARMAX statement, which is equivalent to specifying the option for every MODEL statement: CENTER, DFTEST=, DIF=, DIFX=, DIFY=, LAGMAX=, METHOD=, MINIC=, NOCURRENTX, NOINT, NOPRINT, NSEASON=, P=, PRINT=, PRINTALL, PRINTFORM=, Q=, SCENTER, TREND=, VARDEF=, and XLAG= options.

The following is an example of the options in the PROC VARMAX statement:

```
proc varmax data=one lagmax=3 method=ml;
   model y1-y3 / p=1;
run;
```

BOUND Statement

BOUND *restriction, . . . , restriction* **;**

The BOUND statement sets up linear bounds for parameters when the maximum likelihood method is applied to the estimation of VARMAX, VECM, VARMAX-GARCH, and VEC-ARMAX-GARCH models. Only one BOUND statement is allowed. If you specify more than one *restriction*, separate them with commas. The *restrictions* are specified in the same manner as the *restrictions* in the RESTRICT statement. For information about how to define restrictions by using matrix expressions, operators, and functions, see the section "RESTRICT Statement" on page 2922. Both equality and inequality constraints are allowed in the BOUND statement, although usually equality constraints are specified in the RESTRICT statement and inequality constraints are specified in the BOUND statement.

To use the BOUND statement, you need to know the form of the model. If you do not specify the GARCH statement, the COINTEG statement, or the ECM=, P=, Q=, or XLAG= option in the MODEL statement, then the BOUND statement is not applicable. If you specify the ECM=(NORMALIZE=), METHOD=LS, or PRIOR= option in the MODEL statement, or if you specify the EXOGENEITY, H=, J=, or NORMALIZE= option in the COINTEG statement, the BOUND statement is ignored. Nonlinear restrictions on parameters are not supported.

The following is an example of the BOUND statement for a bivariate (k=2) zero-mean VARMA(1,1) model, which is by default estimated by maximum likelihood method because the MA term is present:

```
proc varmax data=one;
   model y1 y2 / noint p=1 q=1;
   bound -1<=AR<=1, 0<MA;
run;
```

This BOUND statement specifies that all AR parameters must be between -1 and 1 and that all MA parameters must be positive.

You can use the BOUND statement together with the RESTRICT statement, as in the following bivariate (k=2) zero-mean VARMA(1,1) model:

```
proc varmax data=one;
   model y1 y2 / noint p=1 q=1;
   bound AR+MA>=0.001;
   restrict AR(1,1,2) = 0.5;
run;
```

BY Statement

BY *variables* **;**

A BY statement can be used with PROC VARMAX to obtain separate analyses on observations in groups defined by the BY variables.

When a BY statement appears, the procedure expects the input data set to be sorted in order of the BY variables.

If your input data set is not sorted in ascending order, use one of the following alternatives:

- Sort the data using the SORT procedure with a similar BY statement.
- Specify the BY statement option NOTSORTED or DESCENDING in the BY statement for the VARMAX procedure. The NOTSORTED option does not mean that the data are unsorted but rather that the data are arranged in groups (according to values of the BY variables) and that these groups are not necessarily in alphabetical or increasing numeric order.
- Create an index on the BY variables using the DATASETS procedure.

For more information about the BY statement, see in *SAS Language Reference: Concepts*. For more information about the DATASETS procedure, see the discussion in the *Base SAS Procedures Guide*.

The following is an example of the BY statement:

```
proc varmax data=one;
   by region;
   model y1-y3 / p=1;
run;
```

CAUSAL Statement

CAUSAL GROUP1=*(variables)* **GROUP2=***(variables)* **;**

A CAUSAL statement prints the Granger causality test by fitting the VAR(p) model by using all variables defined in GROUP1 and GROUP2. Any number of CAUSAL statements can be specified. The CAUSAL statement proceeds with the MODEL statement and uses the variables and the autoregressive order, p, specified in the MODEL statement. Variables in the GROUP1= and GROUP2= options should be defined in the MODEL statement. If the P=0 option is specified in the MODEL statement, the CAUSAL statement is not applicable.

The null hypothesis of the Granger causality test is that GROUP1 is influenced only by itself, and not by GROUP2. If the hypothesis test fails to reject the null, then the variables listed in GROUP1 might be considered as independent variables.

See the section "VAR and VARX Modeling" on page 2963 for details.

The following is an example of the CAUSAL statement. You specify the CAUSAL statement with the GROUP1= and GROUP2= options.

```
proc varmax data=one;
   model y1-y3 = x1 / p=1;
   causal group1=(x1) group2=(y1-y3);
   causal group1=(y2) group2=(y1 y3);
run;
```

The first CAUSAL statement fits the VAR(1) model by using the variables $y1$, $y2$, $y3$, and $x1$ and tests the null hypothesis that $x1$ causes the other variables, $y1$, $y2$, and $y3$, but the other variables do not cause $x1$. The second CAUSAL statement fits the VAR(1) model by using the variables $y1$, $y3$, and $y2$ and tests the null hypothesis that $y2$ causes the other variables, $y1$ and $y3$, but the other variables do not cause $y2$.

COINTEG Statement

COINTEG RANK=*number* < *options* > **;**

The COINTEG statement fits the vector error correction model to the data, tests the restrictions of the long-run parameters and the adjustment parameters, and tests for weak exogeneity in the long-run parameters. The P= option in the MODEL statement specifies the autoregressive order of the VECM. Only one COINTEG statement is allowed.

The cointegrated system uses maximum likelihood estimation. If there are no moving average (MA) terms specified by the Q= option in the MODEL statement, no GARCH terms specified in the GARCH statement, and no general restrictions specified in the BOUND and RESTRICT statements, then PROC VARMAX applies the maximum likelihood analysis proposed by Johansen and Juselius (1990); Johansen (1995a, b). Otherwise, the likelihood is maximized using an optimizer whose options can be specified in the NLOPTIONS statement.

The following statements fit a VECM(2):

```
proc varmax data=one;
   model y1-y3 / p=2;
   cointeg rank=1;
run;
```

To test restrictions on $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, you specify the J= option and the H= option, respectively. You specify the EXOGENEITY option in the COINTEG statement for tests of weak exogeneity in the long-run parameters.

The following example of the COINTEG statement specifies tests of restrictions on $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, along with tests of weak exogeneity:

```
proc varmax data=one;
   model y1-y3 / p=2;
   cointeg rank=1 h=(1 0, -1 0, 0 1)
           j=(1 0, 0 0, 0 1) exogeneity;
run;
```

You must specify the following option:

RANK=*number*

specifies the cointegration rank of the cointegrated system. The rank of cointegration should be greater than 0 and less than the number of dependent (endogenous) variables. If *number* is different from

the value of the RANK= option specified in the ECM= option in the MODEL statement, the *number* specified here is used for the rank.

You can also specify the following *options* in the COINTEG statement:

ECTREND
specifies the restriction on the drift in the VECM. This option is used in the following cases:

- There is no separate drift in the VECM, but a constant enters only through the error correction term. For example, for VECM(p),

$$\Delta \mathbf{y}_t = \boldsymbol{\alpha}(\boldsymbol{\beta}', \boldsymbol{\beta}_0)(\mathbf{y}'_{t-1}, 1)' + \sum_{i=1}^{p-1} \Phi_i^* \Delta \mathbf{y}_{t-i} + \boldsymbol{\epsilon}_t$$

  An example of the ECTREND option follows:

```
model y1 y2 / p=2;
cointeg rank=1 ectrend;
```

- There is a separate drift and no separate linear trend in the VECM, but a linear trend enters only through the error correction term. For example, for VECM(p),

$$\Delta \mathbf{y}_t = \boldsymbol{\alpha}(\boldsymbol{\beta}', \boldsymbol{\beta}_1)(\mathbf{y}'_{t-1}, t)' + \sum_{i=1}^{p-1} \Phi_i^* \Delta \mathbf{y}_{t-i} + \boldsymbol{\delta}_0 + \boldsymbol{\epsilon}_t$$

  An example of the ECTREND option with the TREND= option follows:

```
model y1 y2 / p=2 trend=linear;
cointeg rank=1 ectrend;
```

If you specify both this option and the NSEASON option in the MODEL statement, then the NSEASON option is ignored. If you specify the NOINT option in the MODEL statement, then this option is ignored.

EXOGENEITY
formulates the likelihood ratio tests for testing weak exogeneity in the long-run parameters. The null hypothesis is that one variable is weakly exogenous for the others.

H=(*matrix*)
specifies the restrictions $\mathbf{H}$ on the $k \times r$ or $(k+1) \times r$ cointegrated coefficient matrix $\tilde{\boldsymbol{\beta}}$ such that $\tilde{\boldsymbol{\beta}} = \mathbf{H}\boldsymbol{\phi}$, where $\mathbf{H}$ is known and $\boldsymbol{\phi}$ is unknown. If you do not specify the ECTREND option, then the cointegrated coefficient matrix $\tilde{\boldsymbol{\beta}}$ is the cointegrating matrix $\boldsymbol{\beta}$ and the $\mathbf{H}$ matrix has dimension $k \times m$. If you specify the ECTREND option, then the cointegrated coefficient matrix $\tilde{\boldsymbol{\beta}}$ is the cointegrating matrix $\boldsymbol{\beta}$ stacked with the coefficient row vector $\boldsymbol{\beta}_0$ or $\boldsymbol{\beta}_1$ for the constant or linear trend in the error correction term, and the $\mathbf{H}$ matrix has dimension $(k+1) \times m$. Here k is the number of dependent variables and m is $r \leq m < k$, where r is defined in the RANK=r option.

For example, consider a VECM(2) with rank equal to 1 on four dependent variables. Then, $\boldsymbol{\beta} = (\beta_{11}, \beta_{21}, \beta_{31}, \beta_{41})'$. To test the null hypothesis $\beta_{11} + \beta_{21} = 0$ (that is, $\mathbf{H}'_\perp \boldsymbol{\beta} = 0$, where $\mathbf{H}_\perp = (1\ -1\ 0\ 0)'$), you can use the following statements to specify the restriction matrix $\mathbf{H}$:

```
model y1-y4 / p=2;
cointeg rank=1 h=(1 0 0, -1 0 0, 0 1 0, 0 0 1);
```

Here the dimension of matrix $\mathbf{H}$ is 4×3 because $k = 4$ and $m = 3$, and each row of the matrix $\mathbf{H}$ is separated by commas. Note that $\mathbf{H}'_{\perp}\mathbf{H} = 0$; that is, the $\mathbf{H}$ and $\mathbf{H}_{\perp}$ matrices are orthogonal.

When the series has no separate deterministic trend, and therefore you specify the ECTREND option, the constant term should be restricted by $\boldsymbol{\alpha}'_{\perp}\boldsymbol{\delta} = 0$. The matrix $\boldsymbol{\alpha}_{\perp}$ is a $k \times (k - r)$ full-rank matrix orthogonal to $\boldsymbol{\alpha}$, such that $rank(\boldsymbol{\alpha}_{\perp}) = k - r$ and $\boldsymbol{\alpha}'_{\perp}\boldsymbol{\alpha} = 0$. The $\tilde{\boldsymbol{\beta}}$ becomes $(\boldsymbol{\beta}', \boldsymbol{\beta}_0)'$ or $\tilde{\boldsymbol{\beta}} = (\beta_{11}, \beta_{21}, \beta_{31}, \beta_{41}, \beta_{11}^{(0)})'$. As for the previous test of $\beta_{11} + \beta_{21} = 0$ (that is, $\mathbf{H}'_{\perp}\tilde{\boldsymbol{\beta}} = 0$, where $\mathbf{H}_{\perp} = (1 \ -1 \ 0 \ 0 \ 0)'$), you can specify the restriction matrix $\mathbf{H}$ as follows:

```
model y1-y4 / p=2;
cointeg rank=1 ectrend
   h=(1 0 0 0, -1 0 0 0, 0 1 0 0, 0 0 1 0, 0 0 0 1);
```

Because the dimension is changed in the $\mathbf{H}_{\perp}$ matrix, the dimension of $\mathbf{H}$ matrix has to be adjusted accordingly.

When the cointegrated system contains three dependent variables and the RANK=2 option is specified, the test of $\beta_{1j} = -\beta_{2j}$ for $j = 1, 2$ can be run with the following restriction matrix $\mathbf{H}$, where $\mathbf{H}_{\perp} = (1 \ 1 \ 0)'$ and $\mathbf{H}'_{\perp}\boldsymbol{\beta} = 0$:

```
cointeg rank=2 h=(1 0, -1 0, 0 1);
```

There are many ways to achieve a matrix that is orthogonal to a particular matrix. The following statements illustrate how to obtain the orthogonal matrix through QR decomposition:

```
proc iml;
   /* For a given matrix H_dot, */
   H_dot = {1 1 0}`;
   /* get its QR decomposition, i.e., H_dot = QR. */
   call qr(Q, R, piv, lindep, H_dot);
   /* Then, the matrix orthogonal to H_dot
      can be extracted from Q. */
   H = Q[,ncol(H_dot)+1:nrow(H_dot)];
   /* Finally, normalize each column of H if necessary. */
   do i = 1 to ncol(H);
      k = 0;
      do j = nrow(H) to 1 by -1;
         if (H[j,i]^=0) then k=j;
      end;
      if (k=0) then
         print "Error: H is not full rank!";
      else
         do j = nrow(H) to 1 by -1;
            H[j,i] = H[j,i] / H[k,i];
         end;
   end;
```

```
   print "The given matrix is:";
   print H_dot;
   print "The matrix orthogonal to it is:";
   print H;
quit;
```

J=(*matrix***)**

specifies the restrictions $\mathbf{J}$ on the $k \times r$ adjustment matrix $\boldsymbol{\alpha}$ such that $\boldsymbol{\alpha} = \mathbf{J}\boldsymbol{\psi}$, where $\mathbf{J}$ is known and $\boldsymbol{\psi}$ is unknown. The $k \times m$ matrix $\mathbf{J}$ is specified by using this option, where k is the number of dependent variables, m is $r \leq m < k$, and r is defined in the RANK=*r* option.

For example, suppose the system contains four variables, the RANK=1 option is specified, and you want to test $\alpha_j = 0$ for $j = 2, 3, 4$ —that is, $\mathbf{J}'_\perp \boldsymbol{\alpha} = 0$, where

$$J_\perp = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Then you can specify the restriction matrix $\mathbf{J}$ as follows:

```
cointeg rank=1 j=(1, 0, 0, 0);
```

Suppose the system contains three variables, the RANK=2 option is specified, and you want to test $\alpha_{2j} = 0$ for $j = 1, 2$—that is, $\mathbf{J}'_\perp \boldsymbol{\alpha} = 0$, where $\mathbf{J}_\perp = (0\ 1\ 0)'$. Then you can specify the restriction matrix $\mathbf{J}$ as follows:

```
cointeg rank=2 j=(1 0, 0 0, 0 1);
```

NLC

specifies the nonlinear constraints that $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are full column rank. Although the constraints are required for a well-defined VECM, only the TECH=QUANEW and TECH=NMSIMP optimization methods in the NLOPTIONS statement support nonlinear constraints. The full-rank constraints are not imposed by default so that other optimization methods, such as TECH=CONGRA or TECH=TRUREG, can be tried. The NLC option works only when numerical optimization is used for estimating VECM (for example, when the BOUND, INITIAL, or RESTRICT statement is specified, or the VEC-ARMA or VEC-ARMA-GARCH model is estimated). That is, the NLC option is ignored if the closed-form solution of parameter estimates and maximum likelihood analysis, which is provided in Johansen and Juselius (1990) and Johansen (1995a, b), can be applied.

NORMALIZE=*variable*

specifies a single dependent (endogenous) *variable* whose cointegrating vectors are normalized. If the *variable* is different from the variable specified in the COINTTEST=(JOHANSEN=) or ECM= option in the MODEL statement, the *variable* in this option is used. If this option is not specified, cointegrating vectors are not normalized.

If the EXOGENEITY, H=, J=, or NORMALIZE= option is specified, the BOUND, GARCH, INITIAL, and RESTRICT statements are all ignored, and the Q= option in the MODEL statement is also ignored.

GARCH Statement

GARCH *options* **;**

The GARCH statement specifies a GARCH-type multivariate conditional heteroscedasticity model.

You can specify the following options:

CORRCONSTANT=ESTIMATE | EXPECT
specifies how to calculate the constant or unconditional correlation matrix in the CCC or DCC GARCH model, respectively. If you specify CORRCONSTANT=EXPECT, the constant conditional correlation matrix in the CCC GARCH model or the unconditional correlation matrix in the DCC GARCH model is calculated through the standardized residuals, given the other parameters. After parameter estimates are output, the constant or unconditional correlation matrix for the CCC or DCC GARCH model is output in the CCCCorrConstant or DCCCorrConstant ODS table, respectively. If you specify CORRCONSTANT=ESTIMATE, the correlation matrix is estimated like all other parameters in the model. By default, CORRCONSTANT=ESTIMATE.

FORM=*value*
specifies the representation for a GARCH model. Valid values are as follows:

BEKK specifies a BEKK representation. This is the default.

CCC specifies a constant conditional correlation representation.

DCC specifies a dynamic conditional correlation representation.

OUTHT=*SAS-data-set*
writes the conditional covariance matrix to an output data set.

P=*number*

P=(*number-list***)**
specifies the order of the process or the subset of GARCH terms to be fitted. For example, you can specify the P=(1,3) option. The P=3 option is equivalent to the P=(1,2,3) option. By default, P=0.

Q=*number*

Q=(*number-list***)**
specifies the order of the process or the subset of ARCH terms to be fitted. This option is required in the GARCH statement. For example, you can specify the Q=(2) option. The Q=2 option is equivalent to the Q=(1,2) option.

SUBFORM=*value*
specifies the type of the univariate GARCH model for each innovation in the CCC or DCC GARCH model. If you specify the FORM=BEKK option, the SUBFORM= option is ignored. The values of the SUBFORM= option are as follows:

EGARCH specifies the exponential GARCH, or EGARCH, model.

GARCH specifies the GARCH model with no constraints.

GJR | TGARCH specifies the GJR GARCH (also called threshold GARCH, or TGARCH) model.

PGARCH specifies the power GARCH, or PGARCH, model.

QGARCH specifies the quadratic GARCH, or QGARCH, model.

By default, SUBFORM=GARCH.

If you specify the ECM=(NORMALIZE=) or PRIOR= option in the MODEL statement, or if you specify the EXOGENEITY, H=, J=, or NORMALIZE= option in the COINTEG statement, the GARCH statement is ignored.

For the VAR(1)–ARCH(1) model,

```
model y1 y2 / p=1;
garch q=1 form=bekk;
```

For the multivariate GARCH(1,1) model,

```
model y1 y2;
garch q=1 p=1 form=ccc;
```

Other multivariate GARCH-type models are

```
model y1 y2 = x1 / xlag=1;
garch q=1;

model y1 y2 / q=1;
garch q=1 p=1;
```

For more information, see the section "Multivariate GARCH Modeling" on page 3006.

ID Statement

ID *variable* **INTERVAL=***value* < **ALIGN=***value* > **;**

The ID statement specifies a variable that identifies observations in the input data set. The datetime variable specified in the ID statement is included in the OUT= data set if the OUTPUT statement is specified. The ID *variable* is usually a SAS datetime variable. The values of the ID variable are extrapolated for the forecast observations based on the value of the INTERVAL= option.

ALIGN= *value*

controls the alignment of SAS dates used to identify output observations. The ALIGN= option allows the following values: BEGINNING | BEG | B, MIDDLE | MID | M, and ENDING | END | E. The default is BEGINNING. The ALIGN= option is used to align the ID variable to the beginning, middle, or end of the time ID interval specified by the INTERVAL= option.

INTERVAL= *value*

specifies the time interval between observations. This option is required in the ID statement. The INTERVAL= option is used in conjunction with the ID variable to check that the input data are in order and have no missing periods. The INTERVAL= option is also used to extrapolate the ID values past the end of the input data when the OUTPUT statement is specified.

The following is an example of the ID statement:

```
proc varmax data=one;
   id date interval=qtr align=mid;
   model y1-y3 / p=1;
run;
```

INITIAL Statement

INITIAL *equation, . . . , equation* **;**

The INITIAL statement sets up the initial parameter values for nonlinear optimization when the maximum likelihood method is applied to the estimation of VARMAX, VECM, VARMAX-GARCH, or VEC-ARMAX-GARCH models. Only one INITIAL statement is allowed. If you specify more than one *equation*, separate them with commas. The *equations* are specified in the same manner as the *restrictions* in the RESTRICT statement. For information about how to define equations by using matrix expressions, operators, and functions, see the section "RESTRICT Statement" on page 2922.

To use the INITIAL statement, you need to know the form of the model. If you do not specify the GARCH statement, the COINTEG statement, or the ECM=, P=, Q=, or XLAG= option in the MODEL statement, then the INITIAL statement is not applicable. If you specify the ECM=(NORMALIZE=), METHOD=LS, or PRIOR= option in the MODEL statement, or if you specify the EXOGENEITY, H=, J=, or NORMALIZE= option in the COINTEG statement, the INITIAL statement is ignored. Nonlinear restrictions on parameters are not supported.

The initial parameter values are the solution of the specified linear equations. If you do not specify initial values for all parameters, the default initial value for any parameter that is not specified in the INITIAL statement is 0, except for the following:

- The diagonal elements of the **COV** parameter matrix are set to ones if the **COV** parameter matrix is to be estimated.
- The diagonal elements of the **GCHC** parameter matrix are set to ones if the **GCHC** parameter matrix is to be estimated and the SUBFORM=EGARCH option is not specified.
- The diagonal elements of the **PACH** parameter matrix are set to ones if the SUBFORM=PGARCH option is specified.

The following is an example of the INITIAL statement for a bivariate (k=2) zero-mean VARMA(1,1) model, which is estimated by the maximum likelihood method by default because a moving average (MA) term is present:

```
proc varmax data=one;
   model y1 y2 / noint p=1 q=1;
   initial AR = 0, MA = 0,
           COV={1 0.5, 0.5 4};
run;
```

Like the RESTRICT statement, the preceding INITIAL statement can be abbreviated as follows:

```
initial AR = MA = 0,
        COV={1 0.5, 0.5 4};
```

or

```
initial AR, MA, COV={1 0.5, 0.5 4};
```

Furthermore, you can omit **AR** and **MA** in the INITIAL statement as follows, because by default the **AR** and **MA** matrices are set to zeros if they are not specified in the INITIAL statement:

```
initial COV={1 0.5, 0.5 4};
```

If you use the INITIAL statement for a vector error correction model (VECM), you must specify initial values for both the **ALPHA** and **BETA** matrices and make sure they are both full rank; otherwise, the INITIAL statement is ignored.

In the following example, the INITIAL statement is ignored because initial values for **ALPHA** and **BETA** are not specified:

```
proc varmax data=one;
   model y1 y2 / noint p=1;
   cointeg rank=1;
   initial cov=I(2)*4;
run;
```

In the following example, the INITIAL statement is ignored because initial values for **ALPHA** are not specified:

```
proc varmax data=one;
   model y1 y2 / noint p=1;
   cointeg rank=1;
   initial beta=1;
run;
```

In the following example, the INITIAL statement is ignored because the initial **BETA** matrix is not full rank:

```
proc varmax data=one;
   model y1 y2 y3 / noint p=1;
   cointeg rank=2;
   initial alpha={1 0, 0 1, 0 0},
           beta ={1 2, 2 4, 3 6};
run;
```

In the following example, the INITIAL statement works fine because the specified initial **ALPHA** and **BETA** matrices are both full rank:

```
proc varmax data=one;
   model y1 y2 y3 / noint p=1;
   cointeg rank=2;
   initial alpha={1 0, 0 1, 0 0},
           beta ={1 2, 2 4, 3 5};
run;
```

MODEL Statement

MODEL *dependents* < = *regressors* >
< , *dependents* < = *regressors* > . . . >
< / *options* > ;

The MODEL statement specifies dependent (endogenous) variables and independent (exogenous) variables for the VARMAX model. The multivariate model can have the same or different independent variables corresponding to the dependent variables. As a special case, the VARMAX procedure allows you to analyze one dependent variable. Only one MODEL statement is allowed.

For example, the following statements are equivalent ways of specifying the multivariate model for the vector $(y1, y2, y3)$:

```
model y1 y2 y3 </options>;
model y1-y3 </options>;
```

The following statements are equivalent ways of specifying the multivariate model with independent variables, where $y1, y2, y3$, and $y4$ are the dependent variables and $x1, x2, x3, x4$, and $x5$ are the independent variables:

```
model y1 y2 y3 y4 = x1 x2 x3 x4 x5 </options>;
model y1 y2 y3 y4 = x1-x5 </options>;
model y1 = x1-x5, y2 = x1-x5, y3 y4 = x1-x5 </options>;
model y1-y4 = x1-x5 </options>;
```

When the multivariate model has different independent variables that correspond to each of the dependent variables, equations are separated by commas (,) and the model can be specified as illustrated by the following MODEL statement:

```
model y1 = x1-x3, y2 = x3-x5, y3 y4 = x1-x5 </options>;
```

The following options can be used in the MODEL statement after a forward slash (/):

CENTER
centers the dependent (endogenous) variables by subtracting their means. Note that centering is done after differencing when the DIF= or DIFY= option is specified. If there are exogenous (independent) variables, this option is not applicable.

```
model y1 y2 / p=1 center;
```

DIF(*variable* **(***number-list***) < ...** *variable* **(***number-list***) >)**

DIF=(*variable* **(***number-list***) < ...** *variable* **(***number-list***) >)**
specifies the degrees of differencing to be applied to the specified dependent or independent variables. The *number-list* must contain one or more numbers, each of which should be greater than zero. The differencing can be the same for all variables, or it can vary among variables. For example, the DIF=(y_1(1,4) y_3(1) x_2(2)) option specifies that the series y_1 is differenced at lag 1 and at lag 4, which is

$$(1 - B^4)(1 - B)y_{1t} = (y_{1t} - y_{1,t-1}) - (y_{1,t-4} - y_{1,t-5})$$

The series y_3 is differenced at lag 1, which is $(y_{3t} - y_{3,t-1})$; and the series x_2 is differenced at lag 2, which is $(x_{2t} - x_{2,t-2})$.

The following uses the data $dy1$, $y2$, $x1$, and $dx2$, where $dy1 = (1 - B)y_{1t}$ and $dx2 = (1 - B)^2 x_{2t}$.

```
model y1 y2 = x1 x2 / p=1 dif=(y1(1) x2(2));
```

DIFX(*number-list***)**

DIFX=(*number-list***)**

specifies the degrees of differencing to be applied to all independent variables. The *number-list* must contain one or more numbers, each of which should be greater than zero. For example, the DIFX=(1) option specifies that all of the independent series are differenced once at lag 1. The DIFX=(1,4) option specifies that all of the independent series are differenced at lag 1 and at lag 4. If independent variables are specified in the DIF= option, then the DIFX= option is ignored.

The following statement uses the data $y1$, $y2$, $dx1$, and $dx2$, where $dx1 = (1 - B)x_{1t}$ and $dx2 = (1 - B)x_{2t}$.

```
model y1 y2 = x1 x2 / p=1 difx(1);
```

DIFY(*number-list***)**

DIFY=(*number-list***)**

specifies the degrees of differencing to be applied to all dependent (endogenous) variables. The *number-list* must contain one or more numbers, each of which should be greater than zero. For details, see the DIFX= option. If dependent variables are specified in the DIF= option, then the DIFY= option is ignored.

```
model y1 y2 / p=1 dify(1);
```

METHOD=*value*

requests the type of estimates to be computed. The possible values of the METHOD= option are as follows:

LS	specifies least squares estimates.
ML	specifies maximum likelihood estimates.
CML	specifies conditional maximum likelihood estimates.

For VARX models, you can apply least squares method, maximum likelihood method, or conditional maximum likelihood method; for VARMAX models, you can apply either maximum likelihood method or conditional maximum likelihood method; for other type of models, namely, vector error correction models, GARCH models, and Bayesian models, the default maximum likelihood method is applied.

```
model y1 y2 / p=1 method=ml;
```

NOCURRENTX

suppresses the current values x_t of the independent variables. In general, the VARX(p, s) model is

$$\mathbf{y}_t = \boldsymbol{\delta} + \sum_{i=1}^{p} \Phi_i \mathbf{y}_{t-i} + \sum_{i=0}^{s} \Theta_i^* \mathbf{x}_{t-i} + \boldsymbol{\epsilon}_t$$

where p is the number of lags of the dependent variables included in the model, and s is the number of lags of the independent variables included in the model, including the contemporaneous values of x_t.

A VARX(1,2) model can be specified as:

```
model y1 y2 = x1 x2 / p=1 xlag=2;
```

If the NOCURRENTX option is specified, it suppresses the current values $\mathbf{x}_t$ and starts with $\mathbf{x}_{t-1}$. The VARX(p, s) model is redefined as:

$$\mathbf{y}_t = \boldsymbol{\delta} + \sum_{i=1}^{p} \Phi_i \mathbf{y}_{t-i} + \sum_{i=1}^{s} \Theta_i^* \mathbf{x}_{t-i} + \boldsymbol{\epsilon}_t$$

This model with $p = 1$ and $s = 2$ can be specified as:

```
model y1 y2 = x1 x2 / p=1 xlag=2 nocurrentx;
```

NOINT

suppresses the intercept parameter $\boldsymbol{\delta}$.

```
model y1 y2 / p=1 noint;
```

NSEASON=*number*

specifies the number of seasonal periods. When the NSEASON=*number* option is specified, (*number* –1) seasonal dummies are added to the regressors. If the NOINT option is specified, the NSEASON= option is not applicable. For more information, see the section "Seasonal Dummies and Time Trends" on page 2969.

```
model y1 y2 / p=1 nseason=4;
```

SCENTER

centers seasonal dummies specified by the NSEASON= option. The centered seasonal dummies are generated by $c - (1/s)$, where c is a seasonal dummy generated by the NSEASON=s option.

```
model y1 y2 / p=1 nseason=4 scenter;
```

TREND=*value*

specifies the degree of deterministic time trend included in the model. Valid values are as follows:

LINEAR	includes a linear time trend as a regressor.
QUAD	includes linear and quadratic time trends as regressors.

The TREND=QUAD option is not applicable for a cointegration analysis. For more information, see the section "Seasonal Dummies and Time Trends" on page 2969.

```
model y1 y2 / p=1 trend=linear;
```

VARDEF=*value*

corrects for the degrees of freedom of the denominator for computing an error covariance matrix for the METHOD=LS option. If the METHOD=ML option is specified, the VARDEF=N option is always used. Valid values are as follows:

DF	specifies that the number of nonmissing observation minus the number of regressors be used.
N	specifies that the number of nonmissing observation be used.

```
model y1 y2 / p=1 vardef=n;
```

Printing Control Options

LAGMAX=*number*

specifies the maximum number of lags for which results are computed and displayed by the PRINT=(CORRX CORRY COVX COVY IARR IMPULSE= IMPULSX= PARCOEF PCANCORR PCORR) options. This option is also used to limit the printed results for the cross covariances and cross-correlations of residuals. The default is LAGMAX=min(12, T-2), where T is the number of nonmissing observations.

```
model y1 y2 / p=1 lagmax=6;
```

NOPRINT

suppresses all printed output.

```
model y1 y2 / p=1 noprint;
```

PRINTALL

requests all printing control options. The options set by the option PRINTALL are DFTEST=, MINIC=, PRINTFORM=BOTH, and PRINT=(CORRB CORRX CORRY COVB COVPE COVX COVY DECOMPOSE DYNAMIC IARR IMPULSE=(ALL) IMPULSX=(ALL) PARCOEF PCANCORR PCORR ROOTS YW).

You can also specify this option as the option ALL.

```
model y1 y2 / p=1 printall;
```

PRINTFORM=*value*

requests the printing format of the output generated by the PRINT= option and cross covariances and cross-correlations of residuals. Valid values are as follows:

BOTH	prints output in both MATRIX and UNIVARIATE forms.
MATRIX	prints output in matrix form. This is the default.
UNIVARIATE	prints output by variables.

```
model y1 y2 / p=1 print=(impulse) printform=univariate;
```

Printing Options

PRINT=(*options***)**

The following options can be used in the PRINT=( ) option. The options are listed within parentheses. If a number in parentheses follows an option listed below, then the option prints the number of lags specified by *number* in parentheses. The default is the number of lags specified by the LAGMAX=*number* option.

CORRB

prints the estimated correlations of the parameter estimates.

CORRX

CORRX(*number***)**

prints the cross-correlation matrices of exogenous (independent) variables. The *number* should be greater than zero.

CORRY

CORRY(*number***)**

prints the cross-correlation matrices of dependent (endogenous) variables. The *number* should be greater than zero.

COVB

prints the estimated covariances of the parameter estimates.

COVPE

COVPE(*number***)**

prints the covariance matrices of *number*-ahead prediction errors for the VARMAX(p,q,s) model. The *number* should be greater than zero. If the DIF= or DIFY= option is specified, the covariance matrices of multistep prediction errors are computed based on the differenced data. This option is not applicable when the PRIOR= option is specified. See the section "Forecasting" on page 2953 for details.

COVX

COVX(*number***)**

prints the cross-covariance matrices of exogenous (independent) variables. The *number* should be greater than zero.

COVY

COVY(*number***)**

prints the cross-covariance matrices of dependent (endogenous) variables. The *number* should be greater than zero.

DECOMPOSE

DECOMPOSE(*number***)**

prints the decomposition of the prediction error covariances using up to the number of lags specified by *number* in parentheses for the VARMA(*p*,*q*) model. The *number* should be greater than zero. It can be interpreted as the contribution of innovations in one variable to the mean squared error of the multistep forecast of another variable. The DECOMPOSE option also prints proportions of the forecast error variance.

If the DIF= or DIFY= option is specified, the covariance matrices of multistep prediction errors are computed based on the differenced data. This option is not applicable when the PRIOR= option is specified. See the section "Forecasting" on page 2953 for details.

DIAGNOSE

prints the residual diagnostics and model diagnostics.

DYNAMIC

prints the contemporaneous relationships among the components of the vector time series.

ESTIMATES

prints the coefficient estimates and a schematic representation of the significance and sign of the parameter estimates.

IARR

IARR(*number***)**

prints the infinite order AR representation of a VARMA process. The *number* should be greater than zero. If the ECM= option or the COINTEG statement is specified, then the reparameterized AR coefficient matrices are printed.

IMPULSE

IMPULSE(*number***)**

IMPULSE=(SIMPLE ACCUM ORTH STDERR ALL)

IMPULSE(*number***)=(SIMPLE ACCUM ORTH STDERR ALL)**

prints the impulse response function. The *number* should be greater than zero. It investigates the response of one variable to an impulse in another variable in a system that involves a number of other variables as well. It is an infinite order MA representation of a VARMA process. See the section "Impulse Response Function" on page 2943 for details.

The following options can be used in the IMPULSE=( ) option. The options are specified within parentheses.

ACCUM	prints the accumulated impulse response function.
ALL	is equivalent to specifying all of SIMPLE, ACCUM, ORTH, and STDERR.
ORTH	prints the orthogonalized impulse response function.
SIMPLE	prints the impulse response function. This is the default.
STDERR	prints the standard errors of the impulse response function, the accumulated impulse response function, or the orthogonalized impulse response function.

If the exogenous variables are used to fit the model, then the STDERR option is ignored.

IMPULSX

IMPULSX(*number*)

IMPULSX=(SIMPLE ACCUM ALL)

IMPULSX(*number*)=(SIMPLE ACCUM ALL)

prints the impulse response function related to exogenous (independent) variables. The *number* should be greater than zero. See the section "Impulse Response Function" on page 2943 for details.

The following options can be used in the IMPULSX=( ) option. The options are specified within parentheses.

ACCUM	prints the accumulated impulse response matrices for the transfer function.
ALL	is equivalent to specifying both SIMPLE and ACCUM.
SIMPLE	prints the impulse response matrices for the transfer function. This is the default.

PARCOEF

PARCOEF(*number*)

prints the partial autoregression coefficient matrices, Φ_{mm} up to the lag *number*. The *number* should be greater than zero. With a VAR process, this option is useful for the identification of the order since the Φ_{mm} have the property that they equal zero for $m > p$ under the hypothetical assumption of a VAR(p) model. See the section "Tentative Order Selection" on page 2958 for details.

PCANCORR

PCANCORR(*number*)

prints the partial canonical correlations of the process at lag m and the test for testing Φ_m=0 for $m > p$ up to the lag *number*. The *number* should be greater than zero. The lag m partial canonical correlations are the canonical correlations between $\mathbf{y}_t$ and $\mathbf{y}_{t-m}$, after adjustment for the dependence of these variables on the intervening values $\mathbf{y}_{t-1}, \ldots, \mathbf{y}_{t-m+1}$. See the section "Tentative Order Selection" on page 2958 for details.

PCORR

PCORR(*number*)

prints the partial correlation matrices. The *number* should be greater than zero. With a VAR process, this option is useful for a tentative order selection by the same property as the partial autoregression coefficient matrices, as described in the PRINT=(PARCOEF) option. See the section "Tentative Order Selection" on page 2958 for details.

ROOTS

prints the eigenvalues of the $kp \times kp$ companion matrix associated with the AR characteristic function $\Phi(B)$, where k is the number of dependent (endogenous) variables, and $\Phi(B)$ is the finite order matrix polynomial in the backshift operator B, such that $B^i \mathbf{y}_t = \mathbf{y}_{t-i}$. These eigenvalues indicate the stationary condition of the process since the stationary condition on the roots of $|\Phi(B)| = 0$ in the VAR(p) model is equivalent to the condition in the corresponding VAR(1) representation that all eigenvalues of the companion matrix be less than one in absolute value. Similarly, you can use this option to check the invertibility of the MA process. In addition, when the GARCH statement is specified, this option prints the roots of the GARCH characteristic polynomials to check covariance stationarity for the GARCH process.

YW

prints Yule-Walker estimates of the preliminary autoregressive model for the dependent (endogenous) variables. The coefficient matrices are printed using the maximum order of the autoregressive process.

Some examples of the PRINT= option are as follows:

```
model y1 y2 / p=1 print=(covy(10) corry(10));
model y1 y2 / p=1 print=(parcoef pcancorr pcorr);
model y1 y2 / p=1 print=(impulse(8) decompose(6) covpe(6));
model y1 y2 / p=1 print=(dynamic roots yw);
```

Lag Specification Options

P=*number*

P=(*number-list***)**

specifies the order of the vector autoregressive process. Subset models of vector autoregressive orders can be specified by listing the desired set of lags. For example, you can specify the P=(1,3,4) option. The P=3 option is equivalent to the P=(1,2,3) option. The default is P=0.

If P=0 and there are no exogenous (independent) variables, then the AR polynomial order is automatically determined by minimizing an information criterion. If P=0 and the PRIOR= or ECM= option or COINTEG statement are specified, then the AR polynomial order is determined automatically.

If the ECM= option or the COINTEG statement is specified, then subset models of vector autoregressive orders are not allowed and the AR maximum order specified is used.

Examples illustrating the P= option follow:

```
model y1 y2 / p=3;
model y1 y2 / p=(1,3);
model y1 y2 / p=(1,3) prior;
```

Q=*number*

Q=(*number-list***)**

specifies the order of the moving-average error process. Subset models of moving-average orders can be specified by listing the desired set of lags. For example, you can specify the Q=(1,5) option. The default is Q=0.

```
model y1 y2 / p=1 q=1;
model y1 y2 / q=(2);
```

XLAG=*number*

XLAG=(*number-list***)**

specifies the lags of exogenous (independent) variables. Subset models of distributed lags can be specified by listing the desired set of lags. For example, XLAG=(2) selects only a lag 2 of the exogenous variables. The default is XLAG=0. To exclude the present values of exogenous variables from the model, the NOCURRENTX option must be used.

```
model y1 y2 = x1-x3 / xlag=2 nocurrentx;
model y1 y2 = x1-x3 / p=1 xlag=(2);
```

Tentative Order Selection Options

MINIC

MINIC=(P=*number* **PERROR=***number* **Q=***number* **TYPE=***value* **)**

prints the information criterion for the appropriate AR and MA tentative order selection.

You can specify the following options within parentheses in the MINIC= option:

P=*number*

P=(p_{min} **:** p_{max}**)**

specifies the range of AR orders to be considered in the tentative order selection. The default is P=(0:5). P=3 is equivalent to P=(0:3).

PERROR=*number*

PERROR=($p_{\epsilon,min}$ **:** $p_{\epsilon,max}$**)**

specifies the range of AR orders for obtaining the error series. The default is PERROR=(p_{max} : $p_{max} + q_{max}$).

Q=*number*

Q=(q_{min} **:** q_{max}**)**

specifies the range of MA orders to be considered in the tentative order selection. The default is Q=(0:5).

TYPE= AIC | AICC | FPE | HQC | SBC

specifies the criterion for the model order selection. Valid criteria are as follows:

AIC	specifies Akaike's information criterion.
AICC	specifies the corrected Akaike's information criterion.
FPE	specifies the final prediction error criterion.
HQC	specifies the Hanna-Quinn criterion.
SBC	specifies the Schwarz Bayesian criterion. You can also specify this value as TYPE=BIC.

By default, TYPE=AICC.

The following examples show how to use the MINIC or MINIC= option:

```
model y1 y2 / minic;
```

```
model y1 y2 / minic=(type=aic p=13);
```

In the selection of AR and MA orders, the model that has the smallest criterion value is chosen. For the definitions of the information criteria used in the MINIC option, see the section "The Minimum Information Criterion (MINIC) Method" on page 2962.

Cointegration Related Options

Two options are related to integrated time series; one is the DFTEST option to test for a unit root and the other is the COINTTEST option to test for cointegration.

DFTEST

DFTEST=(DLAG=*number***)**

DFTEST=(DLAG=(*number***) . . . (***number***))**

prints the Dickey-Fuller unit root tests. The DLAG=(*number*) . . . (*number*) option specifies the regular or seasonal unit root test. Supported values of *number* are in 1, 2, 4, 12. If the *number* is greater than one, a seasonal Dickey-Fuller test is performed. If the TREND= option is specified, the seasonal unit root test is not available. The default is DLAG=1.

For example, the DFTEST=(DLAG=(1)(12)) option produces two tables: the Dickey-Fuller regular unit root test and the seasonal unit root test.

Some examples of the DFTEST= option follow:

```
model y1 y2 / p=2 dftest;
model y1 y2 / p=2 dftest=(dlag=4);
model y1 y2 / p=2 dftest=(dlag=(1)(12));
model y1 y2 / p=2 dftest cointtest;
```

COINTTEST

COINTTEST=(JOHANSEN < (=*options***) > SW < (=***options***) > SIGLEVEL=***number* **)**

specifies the cointegration tests.

You can specify the following options within parentheses in the COINTTEST= option:

JOHANSEN

JOHANSEN=(TYPE=*value* **IORDER=***number* **NORMALIZE=***variable***)**

prints the cointegration rank test for multivariate time series based on Johansen's method. This test is provided when the number of dependent (endogenous) variables is less than or equal to 64. For more information, see the section "Vector Error Correction Modeling" on page 2986.

The VARX(p,s) model can be written as the error correction model

$$\Delta \mathbf{y}_t = \Pi \mathbf{y}_{t-1} + \sum_{i=1}^{p-1} \Phi_i^* \Delta \mathbf{y}_{t-i} + AD_t + \sum_{i=0}^{s} \Theta_i^* \mathbf{x}_{t-i} + \boldsymbol{\epsilon}_t$$

where Π, Φ_i^*, A, and Θ_i^* are coefficient parameters and D_t is a deterministic term such as a constant, a linear trend, or seasonal dummies.

The $I(1)$ model is defined by one reduced-rank condition. If the cointegration rank is $r < k$, then there exist $k \times r$ matrices $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ of rank r such that $\Pi = \boldsymbol{\alpha}\boldsymbol{\beta}'$.

The $I(1)$ model is rewritten as the $I(2)$ model

$$\Delta^2 \mathbf{y}_t = \Pi \mathbf{y}_{t-1} - \Psi \Delta \mathbf{y}_{t-1} + \sum_{i=1}^{p-2} \Psi_i \Delta^2 \mathbf{y}_{t-i} + AD_t + \sum_{i=0}^{s} \Theta_i^* \mathbf{x}_{t-i} + \boldsymbol{\epsilon}_t$$

where $\Psi = I_k - \sum_{i=1}^{p-1} \Phi_i^*$ and $\Psi_i = -\sum_{j=i+1}^{p-1} \Phi_i^*$.

The $I(2)$ model is defined by two reduced-rank conditions. One is that $\Pi = \boldsymbol{\alpha}\boldsymbol{\beta}'$, where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are $k \times r$ matrices of full-rank r. The other is that $\boldsymbol{\alpha}_\perp' \Psi \boldsymbol{\beta}_\perp = \boldsymbol{\xi}\boldsymbol{\eta}'$, where $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ are $(k-r) \times s$ matrices with $s \leq k - r$, and $\boldsymbol{\alpha}_\perp$ and $\boldsymbol{\beta}_\perp$ are $k \times (k-r)$ matrices of full-rank $k - r$ such that $\boldsymbol{\alpha}'\boldsymbol{\alpha}_\perp = 0$ and $\boldsymbol{\beta}'\boldsymbol{\beta}_\perp = 0$.

You can specify the following options within parentheses in the JOHANSEN= option:

IORDER=1 | 2
specifies the integrated order. You can specify the following values:

1 prints the cointegration rank test for an integrated order 1 and prints the long-run parameter, $\boldsymbol{\beta}$, and the adjustment coefficient, $\boldsymbol{\alpha}$. If you specify IORDER=1, then the AR order should be greater than or equal to 1. If you specify P=0 in the MODEL statement, the value of P is set to 1 for the Johansen test.

2 prints the cointegration rank test for integrated orders 1 and 2. If you specify IORDER=2, then the AR order should be greater than or equal to 2. If you specify P=1 and IORDER=2, then the value of IORDER is set to 1; if you specify P=0 and IORDER=2, then the value of P is set to 2.

By default, IORDER=1.

NORMALIZE=*variable*
specifies the dependent (endogenous) *variable* whose cointegration vectors are to be normalized. If the *variable* is different from the variable specified in the COINTEG statement or in the ECM= option in the MODEL statement, then the value specified in the COINTEG statement is used. If you specify this option and you want to estimate an error correction model, then the BOUND, GARCH, INITIAL, and RESTRICT statements are all ignored and the Q= option in the MODEL statement is also ignored.

TYPE=MAX | TRACE
specifies the type of cointegration rank test to be printed. You can specify the following values:

MAX prints the cointegration maximum eigenvalue test.

TRACE prints the cointegration trace test.

By default, TYPE=TRACE. If the NOINT option is not specified, PROC VARMAX prints two different cointegration rank tests in the presence of the unrestricted and restricted deterministic terms (constant or linear trend) models. If you specify IORDER=2, the procedure automatically sets the TYPE=TRACE option.

The following examples illustrate the JOHANSEN= option:

```
model y1 y2 / p=2 cointtest=(johansen=(type=max normalize=y1));

model y1 y2 / p=2 cointtest=(johansen=(iorder=2 normalize=y1));
```

SIGLEVEL=*value*
sets the size (the significance level) of the common trends tests.

The SIGLEVEL=*value* can be set to 0.1, 0.05, or 0.01. By default, SIGLEVEL=0.05.

```
model y1 y2 / p=2 cointtest=(sw siglevel=0.1);

model y1 y2 / p=2 cointtest=(sw siglevel=0.01);
```

SW

SW=(TYPE=*value* **LAG=***number***)**
prints common trends tests for a multivariate time series based on the Stock-Watson method. This test is provided when the number of dependent (endogenous) variables is less than or equal to 6. For more information, see the section "Common Trends" on page 2984.

You can specify the following options within parentheses in the SW= option:

LAG=*number*
specifies the number of lags. The default is LAG=max(1,p) for the TYPE=FILTDIF or TYPE=FILTRES option, where p is the AR maximum order specified by the P= option. The default is LAG=$T^{1/4}$ for the TYPE=KERNEL option, where T is the number of nonmissing observations. If the specified LAG=*number* exceeds the default, then it is replaced by the default.

TYPE=FILTDIF | FILTRES | KERNEL
specifies the type of common trends test to be printed. You can specify the following values:

FILTDIF prints the common trends test based on the filtering method applied to the differenced series.

FILTRES prints the common trends test based on the filtering method applied to the residual series.

KERNEL prints the common trends test based on the kernel method.

By default, TYPE=FILTDIF.

The following examples illustrate the SW option:

```
model y1 y2 / p=2 cointtest=(sw);

model y1 y2 / p=2 cointtest=(sw=(type=kernel));

model y1 y2 / p=2 cointtest=(sw=(type=kernel lag=3));
```

Bayesian VARX Estimation Options

PRIOR

PRIOR=(*prior-options*)

specifies the prior value of parameters for the BVARX(p, s) model. The BVARX model allows for a subset model specification. If the ECM= option or the COINTEG statement is specified with the PRIOR option, the BVECMX(p, s) form is fitted. When the PRIOR option is specified, the Q= option in the MODEL statement is ignored, and the BOUND, GARCH, INITIAL, RESTRICT, and TEST statements are all ignored. For more information, see the section "Bayesian VAR and VARX Modeling" on page 2970.

The following options can be used with the PRIOR=(*prior-options*) option. The *prior-options* are listed within parentheses.

IVAR

IVAR=(*variables*)

specifies an integrated BVAR(p) model. The *variables* should be specified in the MODEL statement as dependent variables. If you use the IVAR option without *variables*, then it sets the overall prior mean of the first lag of each variable equal to one in its own equation and sets all other coefficients to zero. If *variables* are specified, it sets the prior mean of the first lag of the specified variables equal to one in its own equation and sets all other coefficients to zero. When the series $\mathbf{y}_t = (y_1, y_2)'$ follows a bivariate BVAR(2) process, the IVAR or IVAR=(y_1 y_2) option is equivalent to specifying MEAN=(1 0 0 0 0 1 0 0).

If the PRIOR=(MEAN=) or ECM= option or the COINTEG statement is specified, the IVAR= option is ignored.

LAMBDA=*value*

specifies the prior standard deviation of the AR coefficient parameter matrices. It should be a positive number. The default is LAMBDA=1. As the value of the LAMBDA= option is increased, the BVAR(p) model becomes closer to a VAR(p) model.

MEAN=(*vector*)

specifies the mean vector in the prior distribution for the AR coefficients. If the vector is not specified, the prior value is assumed to be a zero vector. See the section "Bayesian VAR and VARX Modeling" on page 2970 for details.

You can specify the mean vector by order of the equation. Let $(\delta, \Phi_1, \ldots, \Phi_p)$ be the parameter sets to be estimated and $\Phi = (\Phi_1, \ldots, \Phi_p)$ be the AR parameter sets. The mean vector is specified by row-wise from Φ; that is, the MEAN=$(\mathrm{vec}(\Phi'))$ option.

For the PRIOR=(mean) option in the BVAR(2),

$$\Phi = \begin{pmatrix} \phi_{1,11} & \phi_{1,12} & \phi_{2,11} & \phi_{2,12} \\ \phi_{1,21} & \phi_{1,22} & \phi_{2,21} & \phi_{2,22} \end{pmatrix} = \begin{pmatrix} 2 & 0.1 & 1 & 0 \\ 0.5 & 3 & 0 & -1 \end{pmatrix}$$

where $\phi_{l,ij}$ is an element of Φ, l is a lag, i is associated with the first dependent variable, and j is associated with the second dependent variable.

```
model y1 y2 / p=2 prior=(mean=(2 0.1 1 0 0.5 3 0 -1));
```

The deterministic terms and exogenous variables are considered to shrink toward zero; you must omit prior means of exogenous variables and deterministic terms such as a constant, seasonal dummies, or trends.

For a Bayesian error correction model estimated when both the ECM= option (or the COINTEG statement) and the PRIOR= option are used, a mean vector for only lagged AR coefficients, Φ_i^*, in terms of regressors $\Delta \mathbf{y}_{t-i}$, for $i = 1, \ldots, (p-1)$ is used in the VECM(p) representation. The diffused prior variance of $\boldsymbol{\alpha}$ is used, since $\boldsymbol{\beta}$ is replaced by $\hat{\boldsymbol{\beta}}$ estimated in a nonconstrained VECM(p) form.

$$\Delta \mathbf{y}_t = \boldsymbol{\alpha} \mathbf{z}_{t-1} + \sum_{i=1}^{p-1} \Phi_i^* \Delta \mathbf{y}_{t-i} + AD_t + \sum_{i=0}^{s} \Theta_i^* \mathbf{x}_{t-i} + \boldsymbol{\epsilon}_t$$

where $\mathbf{z}_t = \boldsymbol{\beta}' \mathbf{y}_t$.

For example, in the case of a bivariate ($k = 2$) BVECM(2) form, the option

$$\mathrm{MEAN} = (\phi_{1,11}^* \; \phi_{1,12}^* \; \phi_{1,21}^* \; \phi_{1,22}^*)$$

where $\phi_{1,ij}^*$ is the (i, j) element of the matrix Φ_1^*.

NREP=*number*

determines the number of repetitions that are used to compute the measure of forecast accuracy. See the equation in the section "Forecasting of BVAR Modeling" on page 2971 for details. The default is NREP=$0.5T$, where T is the number of observations. If NREP is above $0.5T$, it is decreased to $0.5T$; if NREP is below the value of the LEAD= option, it is increased to the value of the LEAD= option.

THETA=*value*

specifies the prior standard deviation of the AR coefficient parameter matrices. The *value* is in the interval (0,1). The default is THETA=0.1. As the value of the THETA= option approaches 1, the specified BVAR(p) model approaches a VAR(p) model.

Some examples of the PRIOR= option follow:

```
model y1 y2 / p=2 prior;
model y1 y2 / p=2 prior=(theta=0.2 lambda=5);
model y1 y2 = x1 / p=2 prior=(theta=0.2 lambda=5);
model y1 y2 = x1 / p=2
   prior=(theta=0.2 lambda=5 mean=(2 0.1 1 0 0.5 3 0 -1));
```

See the section "Bayesian VAR and VARX Modeling" on page 2970 for details.

Vector Error Correction Model Options

ECM=(RANK=*number* **< ECTREND > < NORMALIZE=***variable* **>)**

specifies a vector error correction model.

The ECM= option is obsolete. Use the COINTEG statement instead.

You must specify the following option within parentheses in the ECM= option:

RANK=*number*

specifies the cointegration rank of the cointegrated system. The rank of cointegration should be greater than 0 and less than the number of dependent (endogenous) variables. If *number* is different from the RANK= option specified in the COINTEG statement, the value specified in the COINTEG statement is used for the rank.

You can also specify the following options within parentheses in the ECM= option:

ECTREND

specifies the restriction on the drift in the VECM. This option is used in the following cases:

- There is no separate drift in the VECM, but a constant enters only through the error correction term. For example, for VECM(p),

$$\Delta \mathbf{y}_t = \boldsymbol{\alpha}(\boldsymbol{\beta}', \boldsymbol{\beta}_0)(\mathbf{y}'_{t-1}, 1)' + \sum_{i=1}^{p-1} \Phi_i^* \Delta \mathbf{y}_{t-i} + \boldsymbol{\epsilon}_t$$

  An example of the ECTREND option follows:

```
model y1 y2 / p=2 ecm=(rank=1 ectrend);
```

- There is a separate drift and no separate linear trend in the VECM, but a linear trend enters only through the error correction term. For example, for VECM(p),

$$\Delta \mathbf{y}_t = \boldsymbol{\alpha}(\boldsymbol{\beta}', \boldsymbol{\beta}_1)(\mathbf{y}'_{t-1}, t)' + \sum_{i=1}^{p-1} \Phi_i^* \Delta \mathbf{y}_{t-i} + \boldsymbol{\delta}_0 + \boldsymbol{\epsilon}_t$$

  An example of the ECTREND option with the TREND= option follows:

```
model y1 y2 / p=2 ecm=(rank=1 ectrend) trend=linear;
```

If you specify both this option and the NSEASON option in the MODEL statement, then the NSEASON option is ignored. If you specify the NOINT option in the MODEL statement, then this option is ignored.

NORMALIZE=*variable*

specifies a single dependent (endogenous) *variable* whose cointegrating vectors are normalized. If the *variable* is different from the variable specified in the NORMALIZE= option in the COINTEG statement, the variable specified in the NORMALIZE= option in the COINTEG statement is used. If this option is not specified, cointegrating vectors are not normalized. If you specify this option, then the BOUND, GARCH, INITIAL, and RESTRICT statements are all ignored and the Q= option in the MODEL statement is also ignored.

The following examples illustrate the ECM= option:

```
model y1 y2 / p=2 ecm=(rank=1 normalize=y1);


model y1 y2 / p=2 ecm=(rank=1 ectrend) trend=linear;
```

For more information, see the section "Vector Error Correction Modeling" on page 2986.

NLOPTIONS Statement

NLOPTIONS *options* **;**

The VARMAX procedure uses the nonlinear optimization (NLO) subsystem to perform nonlinear optimization tasks. For a list of all the options in the NLOPTIONS statement, see Chapter 7, "Nonlinear Optimization Methods."

An example of the NLOPTIONS statement is as follows:

```
proc varmax data=one;
   nloptions tech=qn maxit=1000 pall;
   model y1 y2 / p=2;
run;
```

By default, the VARMAX procedure uses the dual quasi-Newton optimization method.

OUTPUT Statement

OUTPUT < *options* > **;**

The OUTPUT statement generates and prints forecasts based on the model estimated in the previous MODEL statement and, optionally, creates an output SAS data set that contains these forecasts.

When the GARCH model is estimated, the upper and lower confidence limits of forecasts are calculated according to the conditional covariance of errors.

ALPHA=*number*
sets the forecast confidence limit size, where *number* is between 0 and 1. When you specify the ALPHA=*number* option, the upper and lower confidence limits define the $100(1-\alpha)\%$ confidence interval. The default is ALPHA=0.05, which produces 95% confidence intervals.

BACK=*number*
specifies the number of observations before the end of the data at which the multistep forecasts begin. The BACK= option value must be less than or equal to the number of observations minus the number of lagged regressors in the model. The default is BACK=0, which means that the forecasts start at the end of the available data.

LEAD=*number*
specifies the number of multistep forecast values to compute. The default is LEAD=12.

NOPRINT
suppresses the printed forecast values of each dependent (endogenous) variable.

OUT=*SAS-data-set*
writes the forecast values to an output data set. If the OUT= option is not included in the OUTPUT statement, then the output data set is named using the DATAn naming convention.

Some examples of the OUTPUT statements follow:

```
proc varmax data=one;
   model y1 y2 / p=2;
   output lead=6 back=2;
run;
```

```
proc varmax data=one;
   model y1 y2 / p=2;
   output out=for noprint;
run;
```

RESTRICT Statement

RESTRICT *restriction, . . . , restriction* **;**

The RESTRICT statement places linear restrictions on the parameters and provides constrained estimation. Only one RESTRICT statement is allowed. If you specify more than one *restriction* in a RESTRICT statement, separate them with commas. Both equality and inequality constraints are allowed in the RESTRICT statement, although usually equality constraints are specified in the RESTRICT statement and inequality constraints are specified in the BOUND statement. If the least squares method is used, the inequality constraints are not applicable.

To use the RESTRICT statement, you need to know the form of the model. If you do not specify the GARCH statement, the COINTEG statement, or the ECM=, P=, Q=, or XLAG= option in the MODEL statement then the RESTRICT statement is not applicable. If you specify the ECM=(NORMALIZE=) option or PRIOR= option in the MODEL statement or if you specify the EXOGENEITY, H=, J=, or NORMALIZE= option in the COINTEG statement, then the RESTRICT statement is ignored. Nonlinear restrictions on parameters are not supported.

Restricted parameter estimates are computed by introducing a Lagrangian parameter for each restriction (Pringle and Rayner 1971). The Lagrangian parameter measures the sensitivity of the sum of squared errors to the restriction. The estimates of these Lagrangian parameters and their significance are printed in the Restrict ODS table.

Matrix Expression

The RESTRICT statement operates on matrices. That is, you can specify the parameter matrices or constant matrices through the RESTRICT statement's built-in operators and functions. You can add elements of the matrices **A** and **B** with the expression **A+B**, and you can perform matrix multiplication with the expression

A*B and elementwise multiplication with the expression **A#B**. You can get the diagonal elements of the matrix **A** through the function **DIAG(A)**, and you can get the $n \times n$ identity matrix through the function **I(n)**.

Each restriction is written as a matrix expression composed of constants, operators, and functions.

Constants

Constants are either scalar constants (such as –1.2, 0.3, and so on) or matrix constants enclosed in braces (such as the 2×2 matrix, **{1 2, 3 4}**, or the 1×3 row vector, **{-0.2 5.3 12}**). Constants also include the dependent variable names and exogenous variable names that represent their index values and are mostly used in the subscripts or function arguments. For example, in the following PROC VARMAX statements, the dependent and exogenous variables have the following index values (based on their orders in the MODEL statement): GDP is equal to 1, CPI to 2, M2 to 3, FFR to 1, and CP to 2. Hence, the function call **AR(2, GDP, {CPI M2})** is equivalent to **AR(2,1,{2 3})**, and **XL(0, CPI, {FFR CP})** is equivalent to **XL(0,2,{1 2})**. For more information about the use of **AR** and **XL** functions to access parameters, see the section "Functions" on page 2924.

```
proc varmax data=macrodata;
   model GDP CPI M2 = FFR CP / p=12 xlag=12;
   restrict AR(2, GDP, {CPI M2}) = 0,
            XL(0, CPI, {FFR CP}) = 0;
run;
```

The matrix constant cannot be the first item in the RESTRICT statement. For example, you cannot specify the following statement:

```
restrict {-0.1 -0.2, -0.3 -0.4} <= AR <= {0.1 0.2, 0.3 0.4};
```

However, you can put the first matrix constant in parentheses and specify the preceding example in the following way:

```
restrict ({-0.1 -0.2, -0.3 -0.4}) <= AR <= {0.1 0.2, 0.3 0.4};
```

Operators

Operators define the operations on operands. Table 42.2 lists all built-in operators supported by the RESTRICT statement.

Table 42.2 Operators

Operator	Name	Description
+	Addition	Adds corresponding matrix elements
=	Comparison, equal	Compares matrix elements
<	Comparison, less than	Compares matrix elements
<=	Comparison, not greater than	Compares matrix elements
>	Comparison, greater than	Compares matrix elements
>=	Comparison, not less than	Compares matrix elements
||	Concatenation, horizontal	Concatenates matrices horizontally
//	Concatenation, vertical	Concatenates matrices vertically
@	Direct product	Takes the direct product of two matrices
:	Index creation	Creates an index vector
#	Multiplication, elementwise	Performs elementwise multiplication

Table 42.2 *continued*

Operator	Name	Description
*	Multiplication, matrix	Performs matrix multiplication
−	Sign reverse	Reverses the signs of elements
[ ]	Subscripts	Selects submatrices
−	Subtraction	Subtracts corresponding matrix elements
`	Transpose	Transposes a matrix

For more information about each operator, see the section "Details of Operators" on page 2929.

Table 42.3 shows the precedence of matrix operators in the RESTRICT statement.

Table 42.3 Operator Precedence

Priority Group	Operators		
I (highest)	[ ] (subscripts)	` (transpose)	
II	− (sign reverse)		
III	*	#	@
IV	− (subtraction)	+	
V	||	//	:
VI (lowest)	=	<	<= > >=

Each restriction can be a compound expression that involves several matrix operators and operands. The rules for evaluating compound expressions are as follows:

- Evaluation follows the order of operator precedence, as described in Table 42.3. Group I has the highest priority; that is, Group I operators are evaluated first. Group II operators are evaluated after Group I operators, and so on. For example, $1 + 2 * 3$ returns 7.
- If neighboring operators in an expression have equal precedence, the expression is evaluated from left to right, except for the Group I operators. For example, $1 - 2 - 3$ returns -4.
- All expressions in parentheses are evaluated first, following the two preceding rules. For example, $3 * (2 + 1)$ returns 9.

Functions

Functions are mainly divided into two categories: one type of function refers to parameters to be estimated, such as `AR(L,I,J)` and `CCC(I,J)`; the other type does not, such as `I(n)` and `DIAG(A)`.

Functions that refer to the parameters are listed in Table 42.4. The arguments for functions can be matrices. The simplest case, scalar arguments, is discussed first. For convenience, the scalar indices `i` and `j` refer to the position of the element in the coefficient matrix, and scalar `l` refers to the lag value.

Table 42.4 Functions That Refer to Parameters

Function	Description
ACH(l,i,j)	ARCH parameter of the lag l value of $\boldsymbol{\epsilon}_t \boldsymbol{\epsilon}_t'$ in a GARCH model
ALPHA(i,j)	The (i, j) element in the adjustment coefficient matrix $\boldsymbol{\alpha}$ for the vector error correction model
AR(l,i,j)	Autoregressive parameter of the lag l value of the jth dependent (endogenous) variable, $y_{j,t-l}$, to the ith dependent variable at time t, y_{it} for models other than error-correction models. For error correction models, **AR(1,i,j)** is the (i, j) element in $\Pi(= \boldsymbol{\alpha}\boldsymbol{\beta}')$ for $\mathbf{y}_{t-1}$, and **AR(l,i,j)**, $l > 1$, is the autoregressive parameter of the lag $(l-1)$ value of the jth differenced dependent (endogenous) variable, $\Delta y_{j,t-(l-1)}$, to the ith differenced dependent variable at time t, Δy_{it}
BETA(i,j)	The (i, j) element in the cointegrating matrix $\boldsymbol{\beta}$ for the vector error correction model
CCC(i,j)	Constant conditional correlation parameter between the ith and jth standardized error processes for the CCC GARCH model
CONST(i)	Intercept parameter of the ith time series, y_{it}
COV(i,j)	Covariance of innovations parameter between the ith and jth error processes when the maximum likelihood method is used for the fitted non-GARCH model
DCCA()	Parameter α in the correlation equation for the DCC GARCH model
DCCB()	Parameter β in the correlation equation for the DCC GARCH model
DCCS(i,j)	Unconditional correlation parameter between the ith and jth standardized error processes for the DCC GARCH model
EACH(l,i,j)	Exponential ARCH parameter of the lag l value of $\epsilon_{it}/\sigma_{it}$ in the CCC or DCC GARCH model when SUBFORM=EGARCH is specified and $i = j$. If $i \neq j$, the value is set to 0
ECCONST(i)	The ith element for the constant in the error correction term for the vector error correction model when the ECTREND option in the COINTEG statement is specified
ECLTREND(i)	The ith element for the linear trend in the error correction term for vector error correction model when the ECTREND option in the COINTEG statement is specified
GCH(l,i,j)	GARCH parameter of the lag l value of the covariance matrix, H_t, in a GARCH model
GCHC(i,j)	Constant parameter of the covariance matrix, H_t, in a GARCH model
LAMBDA(i)	Power parameter for the ith error process in the CCC or DCC GARCH model when SUBFORM=PGARCH is specified
LTREND(i)	Linear trend parameter of the ith time series, y_{it}, when the TREND= option is specified
MA(l,i,j)	Moving average parameter of the lag l value of the jth error process, $\epsilon_{j,t-l}$, to the ith dependent variable at time t, y_{it}
PACH(l,i,j)	Power ARCH parameter of the lag l value of ϵ_{it} in the CCC or DCC GARCH model when SUBFORM=PGARCH is specified and $i = j$. If $i \neq j$, the value is set to 0.

Table 42.4 *continued*

Function	Description
QACH(l,i,j)	Quadratic ARCH center parameter of the lag l value of ϵ_{it} in the CCC or DCC GARCH model when SUBFORM=QGARCH is specified and $i = j$. If $i \neq j$, the value is set to 0.
QTREND(i)	Quadratic trend parameter of the ith time series, y_{it}, when TREND=QUAD is specified
SD(i,j)	Same as **SDUMMY(i, j)**
SDUMMY(i,j)	The jth seasonal dummy of the ith time series at time t, y_{it}, where $j = 1, \ldots, (nseason-1)$, where *nseason* is the value of the NSEASON= option in the MODEL statement
TACH(l,i,j)	Threshold ARCH parameter of the lag l value of $1_{\epsilon_{it}<0}\epsilon_{it}^2$ in the CCC or DCC GARCH model when SUBFORM=GJR is specified and $i = j$. If $i \neq j$, the value is set to 0.
XL(l,i,j)	Exogenous parameter of the lag l value of the jth exogenous (independent) variable, $x_{j,t-l}$, to the ith dependent variable at time t, y_{it}

The functions that refer to parameters, as shown in Table 42.4, accept vector arguments and return the matrix that is constructed by the corresponding parameters. According to the number of arguments, the following list shows what matrix a function returns when the arguments are vectors:

- A function, **FUNC0**, that has zero arguments, always returns the corresponding scalar parameter. **DCCA** and **DCCB** are types of **FUNC0**.
- A function, **FUNC1**, that has one vector argument I, where $I = (i_1\ i_2\ \ldots\ i_{n_I})'$, returns a vector $R = (r_1\ r_2\ \ldots\ r_{n_I})'$, where $r_k =$ **FUNC1(**i_k**)**, $k = 1, ..., n_I$. **CONST**, **ECCONST**, **ECLTREND**, **LAMBDA**, **LTREND**, and **QTREND** are types of **FUNC1**.
- A function, **FUNC2**, that has two vector arguments I and J, where $I = (i_1\ i_2\ \ldots\ i_{n_I})'$ and $J = (j_1\ j_2\ \ldots\ j_{n_J})'$, returns a matrix

$$R = \begin{pmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,n_J} \\ r_{2,1} & r_{2,2} & \cdots & r_{2,n_J} \\ \cdots & & & \\ r_{n_I,1} & r_{n_I,2} & \cdots & r_{n_I,n_J} \end{pmatrix}$$

  where $r_{k,m} =$ **FUNC2(**i_k, j_m**)**, $k = 1, ..., n_I, m = 1, ..., n_J$. **ALPHA**, **BETA**, **CCC**, **COV**, **DCCS**, **GCHC**, **SD**, and **SDUMMY** and types of **FUNC2**.
- A function, **FUNC3**, that has three vector arguments L, I, and J, where $L = (l_1\ l_2\ \ldots\ l_{n_L})'$, $I = (i_1\ i_2\ \ldots\ i_{n_I})'$, and $J = (j_1\ j_2\ \ldots\ j_{n_J})'$, returns a matrix

$$R = \begin{pmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,n_L n_J} \\ r_{2,1} & r_{2,2} & \cdots & r_{2,n_L n_J} \\ \cdots & & & \\ r_{n_I,1} & r_{n_I,2} & \cdots & r_{n_I,n_L n_J} \end{pmatrix}$$

where $r_{k,m}$ = **FUNC3**(l_m, i_k, j_m), $k = 1, ..., n_I, m = 1, ..., n_L n_J$, and l_m and j_m are the quotient and remainder of m divided by n_J, respectively. **ACH**, **AR**, **EACH**, **GCH**, **MA**, **PACH**, **QACH**, **TACH**, and **XL** are types of **FUNC3**.

The functions that refer to parameters can accept empty arguments or omit any number of last arguments. The empty or omitted arguments are replaced with all possible values for those arguments. For example, PROC VARMAX is used to fit a bivariate (k=2) VARX(1,1) model with three exogenous variables as follows:

```
model y1 y2 = x1 x2 x3 / p=1 xlag=3;
```

In order to restrict the third exogenous variable from having an effect on the first dependent variable, and to restrict the first exogenous variable from having an effect on the second dependent variable, you can use the following statement:

```
restrict XL({0 1 2 3}, 1, 3) = 0,
         XL({0 1 2 3}, 2, 1) = 0;
```

Taking advantage of empty arguments, you can specify the preceding example as follows:

```
restrict XL( , 1, 3) = 0,
         XL( , 2, 1) = 0;
```

To get all coefficients of the first lag exogenous variables on dependent variables, you can use **XL(1, {1 2}, {1 2 3})** or **XL(1, ,)** or **XL(1)**. To get all coefficients of exogenous variables on dependent variables, you can use **XL({0 1 2 3}, {1 2}, {1 2 3})**, or **XL(, ,)** or **XL()** or even just **XL**.

Another type of function does not refer to parameters but generates useful matrices. Table 42.5 lists all built-in functions supported by the RESTRICT statement.

Table 42.5 Functions Not Referring to Parameters

Function	Description
DIAG(A)	Creates a diagonal matrix from a vector or extracts the diagonal elements of a matrix
I(n)	Creates an $n \times n$ identity matrix
J(m,n,elem)	Creates an $m \times n$ matrix with all elements equal to **elem**
SHAPE(A,m,n)	Creates a $m \times n$ matrix with elements of matrix **A**

For more information about each function in Table 42.5, see the section "Details of Functions" on page 2933.

Examples

The following examples show how to use the RESTRICT statement.

This example shows a bivariate (k=2) VAR(2) model:

```
proc varmax data=one;
   model y1 y2 / p=2;
   restrict AR(1,1,2)=0, AR(2,1,2)=0.3;
run;
```

The **AR(1,1,2)** and **AR(2,1,2)** parameters are fixed as **AR(1,1,2)=0** and **AR(2,1,2)=0.3**, respectively, and other parameters are to be estimated.

The following example shows a bivariate (k=2) VAR(1) model, estimated using the ML method:

```
proc varmax data=two;
   model y1 y2 = / p=1 method=ml;
   restrict cov(1,1)=cov(2,2), cov(1,2)=0;
run;
```

The **COV(1,1)** and **COV(2,2)** parameters are equal, and the correlation between the two series is fixed at 0. You can also express the preceding restrictions in matrix expressions as follows. This approach is very convenient when the number of dependent variables is large:

```
proc varmax data=two;
   model y1 y2 = / p=1 method=ml;
   restrict cov = cov(1,1)*I(2);
run;
```

When restricting a linear combination of parameters to be 0, you can omit the equal sign. For example, the following two RESTRICT statements are equivalent:

```
restrict AR(1)[1,1]-AR(1)[2,2], 2*MA(1)[1,2]-MA(1)[2,1];

restrict AR(1)[1,1]-AR(1)[2,2] = 0, 2*MA(1)[1,2]-MA(1)[2,1] = 0;
```

The following RESTRICT statement constrains four parameter estimates to be equal:

```
restrict AR(1)[1,1] = AR(1)[1,2],
         AR(1)[1,2] = AR(1)[2,1],
         AR(1)[2,1] = AR(1)[2,2];
```

This restriction can be abbreviated as follows:

```
restrict AR(1)[1,1] = AR(1)[1,2] = AR(1)[2,1] = AR(1)[2,2];
```

Or, in matrix expressions,

```
restrict AR(1,1:2,1:2) = J(2,2,AR(1,1,1));
```

The VARMA representation $A(L)y_t = \Theta(L)\varepsilon_t$, where $A(L) = I_k - A_1L - \cdots - A_pL^p$ and $\Theta(L) = I_k - \Theta_1L - \cdots - \Theta_qL^q$, is said to be in final equation form if $A(L) = a(L)I_k$, where $a(L) = 1 - a_1L - \cdots - a_pL^p$ is a scalar operator with $a_p \neq 0$. If p and k are large, it would be difficult and inconvenient to restrict AR parameters element by element in standard form to estimate the VARMA model in final equation form. However, when you use matrix expressions, the restrictions become very simple, as shown in the following statement for a trivariate ($k = 3$) VARMA(p, q) model, where p might be any positive integer:

```
restrict AR = AR(,1,1) @ I(3);
```

For the vector error correction models, the **AR(1,.,.)** parameters (that is, Π) are not supported in the RESTRICT statement, because **AR(1)** is in fact the product of the estimated parameters $\boldsymbol{\alpha}$ and the transpose of $\boldsymbol{\beta}$. Any linear constraints on **AR(1)** should be regarded as nonlinear constraints on the estimated parameters. For the same reason, the **CONST(.)** or **LTREND(.)** functions are not supported in the RESTRICT statement if the ECTREND option in the COINTEG statement is specified. For example, the following statements are supported:

```
model y1-y4 / p=2;
cointeg rank=1 ectrend;
restrict ALPHA + BETA = 1.0,
         ECCONST;
```

However, neither of the following sets of statements are supported:

```
model y1-y4 / p=2;
cointeg rank=1 ectrend;
restrict AR(1,1,1) = 0;
```

```
model y1-y4 / p=2;
cointeg rank=1 ectrend;
restrict CONST(2) = 0.2;
```

Details of Operators

This section describes all operators that are available in the RESTRICT statement. Each subsection shows how the operator is used, followed by a description of the operator.

Addition Operator: +

matrix1 + matrix2

matrix + scalar

matrix + vector

The addition operator (+) computes a new matrix whose elements are the sums of the corresponding elements of **matrix1** and **matrix2**. If **matrix1** and **matrix2** are both $n \times p$ matrices, then the addition operator adds the element in the ith row and jth column of the first matrix to the element in the ith row and jth column of the second matrix, for $i = 1, \ldots, n$, $j = 1, \ldots, p$. For example, **{1 2 3, 4 5 6} + {7 8 9, 10 11 12}** results in **{8 10 12, 14 16 18}**.

You can also use the addition operator as follows to conveniently add a value to each element of a matrix, to each column of a matrix, or to each row of a matrix:

- When you use the **matrix** + **scalar** form, the scalar value is added to each element of the matrix.
- When you use the **matrix** + **vector** form, the vector is added to each row or column of the $n \times p$ matrix.
  - If you add an $n \times 1$ column vector, each row of the vector is added to each row of the matrix.
  - If you add a $1 \times p$ row vector, each column of the vector is added to each column of the matrix.

For example, you can obtain **{2 3 4, 5 6 7}** from **{1 2 3, 4 5 6} + 1** or **{1 2 3, 4 5 6} + {1 1 1}** or **{1 2 3, 4 5 6} + {1, 1}**.

Comparison Operators: =, <, <=, >, >=

matrix1 = matrix2

matrix1 < matrix2

```
matrix1 <= matrix2

matrix1 > matrix2

matrix1 >= matrix2
```

The comparison operators (=, <, <=, >, >=) compare two matrices element by element and return a list of equivalent restrictions on only scalar constants and parameters.

For example, the RESTRICT statement with matrix expressions

```
restrict AR(1,{1,2},{1,2}) = MA(2,{3,4},{3,4});
```

is transformed into the following equivalent RESTRICT statement with scalar parameters:

```
restrict AR(1,1,1) = MA(2,3,3),
         AR(1,1,2) = MA(2,3,4),
         AR(1,2,1) = MA(2,4,3),
         AR(1,2,2) = MA(2,4,4);
```

You can also use the comparison operators to conveniently compare all elements of a matrix with a scalar:

- If either argument is a scalar, then the VARMAX procedure performs an elementwise comparison between each element of the matrix and the scalar.

You can also compare an $n \times p$ matrix with a row or column vector:

- If the comparison is with an $n \times 1$ column vector, the VARMAX procedure compares each row of the vector to each row of the matrix.
- If the comparison is with a $1 \times p$ row vector, the VARMAX procedure compares each column of the vector to each column of the matrix.

For example, the following statements are equivalent:

```
restrict AR(1,1:2,1:3) >= 0.2;

restrict AR(1,1:2,1:3) >= {0.2, 0.2};

restrict AR(1,1:2,1:3) >= {0.2 0.2 0.2};
```

Concatenation Operator, Horizontal: ||

matrix1 || **matrix2**

The horizontal concatenation operator (||) produces a new matrix by horizontally joining **matrix1** and **matrix2**. The matrices must have the same number of rows, which is also the number of rows in the new matrix. The number of columns in the new matrix is the number of columns in **matrix1** plus the number of columns in **matrix2**.

For example, `{1 1 1, 7 7 7} || {0 0 0, 8 8 8}` returns `{1 1 1 0 0 0, 7 7 7 8 8 8}`.

Concatenation Operator, Vertical: //

matrix1 // matrix2

The vertical concatenation operator (//) produces a new matrix by vertically joining **matrix1** and **matrix2**. The matrices must have the same number of columns, which is also the number of columns in the new matrix. The number of rows in the new matrix is the number of rows in **matrix1** plus the number of rows in **matrix2**.

For example, **{1 1 1} // {0 0 0, 8 8 8}** returns **{1 1 1, 0 0 0, 8 8 8}**.

Direct Product Operator: @

matrix1 @ matrix2

The direct product operator (@) computes a new matrix that is the direct product (also called the *Kronecker product*) of **matrix1** and **matrix2**. For matrices $\mathbf{A}$ and $\mathbf{B}$, the direct product is denoted by $\mathbf{A} \otimes \mathbf{B}$. The number of rows in the new matrix equals the product of the number of rows in **matrix1** and the number of rows in **matrix2**; the number of columns in the new matrix equals the product of the number of columns in **matrix1** and the number of columns in **matrix2**.

Specifically, if $\mathbf{A}$ is an $n \times p$ matrix and $\mathbf{B}$ is a $m \times q$ matrix, then the Kronecker product $\mathbf{A} \otimes \mathbf{B}$ is the following $nm \times pq$ block matrix:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} A_{11}B & \cdots & A_{1p}B \\ \vdots & \ddots & \vdots \\ A_{n1}B & \cdots & A_{np}B \end{bmatrix}$$

For example, **{1 2, 3 4} @ {0 2}** returns **{0 2 0 4, 0 6 0 8}**, and **{0 2} @ {1 2, 3 4}** returns **{0 0 2 4, 0 0 6 8}**. Note that the direct product of two matrices is not commutative.

Index Creation Operator: :

value1 : value2

The index creation operator (:) creates a column vector whose first element is **value1**, whose second element is **value1**+1, and so on, until the last element, which is less than or equal to **value2**.

For example, **3 : 6** returns **{3 4 5 6}**.

If **value1** is greater than **value2**, a reverse-order index is created. For example, **6 : 3** returns **{6 5 4 3}**.

Neither **value1** nor **value2** is required to be an integer.

Multiplication Operator, Elementwise:

matrix1 # matrix2

matrix # scalar

matrix # vector

The elementwise multiplication operator (#) computes a new matrix whose elements are the products of the corresponding elements of **matrix1** and **matrix2**.

For example, **{1 2, 3 4} # {4 8, 0 5}** returns **{4 16, 0 20}**.

In addition to multiplying matrices that have the same dimensions, you can use the elementwise multiplication operator to multiply a matrix and a scalar:

- When either argument is a scalar, each element in **matrix** is multiplied by the scalar value.

When you use the **matrix # vector** form, each row or column of the $n \times p$ matrix is multiplied by a corresponding element of the vector:

- If you multiply by an $n \times 1$ column vector, each row of the matrix is multiplied by the corresponding row of the vector.
- If you multiply by a $1 \times p$ row vector, each column of the matrix is multiplied by the corresponding column of the vector.

For example, a 2×3 matrix can be multiplied on either side by a 2×3, 1×3, 2×1, or 1×1 scalar.

The product of elementwise multiplication is also known as the Schur or Hadamard product. Elementwise multiplication (which uses the # operator) should not be confused with matrix multiplication (which uses the * operator).

*Multiplication Operator, Matrix: **

```
matrix1 * matrix2
```

The matrix multiplication operator (*) computes a new matrix by performing matrix multiplication. The first matrix must have the same number of columns as the second matrix has rows. The new matrix has the same number of rows as the first matrix and the same number of columns as the second matrix. That is, if $\boldsymbol{A}$ is an $n \times p$ matrix and $\boldsymbol{B}$ is a $p \times m$ matrix, then the product $A * B$ is an $n \times m$ matrix. The (i, j) element of the product is the sum $\sum_{k=1}^{p} A_{ik} B_{kj}$.

For example, `{1 2, 3 4} * {1, 2}` returns `{5, 11}`.

Sign Reversal Operator: –

```
– matrix
```

The sign reversal operator (−) computes a new matrix whose elements are formed by reversing the sign of each element in **matrix**. The sign reversal operator is also called the *unary minus* operator.

For example, `–{–1 7 6, 2 0 –8}` returns `{1 –7 –6, –2 0 8}`.

Subscripts: [ ]

```
matrix[rows, columns]

matrix[elements]
```

Subscripts are used with matrices to select submatrices, where **rows**, **columns**, and **elements** are expressions that evaluate to scalars or vectors. If these expressions are numeric, they must contain valid subscript values of rows and columns, or the indices, in the argument matrix.

For example, `{1 2 3, 4 5 6, 7 8 9}[2,3]` returns `6`, `{1 2 3, 4 5 6, 7 8 9}[2,1:3]` returns `{4 5 6}`, and `{1 2 3, 4 5 6, 7 8 9}[,3]` returns `{3, 6, 9}`. Because the VARMAX procedure stores matrices in row-major order, `{11 22 33, 44 55 66, 77 88 99}[{3 5 9}]` returns `{33, 55, 99}`.

Subtraction Operator: –

matrix1 – matrix2

matrix – scalar

matrix – vector

The subtraction operator (−) computes a new matrix whose elements are formed by subtracting the corresponding elements of **matrix2** from those of **matrix1**.

In addition to subtracting conformable matrices, you can also use the subtraction operator to subtract a scalar from a matrix or subtract a vector from a matrix:

- When either argument is a scalar, the VARMAX procedure performs the subtraction between the scalar and each element of the matrix argument. For example, when you use the **matrix – scalar** form, the scalar value is subtracted from each element of the matrix.
- When you use the **matrix – vector** form, the vector is subtracted from each row or column of the $n \times p$ matrix.
  - If you subtract an $n \times 1$ column vector, each row of the vector is subtracted from each row of the matrix.
  - If you subtract a $1 \times p$ row vector, each column of the vector is subtracted from each column of the matrix.

For example, **{1 2 3, 4 5 6} – {1 1 1, 1 1 1}** returns **{0 1 2, 3 4 5}**. The same results can be obtained by **{1 2 3, 4 5 6} – 1** or **{1 2 3, 4 5 6} – {1 1 1}** or **{1 2 3, 4 5 6} – {1, 1}**.

Transpose Operator: `

matrix`

The transpose operator, denoted by the backquote character (`), exchanges the rows and columns of **matrix**, producing the transpose of **matrix**. If v is the value in the ith row and jth column of **matrix**, then the transpose of **matrix** contains v in the jth row and ith column. If **matrix** contains n rows and p columns, the transpose has p rows and n columns.

For example, **{1 2, 3 4, 5 6}`** returns **{1 3 5, 2 4 6}**.

Details of Functions

DIAG Function

DIAG(matrix)

The **DIAG** function creates a diagonal matrix from a vector or extracts the diagonal elements of a matrix. The **matrix** argument can be either a square matrix or a vector.

If **matrix** is a vector, the **DIAG** function creates a matrix whose diagonal elements are the values in the vector. All off-diagonal elements are zeros.

If **matrix** is a square matrix, the **DIAG** function creates a vector from the diagonal elements of the matrix.

For example, **DIAG({1 2 3, 4 5 6, 7 8 9})** returns **{1, 5, 9}**. Also, **DIAG({1 5 9})** or **DIAG({1, 5, 9})** or **DIAG(DIAG({1 2 3, 4 5 6, 7 8 9}))** returns **{1 0 0, 0 5 0, 0 0 9}**.

I Function

I(dim)

The **I** function creates an identity matrix that contains **dim** rows and columns. The diagonal elements of an identity matrix are ones; all other elements are zeros. The value of **dim** must be an integer greater than or equal to 1. Noninteger operands are rounded to the nearest integer.

For example, **I(3)** returns **{1 0 0, 0 1 0, 0 0 1}**.

J Function

J(nrow, ncol, value)

The **J** function creates a matrix that contains **nrow** rows and **ncol** columns, in which all elements are equal to **value**.

The arguments **nrow** and **ncol** are both integers; **value** can be any expression that returns a linear combination of scalar constants and parameters.

For example, **J(2, 3, 1)** returns **{1 1 1, 1 1 1}**. **J(2, 3, 5+2*AR(1,1,1))** returns the same result as **J(2, 3, 1) * (5+2*AR(1,1,1))**.

SHAPE Function

SHAPE(matrix, nrow, ncol)

The **SHAPE** function creates a new matrix from data in **matrix**. The values **nrow** and **ncol** specify the number of rows and columns, respectively, in the new matrix. The **SHAPE** function produces the result matrix by traversing the argument matrix in row-major order until it reaches the specified number of elements. If necessary, the SHAPE function reuses elements.

For example, **SHAPE({1 2 3, 4 5 6}, 3, 2)** returns **{1 2, 3 4, 5 6}**; **SHAPE({1 2 3, 4 5 6}, 5, 2)** returns **{1 2, 3 4, 5 6, 1 2, 3 4}**; and **SHAPE({1 2 3, 4 5 6}, 1, 4)** returns **{1 2 3 4}**.

TEST Statement

TEST *restriction, . . . , restriction* **;**

The TEST statement performs the Wald test for the joint linear hypothesis that is specified in the statement. Each restriction specifies a linear hypothesis to be tested. If you specify more than one *restriction*, separate them with commas. Specify the *restrictions* in the same manner as in the RESTRICT statement. For information about how to define restriction by using matrix expressions, operators, and functions, see the section "RESTRICT Statement" on page 2922. You can specify any number of TEST statements.

To use the TEST statement, you need to know the form of the model. If you do not specify the GARCH statement, the COINTEG statement, or the ECM=, P=, Q=, or XLAG= option in the MODEL statement, then the TEST statement is not applicable. Nonlinear restrictions on parameters are not supported.

For information about the Wald test, see the section "Granger Causality Test" on page 2966.

The following is an example of the TEST statement for a bivariate (k=2) VAR(2) model:

```
proc varmax data=one;
   model y1 y2 / p=2;
   test AR(1,1,2) = 0, AR(2,1,2) = 0;
run;
```

After estimating the parameters, the TEST statement tests the null hypothesis that **AR(1,1,2)=0** and **AR(2,1,2)=0**. Like the RESTRICT statement, the preceding TEST statement can be abbreviated as follows:

```
test AR(1,1,2) = AR(2,1,2) = 0;
```

or

```
test AR(1,1,2), AR(2,1,2);
```

Note that the following statements are different from the preceding statement:

```
test AR(1,1,2);
test AR(2,1,2);
```

These two TEST statements are to test two null hypotheses separately: one is **AR(1,1,2)=0**, and the other is **AR(2,1,2)=0**.

For the vector error correction models, you can test the hypothesis on the **AR(1,.,.)** parameters (that is, Π) by using the TEST statement, because asymptotically these parameters follow a normal distribution and the Wald test can be applied. For the same reason, you can use the **CONST(.)** or **LTREND(.)** function in the TEST statement if the ECTREND option in the COINTEG statement is specified. However, the **BETA(.,.)**, **ECCONST(.)**, and **ECLTREND(.)** functions are not supported in the TEST statement. For example, the following statements are supported,

```
model y1-y4 / p=2;
cointeg rank=1 ectrend;
test AR(1,1,1);
test CONST(2);
```

However, the following statements are not supported,

```
model y1-y4 / p=2;
cointeg rank=1 ectrend;
test BETA(1,1) = BETA(2,1) = 0;
```

or

```
model y1-y4 / p=2;
cointeg rank=1 ectrend;
test ECCONST(1) = 0.2;
```

Details: VARMAX Procedure

Missing Values

The VARMAX procedure currently does not support missing values. PROC VARMAX uses the first contiguous group of observations that have no missing values for any of the MODEL statement variables. Observations at the beginning of the data set that have missing values for any MODEL statement variables are not used or included in the output data set. At the end of the data set, observations can have dependent (endogenous) variables with missing values and independent (exogenous) variables with nonmissing values.

VARMAX Model

The vector autoregressive moving-average model with exogenous variables is called the VARMAX(p,q,s) model. The form of the model can be written as

$$\mathbf{y}_t = \sum_{i=1}^{p} \Phi_i \mathbf{y}_{t-i} + \sum_{i=0}^{s} \Theta_i^* \mathbf{x}_{t-i} + \boldsymbol{\epsilon}_t - \sum_{i=1}^{q} \Theta_i \boldsymbol{\epsilon}_{t-i}$$

where the output variables of interest, $\mathbf{y}_t = (y_{1t}, \ldots, y_{kt})'$, can be influenced by other input variables, $\mathbf{x}_t = (x_{1t}, \ldots, x_{rt})'$, which are determined outside of the system of interest. The variables $\mathbf{y}_t$ are referred to as dependent, response, or endogenous variables, and the variables $\mathbf{x}_t$ are referred to as independent, input, predictor, regressor, or exogenous variables. The unobserved noise variables, $\boldsymbol{\epsilon}_t = (\epsilon_{1t}, \ldots, \epsilon_{kt})'$, are a vector white noise process.

The VARMAX(p,q,s) model can be written

$$\Phi(B)\mathbf{y}_t = \Theta^*(B)\mathbf{x}_t + \Theta(B)\boldsymbol{\epsilon}_t$$

where

$$\begin{aligned} \Phi(B) &= I_k - \Phi_1 B - \cdots - \Phi_p B^p \\ \Theta^*(B) &= \Theta_0^* + \Theta_1^* B + \cdots + \Theta_s^* B^s \\ \Theta(B) &= I_k - \Theta_1 B - \cdots - \Theta_q B^q \end{aligned}$$

are matrix polynomials in B in the backshift operator, such that $B^i \mathbf{y}_t = \mathbf{y}_{t-i}$, the Φ_i and Θ_i are $k \times k$ matrices, and the Θ_i^* are $k \times r$ matrices.

The following assumptions are made:

- $\mathrm{E}(\boldsymbol{\epsilon}_t) = 0$, $\mathrm{E}(\boldsymbol{\epsilon}_t \boldsymbol{\epsilon}_t') = \Sigma$, which is positive-definite, and $\mathrm{E}(\boldsymbol{\epsilon}_t \boldsymbol{\epsilon}_s') = 0$ for $t \neq s$.
- For stationarity and invertibility of the VARMAX process, the roots of $|\Phi(z)| = 0$ and $|\Theta(z)| = 0$ are outside the unit circle.

- The exogenous (independent) variables $\mathbf{x}_t$ are not correlated with residuals $\boldsymbol{\epsilon}_t$, $\mathrm{E}(\mathbf{x}_t\boldsymbol{\epsilon}_t') = 0$. The exogenous variables can be stochastic or nonstochastic. When the exogenous variables are stochastic and their future values are unknown, forecasts of these future values are needed to forecast the future values of the endogenous (dependent) variables. On occasion, future values of the exogenous variables can be assumed to be known because they are deterministic variables. The VARMAX procedure assumes that the exogenous variables are nonstochastic if future values are available in the input data set. Otherwise, the exogenous variables are assumed to be stochastic and their future values are forecasted by assuming that they follow the VARMA(p,q) model, prior to forecasting the endogenous variables, where p and q are the same as in the VARMAX(p,q,s) model.

State Space Representation

Another representation of the VARMAX(p,q,s) model is in the form of a state variable or a state space model, which consists of a state equation

$$\mathbf{z}_t = F\mathbf{z}_{t-1} + K\mathbf{x}_t + G\boldsymbol{\epsilon}_t$$

and an observation equation

$$\mathbf{y}_t = H\mathbf{z}_t$$

where

$$\mathbf{z}_t = \begin{bmatrix} \mathbf{y}_t \\ \vdots \\ \mathbf{y}_{t-p+1} \\ \mathbf{x}_t \\ \vdots \\ \mathbf{x}_{t-s+1} \\ \boldsymbol{\epsilon}_t \\ \vdots \\ \boldsymbol{\epsilon}_{t-q+1} \end{bmatrix}, \quad K = \begin{bmatrix} \Theta_0^* \\ 0_{k\times r} \\ \vdots \\ 0_{k\times r} \\ I_r \\ 0_{r\times r} \\ \vdots \\ 0_{r\times r} \\ 0_{k\times r} \\ \vdots \\ 0_{k\times r} \end{bmatrix}, \quad G = \begin{bmatrix} I_k \\ 0_{k\times k} \\ \vdots \\ 0_{k\times k} \\ 0_{r\times k} \\ \vdots \\ 0_{r\times k} \\ I_{k\times k} \\ 0_{k\times k} \\ \vdots \\ 0_{k\times k} \end{bmatrix}$$

$$F = \begin{bmatrix} \Phi_1 & \cdots & \Phi_{p-1} & \Phi_p & \Theta_1^* & \cdots & \Theta_{s-1}^* & \Theta_s^* & -\Theta_1 & \cdots & -\Theta_{q-1} & -\Theta_q \\ I_k & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & I_k & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 & I_r & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & I_r & 0 & 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & I_k & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & I_k & 0 \end{bmatrix}$$

and

$$H = [I_k, 0_{k\times k}, \ldots, 0_{k\times k}, 0_{k\times r}, \ldots, 0_{k\times r}, 0_{k\times k}, \ldots, 0_{k\times k}]$$

On the other hand, it is assumed that $\mathbf{x}_t$ follows a VARMA(p,q) model

$$\mathbf{x}_t = \sum_{i=1}^{p} A_i \mathbf{x}_{t-i} + \mathbf{a}_t - \sum_{i=1}^{q} C_i \mathbf{a}_{t-i}$$

The model can also be expressed as

$$A(B)\mathbf{x}_t = C(B)\mathbf{a}_t$$

where $A(B) = I_r - A_1 B - \cdots - A_p B^p$ and $C(B) = I_r - C_1 B - \cdots - C_q B^q$ are matrix polynomials in B, and the A_i and C_i are $r \times r$ matrices. Without loss of generality, the AR and MA orders can be taken to be the same as the VARMAX(p,q,s) model, and $\mathbf{a}_t$ and $\boldsymbol{\epsilon}_t$ are independent white noise processes.

Under suitable conditions such as stationarity, $\mathbf{x}_t$ is represented by an infinite order moving-average process

$$\mathbf{x}_t = A(B)^{-1}C(B)\mathbf{a}_t = \Psi^x(B)\mathbf{a}_t = \sum_{j=0}^{\infty} \Psi_j^x \mathbf{a}_{t-j}$$

where $\Psi^x(B) = A(B)^{-1}C(B) = \sum_{j=0}^{\infty} \Psi_j^x B^j$.

The optimal minimum mean squared error (minimum MSE) i-step-ahead forecast of $\mathbf{x}_{t+i}$ is

$$\begin{aligned} \mathbf{x}_{t+i|t} &= \sum_{j=i}^{\infty} \Psi_j^x \mathbf{a}_{t+i-j} \\ \mathbf{x}_{t+i|t+1} &= \mathbf{x}_{t+i|t} + \Psi_{i-1}^x \mathbf{a}_{t+1} \end{aligned}$$

For $i > q$,

$$\mathbf{x}_{t+i|t} = \sum_{j=1}^{p} A_j \mathbf{x}_{t+i-j|t}$$

The VARMAX(p,q,s) model has an absolutely convergent representation as

$$\begin{aligned} \mathbf{y}_t &= \Phi(B)^{-1}\Theta^*(B)\mathbf{x}_t + \Phi(B)^{-1}\Theta(B)\boldsymbol{\epsilon}_t \\ &= \Psi^*(B)\Psi^x(B)\mathbf{a}_t + \Phi(B)^{-1}\Theta(B)\boldsymbol{\epsilon}_t \\ &= V(B)\mathbf{a}_t + \Psi(B)\boldsymbol{\epsilon}_t \end{aligned}$$

or

$$\mathbf{y}_t = \sum_{j=0}^{\infty} V_j \mathbf{a}_{t-j} + \sum_{j=0}^{\infty} \Psi_j \boldsymbol{\epsilon}_{t-j}$$

where $\Psi(B) = \Phi(B)^{-1}\Theta(B) = \sum_{j=0}^{\infty} \Psi_j B^j$, $\Psi^*(B) = \Phi(B)^{-1}\Theta^*(B)$, and $V(B) = \Psi^*(B)\Psi^x(B) = \sum_{j=0}^{\infty} V_j B^j$.

The optimal (minimum MSE) i-step-ahead forecast of $\mathbf{y}_{t+i}$ is

$$\begin{aligned} \mathbf{y}_{t+i|t} &= \sum_{j=i}^{\infty} V_j \mathbf{a}_{t+i-j} + \sum_{j=i}^{\infty} \Psi_j \boldsymbol{\epsilon}_{t+i-j} \\ \mathbf{y}_{t+i|t+1} &= \mathbf{y}_{t+i|t} + V_{i-1}\mathbf{a}_{t+1} + \Psi_{i-1}\boldsymbol{\epsilon}_{t+1} \end{aligned}$$

for $i = 1, \ldots, v$ with $v = \max(p, q+1)$. For $i > q$,

$$\begin{aligned}
\mathbf{y}_{t+i|t} &= \sum_{j=1}^{p} \Phi_j \mathbf{y}_{t+i-j|t} + \sum_{j=0}^{s} \Theta_j^* \mathbf{x}_{t+i-j|t} \\
&= \sum_{j=1}^{p} \Phi_j \mathbf{y}_{t+i-j|t} + \Theta_0^* \mathbf{x}_{t+i|t} + \sum_{j=1}^{s} \Theta_j^* \mathbf{x}_{t+i-j|t} \\
&= \sum_{j=1}^{p} \Phi_j \mathbf{y}_{t+i-j|t} + \Theta_0^* \sum_{j=1}^{p} A_j \mathbf{x}_{t+i-j|t} + \sum_{j=1}^{s} \Theta_j^* \mathbf{x}_{t+i-j|t} \\
&= \sum_{j=1}^{p} \Phi_j \mathbf{y}_{t+i-j|t} + \sum_{j=1}^{u} (\Theta_0^* A_j + \Theta_j^*) \mathbf{x}_{t+i-j|t}
\end{aligned}$$

where $u = \max(p, s)$.

Define $\Pi_j = \Theta_0^* A_j + \Theta_j^*$. For $i = v > q$ with $v = \max(p, q+1)$, you obtain

$$\begin{aligned}
\mathbf{y}_{t+v|t} &= \sum_{j=1}^{p} \Phi_j \mathbf{y}_{t+v-j|t} + \sum_{j=1}^{u} \Pi_j \mathbf{x}_{t+v-j|t} \quad \text{for } u \le v \\
\mathbf{y}_{t+v|t} &= \sum_{j=1}^{p} \Phi_j \mathbf{y}_{t+v-j|t} + \sum_{j=1}^{r} \Pi_j \mathbf{x}_{t+v-j|t} \quad \text{for } u > v
\end{aligned}$$

From the preceding relations, a state equation is

$$\mathbf{z}_{t+1} = F\mathbf{z}_t + K\mathbf{x}_t^* + G\mathbf{e}_{t+1}$$

and an observation equation is

$$\mathbf{y}_t = H\mathbf{z}_t$$

where

$$\mathbf{z}_t = \begin{bmatrix} \mathbf{y}_t \\ \mathbf{y}_{t+1|t} \\ \vdots \\ \mathbf{y}_{t+v-1|t} \\ \mathbf{x}_t \\ \mathbf{x}_{t+1|t} \\ \vdots \\ \mathbf{x}_{t+v-1|t} \end{bmatrix}, \quad \mathbf{x}_t^* = \begin{bmatrix} \mathbf{x}_{t+v-u} \\ \mathbf{x}_{t+v-u+1} \\ \vdots \\ \mathbf{x}_{t-1} \end{bmatrix}, \quad \mathbf{e}_{t+1} = \begin{bmatrix} \mathbf{a}_{t+1} \\ \boldsymbol{\epsilon}_{t+1} \end{bmatrix}$$

$$F = \begin{bmatrix} 0 & I_k & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & I_k & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \Phi_v & \Phi_{v-1} & \Phi_{v-2} & \cdots & \Phi_1 & \Pi_v & \Pi_{v-1} & \Pi_{v-2} & \cdots & \Pi_1 \\ 0 & 0 & 0 & \cdots & 0 & 0 & I_r & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 & I_r & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & A_v & A_{v-1} & A_{v-2} & \cdots & A_1 \end{bmatrix}$$

$$K = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \Pi_u & \Pi_{u-1} & \cdots & \Pi_{v+1} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}, \quad G = \begin{bmatrix} V_0 & I_k \\ V_1 & \Psi_1 \\ \vdots & \vdots \\ V_{v-1} & \Psi_{v-1} \\ I_r & 0_{r\times k} \\ \Psi_1^x & 0_{r\times k} \\ \vdots & \vdots \\ \Psi_{v-1}^x & 0_{r\times k} \end{bmatrix}$$

and

$$H = [I_k, 0_{k\times k}, \ldots, 0_{k\times k}, 0_{k\times r}, \ldots, 0_{k\times r}]$$

Note that the matrix K and the input vector $\mathbf{x}_t^*$ are defined only when $u > v$.

Dynamic Simultaneous Equations Modeling

In the econometrics literature, the VARMAX(p,q,s) model is sometimes written in a form that is slightly different than the one shown in the previous section. This alternative form is referred to as a *dynamic simultaneous equations* model or a *dynamic structural equations* model.

Because $\mathrm{E}(\boldsymbol{\epsilon}_t \boldsymbol{\epsilon}_t') = \Sigma$ is assumed to be positive-definite, there exists a lower triangular matrix A_0 that has ones on the diagonals such that $A_0 \Sigma A_0' = \Sigma^d$, where Σ^d is a diagonal matrix that has positive diagonal elements.

$$A_0 \mathbf{y}_t = \sum_{i=1}^{p} A_i \mathbf{y}_{t-i} + \sum_{i=0}^{s} C_i^* \mathbf{x}_{t-i} + A_0 \boldsymbol{\epsilon}_t - \sum_{i=1}^{q} C_i A_0 \boldsymbol{\epsilon}_{t-i}$$

where $A_i = A_0 \Phi_i$, $C_i^* = A_0 \Theta_i^*$, and $C_i = A_0 \Theta_i A_0^{-1}$.

As an alternative form,

$$A_0\mathbf{y}_t = \sum_{i=1}^{p} A_i\mathbf{y}_{t-i} + \sum_{i=0}^{s} C_i^*\mathbf{x}_{t-i} + \mathbf{a}_t - \sum_{i=1}^{q} C_i\mathbf{a}_{t-i}$$

where $A_i = A_0\Phi_i$, $C_i^* = A_0\Theta_i^*$, $C_i = A_0\Theta_i A_0^{-1}$, and $\mathbf{a}_t = A_0\boldsymbol{\epsilon}_t$. The covariance matrix of $\mathbf{a}_t$ is the diagonal matrix Σ^d. The PRINT=(DYNAMIC) option returns the parameter estimates that result from estimating the model in this form.

A dynamic simultaneous equations model involves a leading (lower triangular) coefficient matrix for $\mathbf{y}_t$ at lag 0 or a leading coefficient matrix for $\boldsymbol{\epsilon}_t$ at lag 0. Such a representation of the VARMAX(p,q,s) model can be more useful in certain circumstances than the standard representation. From the linear combination of the dependent variables obtained by $A_0\mathbf{y}_t$, you can easily see the relationship between the dependent variables in the current time.

The following statements provide the dynamic simultaneous equations of the VAR(1) model.

```
proc iml;
   sig = {1.0  0.5, 0.5 1.25};
   phi = {1.2 -0.5, 0.6 0.3};
   /* simulate the vector time series */
   call varmasim(y,phi) sigma = sig n = 100 seed = 34657;
   cn = {'y1' 'y2'};
   create simul1 from y[colname=cn];
   append from y;
quit;

data simul1;
   set simul1;
   date = intnx( 'year', '01jan1900'd, _n_-1 );
   format date year4.;
run;

proc varmax data=simul1;
   model y1 y2 / p=1 noint print=(dynamic);
run;
```

This is the same data set and model used in the section "Getting Started: VARMAX Procedure" on page 2858. You can compare the results of the VARMA model form and the dynamic simultaneous equations model form.

Figure 42.40 Dynamic Simultaneous Equations (DYNAMIC Option)

The VARMAX Procedure

Covariances of Innovations		
Variable	**y1**	**y2**
y1	1.28875	0.00000
y2	0.00000	1.29578

AR			
Lag	**Variable**	**y1**	**y2**
0	**y1**	1.00000	0.00000
	y2	-0.30845	1.00000
1	**y1**	1.15977	-0.51058
	y2	0.18861	0.54247

Dynamic Model Parameter Estimates						
Equation	**Parameter**	**Estimate**	**Standard Error**	**t Value**	**Pr > |t|**	**Variable**
y1	**AR1_1_1**	1.15977	0.05508	21.06	0.0001	y1(t-1)
	AR1_1_2	-0.51058	0.07140	-7.15	0.0001	y2(t-1)
y2	**AR0_2_1**	0.30845				y1(t)
	AR1_2_1	0.18861	0.05779	3.26	0.0015	y1(t-1)
	AR1_2_2	0.54247	0.07491	7.24	0.0001	y2(t-1)

In Figure 42.4 in the section "Getting Started: VARMAX Procedure" on page 2858, the covariance of $\boldsymbol{\epsilon}_t$ estimated from the VARMAX model form is

$$\Sigma_{\boldsymbol{\epsilon}} = \begin{pmatrix} 1.28875 & 0.39751 \\ 0.39751 & 1.41839 \end{pmatrix}$$

Figure 42.40 shows the results from estimating the model as a dynamic simultaneous equations model. By the decomposition of $\Sigma_{\boldsymbol{\epsilon}}$, you get a diagonal matrix ($\Sigma_{\mathbf{a}}$) and a lower triangular matrix (A_0) such as $\Sigma_{\mathbf{a}} = A_0 \Sigma_{\boldsymbol{\epsilon}} A_0'$ where

$$\Sigma_{\mathbf{a}} = \begin{pmatrix} 1.28875 & 0 \\ 0 & 1.29578 \end{pmatrix} \text{ and } A_0 = \begin{pmatrix} 1 & 0 \\ -0.30845 & 1 \end{pmatrix}$$

The lower triangular matrix (A_0) is shown in the left side of the simultaneous equations model. The parameter estimates in equations system are shown in the right side of the two-equations system.

The simultaneous equations model is written as

$$\begin{pmatrix} 1 & 0 \\ -0.30845 & 1 \end{pmatrix} \mathbf{y}_t = \begin{pmatrix} 1.15977 & -0.51058 \\ 0.18861 & 0.54247 \end{pmatrix} \mathbf{y}_{t-1} + \mathbf{a}_t$$

The resulting two-equation system can be written as

$$\begin{aligned} y_{1t} &= 1.15977y_{1,t-1} - 0.51058y_{2,t-1} + a_{1t} \\ y_{2t} &= 0.30845y_{1t} + 0.18861y_{1,t-1} + 0.54247y_{2,t-1} + a_{2t} \end{aligned}$$

Impulse Response Function

Simple Impulse Response Function (IMPULSE=SIMPLE Option)

The VARMAX(p,q,s) model has a convergent representation

$$\mathbf{y}_t = \Psi^*(B)\mathbf{x}_t + \Psi(B)\boldsymbol{\epsilon}_t$$

where $\Psi^*(B) = \Phi(B)^{-1}\Theta^*(B) = \sum_{j=0}^{\infty} \Psi_j^* B^j$ and $\Psi(B) = \Phi(B)^{-1}\Theta(B) = \sum_{j=0}^{\infty} \Psi_j B^j$.

The elements of the matrices Ψ_j from the operator $\Psi(B)$, called the impulse response, can be interpreted as the response of a variable to a shock in another variable. Let $\psi_{j,in}$ be the (i, n) element of Ψ_j at lag j, where n is the index for the impulse variable, and i is the index for the response variable (impulse $\rightarrow$ response); that is to say, $\psi_{j,in}$ shows the reaction of the i-th variable to a unit shock in variable n, j periods ago, assuming that the effect is not contaminated by other shocks (Lütkepohl 1993). For instance, $\psi_{j,11}$ is an impulse response to $y_{1t} \rightarrow y_{1t}$, and $\psi_{j,12}$ is an impulse response to $y_{2t} \rightarrow y_{1t}$.

Accumulated Impulse Response Function (IMPULSE=ACCUM Option)

The accumulated impulse response function is the cumulative sum of the impulse response function, $\Psi_l^a = \sum_{j=0}^{l} \Psi_j$.

Orthogonalized Impulse Response Function (IMPULSE=ORTH Option)

The MA representation of a VARMA(p,q) model with a standardized white noise innovation process offers another way to interpret a VARMA(p,q) model. Since Σ is positive-definite, there is a lower triangular matrix P such that $\Sigma = PP'$. The alternate MA representation of a VARMA(p,q) model is written as

$$\mathbf{y}_t = \Psi^o(B)\mathbf{u}_t$$

where $\Psi^o(B) = \sum_{j=0}^{\infty} \Psi_j^o B^j$, $\Psi_j^o = \Psi_j P$, and $\mathbf{u}_t = P^{-1}\boldsymbol{\epsilon}_t$.

The elements of the matrices Ψ_j^o, called the *orthogonal impulse response*, can be interpreted as the effects of the components of the standardized shock process $\mathbf{u}_t$ on the process $\mathbf{y}_t$ at lag j.

Impulse Response of Transfer Function (IMPULSX=SIMPLE Option)

The coefficient matrix Ψ_j^* from the transfer function operator $\Psi^*(B)$ can be interpreted as the effects that changes in the exogenous variables $\mathbf{x}_t$ have on the output variable $\mathbf{y}_t$ at lag j; it is called an impulse response matrix in the transfer function.

Accumulated Impulse Response of Transfer Function (IMPULSX=ACCUM Option)

The accumulated impulse response in the transfer function is the cumulative sum of the impulse response in the transfer function, $\Psi_l^{*a} = \sum_{j=0}^{l} \Psi_j^*$.

The asymptotic distributions of the impulse functions can be seen in the section "VAR and VARX Modeling" on page 2963.

The following statements provide the impulse response and the accumulated impulse response in the transfer function for a VARX(1,0) model.

```
proc varmax data=grunfeld plot=impulse;
   model y1-y3 = x1 x2 / p=1 lagmax=5
                         printform=univariate
                         print=(impulsx=(all) estimates);
run;
```

In Figure 42.41, the variables $x1$ and $x2$ are impulses and the variables $y1$, $y2$, and $y3$ are responses. You can read the table matching the pairs of $impulse \rightarrow response$ such as $x1 \rightarrow y1$, $x1 \rightarrow y2$, $x1 \rightarrow y3$, $x2 \rightarrow y1$, $x2 \rightarrow y2$, and $x2 \rightarrow y3$. In the pair of $x1 \rightarrow y1$, you can see the long-run responses of $y1$ to an impulse in $x1$ (the values are 1.69281, 0.35399, 0.09090, and so on for lag 0, lag 1, lag 2, and so on, respectively).

Figure 42.41 Impulse Response in Transfer Function (IMPULSX= Option)

The VARMAX Procedure

Simple Impulse Response of Transfer Function by Variable

Variable Response\Impulse	Lag	x1	x2
y1	0	1.69281	-0.00859
	1	0.35399	0.01727
	2	0.09090	0.00714
	3	0.05136	0.00214
	4	0.04717	0.00072
	5	0.04620	0.00040
y2	0	-6.09850	2.57980
	1	-5.15484	0.45445
	2	-3.04168	0.04391
	3	-2.23797	-0.01376
	4	-1.98183	-0.01647
	5	-1.87415	-0.01453
y3	0	-0.02317	-0.01274
	1	1.57476	-0.01435
	2	1.80231	0.00398
	3	1.77024	0.01062
	4	1.70435	0.01197
	5	1.63913	0.01187

Figure 42.42 shows the responses of $y1$, $y2$, and $y3$ to a forecast error impulse in $x1$.

Figure 42.42 Plot of Impulse Response in Transfer Function

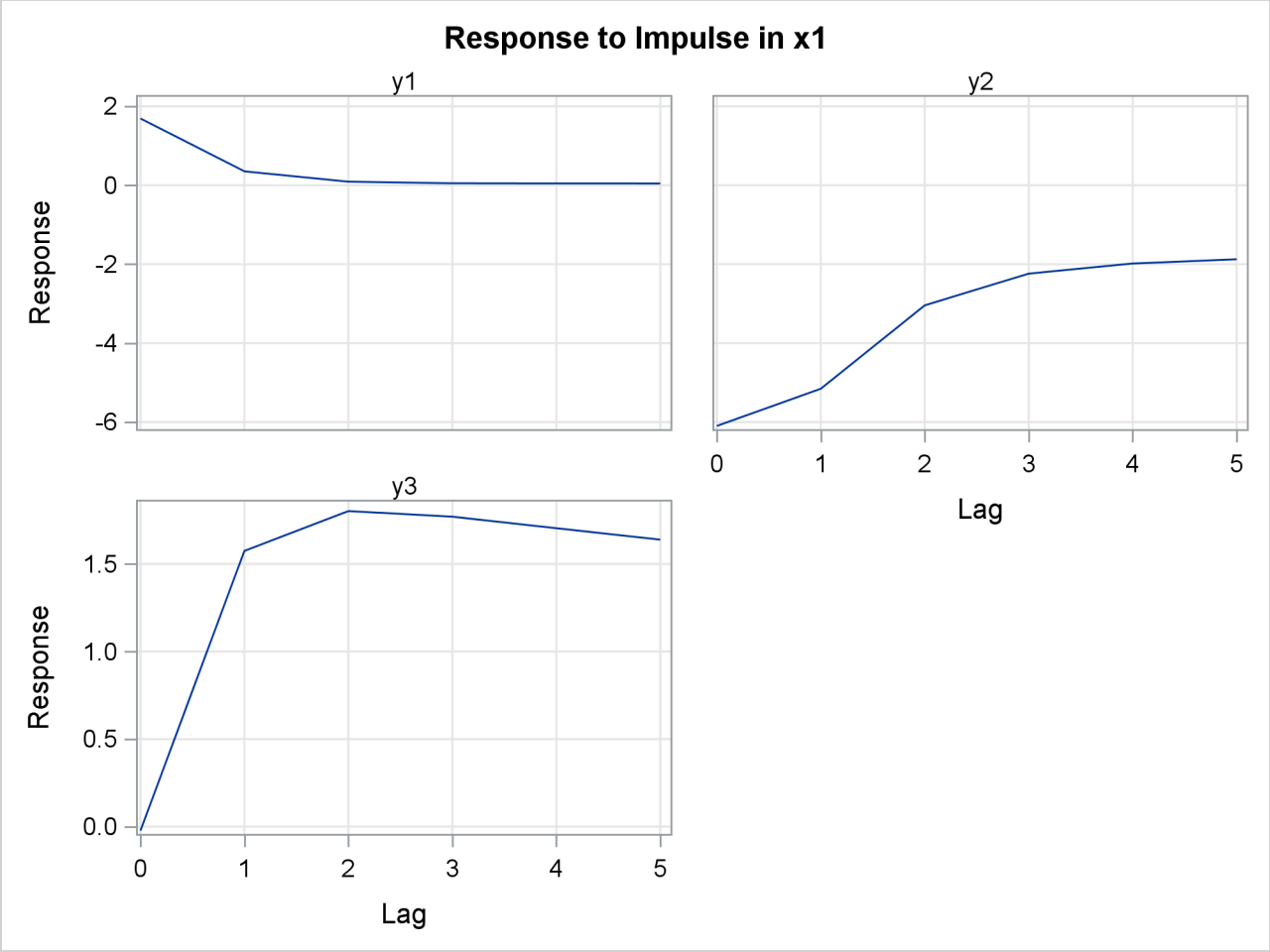


Figure 42.43 shows the accumulated impulse response in transfer function.

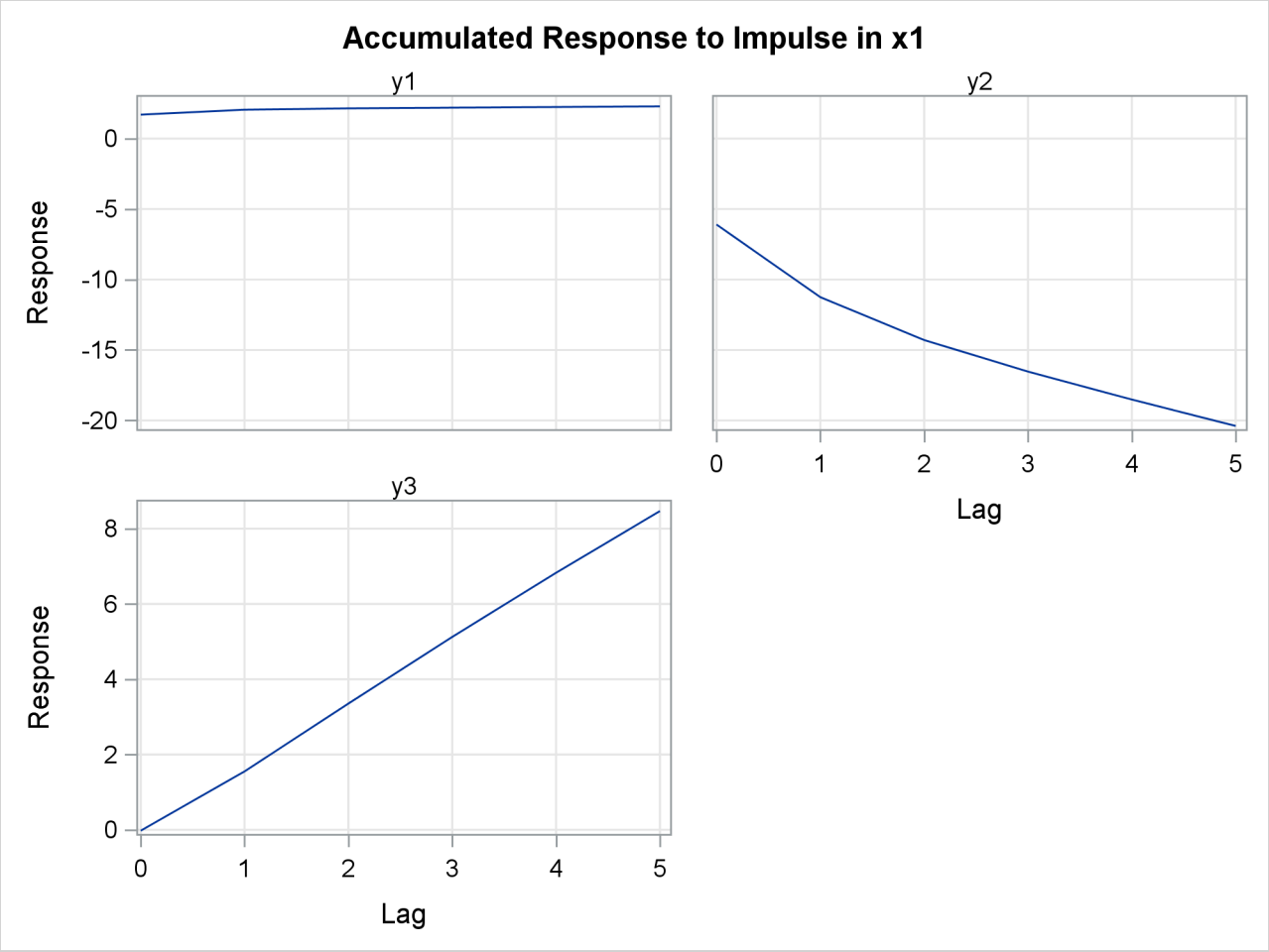
Figure 42.43 Accumulated Impulse Response in Transfer Function (IMPULSX= Option)

Accumulated Impulse Response of Transfer Function by Variable

Variable Response\Impulse	Lag	x1	x2
y1	0	1.69281	-0.00859
	1	2.04680	0.00868
	2	2.13770	0.01582
	3	2.18906	0.01796
	4	2.23623	0.01867
	5	2.28243	0.01907
y2	0	-6.09850	2.57980
	1	-11.25334	3.03425
	2	-14.29502	3.07816
	3	-16.53299	3.06440
	4	-18.51482	3.04793
	5	-20.38897	3.03340
y3	0	-0.02317	-0.01274
	1	1.55159	-0.02709
	2	3.35390	-0.02311
	3	5.12414	-0.01249
	4	6.82848	-0.00052
	5	8.46762	0.01135

Figure 42.44 shows the accumulated responses of $y1$, $y2$, and $y3$ to a forecast error impulse in $x1$.

Figure 42.44 Plot of Accumulated Impulse Response in Transfer Function



The following statements provide the impulse response function, the accumulated impulse response function, and the orthogonalized impulse response function with their standard errors for a VAR(1) model. Parts of the VARMAX procedure output are shown in Figure 42.45, Figure 42.47, and Figure 42.49.

```
proc varmax data=simul1 plot=impulse;
   model y1 y2 / p=1 noint lagmax=5
                 print=(impulse=(all))
                 printform=univariate;
run;
```

Figure 42.45 is the output in a univariate format associated with the PRINT=(IMPULSE=) option for the impulse response function. The keyword STD stands for the standard errors of the elements. The matrix in terms of the lag 0 does not print since it is the identity. In Figure 42.45, the variables $y1$ and $y2$ of the first row are impulses, and the variables $y1$ and $y2$ of the first column are responses. You can read the table matching the $impulse \rightarrow response$ pairs, such as $y1 \rightarrow y1$, $y1 \rightarrow y2$, $y2 \rightarrow y1$, and $y2 \rightarrow y2$. For example, in the pair of $y1 \rightarrow y1$ at lag 3, the response is 0.8055. This represents the impact on y1 of one-unit change in $y1$ after 3 periods. As the lag gets higher, you can see the long-run responses of $y1$ to an impulse in itself.

Figure 42.45 Impulse Response Function (IMPULSE= Option)

The VARMAX Procedure

Simple Impulse Response by Variable

Variable Response\Impulse	Lag	y1	y2
y1	1	1.15977	-0.51058
	STD	0.05508	0.05898
	2	1.06612	-0.78872
	STD	0.10450	0.10702
	3	0.80555	-0.84798
	STD	0.14522	0.14121
	4	0.47097	-0.73776
	STD	0.17191	0.15864
	5	0.14315	-0.52450
	STD	0.18214	0.16115
y2	1	0.54634	0.38499
	STD	0.05779	0.06188
	2	0.84396	-0.13073
	STD	0.08481	0.08556
	3	0.90738	-0.48124
	STD	0.10307	0.09865
	4	0.78943	-0.64856
	STD	0.12318	0.11661
	5	0.56123	-0.65275
	STD	0.14236	0.13482

Figure 42.46 shows the responses of $y1$ and $y2$ to a forecast error impulse in $y1$ with two standard errors.

Figure 42.46 Plot of Impulse Response

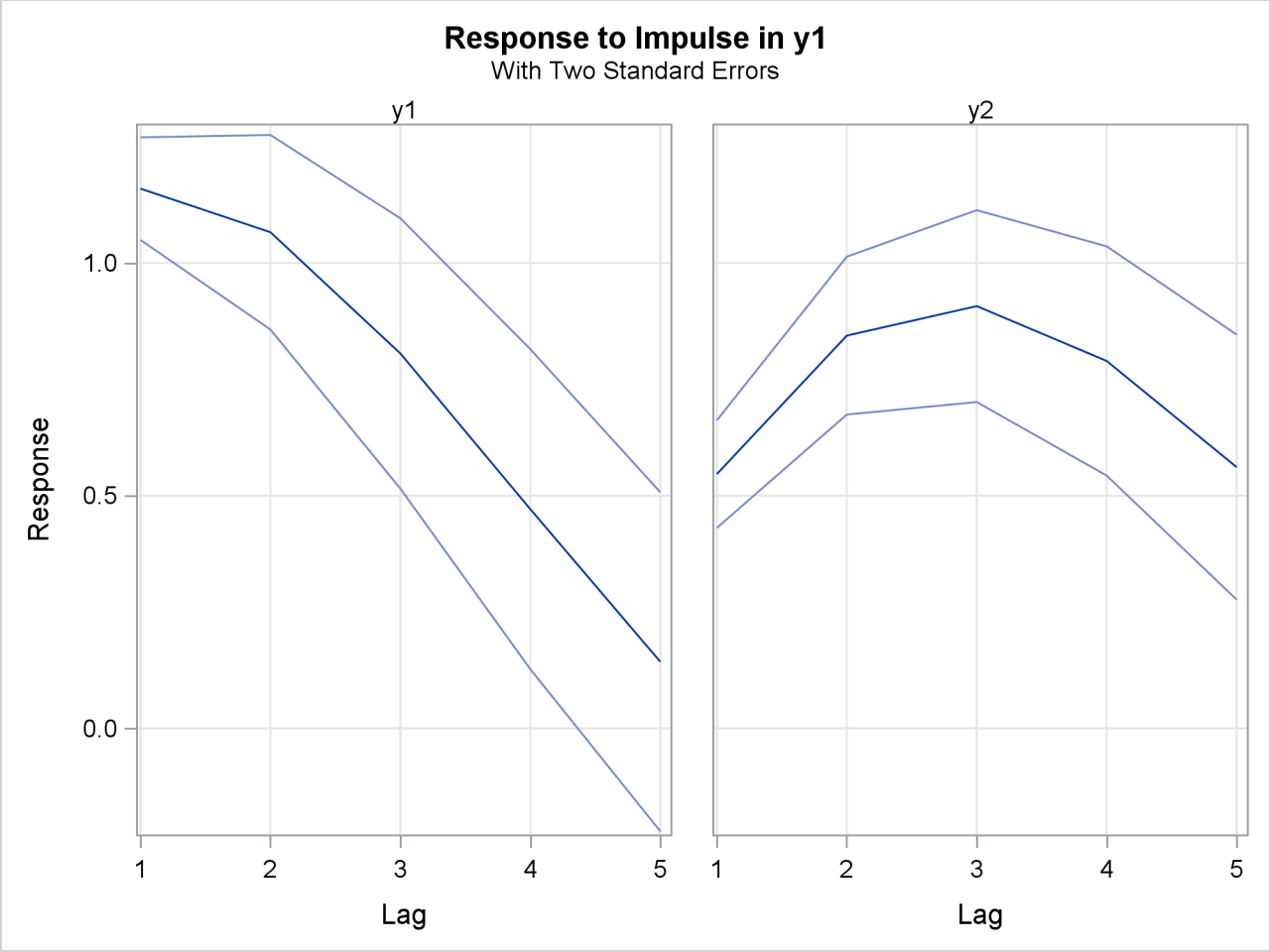


Figure 42.47 is the output in a univariate format associated with the PRINT=(IMPULSE=) option for the accumulated impulse response function. The matrix in terms of the lag 0 does not print since it is the identity.

Figure 42.47 Accumulated Impulse Response Function (IMPULSE= Option)

Accumulated Impulse Response by Variable

Variable Response\Impulse	Lag	y1	y2
y1	1	2.15977	-0.51058
	STD	0.05508	0.05898
	2	3.22589	-1.29929
	STD	0.21684	0.22776
	3	4.03144	-2.14728
	STD	0.52217	0.53649
	4	4.50241	-2.88504
	STD	0.96922	0.97088
	5	4.64556	-3.40953
	STD	1.51137	1.47122
y2	1	0.54634	1.38499
	STD	0.05779	0.06188
	2	1.39030	1.25426
	STD	0.17614	0.18392
	3	2.29768	0.77302
	STD	0.36166	0.36874
	4	3.08711	0.12447
	STD	0.65129	0.65333
	5	3.64834	-0.52829
	STD	1.07510	1.06309

Figure 42.48 shows the accumulated responses of $y1$ and $y2$ to a forecast error impulse in $y1$ with two standard errors.

Figure 42.48 Plot of Accumulated Impulse Response

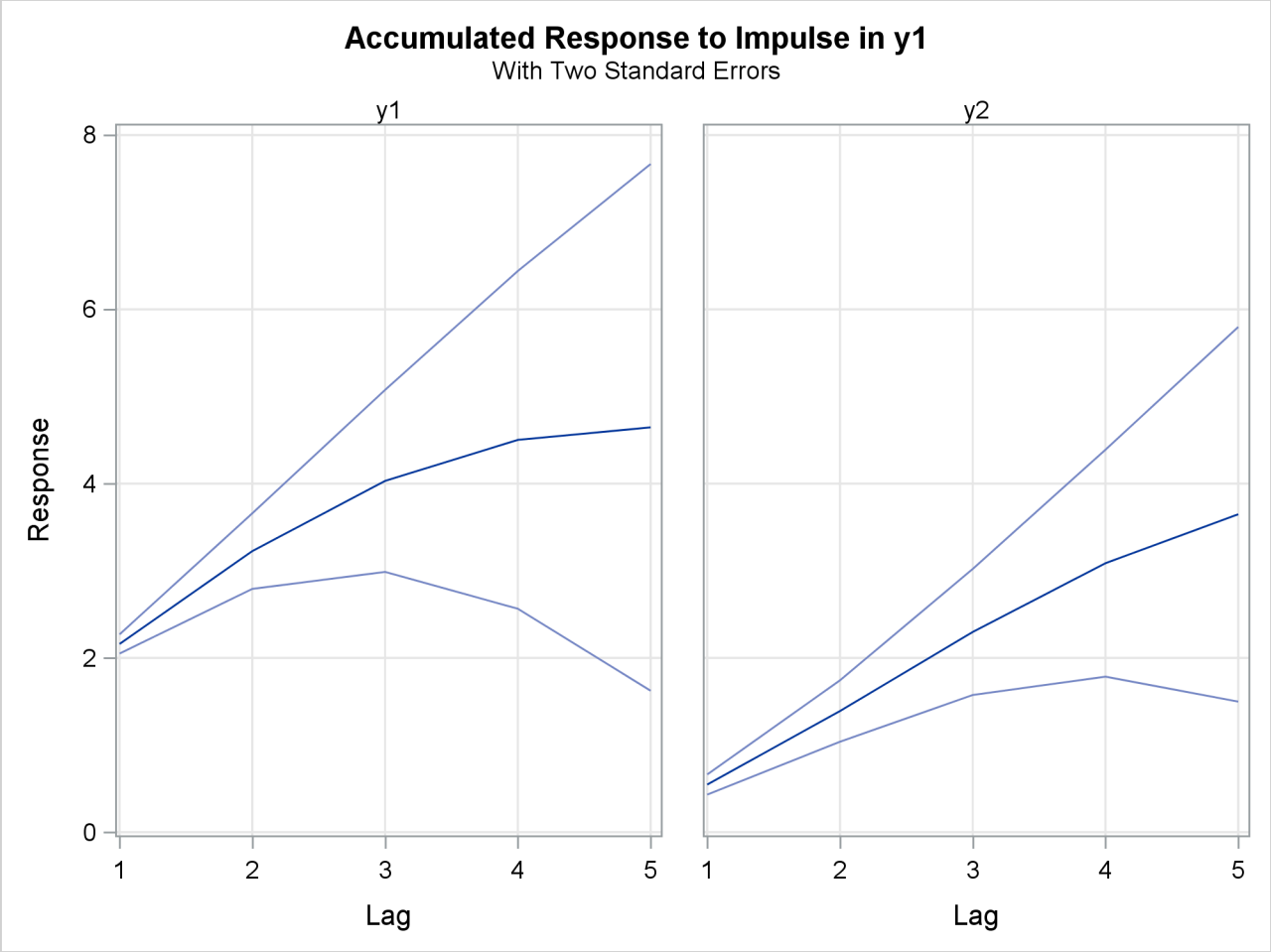


Figure 42.49 is the output in a univariate format associated with the PRINT=(IMPULSE=) option for the orthogonalized impulse response function. The two right-hand side columns, $y1$ and $y2$, represent the $y1\_innovation$ and $y2\_innovation$ variables. These are the impulses variables. The left-hand side column contains responses variables, $y1$ and $y2$. You can read the table by matching the $impulse \rightarrow response$ pairs such as $y1\_innovation \rightarrow y1$, $y1\_innovation \rightarrow y2$, $y2\_innovation \rightarrow y1$, and $y2\_innovation \rightarrow y2$.

Figure 42.49 Orthogonalized Impulse Response Function (IMPULSE= Option)

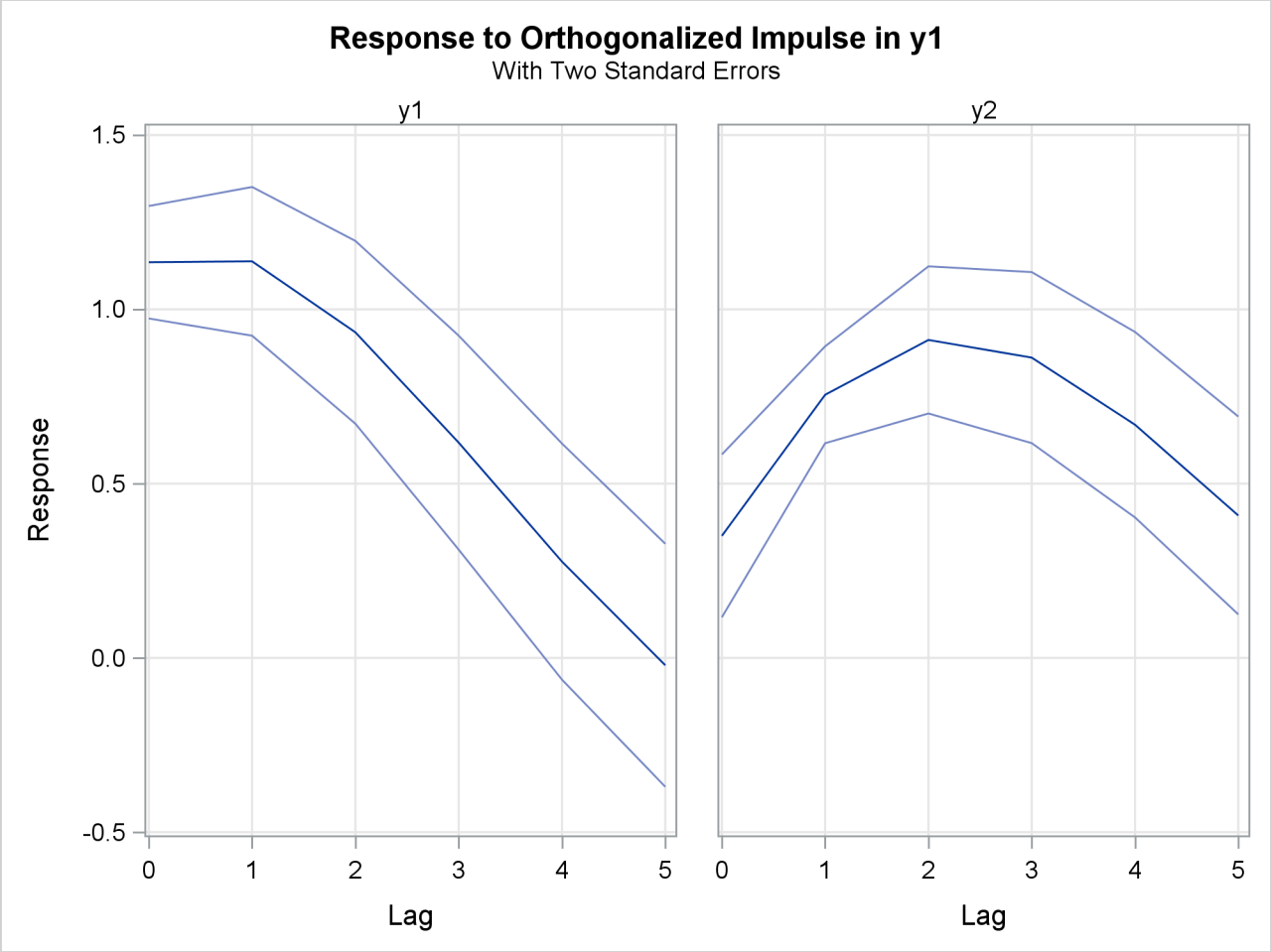
Orthogonalized Impulse Response by Variable

Variable Response\Impulse	Lag	y1	y2
y1	0	1.13523	0.00000
	STD	0.08068	0.00000
	1	1.13783	-0.58120
	STD	0.10666	0.14110
	2	0.93412	-0.89782
	STD	0.13113	0.16776
	3	0.61756	-0.96528
	STD	0.15348	0.18595
	4	0.27633	-0.83981
	STD	0.16940	0.19230
	5	-0.02115	-0.59705
	STD	0.17432	0.18830
y2	0	0.35016	1.13832
	STD	0.11676	0.08855
	1	0.75503	0.43824
	STD	0.06949	0.10937
	2	0.91231	-0.14881
	STD	0.10553	0.13565
	3	0.86158	-0.54780
	STD	0.12266	0.14825
	4	0.66909	-0.73827
	STD	0.13305	0.15846
	5	0.40856	-0.74304
	STD	0.14189	0.16765

In Figure 42.4, there is a positive correlation between ε_{1t} and ε_{2t}. Therefore, shock in $y1$ can be accompanied by a shock in $y2$ in the same period. For example, in the pair of $y1\_innovation \rightarrow y2$, you can see the long-run responses of $y2$ to an impulse in $y1\_innovation$.

Figure 42.50 shows the orthogonalized responses of $y1$ and $y2$ to a forecast error impulse in $y1$ with two standard errors.

Figure 42.50 Plot of Orthogonalized Impulse Response



Forecasting

The optimal (minimum MSE) l-step-ahead forecast of $\mathbf{y}_{t+l}$ is

$$\mathbf{y}_{t+l|t} = \sum_{j=1}^{p} \Phi_j \mathbf{y}_{t+l-j|t} + \sum_{j=0}^{s} \Theta_j^* \mathbf{x}_{t+l-j|t} - \sum_{j=l}^{q} \Theta_j \boldsymbol{\epsilon}_{t+l-j}, \quad l \le q$$

$$\mathbf{y}_{t+l|t} = \sum_{j=1}^{p} \Phi_j \mathbf{y}_{t+l-j|t} + \sum_{j=0}^{s} \Theta_j^* \mathbf{x}_{t+l-j|t}, \quad l > q$$

where $\mathbf{y}_{t+l-j|t} = \mathbf{y}_{t+l-j}$ and $\mathbf{x}_{t+l-j|t} = \mathbf{x}_{t+l-j}$ for $l \le j$. For information about the forecasts $\mathbf{x}_{t+l-j|t}$, see the section "State Space Representation" on page 2937.

Covariance Matrices of Prediction Errors without Exogenous (Independent) Variables

Under the stationarity assumption, the optimal (minimum MSE) l-step-ahead forecast of $\mathbf{y}_{t+l}$ has an infinite moving-average form, $\mathbf{y}_{t+l|t} = \sum_{j=l}^{\infty} \Psi_j \boldsymbol{\epsilon}_{t+l-j}$. The prediction error of the optimal l-step-ahead forecast is $\mathbf{e}_{t+l|t} = \mathbf{y}_{t+l} - \mathbf{y}_{t+l|t} = \sum_{j=0}^{l-1} \Psi_j \boldsymbol{\epsilon}_{t+l-j}$, with zero mean and covariance matrix,

$$\Sigma(l) = \mathrm{Cov}(\mathbf{e}_{t+l|t}) = \sum_{j=0}^{l-1} \Psi_j \Sigma \Psi_j' = \sum_{j=0}^{l-1} \Psi_j^o \Psi_j^{o'}$$

where $\Psi_j^o = \Psi_j P$ with a lower triangular matrix P such that $\Sigma = PP'$. Under the assumption of normality of the $\boldsymbol{\epsilon}_t$, the l-step-ahead prediction error $\mathbf{e}_{t+l|t}$ is also normally distributed as multivariate $N(0, \Sigma(l))$. Hence, it follows that the diagonal elements $\sigma_{ii}^2(l)$ of $\Sigma(l)$ can be used, together with the point forecasts $y_{i,t+l|t}$, to construct l-step-ahead prediction intervals of the future values of the component series, $y_{i,t+l}$.

The following statements use the COVPE option to compute the covariance matrices of the prediction errors for a VAR(1) model. The parts of the VARMAX procedure output are shown in Figure 42.51 and Figure 42.52.

```
proc varmax data=simul1;
   model y1 y2 / p=1 noint lagmax=5
                 printform=both
                 print=(decompose(5) impulse=(all) covpe(5));
run;
```

Figure 42.51 is the output in a matrix format associated with the COVPE option for the prediction error covariance matrices.

Figure 42.51 Covariances of Prediction Errors (COVPE Option)

The VARMAX Procedure

Prediction Error Covariances

Lead	Variable	y1	y2
1	y1	1.28875	0.39751
	y2	0.39751	1.41839
2	y1	2.92119	1.00189
	y2	1.00189	2.18051
3	y1	4.59984	1.98771
	y2	1.98771	3.03498
4	y1	5.91299	3.04856
	y2	3.04856	4.07738
5	y1	6.69463	3.85346
	y2	3.85346	5.07010

Figure 42.52 is the output in a univariate format associated with the COVPE option for the prediction error covariances. This printing format more easily explains the prediction error covariances of each variable.

Figure 42.52 Covariances of Prediction Errors

Prediction Error Covariances by Variable			
Variable	**Lead**	**y1**	**y2**
y1	**1**	1.28875	0.39751
	2	2.92119	1.00189
	3	4.59984	1.98771
	4	5.91299	3.04856
	5	6.69463	3.85346
y2	**1**	0.39751	1.41839
	2	1.00189	2.18051
	3	1.98771	3.03498
	4	3.04856	4.07738
	5	3.85346	5.07010

Covariance Matrices of Prediction Errors in the Presence of Exogenous (Independent) Variables

Exogenous variables can be both stochastic and nonstochastic (deterministic) variables. Considering the forecasts in the VARMAX(p,q,s) model, there are two cases.

When exogenous (independent) variables are stochastic (future values not specified):

As defined in the section "State Space Representation" on page 2937, $\mathbf{y}_{t+l|t}$ has the representation

$$\mathbf{y}_{t+l|t} = \sum_{j=l}^{\infty} V_j \mathbf{a}_{t+l-j} + \sum_{j=l}^{\infty} \Psi_j \boldsymbol{\epsilon}_{t+l-j}$$

and hence

$$\mathbf{e}_{t+l|t} = \sum_{j=0}^{l-1} V_j \mathbf{a}_{t+l-j} + \sum_{j=0}^{l-1} \Psi_j \boldsymbol{\epsilon}_{t+l-j}$$

Therefore, the covariance matrix of the l-step-ahead prediction error is given as

$$\Sigma(l) = \mathrm{Cov}(\mathbf{e}_{t+l|t}) = \sum_{j=0}^{l-1} V_j \Sigma_a V_j' + \sum_{j=0}^{l-1} \Psi_j \Sigma_\epsilon \Psi_j'$$

where Σ_a is the covariance of the white noise series $\mathbf{a}_t$, and $\mathbf{a}_t$ is the white noise series for the VARMA(p,q) model of exogenous (independent) variables, which is assumed not to be correlated with $\boldsymbol{\epsilon}_t$ or its lags.

When future exogenous (independent) variables are specified:

The optimal forecast $\mathbf{y}_{t+l|t}$ of $\mathbf{y}_t$ conditioned on the past information and also on known future values $\mathbf{x}_{t+1}, \ldots, \mathbf{x}_{t+l}$ can be represented as

$$\mathbf{y}_{t+l|t} = \sum_{j=0}^{\infty} \Psi_j^* \mathbf{x}_{t+l-j} + \sum_{j=l}^{\infty} \Psi_j \boldsymbol{\epsilon}_{t+l-j}$$

and the forecast error is

$$\mathbf{e}_{t+l|t} = \sum_{j=0}^{l-1} \Psi_j \boldsymbol{\epsilon}_{t+l-j}$$

Thus, the covariance matrix of the l-step-ahead prediction error is given as

$$\Sigma(l) = \mathrm{Cov}(\mathbf{e}_{t+l|t}) = \sum_{j=0}^{l-1} \Psi_j \Sigma_{\epsilon} \Psi_j'$$

Decomposition of Prediction Error Covariances

In the relation $\Sigma(l) = \sum_{j=0}^{l-1} \Psi_j^o \Psi_j^{o\prime}$, the diagonal elements can be interpreted as providing a decomposition of the l-step-ahead prediction error covariance $\sigma_{ii}^2(l)$ for each component series y_{it} into contributions from the components of the standardized innovations $\boldsymbol{\epsilon}_t$.

If you denote the (i, n) element of Ψ_j^o by $\psi_{j,in}$, the MSE of $y_{i,t+h|t}$ is

$$\mathrm{MSE}(y_{i,t+h|t}) = \mathrm{E}(y_{i,t+h} - y_{i,t+h|t})^2 = \sum_{j=0}^{l-1} \sum_{n=1}^{k} \psi_{j,in}^2$$

Note that $\sum_{j=0}^{l-1} \psi_{j,in}^2$ is interpreted as the contribution of innovations in variable n to the prediction error covariance of the l-step-ahead forecast of variable i.

The proportion, $\omega_{l,in}$, of the l-step-ahead forecast error covariance of variable i accounting for the innovations in variable n is

$$\omega_{l,in} = \sum_{j=0}^{l-1} \psi_{j,in}^2 / \mathrm{MSE}(y_{i,t+h|t})$$

The following statements use the DECOMPOSE option to compute the decomposition of prediction error covariances and their proportions for a VAR(1) model:

```
proc varmax data=simul1;
   model y1 y2 / p=1 noint print=(decompose(15))
                 printform=univariate;
run;
```

The proportions of decomposition of prediction error covariances of two variables are given in Figure 42.53. The output explains that about 91.356% of the one-step-ahead prediction error covariances of the variable y_{2t} is accounted for by its own innovations and about 8.644% is accounted for by y_{1t} innovations.

Figure 42.53 Decomposition of Prediction Error Covariances (DECOMPOSE Option)

Proportions of Prediction Error Covariances by Variable			
Variable	**Lead**	**y1**	**y2**
y1	**1**	1.00000	0.00000
	2	0.88436	0.11564
	3	0.75132	0.24868
	4	0.64897	0.35103
	5	0.58460	0.41540
y2	**1**	0.08644	0.91356
	2	0.31767	0.68233
	3	0.50247	0.49753
	4	0.55607	0.44393
	5	0.53549	0.46451

Forecasting of the Centered Series

If the CENTER option is specified, the sample mean vector is added to the forecast.

Forecasting of the Differenced Series

If dependent (endogenous) variables are differenced, the final forecasts and their prediction error covariances are produced by integrating those of the differenced series. However, if the PRIOR option is specified, the forecasts and their prediction error variances of the differenced series are produced.

Let $\mathbf{z}_t$ be the original series with some appended zero values that correspond to the unobserved past observations. Let $\Delta(B)$ be the $k \times k$ matrix polynomial in the backshift operator that corresponds to the differencing specified by the MODEL statement. The off-diagonal elements of Δ_i are zero, and the diagonal elements can be different. Then $\mathbf{y}_t = \Delta(B)\mathbf{z}_t$.

This gives the relationship

$$\mathbf{z}_t = \Delta^{-1}(B)\mathbf{y}_t = \sum_{j=0}^{\infty} \Lambda_j \mathbf{y}_{t-j}$$

where $\Delta^{-1}(B) = \sum_{j=0}^{\infty} \Lambda_j B^j$ and $\Lambda_0 = I_k$.

The l-step-ahead prediction of $\mathbf{z}_{t+l}$ is

$$\mathbf{z}_{t+l|t} = \sum_{j=0}^{l-1} \Lambda_j \mathbf{y}_{t+l-j|t} + \sum_{j=l}^{\infty} \Lambda_j \mathbf{y}_{t+l-j}$$

The l-step-ahead prediction error of $\mathbf{z}_{t+l}$ is

$$\sum_{j=0}^{l-1} \Lambda_j \left(\mathbf{y}_{t+l-j} - \mathbf{y}_{t+l-j|t}\right) = \sum_{j=0}^{l-1} \left(\sum_{u=0}^{j} \Lambda_u \Psi_{j-u} \right) \boldsymbol{\epsilon}_{t+l-j}$$

Letting $\Sigma_{\mathbf{z}}(0) = 0$, the covariance matrix of the l-step-ahead prediction error of $\mathbf{z}_{t+l}$, $\Sigma_{\mathbf{z}}(l)$, is

$$\begin{aligned}\Sigma_{\mathbf{z}}(l) &= \sum_{j=0}^{l-1}\left(\sum_{u=0}^{j}\Lambda_u\Psi_{j-u}\right)\Sigma_\epsilon\left(\sum_{u=0}^{j}\Lambda_u\Psi_{j-u}\right)' \\ &= \Sigma_{\mathbf{z}}(l-1)+\left(\sum_{j=0}^{l-1}\Lambda_j\Psi_{l-1-j}\right)\Sigma_\epsilon\left(\sum_{j=0}^{l-1}\Lambda_j\Psi_{l-1-j}\right)'\end{aligned}$$

If there are stochastic exogenous (independent) variables, the covariance matrix of the l-step-ahead prediction error of $\mathbf{z}_{t+l}$, $\Sigma_{\mathbf{z}}(l)$, is

$$\begin{aligned}\Sigma_{\mathbf{z}}(l) &= \Sigma_{\mathbf{z}}(l-1)+\left(\sum_{j=0}^{l-1}\Lambda_j\Psi_{l-1-j}\right)\Sigma_\epsilon\left(\sum_{j=0}^{l-1}\Lambda_j\Psi_{l-1-j}\right)' \\ &\quad +\left(\sum_{j=0}^{l-1}\Lambda_j V_{l-1-j}\right)\Sigma_a\left(\sum_{j=0}^{l-1}\Lambda_j V_{l-1-j}\right)'\end{aligned}$$

Tentative Order Selection

Sample Cross-Covariance and Cross-Correlation Matrices

Given a stationary multivariate time series $\mathbf{y}_t$, cross-covariance matrices are

$$\Gamma(l) = \mathrm{E}[(\mathbf{y}_t-\boldsymbol{\mu})(\mathbf{y}_{t+l}-\boldsymbol{\mu})']$$

where $\boldsymbol{\mu} = \mathrm{E}(\mathbf{y}_t)$, and cross-correlation matrices are

$$\rho(l) = D^{-1}\Gamma(l)D^{-1}$$

where D is a diagonal matrix with the standard deviations of the components of $\mathbf{y}_t$ on the diagonal.

The sample cross-covariance matrix at lag l, denoted as $C(l)$, is computed as

$$\hat{\Gamma}(l) = C(l) = \frac{1}{T}\sum_{t=1}^{T-l}\tilde{\mathbf{y}}_t\tilde{\mathbf{y}}'_{t+l}$$

where $\tilde{\mathbf{y}}_t$ is the centered data and T is the number of nonmissing observations. Thus, the (i,j) element of $\hat{\Gamma}(l)$ is $\hat{\gamma}_{ij}(l) = c_{ij}(l)$. The sample cross-correlation matrix at lag l is computed as

$$\hat{\rho}_{ij}(l) = c_{ij}(l)/[c_{ii}(0)c_{jj}(0)]^{1/2},\quad i,j = 1,\ldots,k$$

The following statements use the CORRY option to compute the sample cross-correlation matrices and their summary indicator plots in terms of $+$, $-$, and $\cdot$, where + indicates significant positive cross-correlations, $-$ indicates significant negative cross-correlations, and $\cdot$ indicates insignificant cross-correlations.

```
proc varmax data=simul1;
   model y1 y2 / p=1 noint lagmax=3 print=(corry)
                 printform=univariate;
run;
```

Figure 42.54 shows the sample cross-correlation matrices of y_{1t} and y_{2t}. As shown, the sample autocorrelation functions for each variable decay quickly, but are significant with respect to two standard errors.

Figure 42.54 Cross-Correlations (CORRY Option)

The VARMAX Procedure

Cross Correlations of Dependent Series by Variable

Variable	Lag	y1	y2
y1	0	1.00000	0.67041
	1	0.83143	0.84330
	2	0.56094	0.81972
	3	0.26629	0.66154
y2	0	0.67041	1.00000
	1	0.29707	0.77132
	2	-0.00936	0.48658
	3	-0.22058	0.22014

Schematic Representation of Cross Correlations

Variable/Lag	0	1	2	3
y1	++	++	++	++
y2	++	++	.+	−+

+ is > 2*std error, - is < -2*std error, . is between

Partial Autoregressive Matrices

For each $m = 1, 2, \ldots, p$ you can define a sequence of matrices Φ_{mm}, which is called the partial autoregression matrices of lag m, as the solution for Φ_{mm} to the Yule-Walker equations of order m,

$$\Gamma(l) = \sum_{i=1}^{m} \Gamma(l-i)\Phi_{im}', \quad l = 1, 2, \ldots, m$$

The sequence of the partial autoregression matrices Φ_{mm} of order m has the characteristic property that if the process follows the AR(p), then $\Phi_{pp} = \Phi_p$ and $\Phi_{mm} = 0$ for $m > p$. Hence, the matrices Φ_{mm} have the cutoff property for a VAR(p) model, and so they can be useful in the identification of the order of a pure VAR model.

The following statements use the PARCOEF option to compute the partial autoregression matrices:

```
proc varmax data=simul1;
   model y1 y2 / p=1 noint lagmax=3
                 printform=univariate
                print=(corry parcoef pcorr
                       pcancorr roots);
run;
```

Figure 42.55 shows that the model can be obtained by an AR order $m = 1$ since partial autoregression matrices are insignificant after lag 1 with respect to two standard errors. The matrix for lag 1 is the same as the Yule-Walker autoregressive matrix.

Figure 42.55 Partial Autoregression Matrices (PARCOEF Option)

The VARMAX Procedure

Partial Autoregression			
Lag	Variable	y1	y2
1	y1	1.14844	-0.50954
	y2	0.54985	0.37409
2	y1	-0.00724	0.05138
	y2	0.02409	0.05909
3	y1	-0.02578	0.03885
	y2	-0.03720	0.10149

Schematic Representation of Partial Autoregression			
Variable/Lag	1	2	3
y1	+-	. .	. .
y2	++	. .	. .
+ is > 2*std error, - is < -2*std error, . is between			

Partial Correlation Matrices

Define the forward autoregression

$$\mathbf{y}_t = \sum_{i=1}^{m-1} \Phi_{i,m-1}\mathbf{y}_{t-i} + \mathbf{u}_{m,t}$$

and the backward autoregression

$$\mathbf{y}_{t-m} = \sum_{i=1}^{m-1} \Phi^*_{i,m-1}\mathbf{y}_{t-m+i} + \mathbf{u}^*_{m,t-m}$$

The matrices $P(m)$ defined by Ansley and Newbold (1979) are given by

$$P(m) = \Sigma^{*1/2}_{m-1}\Phi'_{mm}\Sigma^{-1/2}_{m-1}$$

where

$$\Sigma_{m-1} = \mathrm{Cov}(\mathbf{u}_{m,t}) = \Gamma(0) - \sum_{i=1}^{m-1}\Gamma(-i)\Phi'_{i,m-1}$$

and

$$\Sigma^*_{m-1} = \mathrm{Cov}(\mathbf{u}^*_{m,t-m}) = \Gamma(0) - \sum_{i=1}^{m-1} \Gamma(m-i)\Phi^{*'}_{m-i,m-1}$$

$P(m)$ are the partial cross-correlation matrices at lag m between the elements of $\mathbf{y}_t$ and $\mathbf{y}_{t-m}$, given $\mathbf{y}_{t-1}, \ldots, \mathbf{y}_{t-m+1}$. The matrices $P(m)$ have the cutoff property for a VAR(p) model, and so they can be useful in the identification of the order of a pure VAR structure.

The following statements use the PCORR option to compute the partial cross-correlation matrices:

```
proc varmax data=simul1;
   model y1 y2 / p=1 noint lagmax=3
                 print=(pcorr)
                 printform=univariate;
run;
```

The partial cross-correlation matrices in Figure 42.56 are insignificant after lag 1 with respect to two standard errors. This indicates that an AR order of $m = 1$ can be an appropriate choice.

Figure 42.56 Partial Correlations (PCORR Option)

The VARMAX Procedure

Partial Cross Correlations by Variable

Variable	Lag	y1	y2
y1	1	0.80348	0.42672
	2	0.00276	0.03978
	3	-0.01091	0.00032
y2	1	-0.30946	0.71906
	2	0.04676	0.07045
	3	0.01993	0.10676

Schematic Representation of Partial Cross Correlations

Variable/Lag	1	2	3
y1	++	. .	. .
y2	−+	. .	. .

+ is > 2*std error, - is < -2*std error, . is between

Partial Canonical Correlation Matrices

The partial canonical correlations at lag m between the vectors $\mathbf{y}_t$ and $\mathbf{y}_{t-m}$, given $\mathbf{y}_{t-1}, \ldots, \mathbf{y}_{t-m+1}$, are $1 \geq \rho_1(m) \geq \rho_2(m) \cdots \geq \rho_k(m)$. The partial canonical correlations are the canonical correlations between the residual series $\mathbf{u}_{m,t}$ and $\mathbf{u}^*_{m,t-m}$, where $\mathbf{u}_{m,t}$ and $\mathbf{u}^*_{m,t-m}$ are defined in the previous section. Thus, the squared partial canonical correlations $\rho^2_i(m)$ are the eigenvalues of the matrix

$$\{\mathrm{Cov}(\mathbf{u}_{m,t})\}^{-1}\mathrm{E}(\mathbf{u}_{m,t}\mathbf{u}^{*'}_{m,t-m})\{\mathrm{Cov}(\mathbf{u}^*_{m,t-m})\}^{-1}\mathrm{E}(\mathbf{u}^*_{m,t-m}\mathbf{u}'_{m,t}) = \Phi^{*'}_{mm}\Phi'_{mm}$$

It follows that the test statistic to test for $\Phi_m = 0$ in the VAR model of order $m > p$ is approximately

$$(T-m)\ \mathrm{tr}\ \{\Phi^{*'}_{mm}\Phi'_{mm}\} \approx (T-m)\sum_{i=1}^{k}\rho_i^2(m)$$

and has an asymptotic chi-square distribution with k^2 degrees of freedom for $m > p$.

The following statements use the PCANCORR option to compute the partial canonical correlations:

```
proc varmax data=simul1;
   model y1 y2 / p=1 noint lagmax=3 print=(pcancorr);
run;
```

Figure 42.57 shows that the partial canonical correlations $\rho_i(m)$ between $\mathbf{y}_t$ and $\mathbf{y}_{t-m}$ are $\{0.918, 0.773\}$, $\{0.092, 0.018\}$, and $\{0.109, 0.011\}$ for lags m =1 to 3. After lag m =1, the partial canonical correlations are insignificant with respect to the 0.05 significance level, indicating that an AR order of $m = 1$ can be an appropriate choice.

Figure 42.57 Partial Canonical Correlations (PCANCORR Option)

The VARMAX Procedure

Partial Canonical Correlations

Lag	Correlation1	Correlation2	DF	Chi-Square	Pr > ChiSq
1	0.91783	0.77335	4	142.61	<.0001
2	0.09171	0.01816	4	0.86	0.9307
3	0.10861	0.01078	4	1.16	0.8854

The Minimum Information Criterion (MINIC) Method

The minimum information criterion (MINIC) method can tentatively identify the orders of a VARMA(p,q) process (Spliid 1983; Koreisha and Pukkila 1989; Quinn 1980). The first step of this method is to obtain estimates of the innovations series, $\boldsymbol{\epsilon}_t$, from the VAR(p_ϵ), where p_ϵ is chosen sufficiently large. The choice of the autoregressive order, p_ϵ, is determined by use of a selection criterion. From the selected VAR(p_ϵ) model, you obtain estimates of residual series

$$\tilde{\boldsymbol{\epsilon}}_t = \mathbf{y}_t - \sum_{i=1}^{p_\epsilon}\hat{\Phi}_i^{p_\epsilon}\mathbf{y}_{t-i} - \hat{\boldsymbol{\delta}}^{p_\epsilon},\quad t = p_\epsilon+1,\ldots,T$$

In the second step, you select the order (p,q) of the VARMA model for p in $(p_{min} : p_{max})$ and q in $(q_{min} : q_{max})$

$$\mathbf{y}_t = \boldsymbol{\delta} + \sum_{i=1}^{p}\Phi_i\mathbf{y}_{t-i} - \sum_{i=1}^{q}\Theta_i\tilde{\boldsymbol{\epsilon}}_{t-i} + \boldsymbol{\epsilon}_t$$

which minimizes a selection criterion like SBC or HQ.

According to Lütkepohl (1993), the information criteria, namely Akaike's information criterion (AIC), the corrected Akaike's information criterion (AICC), the final prediction error criterion (FPE), the Hannan-Quinn criterion (HQC), and the Schwarz Bayesian criterion (SBC), are defined as

$$
\begin{aligned}
\text{AIC} &= \log(|\tilde{\Sigma}|) + 2r_b k/T \\
\text{AICC} &= \log(|\tilde{\Sigma}|) + 2r_b k/(T - r_b) \\
\text{FPE} &= \left(\frac{T + r_b}{T - r_b}\right)^k |\tilde{\Sigma}| \\
\text{HQC} &= \log(|\tilde{\Sigma}|) + 2r_b k \log(\log(T))/T \\
\text{SBC} &= \log(|\tilde{\Sigma}|) + r_b k \log(T)/T
\end{aligned}
$$

where $\tilde{\Sigma}$ is the maximum likelihood estimate of the innovation covariance matrix Σ, r_b is the number of parameters in each mean equation, k is the number of dependent variables, and T is the number of observations used to estimate the model. Compared to the definitions of AIC, AICC, HQC, and SBC discussed in the section "Multivariate Model Diagnostic Checks" on page 2981, the preceding definitions omit some constant terms and are normalized by T. More specifically, only the parameters in each of the mean equations are counted; the parameters in the innovation covariance matrix Σ are not counted.

The following statements use the MINIC= option to compute a table that contains the information criterion associated with various AR and MA orders:

```
proc varmax data=simul1;
   model y1 y2 / p=1 noint minic=(p=3 q=3);
run;
```

Figure 42.58 shows the output associated with the MINIC= option. The criterion takes the smallest value at AR order 1.

Figure 42.58 MINIC= Option

The VARMAX Procedure

Minimum Information Criterion Based on AICC

Lag	MA 0	MA 1	MA 2	MA 3
AR 0	3.3574947	3.0331352	2.7080996	2.3049869
AR 1	0.5544431	0.6146887	0.6771732	0.7517968
AR 2	0.6369334	0.6729736	0.7610413	0.8481559
AR 3	0.7235629	0.7551756	0.8053765	0.8654079

VAR and VARX Modeling

The pth-order VAR process is written as

$$
\mathrm{y}_t - \boldsymbol{\mu} = \sum_{i=1}^{p} \Phi_i (\mathrm{y}_{t-i} - \boldsymbol{\mu}) + \boldsymbol{\epsilon}_t \quad \text{or} \quad \Phi(B)(\mathrm{y}_t - \boldsymbol{\mu}) = \boldsymbol{\epsilon}_t
$$

with $\Phi(B) = I_k - \sum_{i=1}^{p} \Phi_i B^i$.

Equivalently, it can be written as

$$\mathbf{y}_t = \boldsymbol{\delta} + \sum_{i=1}^{p} \Phi_i \mathbf{y}_{t-i} + \boldsymbol{\epsilon}_t \quad \text{or} \quad \Phi(B)\mathbf{y}_t = \boldsymbol{\delta} + \boldsymbol{\epsilon}_t$$

with $\boldsymbol{\delta} = (I_k - \sum_{i=1}^{p} \Phi_i)\boldsymbol{\mu}$.

Stationarity

For stationarity, the VAR process must be expressible in the convergent causal infinite MA form as

$$\mathbf{y}_t = \boldsymbol{\mu} + \sum_{j=0}^{\infty} \Psi_j \boldsymbol{\epsilon}_{t-j}$$

where $\Psi(B) = \Phi(B)^{-1} = \sum_{j=0}^{\infty} \Psi_j B^j$ with $\sum_{j=0}^{\infty} ||\Psi_j|| < \infty$, where $||A||$ denotes a norm for the matrix A such as $||A||^2 = \mathrm{tr}\{A'A\}$. The matrix Ψ_j can be recursively obtained from the relation $\Phi(B)\Psi(B) = I$; it is

$$\Psi_j = \Phi_1 \Psi_{j-1} + \Phi_2 \Psi_{j-2} + \cdots + \Phi_p \Psi_{j-p}$$

where $\Psi_0 = I_k$ and $\Psi_j = 0$ for $j < 0$.

The stationarity condition is satisfied if all roots of $|\Phi(z)| = 0$ are outside of the unit circle. The stationarity condition is equivalent to the condition in the corresponding VAR(1) representation, $\mathbf{Y}_t = \Phi \mathbf{Y}_{t-1} + \boldsymbol{\varepsilon}_t$, that all eigenvalues of the $kp \times kp$ companion matrix Φ be less than one in absolute value, where $\mathbf{Y}_t = (\mathbf{y}_t', \ldots, \mathbf{y}_{t-p+1}')'$, $\boldsymbol{\varepsilon}_t = (\boldsymbol{\epsilon}_t', 0', \ldots, 0')'$, and

$$\Phi = \begin{bmatrix} \Phi_1 & \Phi_2 & \cdots & \Phi_{p-1} & \Phi_p \\ I_k & 0 & \cdots & 0 & 0 \\ 0 & I_k & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & I_k & 0 \end{bmatrix}$$

If the stationarity condition is not satisfied, a nonstationary model (a differenced model or an error correction model) might be more appropriate.

The following statements estimate a VAR(1) model and use the ROOTS option to compute the characteristic polynomial roots:

```
proc varmax data=simul1;
   model y1 y2 / p=1 noint print=(roots);
run;
```

Figure 42.59 shows the output associated with the ROOTS option, which indicates that the series is stationary since the modulus of the eigenvalue is less than one.

Figure 42.59 Stationarity (ROOTS Option)

The VARMAX Procedure

Roots of AR Characteristic Polynomial

Index	Real	Imaginary	Modulus	Radian	Degree
1	0.77238	0.35899	0.8517	0.4351	24.9284
2	0.77238	-0.35899	0.8517	-0.4351	-24.9284

Parameter Estimation

Consider the stationary VAR(p) model

$$\mathbf{y}_t = \boldsymbol{\delta} + \sum_{i=1}^{p} \Phi_i \mathbf{y}_{t-i} + \boldsymbol{\epsilon}_t$$

where $\mathbf{y}_{-p+1}, \ldots, \mathbf{y}_0$ are assumed to be available (for convenience of notation). This can be represented by the general form of the multivariate linear model,

$$Y = XB + E \quad \text{or} \quad \mathbf{y} = (X \otimes I_k)\boldsymbol{\beta} + \mathbf{e}$$

where

$$\begin{aligned} Y &= (\mathbf{y}_1, \ldots, \mathbf{y}_T)' \\ B &= (\boldsymbol{\delta}, \Phi_1, \ldots, \Phi_p)' \\ X &= (X_0, \ldots, X_{T-1})' \\ X_t &= (1, \mathbf{y}_t', \ldots, \mathbf{y}_{t-p+1}')' \\ E &= (\boldsymbol{\epsilon}_1, \ldots, \boldsymbol{\epsilon}_T)' \\ \mathbf{y} &= \mathrm{vec}(Y') \\ \boldsymbol{\beta} &= \mathrm{vec}(B') \\ \mathbf{e} &= \mathrm{vec}(E') \end{aligned}$$

with *vec* denoting the column stacking operator.

The conditional least squares estimator of $\boldsymbol{\beta}$ is

$$\hat{\boldsymbol{\beta}} = ((X'X)^{-1}X' \otimes I_k)\mathbf{y}$$

and the estimate of Σ is

$$\hat{\Sigma} = (T - (kp + 1))^{-1} \sum_{t=1}^{T} \hat{\boldsymbol{\epsilon}}_t \hat{\boldsymbol{\epsilon}}_t'$$

where $\hat{\boldsymbol{\epsilon}}_t$ is the residual vectors. Consistency and asymptotic normality of the LS estimator are that

$$\sqrt{T}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}) \xrightarrow{d} N(0, \Gamma_p^{-1} \otimes \Sigma)$$

where $X'X/T$ converges in probability to Γ_p and $\xrightarrow{d}$ denotes convergence in distribution.

The (conditional) maximum likelihood estimator in the VAR(p) model is equal to the (conditional) least squares estimator on the assumption of normality of the error vectors.

Asymptotic Distributions of Impulse Response Functions

As before, *vec* denotes the column stacking operator and *vech* is the corresponding operator that stacks the elements on and below the diagonal. For any $k \times k$ matrix A, the commutation matrix K_k is defined as $K_k \mathrm{vec}(A) = \mathrm{vec}(A')$; the duplication matrix D_k is defined as $D_k \mathrm{vech}(A) = \mathrm{vec}(A)$; the elimination matrix L_k is defined as $L_k \mathrm{vec}(A) = \mathrm{vech}(A)$.

The asymptotic distribution of the impulse response function (Lütkepohl 1993) is

$$\sqrt{T}\mathrm{vec}(\hat{\Psi}_j - \Psi_j) \xrightarrow{d} N(0, G_j \Sigma_{\boldsymbol{\beta}} G_j') \quad j = 1, 2, \ldots$$

where $\Sigma_{\boldsymbol{\beta}} = \Gamma_p^{-1} \otimes \Sigma$ and

$$G_j = \frac{\partial \mathrm{vec}(\Psi_j)}{\partial \boldsymbol{\beta}'} = \sum_{i=0}^{j-1} \mathbf{J}(\boldsymbol{\Phi}')^{j-1-i} \otimes \Psi_i$$

where $\mathbf{J} = [I_k, 0, \ldots, 0]$ is a $k \times kp$ matrix and $\boldsymbol{\Phi}$ is a $kp \times kp$ companion matrix.

The asymptotic distribution of the accumulated impulse response function is

$$\sqrt{T}\mathrm{vec}(\hat{\Psi}_l^a - \Psi_l^a) \xrightarrow{d} N(0, F_l \Sigma_{\boldsymbol{\beta}} F_l') \quad l = 1, 2, \ldots$$

where $F_l = \sum_{j=1}^{l} G_j$.

The asymptotic distribution of the orthogonalized impulse response function is

$$\sqrt{T}\mathrm{vec}(\hat{\Psi}_j^o - \Psi_j^o) \xrightarrow{d} N(0, C_j \Sigma_{\boldsymbol{\beta}} C_j' + \bar{C}_j \Sigma_{\boldsymbol{\sigma}} \bar{C}_j') \quad j = 0, 1, 2, \ldots$$

where $C_0 = 0$, $C_j = (\Psi_0^{o\prime} \otimes I_k) G_j$, $\bar{C}_j = (I_k \otimes \Psi_j) H$,

$$H = \frac{\partial \mathrm{vec}(\Psi_0^o)}{\partial \boldsymbol{\sigma}'} = L_k' \{L_k (I_{k^2} + K_k)(\Psi_0^o \otimes I_k) L_k'\}^{-1}$$

and $\Sigma_{\boldsymbol{\sigma}} = 2D_k^+ (\Sigma \otimes \Sigma) D_k^{+\prime}$ with $D_k^+ = (D_k' D_k)^{-1} D_k'$ and $\boldsymbol{\sigma} = \mathrm{vech}(\Sigma)$.

Granger Causality Test

Let $\mathbf{y}_t$ be arranged and partitioned in subgroups $\mathbf{y}_{1t}$ and $\mathbf{y}_{2t}$ with dimensions k_1 and k_2, respectively ($k = k_1 + k_2$); that is, $\mathbf{y}_t = (\mathbf{y}_{1t}', \mathbf{y}_{2t}')'$ with the corresponding white noise process $\boldsymbol{\epsilon}_t = (\boldsymbol{\epsilon}_{1t}', \boldsymbol{\epsilon}_{2t}')'$. Consider the VAR($p$) model with partitioned coefficients $\Phi_{ij}(B)$ for $i, j = 1, 2$ as follows:

$$\begin{bmatrix} \Phi_{11}(B) & \Phi_{12}(B) \\ \Phi_{21}(B) & \Phi_{22}(B) \end{bmatrix} \begin{bmatrix} \mathbf{y}_{1t} \\ \mathbf{y}_{2t} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\delta}_1 \\ \boldsymbol{\delta}_2 \end{bmatrix} + \begin{bmatrix} \boldsymbol{\epsilon}_{1t} \\ \boldsymbol{\epsilon}_{2t} \end{bmatrix}$$

The variables $\mathbf{y}_{1t}$ are said to cause $\mathbf{y}_{2t}$, but $\mathbf{y}_{2t}$ do not cause $\mathbf{y}_{1t}$ if $\Phi_{12}(B) = 0$. The implication of this model structure is that future values of the process $\mathbf{y}_{1t}$ are influenced only by its own past and not by the past of $\mathbf{y}_{2t}$, where future values of $\mathbf{y}_{2t}$ are influenced by the past of both $\mathbf{y}_{1t}$ and $\mathbf{y}_{2t}$. If the future $\mathbf{y}_{1t}$ are not influenced by the past values of $\mathbf{y}_{2t}$, then it can be better to model $\mathbf{y}_{1t}$ separately from $\mathbf{y}_{2t}$.

Consider testing $H_0: C\boldsymbol{\beta} = c$, where C is a $s \times (k^2 p + k)$ matrix of rank s and c is an s-dimensional vector where $s = k_1 k_2 p$. Assuming that

$$\sqrt{T}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}) \xrightarrow{d} N(0, \Gamma_p^{-1} \otimes \Sigma)$$

you get the Wald statistic

$$T(C\hat{\boldsymbol{\beta}} - c)'[C(\hat{\Gamma}_p^{-1} \otimes \hat{\Sigma})C']^{-1}(C\hat{\boldsymbol{\beta}} - c) \xrightarrow{d} \chi^2(s)$$

For the Granger causality test, the matrix C consists of zeros or ones and c is the zero vector. See Lütkepohl (1993) for more details of the Granger causality test.

VARX Modeling

The vector autoregressive model with exogenous variables is called the VARX(p,s) model. The form of the VARX(p,s) model can be written as

$$\mathbf{y}_t = \boldsymbol{\delta} + \sum_{i=1}^{p} \Phi_i \mathbf{y}_{t-i} + \sum_{i=0}^{s} \Theta_i^* \mathbf{x}_{t-i} + \boldsymbol{\epsilon}_t$$

The parameter estimates can be obtained by representing the general form of the multivariate linear model,

$$Y = XB + E \quad \text{or} \quad \mathbf{y} = (X \otimes I_k)\boldsymbol{\beta} + \mathbf{e}$$

where

$$\begin{aligned}
Y &= (\mathbf{y}_1, \ldots, \mathbf{y}_T)' \\
B &= (\boldsymbol{\delta}, \Phi_1, \ldots, \Phi_p, \Theta_0^*, \ldots, \Theta_s^*)' \\
X &= (X_0, \ldots, X_{T-1})' \\
X_t &= (1, \mathbf{y}_t', \ldots, \mathbf{y}_{t-p+1}', \mathbf{x}_{t+1}', \ldots, \mathbf{x}_{t-s+1}')' \\
E &= (\boldsymbol{\epsilon}_1, \ldots, \boldsymbol{\epsilon}_T)' \\
\mathbf{y} &= \text{vec}(Y') \\
\boldsymbol{\beta} &= \text{vec}(B') \\
\mathbf{e} &= \text{vec}(E')
\end{aligned}$$

The conditional least squares estimator of β can be obtained by using the same method in a VAR(p) modeling. If the multivariate linear model has different independent variables that correspond to dependent variables, the SUR (seemingly unrelated regression) method is used to improve the regression estimates.

The following example fits the ordinary regression model:

```
proc varmax data=one;
   model y1-y3 = x1-x5;
run;
```

This is equivalent to the REG procedure in the SAS/STAT software:

```
proc reg data=one;
   model y1 = x1-x5;
   model y2 = x1-x5;
   model y3 = x1-x5;
run;
```

The following example fits the second-order lagged regression model:

```
proc varmax data=two;
   model y1 y2 = x / xlag=2;
run;
```

This is equivalent to the REG procedure in the SAS/STAT software:

```
data three;
   set two;
   xlag1 = lag1(x);
   xlag2 = lag2(x);
run;

proc reg data=three;
   model y1 = x xlag1 xlag2;
   model y2 = x xlag1 xlag2;
run;
```

The following example fits the ordinary regression model with different regressors:

```
proc varmax data=one;
   model y1 = x1-x3, y2 = x2 x3;
run;
```

This is equivalent to the following SYSLIN procedure statements:

```
proc syslin data=one vardef=df sur;
   endogenous y1 y2;
   model y1 = x1-x3;
   model y2 = x2 x3;
run;
```

From the output in Figure 42.21 in the section "Getting Started: VARMAX Procedure" on page 2858, you can see that the parameters, XL0_1_2, XL0_2_1, XL0_3_1, and XL0_3_2 associated with the exogenous variables, are not significant. The following example fits the VARX(1,0) model with different regressors:

```
proc varmax data=grunfeld;
   model y1 = x1, y2 = x2, y3 / p=1 print=(estimates);
run;
```

Figure 42.60 Parameter Estimates for the VARX(1, 0) Model

The VARMAX Procedure

		XLag	
Lag	Variable	x1	x2
0	y1	1.83231	_
	y2	_	2.42110
	y3	_	_

As you can see in Figure 42.60, the symbol '_' in the elements of matrix corresponds to endogenous variables that do not take the denoted exogenous variables.

Seasonal Dummies and Time Trends

You can use the NSEASON= option to introduce seasonal dummies into the model, and the TREND= option to introduce linear trend or both linear and quadratic trends into the model. The definition of the seasonal dummies and trends starts from the first observation after skipping the presample and the observations that have missing values. The size of the presample is $\max(p, s)$, where p is the maximum number of lags of AR terms and s is the maximum number of lags of exogenous variables; that is, the presample contains $\{\mathbf{y}_{-l+1}, \mathbf{x}_{-l+1}, \ldots, \mathbf{y}_0, \mathbf{x}_0\}$, where $l = max(p, s)$.

The following statements fit a bivariate VARX(1, 2) model that has four seasonal periods and both linear and quadratic time trends:

```
data One;
   format date date9.;
   do obs = 1 to 100;
      date=intnx('quarter','01Jan1990'd,obs-1);
      y1 = normal(1); y2 = normal(1); x = normal(1);
      output;
   end;
run;

proc varmax data=One;
   model y1 y2 = x / nseason=4 xlag=2 p=1 trend=quad;
run;
```

In the following statements, the seasonal dummies and time trends are explicitly defined in the data set, together with the lags of dependent and exogenous variables, and then the equivalent model is fit by the REG procedure in SAS/STAT software:

```
data Two;
   set one;
   y1lag1 = lag(y1); y2lag1 = lag(y2);
   xlag1 = lag(x); xlag2 = lag2(x);
   if (obs>2) then do;
      ltrend = obs - 2;
      qtrend = ltrend * ltrend;
      const = 1;
      if (mod(ltrend-2,4)=0) then sd1 = 1;
      else sd1 = 0;
      if (mod(ltrend-3,4)=0) then sd2 = 1;
      else sd2 = 0;
      if (mod(ltrend-4,4)=0) then sd3 = 1;
      else sd3 = 0;
   end;
run;

proc reg data=Two(firstobs=3);
   model y1 = const sd1 sd2 sd3 ltrend qtrend
              x xlag1 xlag2 y1lag1 y2lag1 / noint;
   model y2 = const sd1 sd2 sd3 ltrend qtrend
              x xlag1 xlag2 y1lag1 y2lag1 / noint;
run;
```

The first 11 observations in data set Two are output in Figure 42.61 to show what the seasonal dummies and linear and quadratic time trends look like.

```
proc print data=Two(obs=11);
   var date const sd1 sd2 sd3 ltrend qtrend;
run;
```

Figure 42.61 The First 11 Observations in Data Set Two

Obs	date	const	sd1	sd2	sd3	ltrend	qtrend
1	01JAN1990	.	.	.	.	.	.
2	01APR1990	.	.	.	.	.	.
3	01JUL1990	1	0	0	0	1	1
4	01OCT1990	1	1	0	0	2	4
5	01JAN1991	1	0	1	0	3	9
6	01APR1991	1	0	0	1	4	16
7	01JUL1991	1	0	0	0	5	25
8	01OCT1991	1	1	0	0	6	36
9	01JAN1992	1	0	1	0	7	49
10	01APR1992	1	0	0	1	8	64
11	01JUL1992	1	0	0	0	9	81

Bayesian VAR and VARX Modeling

Consider the VAR(p) model

$$\mathbf{y}_t = \boldsymbol{\delta} + \Phi_1 \mathbf{y}_{t-1} + \cdots + \Phi_p \mathbf{y}_{t-p} + \boldsymbol{\epsilon}_t$$

or

$$\mathbf{y} = (X \otimes I_k)\boldsymbol{\beta} + \mathbf{e}$$

When the parameter vector $\boldsymbol{\beta}$ has a prior multivariate normal distribution with known mean $\boldsymbol{\beta}^*$ and covariance matrix V_β, the prior density is written as

$$f(\boldsymbol{\beta}) = (\frac{1}{2\pi})^{k^2 p/2} |V_\beta|^{-1/2} \exp[-\frac{1}{2}(\boldsymbol{\beta} - \boldsymbol{\beta}^*) V_\beta^{-1} (\boldsymbol{\beta} - \boldsymbol{\beta}^*)]$$

The likelihood function for the Gaussian process becomes

$$\begin{aligned} \ell(\boldsymbol{\beta}|\mathbf{y}) &= (\frac{1}{2\pi})^{kT/2} |I_T \otimes \Sigma|^{-1/2} \times \\ &\quad \exp[-\frac{1}{2}(\mathbf{y} - (X \otimes I_k)\boldsymbol{\beta})'(I_T \otimes \Sigma^{-1})(\mathbf{y} - (X \otimes I_k)\boldsymbol{\beta})] \end{aligned}$$

Therefore, the posterior density is derived as

$$f(\boldsymbol{\beta}|\mathbf{y}) \propto \exp[-\frac{1}{2}(\boldsymbol{\beta} - \bar{\boldsymbol{\beta}})' \bar{\Sigma}_\beta^{-1} (\boldsymbol{\beta} - \bar{\boldsymbol{\beta}})]$$

where the posterior mean is

$$\bar{\boldsymbol{\beta}} = [V_\beta^{-1} + (X'X \otimes \Sigma^{-1})]^{-1}[V_\beta^{-1}\boldsymbol{\beta}^* + (X' \otimes \Sigma^{-1})\mathbf{y}]$$

and the posterior covariance matrix is

$$\bar{\Sigma}_\beta = [V_\beta^{-1} + (X'X \otimes \Sigma^{-1})]^{-1}$$

In practice, the prior mean $\boldsymbol{\beta}^*$ and the prior variance V_β need to be specified. If all the parameters are considered to shrink toward zero, the null prior mean should be specified. According to Litterman (1986), the prior variance can be given by

$$v_{ij}(l) = \begin{cases} (\lambda/l)^2 & \text{if } i = j \\ (\lambda\theta\sigma_{ii}/l\sigma_{jj})^2 & \text{if } i \neq j \end{cases}$$

where $v_{ij}(l)$ is the prior variance of the (i, j) element of Φ_l, λ is the prior standard deviation of the diagonal elements of Φ_l, θ is a constant in the interval $(0, 1)$, and σ_{ii}^2 is the ith diagonal element of Σ. The deterministic terms have diffused prior variance. In practice, you replace the σ_{ii}^2 by the diagonal element of the ML estimator of Σ in the nonconstrained model.

For example, for a bivariate BVAR(2) model,

$$\begin{aligned} y_{1t} &= 0 + \phi_{1,11}y_{1,t-1} + \phi_{1,12}y_{2,t-1} + \phi_{2,11}y_{1,t-2} + \phi_{2,12}y_{2,t-2} + \epsilon_{1t} \\ y_{2t} &= 0 + \phi_{1,21}y_{1,t-1} + \phi_{1,22}y_{2,t-1} + \phi_{2,21}y_{1,t-2} + \phi_{2,22}y_{2,t-2} + \epsilon_{2t} \end{aligned}$$

with the prior covariance matrix

$$\begin{aligned} V_\beta = \text{Diag} \quad (\quad & \infty, \lambda^2, (\lambda\theta\sigma_1/\sigma_2)^2, (\lambda/2)^2, (\lambda\theta\sigma_1/2\sigma_2)^2, \\ & \infty, (\lambda\theta\sigma_2/\sigma_1)^2, \lambda^2, (\lambda\theta\sigma_2/2\sigma_1)^2, (\lambda/2)^2 \quad) \end{aligned}$$

For the Bayesian estimation of integrated systems, the prior mean is set to the first lag of each variable equal to one in its own equation and all other coefficients at zero. For example, for a bivariate BVAR(2) model,

$$\begin{aligned} y_{1t} &= 0 + 1\ y_{1,t-1} + 0\ y_{2,t-1} + 0\ y_{1,t-2} + 0\ y_{2,t-2} + \epsilon_{1t} \\ y_{2t} &= 0 + 0\ y_{1,t-1} + 1\ y_{2,t-1} + 0\ y_{1,t-2} + 0\ y_{2,t-2} + \epsilon_{2t} \end{aligned}$$

Forecasting of BVAR Modeling

The mean squared error (MSE) is used to measure forecast accuracy (Litterman 1986). The MSE of the s-step-ahead forecast is

$$MSE_s = \frac{1}{J-s+1}\sum_{j=1}^{J-s+1}(A_{t_j} - F_{t_j}^s)^2$$

where J is the number specified by NREP= option, t_j is the time index of the observation to be forecasted in repetition j, A_{t_j} is the actual value at time t_j, and $F_{t_j}^s$ is the forecast made s periods earlier. If there are not enough observations, some MSEs might not be calculated.

Bayesian VARX Modeling

The Bayesian vector autoregressive model with exogenous variables is called the BVARX(p,s) model. The form of the BVARX(p,s) model can be written as

$$\mathbf{y}_t = \boldsymbol{\delta} + \sum_{i=1}^{p} \Phi_i \mathbf{y}_{t-i} + \sum_{i=0}^{s} \Theta_i^* \mathbf{x}_{t-i} + \boldsymbol{\epsilon}_t$$

The parameter estimates can be obtained by representing the general form of the multivariate linear model,

$$\mathbf{y} = (X \otimes I_k)\boldsymbol{\beta} + \mathbf{e}$$

The prior means for the AR coefficients are the same as those specified in BVAR(p). The prior means for the exogenous coefficients are set to zero.

Some examples of the Bayesian VARX model are as follows:

```
model y1 y2 = x1 / p=1 xlag=1 prior;

model y1 y2 = x1 / p=(1 3) xlag=1 nocurrentx
                   prior=(lambda=0.9 theta=0.1);
```

VARMA and VARMAX Modeling

A zero-mean VARMA(p, q) process is written as

$$\mathbf{y}_t = \sum_{i=1}^{p} \Phi_i \mathbf{y}_{t-i} + \boldsymbol{\epsilon}_t - \sum_{i=1}^{q} \Theta_i \boldsymbol{\epsilon}_{t-i}$$

or

$$\Phi(B)\mathbf{y}_t = \Theta(B)\boldsymbol{\epsilon}_t$$

where $\Phi(B) = I_k - \sum_{i=1}^{p} \Phi_i B^i$ and $\Theta(B) = I_k - \sum_{i=1}^{q} \Theta_i B^i$.

Stationarity and Invertibility

For stationarity and invertibility of the VARMA process, the roots of $|\Phi(z)| = 0$ and $|\Theta(z)| = 0$ are outside the unit circle.

Parameter Estimation

Under the assumption of normality of the $\boldsymbol{\epsilon}_t$ with zero-mean vector and nonsingular covariance matrix Σ, the conditional (approximate) log-likelihood function of a zero-mean VARMA(p,q) model is considered.

Define $Y = (\mathbf{y}_1, \ldots, \mathbf{y}_T)'$ and $E = (\boldsymbol{\epsilon}_1, \ldots, \boldsymbol{\epsilon}_T)'$ with $B^i Y = (\mathbf{y}_{1-i}, \ldots, \mathbf{y}_{T-i})'$ and $B^i E = (\boldsymbol{\epsilon}_{1-i}, \ldots, \boldsymbol{\epsilon}_{T-i})'$; define $\mathbf{y} = \mathrm{vec}(Y')$ and $\mathbf{e} = \mathrm{vec}(E')$. Then

$$\mathbf{y} - \sum_{i=1}^{p}(I_T \otimes \Phi_i)B^i\mathbf{y} = \mathbf{e} - \sum_{i=1}^{q}(I_T \otimes \Theta_i)B^i\mathbf{e}$$

where $B^i\mathbf{y} = \mathrm{vec}[(B^iY)']$ and $B^i\mathbf{e} = \mathrm{vec}[(B^iE)']$.

Then, the conditional (approximate) log-likelihood function can be written as (Reinsel 1997)

$$\begin{aligned}\ell &= -\frac{T}{2}\log|\Sigma| - \frac{1}{2}\sum_{t=1}^{T}\boldsymbol{\epsilon}_t'\Sigma^{-1}\boldsymbol{\epsilon}_t \\ &= -\frac{T}{2}\log|\Sigma| - \frac{1}{2}\mathbf{w}'\Theta'^{-1}(I_T \otimes \Sigma^{-1})\Theta^{-1}\mathbf{w}\end{aligned}$$

where $\mathbf{w} = \mathbf{y} - \sum_{i=1}^{p}(I_T \otimes \Phi_i)B^i\mathbf{y}$ and Θ is such that $\mathbf{e} - \sum_{i=1}^{q}(I_T \otimes \Theta_i)B^i\mathbf{e} = \Theta\mathbf{e}$. You can specify METHOD=CML in the MODEL statement to apply conditional maximum likelihood estimation.

For the exact log-likelihood function of a VARMA model, the VARMA model is transformed into the equivalent state space form and then the Kalman filtering method is applied.

The state space form of the zero-mean VARMA(p,q) model consists of a state equation

$$\mathbf{z}_t = F\mathbf{z}_{t-1} + G\boldsymbol{\epsilon}_t$$

and an observation equation

$$\mathbf{y}_t = H\mathbf{z}_t$$

where

$$\mathbf{z}_t = (\mathbf{y}_t', \mathbf{y}_{t-1}', \ldots, \mathbf{y}_{t-(v-1)}', \boldsymbol{\epsilon}_t', \boldsymbol{\epsilon}_{t-1}, \ldots, \boldsymbol{\epsilon}_{t-(q-1)}')'$$

$$F = \begin{bmatrix} \Phi_1 & \cdots & \Phi_{v-1} & \Phi_v & -\Theta_1 & \cdots & -\Theta_{q-1} & -\Theta_q \\ I_k & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \ddots & 0 & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & I_k & 0 & 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 & I_k & \cdots & 0 & 0 \\ \vdots & \ddots & 0 & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & I_k & 0 \end{bmatrix}, \quad G = \begin{bmatrix} I_k \\ 0_{k(v-1)\times k} \\ I_k \\ 0_{k(q-1)\times k} \end{bmatrix}$$

and

$$H = [I_k, 0_{k(v+q-1)\times k}]$$

where $v = \max(p, 1)$ and $\Phi_i = 0$ for $i > p$.

The Kalman filtering approach is used to evaluate the likelihood function. The updating equation is

$$\hat{\mathbf{z}}_{t|t} = \hat{\mathbf{z}}_{t|t-1} + K_t \boldsymbol{\epsilon}_{t|t-1}$$

where

$$K_t = P_{t|t-1} H' [H P_{t|t-1} H']^{-1}$$

The prediction equation is

$$\hat{\mathbf{z}}_{t|t-1} = F \hat{\mathbf{z}}_{t-1|t-1}, \quad P_{t|t-1} = F P_{t-1|t-1} F' + G \Sigma G'$$

where $P_{t|t} = [I - K_t H] P_{t|t-1}$ for $t = 1, 2, \ldots, n$.

The log-likelihood function can be expressed as

$$\ell = -\frac{1}{2} \sum_{t=1}^{T} [\log |\Sigma_{t|t-1}| + (\mathbf{y}_t - \hat{\mathbf{y}}_{t|t-1})' \Sigma_{t|t-1}^{-1} (\mathbf{y}_t - \hat{\mathbf{y}}_{t|t-1})]$$

where $\hat{\mathbf{y}}_{t|t-1}$ and $\Sigma_{t|t-1}$ are determined recursively from the Kalman filtering method. To construct the likelihood function from Kalman filtering, you obtain $\hat{\mathbf{y}}_{t|t-1} = H \hat{\mathbf{z}}_{t|t-1}$, $\hat{\boldsymbol{\epsilon}}_{t|t-1} = \mathbf{y}_t - \hat{\mathbf{y}}_{t|t-1}$, and $\Sigma_{t|t-1} = H P_{t|t-1} H'$.

When you specify METHOD=ML in the MODEL statement, the exact log likelihood is evaluated and used in the maximum likelihood estimation.

Define the vector $\boldsymbol{\beta}$ as

$$\boldsymbol{\beta} = (\phi_1', \ldots, \phi_p', \theta_1', \ldots, \theta_q', \text{vech}(\Sigma))'$$

where $\phi_i = \text{vec}(\Phi_i)$ and $\theta_i = \text{vec}(\Theta_i)$. All elements of $\boldsymbol{\beta}$ are estimated through the preceding (conditional) maximum likelihood method. The estimates of $\Phi_i, i = 1, ..., p$, and $\Theta_i, i = 1, ..., q$, are output in the ParameterEstimates ODS table. The estimates of the covariance matrix (Σ) are output in the CovarianceParameterEstimates ODS table. If you specify the OUTEST=, OUTCOV, PRINT=(COVB), or PRINT=(CORRB) option, you can see all elements of $\boldsymbol{\beta}$, including the covariance matrix Σ, in the parameter estimates, covariance of parameter estimates, or correlation of parameter estimates. You can also apply the BOUND, INITIAL, RESTRICT, and TEST statements to any elements of $\boldsymbol{\beta}$, including the covariance matrix Σ. For more information, see the syntax of the corresponding statement.

The (conditional) log-likelihood equations are solved by iterative numerical methods such as quasi-Newton optimization. The starting values for the AR and MA parameters are obtained from the least squares estimates. Although the small-sample properties of CML estimates might not be as good as the ML estimates, the CML method is much faster than the ML method. Depending on the sample size and number of parameters to be estimated, the CML method can be hundreds or even thousands of times faster than the ML method. In the following example code, the CML method is about 100 times faster than the ML method, with very similar estimation and forecast results:

```
proc iml;
   phi = (0.9 * I(4)) // (-0.7* I(4));
   theta = 0.8 * I(4);
   sig = I(4);
   /* to simulate the vector time series */
   call varmasim(y,phi,theta) sigma=sig n=400 seed=2;
```

```
   cn = {'y1' 'y2' 'y3' 'y4'};
   create simul6 from y[colname=cn];
   append from y;
   close;
quit;

proc varmax data=simul6;
   model y1 y2 y3 y4 / noint p=2 q=1 method=cml;
   nloptions pall maxit=5000 tech=qn;
   output out=ocml back=12 lead=24;
run;

proc varmax data=simul6;
   model y1 y2 y3 y4 / noint p=2 q=1 method=ml;
   nloptions pall maxit=5000 tech=qn;
   output out=oml back=12 lead=24;
run;
```

Asymptotic Distribution of the Parameter Estimates

Under the assumptions of stationarity and invertibility for the VARMA model and the assumption that $\boldsymbol{\epsilon}_t$ is a white noise process, $\hat{\boldsymbol{\beta}}$ is a consistent estimator for $\boldsymbol{\beta}$ and $\sqrt{T}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})$ converges in distribution to the multivariate normal $N(0, V^{-1})$ as $T \to \infty$, where V is the asymptotic information matrix of $\boldsymbol{\beta}$.

Asymptotic Distributions of Impulse Response Functions

Defining the vector $\boldsymbol{\beta}$

$$\boldsymbol{\beta} = (\phi_1', \ldots, \phi_p', \theta_1', \ldots, \theta_q')'$$

the asymptotic distribution of the impulse response function for a VARMA(p, q) model is

$$\sqrt{T}\mathrm{vec}(\hat{\Psi}_j - \Psi_j) \xrightarrow{d} N(0, G_j \Sigma_{\boldsymbol{\beta}} G_j') \quad j = 1, 2, \ldots$$

where $\Sigma_{\boldsymbol{\beta}}$ is the covariance matrix of the parameter estimates and

$$G_j = \frac{\partial \mathrm{vec}(\Psi_j)}{\partial \boldsymbol{\beta}'} = \sum_{i=0}^{j-1} \mathbf{H}'(\mathbf{A}')^{j-1-i} \otimes \mathbf{J}\mathbf{A}^i\mathbf{J}'$$

where $\mathbf{H} = [I_k, 0, \ldots, 0, I_k, 0, \ldots, 0]'$ is a $k(p+q) \times k$ matrix with the second I_k following after p block matrices; $\mathbf{J} = [I_k, 0, \ldots, 0]$ is a $k \times k(p+q)$ matrix; $\mathbf{A}$ is a $k(p+q) \times k(p+q)$ matrix,

$$\mathbf{A} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

where

$$A_{11} = \begin{bmatrix} \Phi_1 & \Phi_2 & \cdots & \Phi_{p-1} & \Phi_p \\ I_k & 0 & \cdots & 0 & 0 \\ 0 & I_k & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & I_k & 0 \end{bmatrix} \quad A_{12} = \begin{bmatrix} -\Theta_1 & \cdots & -\Theta_{q-1} & -\Theta_q \\ 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 \end{bmatrix}$$

A_{21} is a $kq \times kp$ zero matrix, and

$$A_{22} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ I_k & 0 & \cdots & 0 & 0 \\ 0 & I_k & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & I_k & 0 \end{bmatrix}$$

An Example of a VARMA(1,1) Model

Consider a VARMA(1,1) model with mean zero,

$$\mathrm{y}_t = \Phi_1 \mathrm{y}_{t-1} + \boldsymbol{\epsilon}_t - \Theta_1 \boldsymbol{\epsilon}_{t-1}$$

where $\boldsymbol{\epsilon}_t$ is the white noise process with a mean zero vector and the positive-definite covariance matrix Σ.

The following IML procedure statements simulate a bivariate vector time series from this model to provide test data for the VARMAX procedure:

```
proc iml;
   sig = {1.0  0.5, 0.5 1.25};
   phi = {1.2 -0.5, 0.6 0.3};
   theta = {0.5 -0.2, 0.1 0.3};
   /* to simulate the vector time series */
   call varmasim(y,phi,theta) sigma=sig n=100 seed=34657;
   cn = {'y1' 'y2'};
   create simul3 from y[colname=cn];
   append from y;
run;
```

The following statements fit a VARMA(1,1) model to the simulated data. You specify the order of the autoregressive model by using the P= option and specify the order of moving-average model by using the Q= option. You specify the quasi-Newton optimization in the NLOPTIONS statement as an optimization method.

```
proc varmax data=simul3;
   nloptions tech=qn;
   model y1 y2 / p=1 q=1 noint print=(estimates);
run;
```

Figure 42.62 shows the initial values of parameters. The initial values were estimated by using the least squares method.

Figure 42.62 Start Parameter Estimates for the VARMA(1, 1) Model

The VARMAX Procedure

Optimization Start Parameter Estimates			
N	Parameter	Estimate	Gradient Objective Function
1	AR1_1_1	0.964299	-2.357098
2	AR1_2_1	0.481620	-3.773499
3	AR1_1_2	-0.363819	1.865051
4	AR1_2_2	0.457378	-10.778568
5	MA1_1_1	0.244355	-2.552198
6	MA1_2_1	-0.034093	2.716227
7	MA1_1_2	-0.006261	-0.147004
8	MA1_2_2	0.444636	0.141839
9	COV1_1	1.353584	2.765550
10	COV1_2	0.415649	-1.389416
11	COV2_2	1.445260	2.581735

Figure 42.63 shows the default option settings for the quasi-Newton optimization technique.

Figure 42.63 Default Criteria for the quasi-Newton Optimization

Minimum Iterations	0
Maximum Iterations	200
Maximum Function Calls	2000
ABSGCONV Gradient Criterion	0.00001
GCONV Gradient Criterion	1E-8
ABSFCONV Function Criterion	0
FCONV Function Criterion	2.220446E-16
FCONV2 Function Criterion	0
FSIZE Parameter	0
ABSXCONV Parameter Change Criterion	0
XCONV Parameter Change Criterion	0
XSIZE Parameter	0
ABSCONV Function Criterion	-1.34078E154
Line Search Method	2
Starting Alpha for Line Search	1
Line Search Precision LSPRECISION	0.4
DAMPSTEP Parameter for Line Search	.
Singularity Tolerance (SINGULAR)	1E-8

Figure 42.64 shows the iteration history of parameter estimates.

Figure 42.64 Iteration History of Parameter Estimates

Iteration	Restarts	Function Calls	Active Constraints	Objective Function	Objective Function Change	Max Abs Gradient Element	Step Size	Slope of Search Direction
1	0	3	0	121.22330	0.1526	5.2001	0.00384	-78.688
2	0	5	0	120.97740	0.2459	6.2584	3.214	-0.156
3	0	6	0	120.58286	0.3945	4.1004	0.948	-0.648
4	0	7	0	120.43152	0.1513	3.7834	1.000	-0.346
5	0	8	0	120.32992	0.1016	6.3797	1.000	-0.243
6	0	10	0	120.26832	0.0616	3.1048	0.407	-0.304
7	0	12	0	120.23311	0.0352	1.0747	0.983	-0.0731
8	0	14	0	120.22264	0.0105	0.6370	1.518	-0.0127
9	0	15	0	120.21560	0.00704	1.3563	4.650	-0.0056
10	0	16	0	120.21281	0.00279	1.2963	2.102	-0.0084
11	0	17	0	120.20951	0.00330	0.1634	1.139	-0.0061
12	0	19	0	120.20896	0.000542	0.1349	2.591	-0.0004
13	0	21	0	120.20884	0.000123	0.0662	1.883	-0.0001
14	0	22	0	120.20875	0.000093	0.1399	4.120	-0.0001
15	0	24	0	120.20871	0.000037	0.00917	1.073	-0.0001
16	0	26	0	120.20871	1.643E-6	0.00858	2.115	-155E-8
17	0	27	0	120.20871	7.704E-7	0.00543	5.409	-759E-9

Figure 42.65 shows the final parameter estimates.

Figure 42.65 Results of Parameter Estimates for the VARMA(1, 1) Model

The VARMAX Procedure

Optimization Results

Parameter Estimates

N	Parameter	Estimate	Gradient Objective Function
1	**AR1_1_1**	1.020117	0.003641
2	**AR1_2_1**	0.393557	0.000140
3	**AR1_1_2**	-0.388708	0.001311
4	**AR1_2_2**	0.551644	0.002479
5	**MA1_1_1**	0.330598	0.000131
6	**MA1_2_1**	-0.166999	0.000086321
7	**MA1_1_2**	-0.032507	-0.001133
8	**MA1_2_2**	0.587232	-0.000523
9	**COV1_1**	1.253624	0.005429
10	**COV1_2**	0.382094	-0.001152
11	**COV2_2**	1.322424	-0.000535

Figure 42.66 shows the AR coefficient matrix in terms of lag 1, the MA coefficient matrix in terms of lag 1, the parameter estimates, and their significance, which is one indication of how well the model fits the data.

Figure 42.66 Parameter Estimates for the VARMA(1, 1) Model

The VARMAX Procedure

Type of Model	VARMA(1,1)
Estimation Method	Maximum Likelihood Estimation

		AR	
Lag	**Variable**	**y1**	**y2**
1	**y1**	1.02012	-0.38871
	y2	0.39356	0.55164

		MA	
Lag	**Variable**	**e1**	**e2**
1	**y1**	0.33060	-0.03251
	y2	-0.16700	0.58723

Schematic Representation		
Variable/Lag	**AR1**	**MA1**
y1	+-	+.
y2	++	.+

+ is > 2*std error, - is < -2*std error, . is between, * is N/A

Model Parameter Estimates

Equation	Parameter	Estimate	Standard Error	t Value	Pr > |t|	Variable
y1	**AR1_1_1**	1.02012	0.10076	10.12	0.0001	y1(t-1)
	AR1_1_2	-0.38871	0.09557	-4.07	0.0001	y2(t-1)
	MA1_1_1	0.33060	0.14389	2.30	0.0237	e1(t-1)
	MA1_1_2	-0.03251	0.14146	-0.23	0.8187	e2(t-1)
y2	**AR1_2_1**	0.39356	0.10210	3.85	0.0002	y1(t-1)
	AR1_2_2	0.55164	0.08536	6.46	0.0001	y2(t-1)
	MA1_2_1	-0.16700	0.15801	-1.06	0.2931	e1(t-1)
	MA1_2_2	0.58723	0.14372	4.09	0.0001	e2(t-1)

Covariance Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Pr > |t|
COV1_1	1.25362	0.17788	7.05	0.0001
COV1_2	0.38209	0.13484	2.83	0.0056
COV2_2	1.32242	0.18829	7.02	0.0001

The fitted VARMA(1,1) model with estimated standard errors in parentheses is given as

$$\mathbf{y}_t = \begin{pmatrix} 1.01846 & -0.38682 \\ (0.10256) & (0.09644) \\ 0.39182 & 0.55281 \\ (0.10062) & (0.08422) \end{pmatrix} \mathbf{y}_{t-1} + \boldsymbol{\epsilon}_t - \begin{pmatrix} 0.32292 & -0.02160 \\ (0.14524) & (0.14203) \\ -0.16501 & 0.58576 \\ (0.15704) & (0.14115) \end{pmatrix} \boldsymbol{\epsilon}_{t-1}$$

and

$$\boldsymbol{\epsilon}_t \sim \text{iid } N(0, \begin{pmatrix} 1.25202 & 0.37950 \\ (0.17697) & (0.13401) \\ 0.37950 & 1.31315 \\ (0.13401) & (0.18610) \end{pmatrix}$$

VARMAX Modeling

A general VARMAX(p, q, s) process is written as

$$\mathbf{y}_t = \boldsymbol{\delta}_t + \sum_{i=1}^{p} \Phi_i \mathbf{y}_{t-i} + \boldsymbol{\epsilon}_t - \sum_{i=1}^{q} \Theta_i \boldsymbol{\epsilon}_{t-i}$$

or

$$\Phi(B)\mathbf{y}_t = \boldsymbol{\delta}_t + \Theta(B)\boldsymbol{\epsilon}_t$$

where $\Phi(B) = I_k - \sum_{i=1}^{p} \Phi_i B^i$ and $\Theta(B) = I_k - \sum_{i=1}^{q} \Theta_i B^i$. The vector $\boldsymbol{\delta}_t$ consists of all possible deterministic terms, namely constant, seasonal dummies, linear trend, quadratic trend, and exogenous variables. The vector $\boldsymbol{\delta}_t = \Delta \mathbf{c}_t$, where $\mathbf{c}_t = (D_t' \ \mathbf{x}_t' \ \ldots \ \mathbf{x}_{t-s}')'$; $D_t = (1 \ d_{t,1} \ \ldots \ d_{t,n_s-1} \ t \ t^2)'$; $d_{t,i}, i = 1, \ldots, n_s - 1$, are seasonal dummies and n_s is based on the NSEASON= option; $\Delta = (A \ \Theta_0^* \ \ldots \Theta_s^*)$; A is the parameter matrix corresponding to D_t and Θ_i^* for $\mathbf{x}_{t-i}, i = 0, \ldots, s$.

The state space form of the VARMAX(p,q,s) model consists of a state equation

$$\mathbf{z}_t = F\mathbf{z}_{t-1} + \mathbf{w}_t + G\boldsymbol{\epsilon}_t$$

and an observation equation

$$\mathbf{y}_t = H\mathbf{z}_t$$

where

$$\mathbf{z}_t = (\mathbf{y}_t', \mathbf{y}_{t-1}', \ldots, \mathbf{y}_{t-(v-1)}', \boldsymbol{\epsilon}_t', \boldsymbol{\epsilon}_{t-1}, \ldots, \boldsymbol{\epsilon}_{t-(q-1)}', \mathbf{c}_{t+1}')'$$

$$F = \begin{bmatrix} \Phi_1 & \cdots & \Phi_{v-1} & \Phi_v & -\Theta_1 & \cdots & -\Theta_{q-1} & -\Theta_q & \Delta \\ I_k & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & 0 & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & I_k & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & \cdots & 0 & 0 & I_k & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & 0 & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & I_k & 0 & 0 \end{bmatrix}, \quad G = \begin{bmatrix} I_k \\ 0_{k(v-1)\times k} \\ I_k \\ 0_{k(q-1)\times k} \\ 0_{u\times k} \end{bmatrix}$$

and

$$H = [I_k, 0_{(k(v+q-1)+u)\times k}]$$

where $v = \max(p, 1)$, $\Phi_i = 0$ for $i > p$, and u is the dimension of c_t.

Kalman filtering is used to evaluate the likelihood function. The updating equation is

$$\hat{\mathbf{z}}_{t|t} = \hat{\mathbf{z}}_{t|t-1} + K_t \boldsymbol{\epsilon}_{t|t-1}$$

where

$$K_t = P_{t|t-1} H' [H P_{t|t-1} H']^{-1}$$

The prediction equation is

$$\hat{\mathbf{z}}_{t|t-1} = F \hat{\mathbf{z}}_{t-1|t-1} + \mathbf{w}_t, \quad P_{t|t-1} = F P_{t-1|t-1} F' + G \Sigma G'$$

where $P_{t|t} = [I - K_t H] P_{t|t-1}$ for $t = 1, 2, \ldots, n$.

The log-likelihood function can be expressed as

$$\ell = -\frac{1}{2} \sum_{t=1}^{T} [\log |\Sigma_{t|t-1}| + (\mathbf{y}_t - \hat{\mathbf{y}}_{t|t-1})' \Sigma_{t|t-1}^{-1} (\mathbf{y}_t - \hat{\mathbf{y}}_{t|t-1})]$$

where $\hat{\mathbf{y}}_{t|t-1}$ and $\Sigma_{t|t-1}$ are determined recursively from Kalman filtering. To construct the likelihood function from Kalman filtering, you obtain $\hat{\mathbf{y}}_{t|t-1} = H \hat{\mathbf{z}}_{t|t-1}$, $\hat{\boldsymbol{\epsilon}}_{t|t-1} = \mathbf{y}_t - \hat{\mathbf{y}}_{t|t-1}$, and $\Sigma_{t|t-1} = H P_{t|t-1} H'$.

In the preceding state space form of a VARMAX model, the exogenous variables are treated as determined terms, which implies that the values of the exogenous variables must be provided to forecast the out-of-sample dependent variables. If you do not have the future values of the exogenous variables, either you predict the exogenous variables in a separate model, or you express both the exogenous variables and the dependent variables in one combined model and predict them together (Reinsel 1997).

The dimension of the state space vector of the Kalman filtering method for the VARMAX(*p*,*q*,*s*) model might be large, so it might take a lot of time and memory for computing.

Two examples of VARMAX modeling follow:

```
model y1 y2 = x1 / q=1;
nloptions tech=qn;

model y1 y2 = x1 / p=1 q=1 xlag=1 nocurrentx;
nloptions tech=qn;
```

Model Diagnostic Checks

Multivariate Model Diagnostic Checks

Log Likelihood

The log-likelihood function for the fitted model is reported in the LogLikelihood ODS table. The log-likelihood functions for different models are defined as follows:

- For VARMAX models that are estimated through the (conditional) maximum likelihood method, see the section "VARMA and VARMAX Modeling" on page 2972.

- For Bayesian VAR and VARX models, see the section "Bayesian VAR and VARX Modeling" on page 2970.
- For (Bayesian) vector error correction models, see the section "Vector Error Correction Modeling" on page 2986.
- For multivariate GARCH models, see the section "Multivariate GARCH Modeling" on page 3006.
- For VAR and VARX models that are estimated through the least squares (LS) method, the log likelihood is defined as

$$\ell = -\frac{1}{2}(T \log|\tilde{\Sigma}| + kT)$$

where $\tilde{\Sigma}$ is the maximum likelihood estimate of the innovation covariance matrix, k is the number of dependent variables, and T is the number of observations used in the estimation.

Information Criteria

The information criteria include Akaike's information criterion (AIC), the corrected Akaike's information criterion (AICC), the final prediction error criterion (FPE), the Hannan-Quinn criterion (HQC), and the Schwarz Bayesian criterion (SBC, also referred to as BIC). These criteria are defined as

$$\begin{aligned}
\text{AIC} &= -2\ell + 2r \\
\text{AICC} &= -2\ell + 2rT/(T-r-1) \\
\text{FPE} &= (\frac{T+r_b}{T-r_b})^k|\tilde{\Sigma}| \\
\text{HQC} &= -2\ell + 2r\log(\log(T)) \\
\text{SBC} &= -2\ell + r\log(T)
\end{aligned}$$

where ℓ is the log likelihood, r is the total number of parameters in the model, k is the number of dependent variables, T is the number of observations that are used to estimate the model, r_b is the number of parameters in each mean equation, and $\tilde{\Sigma}$ is the maximum likelihood estimate of Σ. As suggested by Burnham and Anderson (2004) for least squares estimation, the total number of parameters, r, must include the parameters in the innovation covariance matrix. When comparing models, choose the model that has the smallest criterion values.

See Figure 42.4 earlier in this chapter for an example of the output.

Portmanteau Statistic

The Portmanteau statistic, Q_s, is used to test whether correlation remains on the model residuals. The null hypothesis is that the residuals are uncorrelated. Let $C_\epsilon(l)$ be the residual cross-covariance matrices, $\hat{\rho}_\epsilon(l)$ be the residual cross-correlation matrices as

$$C_\epsilon(l) = T^{-1}\sum_{t=1}^{T-l}\boldsymbol{\epsilon}_t\boldsymbol{\epsilon}'_{t+l}$$

and

$$\hat{\rho}_\epsilon(l) = \hat{V}_\epsilon^{-1/2} C_\epsilon(l) \hat{V}_\epsilon^{-1/2} \text{ and } \hat{\rho}_\epsilon(-l) = \hat{\rho}_\epsilon(l)'$$

where $\hat{V}_\epsilon = \text{Diag}(\hat{\sigma}_{11}^2, \ldots, \hat{\sigma}_{kk}^2)$ and $\hat{\sigma}_{ii}^2$ are the diagonal elements of $\hat{\Sigma}$. The multivariate portmanteau test defined in Hosking (1980) is

$$Q_s = T^2 \sum_{l=1}^{s} (T-l)^{-1} \text{tr}\{\hat{\rho}_\epsilon(l)\hat{\rho}_\epsilon(0)^{-1}\hat{\rho}_\epsilon(-l)\hat{\rho}_\epsilon(0)^{-1}\}$$

The statistic Q_s has approximately the chi-square distribution with $k^2(s-p-q)$ degrees of freedom. An example of the output is displayed in Figure 42.7.

Univariate Model Diagnostic Checks

There are various ways to perform diagnostic checks for a univariate model. For details, see the section "Testing for Nonlinear Dependence: Heteroscedasticity Tests" on page 422 in Chapter 9, "The AUTOREG Procedure." An example of the output is displayed in Figure 42.8 and Figure 42.9.

- Durbin-Watson (DW) statistics: The DW test statistics test for the first order autocorrelation in the residuals.
- Jarque-Bera normality test: This test is helpful in determining whether the model residuals represent a white noise process. This tests the null hypothesis that the residuals have normality.
- *F* tests for autoregressive conditional heteroscedastic (ARCH) disturbances: *F* test statistics test for the heteroscedastic disturbances in the residuals. This tests the null hypothesis that the residuals have equal covariances
- *F* tests for AR disturbance: These test statistics are computed from the residuals of the univariate AR(1), AR(1,2), AR(1,2,3) and AR(1,2,3,4) models to test the null hypothesis that the residuals are uncorrelated.

Cointegration

This section briefly introduces the concepts of cointegration (Johansen 1995a).

Definition 1. (Engle and Granger 1987): If a series y_t with no deterministic components can be represented by a stationary and invertible ARMA process after differencing d times, the series is integrated of order d, that is, $y_t \sim I(d)$.

Definition 2. (Engle and Granger 1987): If all elements of the vector $\mathbf{y}_t$ are $I(d)$ and there exists a cointegrating vector $\boldsymbol{\beta} \neq 0$ such that $\boldsymbol{\beta}'\mathbf{y}_t \sim I(d-b)$ for any $b > 0$, the vector process is said to be cointegrated $CI(d,b)$.

A simple example of a cointegrated process is the following bivariate system:

$$\begin{aligned} y_{1t} &= \gamma y_{2t} + \epsilon_{1t} \\ y_{2t} &= y_{2,t-1} + \epsilon_{2t} \end{aligned}$$

with ϵ_{1t} and ϵ_{2t} being uncorrelated white noise processes. In the second equation, y_{2t} is a random walk, $\Delta y_{2t} = \epsilon_{2t}$, $\Delta \equiv 1 - B$. Differencing the first equation results in

$$\Delta y_{1t} = \gamma \Delta y_{2t} + \Delta \epsilon_{1t} = \gamma \epsilon_{2t} + \epsilon_{1t} - \epsilon_{1,t-1}$$

Thus, both y_{1t} and y_{2t} are $I(1)$ processes, but the linear combination $y_{1t} - \gamma y_{2t}$ is stationary. Hence $\mathbf{y}_t = (y_{1t}, y_{2t})'$ is cointegrated with a cointegrating vector $\boldsymbol{\beta} = (1, -\gamma)'$.

In general, if the vector process $\mathbf{y}_t$ has k components, then there can be more than one cointegrating vector $\boldsymbol{\beta}'$. It is assumed that there are r linearly independent cointegrating vectors with $r < k$, which make the $k \times r$ matrix $\boldsymbol{\beta}$. The rank of matrix $\boldsymbol{\beta}$ is r, which is called the *cointegration rank* of $\mathbf{y}_t$.

Common Trends

This section briefly discusses the implication of cointegration for the moving-average representation. Let $\mathbf{y}_t$ be cointegrated $CI(1, 1)$, then $\Delta \mathbf{y}_t$ has the Wold representation:

$$\Delta \mathbf{y}_t = \boldsymbol{\delta} + \Psi(B)\boldsymbol{\epsilon}_t$$

where $\boldsymbol{\epsilon}_t$ is $iid(0, \Sigma)$, $\Psi(B) = \sum_{j=0}^{\infty} \Psi_j B^j$ with $\Psi_0 = I_k$, and $\sum_{j=0}^{\infty} j|\Psi_j| < \infty$.

Assume that $\boldsymbol{\epsilon}_t = 0$ if $t \leq 0$ and $\mathbf{y}_0$ is a nonrandom initial value. Then the difference equation implies that

$$\mathbf{y}_t = \mathbf{y}_0 + \boldsymbol{\delta} t + \Psi(1) \sum_{i=0}^{t} \boldsymbol{\epsilon}_i + \Psi^*(B)\boldsymbol{\epsilon}_t$$

where $\Psi^*(B) = (1 - B)^{-1}(\Psi(B) - \Psi(1))$ and $\Psi^*(B)$ is absolutely summable.

Assume that the rank of $\Psi(1)$ is $m = k - r$. When the process $\mathbf{y}_t$ is cointegrated, there is a cointegrating $k \times r$ matrix $\boldsymbol{\beta}$ such that $\boldsymbol{\beta}'\mathbf{y}_t$ is stationary.

Premultiplying $\mathbf{y}_t$ by $\boldsymbol{\beta}'$ results in

$$\boldsymbol{\beta}'\mathbf{y}_t = \boldsymbol{\beta}'\mathbf{y}_0 + \boldsymbol{\beta}'\Psi^*(B)\boldsymbol{\epsilon}_t$$

because $\boldsymbol{\beta}'\Psi(1) = 0$ and $\boldsymbol{\beta}'\boldsymbol{\delta} = 0$.

Stock and Watson (1988) showed that the cointegrated process $\mathbf{y}_t$ has a common trends representation derived from the moving-average representation. Since the rank of $\Psi(1)$ is $m = k - r$, there is a $k \times r$ matrix H_1 with rank r such that $\Psi(1)H_1 = 0$. Let H_2 be a $k \times m$ matrix with rank m such that $H_2'H_1 = 0$; then $A = C(1)H_2$ has rank m. The $H = (H_1, H_2)$ has rank k. By construction of $\mathbf{H}$,

$$\Psi(1)H = [0, A] = AS_m$$

where $S_m = (0_{m\times r}, I_m)$. Since $\boldsymbol{\beta}'\Psi(1) = 0$ and $\boldsymbol{\beta}'\boldsymbol{\delta} = 0$, $\boldsymbol{\delta}$ lies in the column space of $\Psi(1)$ and can be written

$$\boldsymbol{\delta} = \Psi(1)\tilde{\boldsymbol{\delta}}$$

where $\tilde{\boldsymbol{\delta}}$ is a k-dimensional vector. The common trends representation is written as

$$\begin{aligned}
\mathbf{y}_t &= \mathbf{y}_0 + \Psi(1)[\tilde{\boldsymbol{\delta}}t + \sum_{i=0}^{t}\boldsymbol{\epsilon}_i] + \Psi^*(B)\boldsymbol{\epsilon}_t \\
&= \mathbf{y}_0 + \Psi(1)H[H^{-1}\tilde{\delta}t + H^{-1}\sum_{i=0}^{t}\boldsymbol{\epsilon}_i] + \mathbf{a}_t \\
&= \mathbf{y}_0 + A\boldsymbol{\tau}_t + \mathbf{a}_t
\end{aligned}$$

and

$$\boldsymbol{\tau}_t = \pi + \boldsymbol{\tau}_{t-1} + \mathbf{v}_t$$

where $\mathbf{a}_t = \Psi^*(B)\boldsymbol{\epsilon}_t$, $\pi = S_m H^{-1}\tilde{\boldsymbol{\delta}}$, $\boldsymbol{\tau}_t = S_m[H^{-1}\tilde{\boldsymbol{\delta}}t + H^{-1}\sum_{i=0}^{t}\boldsymbol{\epsilon}_i]$, and $\mathbf{v}_t = S_m H^{-1}\boldsymbol{\epsilon}_t$.

Stock and Watson showed that the common trends representation expresses $\mathbf{y}_t$ as a linear combination of m random walks ($\boldsymbol{\tau}_t$) with drift π plus $I(0)$ components ($\mathbf{a}_t$).

Test for the Common Trends

Stock and Watson (1988) proposed statistics for common trends testing. The null hypothesis is that the k-dimensional time series $\mathbf{y}_t$ has m common stochastic trends, where $m \leq k$ and the alternative is that it has s common trends, where $s < m$. The test procedure of m versus s common stochastic trends is performed based on the first-order serial correlation matrix of $\mathbf{y}_t$. Let $\boldsymbol{\beta}_\perp$ be a $k \times m$ matrix orthogonal to the cointegrating matrix such that $\boldsymbol{\beta}_\perp'\boldsymbol{\beta} = 0$ and $\boldsymbol{\beta}_\perp\boldsymbol{\beta}_\perp' = I_m$. Let $\mathbf{z}_t = \boldsymbol{\beta}'\mathbf{y}_t$ and $\mathbf{w}_t = \boldsymbol{\beta}_\perp'\mathbf{y}_t$. Then

$$\mathbf{w}_t = \boldsymbol{\beta}_\perp'\mathbf{y}_0 + \boldsymbol{\beta}_\perp'\boldsymbol{\delta}t + \boldsymbol{\beta}_\perp'\Psi(1)\sum_{i=0}^{t}\boldsymbol{\epsilon}_i + \boldsymbol{\beta}_\perp'\Psi^*(B)\boldsymbol{\epsilon}_t$$

Combining the expression of $\mathbf{z}_t$ and $\mathbf{w}_t$,

$$\begin{aligned}
\begin{bmatrix} \mathbf{z}_t \\ \mathbf{w}_t \end{bmatrix} &= \begin{bmatrix} \boldsymbol{\beta}'\mathbf{y}_0 \\ \boldsymbol{\beta}_\perp'\mathbf{y}_0 \end{bmatrix} + \begin{bmatrix} 0 \\ \boldsymbol{\beta}_\perp'\boldsymbol{\delta} \end{bmatrix} t + \begin{bmatrix} 0 \\ \boldsymbol{\beta}_\perp'\Psi(1) \end{bmatrix} \sum_{i=1}^{t}\boldsymbol{\epsilon}_i \\
&+ \begin{bmatrix} \boldsymbol{\beta}'\Psi^*(B) \\ \boldsymbol{\beta}_\perp'\Psi^*(B) \end{bmatrix} \boldsymbol{\epsilon}_t
\end{aligned}$$

The Stock-Watson common trends test is performed based on the component $\mathbf{w}_t$ by testing whether $\boldsymbol{\beta}_\perp'\Psi(1)$ has rank m against rank s.

The following statements perform the Stock-Watson test for common trends:

```
proc iml;
   sig = 100*i(2);
   phi = {-0.2 0.1, 0.5 0.2, 0.8 0.7, -0.4 0.6};
   call varmasim(y,phi) sigma=sig n=100 initial=0
                        seed=45876;
   cn = {'y1' 'y2'};
   create simul2 from y[colname=cn];
   append from y;
quit;

data simul2;
   set simul2;
   date = intnx( 'year', '01jan1900'd, _n_-1 );
   format date year4. ;
run;

proc varmax data=simul2;
   model y1 y2 / p=2 cointtest=(sw);
run;
```

In Figure 42.67, the first column is the null hypothesis that $\mathbf{y}_t$ has $m \leq k$ common trends; the second column is the alternative hypothesis that $\mathbf{y}_t$ has $s < m$ common trends; the third column contains the eigenvalues used for the test statistics; the fourth column contains the test statistics using AR(p) filtering of the data. The table shows the output of the case $p = 2$.

Figure 42.67 Common Trends Test (COINTTEST=(SW) Option)

The VARMAX Procedure

Common Trend Test

H0: Rank=m	H1: Rank=s	Eigenvalue	Filter	5% Critical Value	Lag
1	**0**	1.000906	0.09	-14.10	2
2	**0**	0.996763	-0.32	-8.80	
	1	0.648908	-35.11	-23.00	

The test statistic for testing for 2 versus 1 common trends is more negative (–35.1) than the critical value (–23.0). Therefore, the test rejects the null hypothesis, which means that the series has a single common trend.

Vector Error Correction Modeling

This section discusses the implication of cointegration for the autoregressive representation.

Consider the vector autoregressive process that has Gaussian errors defined by

$$\mathbf{y}_t = \sum_{i=1}^{p} \Phi_i \mathbf{y}_{t-i} + \boldsymbol{\epsilon}_t$$

or

$$\Phi(B)\mathbf{y}_t = \boldsymbol{\epsilon}_t$$

where the initial values, $\mathbf{y}_{-p+1}, \ldots, \mathbf{y}_0$, are fixed and $\boldsymbol{\epsilon}_t \sim N(0, \Sigma)$. The AR operator $\Phi(B)$ can be re-expressed as

$$\Phi(B) = \Phi^*(B)(1 - B) + \Phi(1)B$$

where

$$\Phi(1) = I_k - \Phi_1 - \Phi_2 - \cdots - \Phi_p, \Phi^*(B) = I_k - \sum_{i=1}^{p-1} \Phi_i^* B^i, \Phi_i^* = -\sum_{j=i+1}^{p} \Phi_j$$

The vector error correction model (VECM), also called the vector equilibrium correction model, is defined as

$$\Phi^*(B)(1 - B)\mathbf{y}_t = \boldsymbol{\alpha}\boldsymbol{\beta}'\mathbf{y}_{t-1} + \boldsymbol{\epsilon}_t$$

or

$$\Delta\mathbf{y}_t = \boldsymbol{\alpha}\boldsymbol{\beta}'\mathbf{y}_{t-1} + \sum_{i=1}^{p-1} \Phi_i^* \Delta\mathbf{y}_{t-i} + \boldsymbol{\epsilon}_t$$

where $\boldsymbol{\alpha}\boldsymbol{\beta}' = -\Phi(1)$.

Granger Representation Theorem Engle and Granger (1987) define

$$\Pi(z) \equiv (1 - z)I_k - \boldsymbol{\alpha}\boldsymbol{\beta}'z - \sum_{i=1}^{p-1} \Phi_i^*(1 - z)z^i$$

and the following assumptions hold:

1. $|\Pi(z)| = 0 \Rightarrow |z| > 1$ or $z = 1$.
2. The number of unit roots, $z = 1$, is exactly $k - r$.
3. $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are $k \times r$ matrices, and their ranks are both r.

Then y_t has the representation

$$y_t = C\sum_{i=1}^{t} \boldsymbol{\epsilon}_i + C^*(B)\boldsymbol{\epsilon}_t + y_0^*$$

where the Granger representation coefficient, C, is

$$C = \boldsymbol{\beta}_\perp \left[\boldsymbol{\alpha}'_\perp \Phi(1)\boldsymbol{\beta}_\perp\right]^{-1} \boldsymbol{\alpha}'_\perp$$

where the full-rank $k \times r$ matrix $\boldsymbol{\beta}_\perp$ is orthogonal to $\boldsymbol{\beta}$ and the full-rank $k \times r$ matrix $\boldsymbol{\alpha}_\perp$ is orthogonal to $\boldsymbol{\alpha}$. $C^*(B)\boldsymbol{\epsilon}_t = \sum_{j=1}^{\infty} C_j^* \boldsymbol{\epsilon}_{t-j}$ is an $I(0)$ process, and y_0^* depends on the initial values.

The Granger representation coefficient C can be defined only when the $(k-r) \times (k-r)$ matrix $\boldsymbol{\alpha}'_{\perp}\Phi(1)\boldsymbol{\beta}_{\perp}$ is invertible.

One motivation for the VECM(p) form is to consider the relation $\boldsymbol{\beta}'\mathbf{y}_t = \mathbf{c}$ as defining the underlying economic relations. Assume that agents react to the disequilibrium error $\boldsymbol{\beta}'\mathbf{y}_t - \mathbf{c}$ through the adjustment coefficient $\boldsymbol{\alpha}$ to restore equilibrium. The cointegrating vector, $\boldsymbol{\beta}$, is sometimes called the long-run parameter.

Consider a vector error correction model that has a deterministic term, D_t, which can contain a constant, a linear trend, and seasonal dummy variables. Exogenous variables can also be included in the model. The model has the form

$$\Delta\mathbf{y}_t = \Pi\mathbf{y}_{t-1} + \sum_{i=1}^{p-1}\Phi_i^*\Delta\mathbf{y}_{t-i} + AD_t + \sum_{i=0}^{s}\Theta_i^*\mathbf{x}_{t-i} + \boldsymbol{\epsilon}_t$$

where $\Pi = \boldsymbol{\alpha}\boldsymbol{\beta}'$.

The alternative vector error correction representation considers the error correction term at lag $t-p$ and is written as

$$\Delta\mathbf{y}_t = \sum_{i=1}^{p-1}\Phi_i^{\sharp}\Delta\mathbf{y}_{t-i} + \Pi^{\sharp}\mathbf{y}_{t-p} + AD_t + \sum_{i=0}^{s}\Theta_i^*\mathbf{x}_{t-i} + \boldsymbol{\epsilon}_t$$

If the matrix Π has a full rank ($r = k$), all components of $\mathbf{y}_t$ are $I(0)$. On the other hand, $\mathbf{y}_t$ are stationary in difference if $rank(\Pi) = 0$. When the rank of the matrix Π is $r < k$, there are $k-r$ linear combinations that are nonstationary and r stationary cointegrating relations. Note that the linearly independent vector $\mathbf{z}_t = \boldsymbol{\beta}'\mathbf{y}_t$ is stationary and this transformation is not unique unless $r = 1$. There does not exist a unique cointegrating matrix $\boldsymbol{\beta}$ because the coefficient matrix Π can also be decomposed as

$$\Pi = \boldsymbol{\alpha}MM^{-1}\boldsymbol{\beta}' = \boldsymbol{\alpha}^*\boldsymbol{\beta}^{*'}$$

where M is an $r \times r$ nonsingular matrix.

Test for Cointegration

The cointegration rank test determines the linearly independent columns of Π. Johansen and Juselius proposed the cointegration rank test by using the reduced rank regression (Johansen 1988, 1995b; Johansen and Juselius 1990).

Different Specifications of Deterministic Trends

When you construct the VECM(p) form from the VAR(p) model, the deterministic terms in the VECM(p) form can differ from those in the VAR(p) model. When there are deterministic cointegrated relationships among variables, deterministic terms in the VAR(p) model are not present in the VECM(p) form. On the other hand, if there are stochastic cointegrated relationships in the VAR(p) model, deterministic terms appear in the VECM(p) form via the error correction term or as an independent term in the VECM(p) form. There are five different specifications of deterministic trends in the VECM(p) form.

- **Case 1:** There is no separate drift in the VECM(p) form.

$$\Delta \mathbf{y}_t = \boldsymbol{\alpha}\boldsymbol{\beta}'\mathbf{y}_{t-1} + \sum_{i=1}^{p-1} \Phi_i^* \Delta \mathbf{y}_{t-i} + \boldsymbol{\epsilon}_t$$

- **Case 2:** There is no separate drift in the VECM(p) form, but a constant enters only via the error correction term.

$$\Delta \mathbf{y}_t = \boldsymbol{\alpha}(\boldsymbol{\beta}', \boldsymbol{\beta}_0)(\mathbf{y}'_{t-1}, 1)' + \sum_{i=1}^{p-1} \Phi_i^* \Delta \mathbf{y}_{t-i} + \boldsymbol{\epsilon}_t$$

- **Case 3:** There is a separate drift and no separate linear trend in the VECM(p) form.

$$\Delta \mathbf{y}_t = \boldsymbol{\alpha}\boldsymbol{\beta}'\mathbf{y}_{t-1} + \sum_{i=1}^{p-1} \Phi_i^* \Delta \mathbf{y}_{t-i} + \boldsymbol{\delta}_0 + \boldsymbol{\epsilon}_t$$

- **Case 4:** There is a separate drift and no separate linear trend in the VECM(p) form, but a linear trend enters only via the error correction term.

$$\Delta \mathbf{y}_t = \boldsymbol{\alpha}(\boldsymbol{\beta}', \boldsymbol{\beta}_1)(\mathbf{y}'_{t-1}, t)' + \sum_{i=1}^{p-1} \Phi_i^* \Delta \mathbf{y}_{t-i} + \boldsymbol{\delta}_0 + \boldsymbol{\epsilon}_t$$

- **Case 5:** There is a separate linear trend in the VECM(p) form.

$$\Delta \mathbf{y}_t = \boldsymbol{\alpha}\boldsymbol{\beta}'\mathbf{y}_{t-1} + \sum_{i=1}^{p-1} \Phi_i^* \Delta \mathbf{y}_{t-i} + \boldsymbol{\delta}_0 + \boldsymbol{\delta}_1 t + \boldsymbol{\epsilon}_t$$

First, focus on Cases 1, 3, and 5 to test the null hypothesis that there are at most r cointegrating vectors. Let

$$\begin{aligned}
Z_{0t} &= \Delta \mathbf{y}_t \\
Z_{1t} &= \mathbf{y}_{t-1} \\
Z_{2t} &= [\Delta \mathbf{y}'_{t-1}, \ldots, \Delta \mathbf{y}'_{t-p+1}, D_t]' \\
Z_0 &= [Z_{01}, \ldots, Z_{0T}]' \\
Z_1 &= [Z_{11}, \ldots, Z_{1T}]' \\
Z_2 &= [Z_{21}, \ldots, Z_{2T}]'
\end{aligned}$$

where D_t can be empty for Case 1, 1 for Case 3, and $(1, t)$ for Case 5.

In Case 2, Z_{1t} and Z_{2t} are defined as

$$\begin{aligned}
Z_{1t} &= [\mathbf{y}'_{t-1}, 1]' \\
Z_{2t} &= [\Delta \mathbf{y}'_{t-1}, \ldots, \Delta \mathbf{y}'_{t-p+1}]'
\end{aligned}$$

In Case 4, Z_{1t} and Z_{2t} are defined as

$$\begin{aligned} Z_{1t} &= [\mathbf{y}'_{t-1}, t]' \\ Z_{2t} &= [\Delta\mathbf{y}'_{t-1}, \ldots, \Delta\mathbf{y}'_{t-p+1}, 1]' \end{aligned}$$

Let Ψ be the matrix of parameters consisting of $\Phi^*_1, \ldots, \Phi^*_{p-1}$, A, and $\Theta^*_0, \ldots, \Theta^*_s$, where parameter A corresponds with the regressors D_t. Then the VECM(p) form is rewritten in these variables as

$$Z_{0t} = \boldsymbol{\alpha}\boldsymbol{\beta}' Z_{1t} + \Psi Z_{2t} + \boldsymbol{\epsilon}_t$$

The log-likelihood function is given by

$$\begin{aligned} \ell &= -\frac{kT}{2}\log 2\pi - \frac{T}{2}\log|\Sigma| \\ &\quad -\frac{1}{2}\sum_{t=1}^{T}(Z_{0t} - \boldsymbol{\alpha}\boldsymbol{\beta}' Z_{1t} - \Psi Z_{2t})'\Sigma^{-1}(Z_{0t} - \boldsymbol{\alpha}\boldsymbol{\beta}' Z_{1t} - \Psi Z_{2t}) \end{aligned}$$

The residuals, R_{0t} and R_{1t}, are obtained by regressing Z_{0t} and Z_{1t} on Z_{2t}, respectively. The regression equation of residuals is

$$R_{0t} = \boldsymbol{\alpha}\boldsymbol{\beta}' R_{1t} + \hat{\boldsymbol{\epsilon}}_t$$

The crossproducts matrices are computed

$$S_{ij} = \frac{1}{T}\sum_{t=1}^{T} R_{it} R'_{jt}, \quad i, j = 0, 1$$

Then the maximum likelihood estimator for $\boldsymbol{\beta}$ is obtained from the eigenvectors that correspond to the r largest eigenvalues of the following equation:

$$|\lambda S_{11} - S_{10} S_{00}^{-1} S_{01}| = 0$$

The eigenvalues of the preceding equation are squared canonical correlations between R_{0t} and R_{1t}, and the eigenvectors that correspond to the r largest eigenvalues are the r linear combinations of $\mathbf{y}_{t-1}$, which have the largest squared partial correlations with the stationary process $\Delta\mathbf{y}_t$ after correcting for lags and deterministic terms. Such an analysis calls for a reduced rank regression of $\Delta\mathbf{y}_t$ on $\mathbf{y}_{t-1}$ corrected for $(\Delta\mathbf{y}_{t-1}, \ldots, \Delta\mathbf{y}_{t-p+1}, D_t)$, as discussed by Anderson (1951). Johansen (1988) suggests two test statistics to test the null hypothesis that there are at most r cointegrating vectors

$$\mathrm{H}_0 : \lambda_i = 0 \text{ for } i = r+1, \ldots, k$$

Trace Test

The trace statistic for testing the null hypothesis that there are at most r cointegrating vectors is as follows:

$$\lambda_{trace} = -T \sum_{i=r+1}^{k} \log(1-\lambda_i)$$

The asymptotic distribution of this statistic is given by

$$tr\left\{\int_0^1 (dW)\tilde{W}'\left(\int_0^1 \tilde{W}\tilde{W}'dr\right)^{-1}\int_0^1 \tilde{W}(dW)'\right\}$$

where $tr(A)$ is the trace of a matrix A, W is the $k-r$ dimensional Brownian motion, and $\tilde{W}$ is the Brownian motion itself, or the demeaned or detrended Brownian motion according to the different specifications of deterministic trends in the vector error correction model.

Maximum Eigenvalue Test

The maximum eigenvalue statistic for testing the null hypothesis that there are at most r cointegrating vectors is as follows:

$$\lambda_{max} = -T\log(1-\lambda_{r+1})$$

The asymptotic distribution of this statistic is given by

$$max\{\int_0^1 (dW)\tilde{W}'(\int_0^1 \tilde{W}\tilde{W}'dr)^{-1}\int_0^1 \tilde{W}(dW)'\}$$

where $max(A)$ is the maximum eigenvalue of a matrix A. Osterwald-Lenum (1992) provided detailed tables of the critical values of these statistics.

The following statements use the JOHANSEN option to compute the Johansen cointegration rank trace test of integrated order 1:

```
proc varmax data=simul2;
   model y1 y2 / p=2 cointtest=(johansen=(normalize=y1));
run;
```

Figure 42.68 shows the output based on the model specified in the MODEL statement. An intercept term is assumed. In the "Cointegration Rank Test Using Trace" table, the column Drift in ECM indicates that there is no separate drift in the error correction model, and the column Drift in Process indicates that the process has a constant drift before differencing. The "Cointegration Rank Test Using Trace" table shows the trace statistics and p-values based on Case 3, and the "Cointegration Rank Test Using Trace under Restriction" table shows the trace statistics and p-values based on Case 2. For a specified significance level, such as 5%, the output indicates that the null hypothesis that the series are not cointegrated (H0: Rank = 0) can be rejected, because the p-values for both Case 2 and Case 3 are less than 0.05. The output also shows that the null hypothesis that the series are cointegrated with rank 1 (H0: Rank = 1) cannot be rejected for either Case 2 or Case 3, because the p-values for these tests are both greater than 0.05.

Figure 42.68 Cointegration Rank Test (COINTTEST=(JOHANSEN=) Option)

The VARMAX Procedure

Cointegration Rank Test Using Trace

H0: Rank=r	H1: Rank>r	Eigenvalue	Trace	Pr > Trace	Drift in ECM	Drift in Process
0	0	0.4644	61.7522	<.0001	Constant	Linear
1	1	0.0056	0.5552	0.4559		

Cointegration Rank Test Using Trace Under Restriction

H0: Rank=r	H1: Rank>r	Eigenvalue	Trace	Pr > Trace	Drift in ECM	Drift in Process
0	0	0.5209	76.3788	<.0001	Constant	Constant
1	1	0.0426	4.2680	0.3741		

Figure 42.69 shows which result, either Case 2 (the hypothesis H0) or Case 3 (the hypothesis H1), is appropriate depending on the significance level. Since the cointegration rank is chosen to be 1 by the result in Figure 42.68, look at the last row that corresponds to rank=1. Since the *p*-value is 0.054, the Case 2 cannot be rejected at the significance level 5%, but it can be rejected at the significance level 10%. For modeling of the two Case 2 and Case 3, see Figure 42.72 and Figure 42.73.

Figure 42.69 Cointegration Rank Test, Continued

Hypothesis of the Restriction

Hypothesis	Drift in ECM	Drift in Process
H0(Case 2)	Constant	Constant
H1(Case 3)	Constant	Linear

Hypothesis Test of the Restriction

Rank	Eigenvalue	Restricted Eigenvalue	DF	Chi-Square	Pr > ChiSq
0	0.4644	0.5209	2	14.63	0.0007
1	0.0056	0.0426	1	3.71	0.0540

Figure 42.70 shows the estimates of long-run parameter (Beta) and adjustment coefficients (Alpha) based on Case 3.

Figure 42.70 Cointegration Rank Test, Continued

Beta

Variable	1	2
y1	1.00000	1.00000
y2	-2.04869	-0.02854

Alpha

Variable	1	2
y1	-0.46421	-0.00502
y2	0.17535	-0.01275

Using the NORMALIZE= option, the first row of the "Beta" table has 1. Considering that the cointegration rank is 1, the long-run relationship of the series is

$$\begin{aligned}
\boldsymbol{\beta}' y_t &= \begin{bmatrix} 1 & -2.04869 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \\
&= y_{1t} - 2.04869 y_{2t} \\
y_{1t} &= 2.04869 y_{2t}
\end{aligned}$$

Figure 42.71 shows the estimates of long-run parameter (Beta) and adjustment coefficients (Alpha) based on Case 2.

Figure 42.71 Cointegration Rank Test, Continued

Beta Under Restriction		
Variable	**1**	**2**
y1	1.00000	1.00000
y2	-2.04366	-2.75773
1	6.75919	101.37051

Alpha Under Restriction		
Variable	**1**	**2**
y1	-0.48015	0.01091
y2	0.12538	0.03722

Considering that the cointegration rank is 1, the long-run relationship of the series is

$$\begin{aligned}
\boldsymbol{\beta}' y_t &= \begin{bmatrix} 1 & -2.04366 & 6.75919 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ 1 \end{bmatrix} \\
&= y_{1t} - 2.04366\, y_{2t} + 6.75919 \\
y_{1t} &= 2.04366\, y_{2t} - 6.75919
\end{aligned}$$

Estimation of Vector Error Correction Model

The preceding log-likelihood function is maximized for

$$\begin{aligned}
\hat{\boldsymbol{\beta}} &= S_{11}^{-1/2}[v_1, \ldots, v_r] \\
\hat{\boldsymbol{\alpha}} &= S_{01}\hat{\boldsymbol{\beta}}(\hat{\boldsymbol{\beta}}' S_{11} \hat{\boldsymbol{\beta}})^{-1} \\
\hat{\Pi} &= \hat{\boldsymbol{\alpha}}\hat{\boldsymbol{\beta}}' \\
\hat{\Psi}' &= (Z_2' Z_2)^{-1} Z_2' (Z_0 - Z_1 \hat{\Pi}') \\
\hat{\Sigma} &= (Z_0 - Z_2\hat{\Psi}' - Z_1\hat{\Pi}')'(Z_0 - Z_2\hat{\Psi}' - Z_1\hat{\Pi}')/T
\end{aligned}$$

The estimators of the orthogonal complements of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are

$$\hat{\boldsymbol{\beta}}_\perp = S_{11}[v_{r+1}, \ldots, v_k]$$

and

$$\hat{\boldsymbol{\alpha}}_{\perp} = S_{00}^{-1} S_{01}[v_{r+1}, \ldots, v_k]$$

Let ϑ denote the parameter vector $(\mathrm{vec}(\boldsymbol{\alpha}, \Psi)', \mathrm{vech}(\Sigma)')'$. The covariance of parameter estimates $\hat{\vartheta}$ is obtained as the inverse of the negative Hessian matrix $H \equiv \frac{\partial^2 \ell}{\partial \vartheta \partial \vartheta'}$. Because $\hat{\Pi} = \hat{\boldsymbol{\alpha}}\hat{\boldsymbol{\beta}}'$, the variance of $\hat{\Pi}$ and the covariance between $\hat{\Pi}$ and $\hat{\vartheta}$ are calculated as follows:

$$\mathrm{cov}(\mathrm{vec}(\hat{\Pi}), \mathrm{vec}(\hat{\Pi})) = (\hat{\boldsymbol{\beta}} \otimes I_k)\mathrm{cov}(\mathrm{vec}(\hat{\boldsymbol{\alpha}}), \mathrm{vec}(\hat{\boldsymbol{\alpha}}))(\hat{\boldsymbol{\beta}} \otimes I_k)'$$

$$\mathrm{cov}(\mathrm{vec}(\hat{\Pi}), \hat{\vartheta}) = (\hat{\boldsymbol{\beta}} \otimes I_k)\mathrm{cov}(\mathrm{vec}(\hat{\boldsymbol{\alpha}}), \hat{\vartheta})$$

For Case 2 (Case 4), because the coefficient vector $\hat{\boldsymbol{\delta}}_0$ ($\hat{\boldsymbol{\delta}}_1$) for the constant term (the linear trend term) is the product of $\hat{\boldsymbol{\alpha}}$ and $\hat{\boldsymbol{\beta}}_0$ ($\hat{\boldsymbol{\beta}}_1$), the variance of $\hat{\boldsymbol{\delta}}_0$ ($\hat{\boldsymbol{\delta}}_1$) and the covariance between $\hat{\boldsymbol{\delta}}_0$ ($\hat{\boldsymbol{\delta}}_1$) and $\hat{\vartheta}$ are calculated as follows:

$$\mathrm{cov}(\hat{\boldsymbol{\delta}}_i, \hat{\boldsymbol{\delta}}_i) = (\hat{\boldsymbol{\beta}}_i' \otimes I_k)\mathrm{cov}(\mathrm{vec}(\hat{\boldsymbol{\alpha}}), \mathrm{vec}(\hat{\boldsymbol{\alpha}}))(\hat{\boldsymbol{\beta}}_i' \otimes I_k)', \quad i = 0 \text{ or } 1$$

$$\mathrm{cov}(\hat{\boldsymbol{\delta}}_i, \hat{\vartheta}) = (\hat{\boldsymbol{\beta}}_i' \otimes I_k)\mathrm{cov}(\mathrm{vec}(\hat{\boldsymbol{\alpha}}), \hat{\vartheta}), \quad i = 0 \text{ or } 1$$

The following statements are examples of fitting the five different cases of the vector error correction models mentioned in the previous section.

For fitting Case 1,

```
model y1 y2 / p=2 noint;
cointeg rank=1 normalize=y1;
```

For fitting Case 2,

```
model y1 y2 / p=2;
cointeg rank=1 normalize=y1 ectrend;
```

For fitting Case 3,

```
model y1 y2 / p=2;
cointeg rank=1 normalize=y1;
```

For fitting Case 4,

```
model y1 y2 / p=2 trend=linear;
cointeg rank=1 normalize=y1 ectrend;
```

For fitting Case 5,

```
model y1 y2 / p=2 trend=linear;
cointeg rank=1 normalize=y1;
```

In the previous example, the output from the COINTTEST=(JOHANSEN) option shown in Figure 42.69 indicates that you can fit the model by using either Case 2 or Case 3 because the test of the restriction was not significant at the 0.05 level, but was significant at the 0.10 level. Following both models are fit to show the differences in the displayed output. Figure 42.72 is for Case 2, and Figure 42.73 is for Case 3.

For Case 2,

```
proc varmax data=simul2;
   model y1 y2 / p=2 print=(estimates);
   cointeg rank=1 normalize=y1 ectrend;
run;
```

Figure 42.72 Parameter Estimation with the ECTREND Option

The VARMAX Procedure

Parameter Alpha * Beta' Estimates

Variable	y1	y2	1
y1	-0.48015	0.98126	-3.24543
y2	0.12538	-0.25624	0.84748

AR Coefficients of Differenced Lag

DIF Lag	Variable	y1	y2
1	y1	-0.72759	-0.77463
	y2	0.38982	-0.55173

Model Parameter Estimates

Equation	Parameter	Estimate	Standard Error	t Value	Pr > |t|	Variable
D_y1	CONST1	-3.24543	0.33022	-9.83	<.0001	1, EC
	AR1_1_1	-0.48015	0.04886	-9.83	<.0001	y1(t-1)
	AR1_1_2	0.98126	0.09984	9.83	<.0001	y2(t-1)
	AR2_1_1	-0.72759	0.04623	-15.74	<.0001	D_y1(t-1)
	AR2_1_2	-0.77463	0.04978	-15.56	<.0001	D_y2(t-1)
D_y2	CONST2	0.84748	0.35394	2.39	0.0187	1, EC
	AR1_2_1	0.12538	0.05236	2.39	0.0187	y1(t-1)
	AR1_2_2	-0.25624	0.10702	-2.39	0.0187	y2(t-1)
	AR2_2_1	0.38982	0.04955	7.87	<.0001	D_y1(t-1)
	AR2_2_2	-0.55173	0.05336	-10.34	<.0001	D_y2(t-1)

Figure 42.72 can be reported as follows:

$$\Delta \mathbf{y}_t = \begin{bmatrix} -0.48015 & 0.98126 & -3.24543 \\ 0.12538 & -0.25624 & 0.84748 \end{bmatrix} \begin{bmatrix} y_{1,t-1} \\ y_{2,t-1} \\ 1 \end{bmatrix} + \begin{bmatrix} -0.72759 & -0.77463 \\ 0.38982 & -0.55173 \end{bmatrix} \Delta \mathbf{y}_{t-1} + \boldsymbol{\epsilon}_t$$

The keyword “EC” in the “Model Parameter Estimates” table means that the ECTREND option is used for fitting the model.

For fitting Case 3,

```
proc varmax data=simul2;
   model y1 y2 / p=2 print=(estimates);
   cointeg rank=1 normalize=y1;
run;
```

Figure 42.73 Parameter Estimation without the ECTREND Option

The VARMAX Procedure

Parameter Alpha * Beta' Estimates		
Variable	**y1**	**y2**
y1	-0.46421	0.95103
y2	0.17535	-0.35923

AR Coefficients of Differenced Lag			
DIF Lag	**Variable**	**y1**	**y2**
1	**y1**	-0.74052	-0.76305
	y2	0.34820	-0.51194

Model Parameter Estimates						
Equation	**Parameter**	**Estimate**	**Standard Error**	**t Value**	**Pr > |t|**	**Variable**
D_y1	**CONST1**	-2.60825	1.32398	-1.97	0.0518	1
	AR1_1_1	-0.46421	0.05474	-8.48	<.0001	y1(t-1)
	AR1_1_2	0.95103	0.11215	8.48	<.0001	y2(t-1)
	AR2_1_1	-0.74052	0.05060	-14.63	<.0001	D_y1(t-1)
	AR2_1_2	-0.76305	0.05352	-14.26	<.0001	D_y2(t-1)
D_y2	**CONST2**	3.43005	1.39587	2.46	0.0159	1
	AR1_2_1	0.17535	0.05771	3.04	0.0031	y1(t-1)
	AR1_2_2	-0.35923	0.11824	-3.04	0.0031	y2(t-1)
	AR2_2_1	0.34820	0.05335	6.53	<.0001	D_y1(t-1)
	AR2_2_2	-0.51194	0.05643	-9.07	<.0001	D_y2(t-1)

Figure 42.73 can be reported as follows:

$$\Delta \mathbf{y}_t = \begin{bmatrix} -0.46421 & 0.95103 \\ 0.17535 & -0.35293 \end{bmatrix} \mathbf{y}_{t-1} + \begin{bmatrix} -0.74052 & -0.76305 \\ 0.34820 & -0.51194 \end{bmatrix} \Delta \mathbf{y}_{t-1} + \begin{bmatrix} -2.60825 \\ 3.43005 \end{bmatrix} + \boldsymbol{\epsilon}_t$$

A Test for the Long-run Relations

Consider the example with the variables m_t log real money, y_t log real income, i_t^d deposit interest rate, and i_t^b bond interest rate. It seems a natural hypothesis that in the long-run relation, money and income have equal coefficients with opposite signs. This can be formulated as the hypothesis that the cointegrated relation contains only m_t and y_t through $m_t - y_t$. For the analysis, you can express these restrictions in the

parameterization of **H** such that $\boldsymbol{\beta} = H\phi$, where **H** is a known $k \times s$ matrix and ψ is the $s \times r (r \leq s < k)$ parameter matrix to be estimated. For this example, **H** is given by

$$H = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Restriction $H_0: \boldsymbol{\beta} = H\phi$

When the linear restriction $\boldsymbol{\beta} = H\phi$ is given, it implies that the same restrictions are imposed on all cointegrating vectors. You obtain the maximum likelihood estimator of $\boldsymbol{\beta}$ by reduced rank regression of $\Delta \mathbf{y}_t$ on $H\mathbf{y}_{t-1}$ corrected for $(\Delta \mathbf{y}_{t-1}, \ldots, \Delta \mathbf{y}_{t-p+1}, D_t)$, solving the following equation

$$|\rho H' S_{11} H - H' S_{10} S_{00}^{-1} S_{01} H| = 0$$

for the eigenvalues $1 > \rho_1 > \cdots > \rho_s > 0$ and eigenvectors $(v_1, \ldots, v_s)$, S_{ij} given in the preceding section. Then choose $\hat{\phi} = (v_1, \ldots, v_r)$ that corresponds to the r largest eigenvalues, and the $\hat{\boldsymbol{\beta}}$ is $H\hat{\phi}$.

The test statistic for $H_0: \boldsymbol{\beta} = H\phi$ is given by

$$T \sum_{i=1}^{r} \log\{(1 - \rho_i)/(1 - \lambda_i)\} \xrightarrow{d} \chi^2_{r(k-s)}$$

If the series has no deterministic trend, the constant term should be restricted by $\boldsymbol{\alpha}'_{\perp} \boldsymbol{\delta}_0 = 0$ as in Case 2. Then **H** is given by

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The following statements test that $2\,\beta_1 + \beta_2 = 0$:

```
proc varmax data=simul2;
   model y1 y2 / p=2;
   cointeg rank=1 h=(1,-2) normalize=y1;
run;
```

Figure 42.74 shows the results of testing $H_0: 2\beta_1 + \beta_2 = 0$. The input **H** matrix is $H = (1 - 2)'$. The adjustment coefficient is reestimated under the restriction, and the test indicates that you cannot reject the null hypothesis.

Figure 42.74 Testing of Linear Restriction (H= Option)

The VARMAX Procedure

Beta Under Restriction	
Variable	**1**
y1	1.00000
y2	-2.00000

Alpha Under Restriction	
Variable	**1**
y1	-0.47404
y2	0.17534

Hypothesis Test					
Index	**Eigenvalue**	**Restricted Eigenvalue**	**DF**	**Chi-Square**	**Pr > ChiSq**
1	0.4644	0.4616	1	0.51	0.4738

Test for the Weak Exogeneity and Restrictions of Alpha

Consider a vector error correction model:

$$\Delta \mathbf{y}_t = \boldsymbol{\alpha}\boldsymbol{\beta}'\mathbf{y}_{t-1} + \sum_{i=1}^{p-1} \Phi_i^* \Delta \mathbf{y}_{t-i} + A D_t + \boldsymbol{\epsilon}_t$$

Divide the process $\mathbf{y}_t$ into $(\mathbf{y}_{1t}', \mathbf{y}_{2t}')'$ with dimension k_1 and k_2 and the Σ into

$$\Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}$$

Similarly, the parameters can be decomposed as follows:

$$\boldsymbol{\alpha} = \begin{bmatrix} \boldsymbol{\alpha}_1 \\ \boldsymbol{\alpha}_2 \end{bmatrix} \quad \Phi_i^* = \begin{bmatrix} \Phi_{1i}^* \\ \Phi_{2i}^* \end{bmatrix} \quad A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$$

Then the VECM(p) form can be rewritten by using the decomposed parameters and processes:

$$\begin{bmatrix} \Delta \mathbf{y}_{1t} \\ \Delta \mathbf{y}_{2t} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\alpha}_1 \\ \boldsymbol{\alpha}_2 \end{bmatrix} \boldsymbol{\beta}' \mathbf{y}_{t-1} + \sum_{i=1}^{p-1} \begin{bmatrix} \Phi_{1i}^* \\ \Phi_{2i}^* \end{bmatrix} \Delta \mathbf{y}_{t-i} + \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} D_t + \begin{bmatrix} \boldsymbol{\epsilon}_{1t} \\ \boldsymbol{\epsilon}_{2t} \end{bmatrix}$$

The conditional model for $\mathbf{y}_{1t}$ given $\mathbf{y}_{2t}$ is

$$\begin{aligned} \Delta \mathbf{y}_{1t} &= \omega \Delta \mathbf{y}_{2t} + (\alpha_1 - \omega\alpha_2)\boldsymbol{\beta}'\mathbf{y}_{t-1} + \sum_{i=1}^{p-1} (\Phi_{1i}^* - \omega\Phi_{2i}^*)\Delta \mathbf{y}_{t-i} \\ &\quad + (A_1 - \omega A_2) D_t + \boldsymbol{\epsilon}_{1t} - \omega\boldsymbol{\epsilon}_{2t} \end{aligned}$$

and the marginal model of $\mathbf{y}_{2t}$ is

$$\Delta \mathbf{y}_{2t} = \alpha_2 \boldsymbol{\beta}' \mathbf{y}_{t-1} + \sum_{i=1}^{p-1} \Phi_{2i}^* \Delta \mathbf{y}_{t-i} + A_2 D_t + \boldsymbol{\epsilon}_{2t}$$

where $\omega = \Sigma_{12} \Sigma_{22}^{-1}$.

The test of weak exogeneity of $\mathbf{y}_{2t}$ for the parameters $(\alpha_1, \boldsymbol{\beta})$ determines whether $\alpha_2 = 0$. Weak exogeneity means that there is no information about $\boldsymbol{\beta}$ in the marginal model or that the variables $\mathbf{y}_{2t}$ do not react to a disequilibrium.

Restriction $H_0: \boldsymbol{\alpha} = J\psi$

Consider the null hypothesis $H_0: \boldsymbol{\alpha} = J\psi$, where J is a $k \times m$ matrix with $r \leq m < k$.

From the previous residual regression equation

$$R_{0t} = \boldsymbol{\alpha}\boldsymbol{\beta}' R_{1t} + \hat{\boldsymbol{\epsilon}}_t = J\psi\boldsymbol{\beta}' R_{1t} + \hat{\boldsymbol{\epsilon}}_t$$

you can obtain

$$\begin{aligned} \bar{J}' R_{0t} &= \psi\boldsymbol{\beta}' R_{1t} + \bar{J}'\hat{\boldsymbol{\epsilon}}_t \\ J_\perp' R_{0t} &= J_\perp'\hat{\boldsymbol{\epsilon}}_t \end{aligned}$$

where $\bar{J} = J(J'J)^{-1}$ and $J_\perp$ is orthogonal to J such that $J_\perp' J = 0$.

Define

$$\Sigma_{JJ_\perp} = \bar{J}'\Sigma J_\perp \text{ and } \Sigma_{J_\perp J_\perp} = J_\perp'\Sigma J_\perp$$

and let $\omega = \Sigma_{JJ_\perp}\Sigma_{J_\perp J_\perp}^{-1}$. Then $\bar{J}' R_{0t}$ can be written as

$$\bar{J}' R_{0t} = \psi\boldsymbol{\beta}' R_{1t} + \omega J_\perp' R_{0t} + \bar{J}'\hat{\boldsymbol{\epsilon}}_t - \omega J_\perp'\hat{\boldsymbol{\epsilon}}_t$$

Using the marginal distribution of $J_\perp' R_{0t}$ and the conditional distribution of $\bar{J}' R_{0t}$, the new residuals are computed as

$$\begin{aligned} \tilde{R}_{Jt} &= \bar{J}' R_{0t} - S_{JJ_\perp} S_{J_\perp J_\perp}^{-1} J_\perp' R_{0t} \\ \tilde{R}_{1t} &= R_{1t} - S_{1J_\perp} S_{J_\perp J_\perp}^{-1} J_\perp' R_{0t} \end{aligned}$$

where

$$S_{JJ_\perp} = \bar{J}' S_{00} J_\perp, \quad S_{J_\perp J_\perp} = J_\perp' S_{00} J_\perp, \quad \text{and} \quad S_{J_\perp 1} = J_\perp' S_{01}$$

In terms of $\tilde{R}_{Jt}$ and $\tilde{R}_{1t}$, the MLE of $\boldsymbol{\beta}$ is computed by using the reduced rank regression. Let

$$S_{ij.J_\perp} = \frac{1}{T} \sum_{t=1}^{T} \tilde{R}_{it} \tilde{R}_{jt}', \quad \text{for } i, j = 1, J$$

Under the null hypothesis $H_0: \boldsymbol{\alpha} = J\psi$, the MLE $\tilde{\boldsymbol{\beta}}$ is computed by solving the equation

$$|\rho S_{11.J_\perp} - S_{1J.J_\perp} S^{-1}_{JJ.J_\perp} S_{J1.J_\perp}| = 0$$

Then $\tilde{\boldsymbol{\beta}} = (v_1, \ldots, v_r)$, where the eigenvectors correspond to the r largest eigenvalues and are normalized such that $\tilde{\boldsymbol{\beta}}' S_{11.J_\perp} \tilde{\boldsymbol{\beta}} = I_r$; $\tilde{\boldsymbol{\alpha}} = J S_{J1.J_\perp} \tilde{\boldsymbol{\beta}}$. The likelihood ratio test for $H_0: \boldsymbol{\alpha} = J\psi$ is

$$T \sum_{i=1}^{r} \log\{(1-\rho_i)/(1-\lambda_i)\} \xrightarrow{d} \chi^2_{r(k-m)}$$

See Theorem 6.1 in Johansen and Juselius (1990) for more details.

The test of weak exogeneity of $\mathbf{y}_{2t}$ is a special case of the test $\boldsymbol{\alpha} = J\psi$, considering $J = (I_{k_1}, 0)'$. Consider the previous example with four variables (m_t, y_t, i^b_t, i^d_t). If $r = 1$, you formulate the weak exogeneity of (y_t, i^b_t, i^d_t) for m_t as $J = [0, I_3]'$ and the weak exogeneity of i^d_t for (m_t, y_t, i^b_t) as $J = [I_3, 0]'$.

The following statements test the weak exogeneity of other variables, assuming $r = 1$:

```
proc varmax data=simul2;
   model y1 y2 / p=2;
   cointeg rank=1 exogeneity normalize=y1;
run;
```

Figure 42.75 shows that each variable is not the weak exogeneity of other variable.

Figure 42.75 Testing of Weak Exogeneity (EXOGENEITY Option)

The VARMAX Procedure

Testing Weak Exogeneity of Each Variables

Variable	DF	Chi-Square	Pr > ChiSq
y1	1	53.46	<.0001
y2	1	8.76	0.0031

General Tests and Restrictions on Parameters

The previous sections discuss some special forms of tests on $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$, namely the long-run relations that are expressed in the form $H_0: \boldsymbol{\beta} = \mathbf{H}\boldsymbol{\phi}$, the weak exogeneity test, and the null hypotheses on $\boldsymbol{\alpha}$ in the form $H_0: \boldsymbol{\alpha} = \mathbf{J}\boldsymbol{\psi}$. In fact, with the help of the RESTRICT and BOUND statements, you can estimate the models that have linear restrictions on any parameters to be estimated, which means that you can implement the likelihood ratio (LR) test for any linear relationship between the parameters.

The restricted error correction model must be estimated through numerical optimization. You might need to use the NLOPTIONS statement to try different options for the optimizer and the INITIAL statement to try different starting points. This is essentially important because the $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are usually not identifiable.

You can also use the TEST statement to apply the Wald test for any linear relationships between parameters that are not long-run . Even more, you can test the constraints on $\Pi(= \boldsymbol{\alpha}\boldsymbol{\beta}')$ and $\boldsymbol{\delta}_0(= \boldsymbol{\alpha}\boldsymbol{\beta}_0)$ in Case 2 or $\boldsymbol{\delta}_1(= \boldsymbol{\alpha}\boldsymbol{\beta}_1)$ in Case 4 when the constant term or linear trend is restricted to the error correction term.

For more information and examples, see the section "Example 42.3: Analysis of Restricted Cointegrated Systems" on page 3055.

Forecasting of the VECM

Consider the cointegrated moving-average representation of the differenced process of $\mathbf{y}_t$

$$\Delta \mathbf{y}_t = \boldsymbol{\delta} + \Psi(B)\boldsymbol{\epsilon}_t$$

Assume that $\mathbf{y}_0 = 0$. The linear process $\mathbf{y}_t$ can be written as

$$\mathbf{y}_t = \boldsymbol{\delta} t + \sum_{i=1}^{t} \sum_{j=0}^{t-i} \Psi_j \boldsymbol{\epsilon}_i$$

Therefore, for any $l > 0$,

$$\mathbf{y}_{t+l} = \boldsymbol{\delta}(t+l) + \sum_{i=1}^{t} \sum_{j=0}^{t+l-i} \Psi_j \boldsymbol{\epsilon}_i + \sum_{i=1}^{l} \sum_{j=0}^{l-i} \Psi_j \boldsymbol{\epsilon}_{t+i}$$

The l-step-ahead forecast is derived from the preceding equation:

$$\mathbf{y}_{t+l|t} = (t+l) + \sum_{i=1}^{t} \sum_{j=0}^{t+l-i} \Psi_j \boldsymbol{\epsilon}_i$$

Note that

$$\lim_{l \to \infty} \boldsymbol{\beta}' \mathbf{y}_{t+l|t} = 0$$

since $\lim_{l\to\infty} \sum_{j=0}^{t+l-i} \Psi_j = \Psi(1)$ and $\boldsymbol{\beta}'\Psi(1) = 0$. The long-run forecast of the cointegrated system shows that the cointegrated relationship holds, although there might exist some deviations from the equilibrium status in the short-run. The covariance matrix of the predict error $\mathbf{e}_{t+l|t} = \mathbf{y}_{t+l} - \mathbf{y}_{t+l|t}$ is

$$\Sigma(l) = \sum_{i=1}^{l} [(\sum_{j=0}^{l-i} \Psi_j) \Sigma (\sum_{j=0}^{l-i} \Psi_j')]$$

When the linear process is represented as a VECM(p) model, you can obtain

$$\Delta \mathbf{y}_t = \Pi \mathbf{y}_{t-1} + \sum_{j=1}^{p-1} \Phi_j^* \Delta \mathbf{y}_{t-j} + \boldsymbol{\delta} + \boldsymbol{\epsilon}_t$$

The transition equation is defined as

$$\mathbf{z}_t = F \mathbf{z}_{t-1} + \mathbf{e}_t$$

where $\mathbf{z}_t = (\mathbf{y}'_{t-1}, \Delta\mathbf{y}'_t, \Delta\mathbf{y}'_{t-1}, \cdots, \Delta\mathbf{y}'_{t-p+2})'$ is a state vector and the transition matrix is

$$F = \begin{bmatrix} I_k & I_k & 0 & \cdots & 0 \\ \Pi & (\Pi + \Phi_1^*) & \Phi_2^* & \cdots & \Phi_{p-1}^* \\ 0 & I_k & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & I_k & 0 \end{bmatrix}$$

where 0 is a $k \times k$ zero matrix. The observation equation can be written

$$\mathbf{y}_t = \boldsymbol{\delta} t + H\mathbf{z}_t$$

where $H = [I_k, I_k, 0, \ldots, 0]$.

The l-step-ahead forecast is computed as

$$\mathbf{y}_{t+l|t} = \boldsymbol{\delta}(t+l) + HF^l\mathbf{z}_t$$

Cointegration with Exogenous Variables

The error correction model with exogenous variables can be written as follows:

$$\Delta\mathbf{y}_t = \boldsymbol{\alpha}\boldsymbol{\beta}'\mathbf{y}_{t-1} + \sum_{i=1}^{p-1}\Phi_i^*\Delta\mathbf{y}_{t-i} + AD_t + \sum_{i=0}^{s}\Theta_i^*\mathbf{x}_{t-i} + \boldsymbol{\epsilon}_t$$

The following statements demonstrate how to fit VECMX(p, s), where $p = 2$ and $s = 1$ from the P=2 and XLAG=1 options:

```
proc varmax data=simul3;
   model y1 y2 = x1 / p=2 xlag=1;
   cointeg rank=1;
run;
```

The following statements demonstrate how to BVECMX(2,1):

```
proc varmax data=simul3;
   model y1 y2 = x1 / p=2 xlag=1
                      prior=(lambda=0.9 theta=0.1);
   cointeg rank=1;
run;
```

I(2) Model

The VARX(p,s) model can be written in the error correction form:

$$\Delta\mathbf{y}_t = \boldsymbol{\alpha}\boldsymbol{\beta}'\mathbf{y}_{t-1} + \sum_{i=1}^{p-1}\Phi_i^*\Delta\mathbf{y}_{t-i} + AD_t + \sum_{i=0}^{s}\Theta_i^*\mathbf{x}_{t-i} + \boldsymbol{\epsilon}_t$$

Let $\Phi^* = I_k - \sum_{i=1}^{p-1} \Phi_i^*$.

If $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ have full-rank r, and $rank(\boldsymbol{\alpha}'_\perp \Phi^* \boldsymbol{\beta}_\perp) = k - r$, then $\mathbf{y}_t$ is an $I(1)$ process.

If the condition $rank(\boldsymbol{\alpha}'_\perp \Phi^* \boldsymbol{\beta}_\perp) = k - r$ fails and $\boldsymbol{\alpha}'_\perp \Phi^* \boldsymbol{\beta}_\perp$ has reduced-rank $\boldsymbol{\alpha}'_\perp \Phi^* \boldsymbol{\beta}_\perp = \boldsymbol{\xi}\boldsymbol{\eta}'$ where $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ are $(k-r) \times s$ matrices with $s \leq k - r$, then $\boldsymbol{\alpha}_\perp$ and $\boldsymbol{\beta}_\perp$ are defined as $k \times (k-r)$ matrices of full rank such that $\boldsymbol{\alpha}'\boldsymbol{\alpha}_\perp = 0$ and $\boldsymbol{\beta}'\boldsymbol{\beta}_\perp = 0$.

If $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ have full-rank s, then the process $\mathbf{y}_t$ is $I(2)$, which has the implication of $I(2)$ model for the moving-average representation.

$$\mathbf{y}_t = B_0 + B_1 t + C_2 \sum_{j=1}^{t} \sum_{i=1}^{j} \boldsymbol{\epsilon}_i + C_1 \sum_{i=1}^{t} \boldsymbol{\epsilon}_i + C_0(B)\boldsymbol{\epsilon}_t$$

The matrices C_1, C_2, and $C_0(B)$ are determined by the cointegration properties of the process, and B_0 and B_1 are determined by the initial values. For details, see Johansen (1995b).

The implication of the $I(2)$ model for the autoregressive representation is given by

$$\Delta^2 \mathbf{y}_t = \Pi \mathbf{y}_{t-1} - \Phi^* \Delta \mathbf{y}_{t-1} + \sum_{i=1}^{p-2} \Psi_i \Delta^2 \mathbf{y}_{t-i} + AD_t + \sum_{i=0}^{s} \Theta_i^* \mathbf{x}_{t-i} + \boldsymbol{\epsilon}_t$$

where $\Psi_i = -\sum_{j=i+1}^{p-1} \Phi_i^*$ and $\Phi^* = I_k - \sum_{i=1}^{p-1} \Phi_i^*$.

Test for I(2)

The $I(2)$ cointegrated model is given by the following parameter restrictions:

$$H_{r,s}: \Pi = \boldsymbol{\alpha}\boldsymbol{\beta}' \text{ and } \boldsymbol{\alpha}'_\perp \Phi^* \boldsymbol{\beta}_\perp = \boldsymbol{\xi}\boldsymbol{\eta}'$$

where $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ are $(k-r) \times s$ matrices with $0 \leq s \leq k - r$. Let H_r^0 represent the $I(1)$ model where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ have full-rank r, let $H_{r,s}^0$ represent the $I(2)$ model where $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ have full-rank s, and let $H_{r,s}$ represent the $I(2)$ model where $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ have rank $\leq s$. The following table shows the relation between the $I(1)$ models and the $I(2)$ models.

Table 42.6 Relation between the $I(1)$ and $I(2)$ Models

	$I(2)$										$I(1)$
$r \backslash k-r-s$	k		$k-1$		$\cdots$		1				
0	H_{00}	$\subset$	H_{01}	$\subset$	$\cdots$	$\subset$	$H_{0,k-1}$	$\subset$	H_{0k}	$=$	H_0^0
1			H_{10}	$\subset$	$\cdots$	$\subset$	$H_{1,k-2}$	$\subset$	$H_{1,k-1}$	$=$	H_1^0
$\vdots$							$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$k-1$							$H_{k-1,0}$	$\subset$	$H_{k-1,1}$	$=$	H_{k-1}^0

Johansen (1995b) proposed the two-step procedure to analyze the $I(2)$ model. In the first step, the values of $(r, \boldsymbol{\alpha}, \boldsymbol{\beta})$ are estimated using the reduced rank regression analysis, performing the regression analysis $\Delta^2 \mathbf{y}_t$,

$\Delta \mathbf{y}_{t-1}$, and $\mathbf{y}_{t-1}$ on $\Delta^2 \mathbf{y}_{t-1}, \ldots, \Delta^2 \mathbf{y}_{t-p+2}$, and D_t. This gives residuals R_{0t}, R_{1t}, and R_{2t}, and residual product moment matrices

$$M_{ij} = \frac{1}{T} \sum_{t=1}^{T} R_{it} R'_{jt} \quad \text{for } i, j = 0, 1, 2$$

Perform the reduced rank regression analysis $\Delta^2 \mathbf{y}_t$ on $\mathbf{y}_{t-1}$ corrected for $\Delta \mathbf{y}_{t-1}$, $\Delta^2 \mathbf{y}_{t-1}, \ldots, \Delta^2 \mathbf{y}_{t-p+2}$, and D_t, and solve the eigenvalue problem of the equation

$$|\lambda M_{22.1} - M_{20.1} M_{00.1}^{-1} M_{02.1}| = 0$$

where $M_{ij.1} = M_{ij} - M_{i1} M_{11}^{-1} M_{1j}$ for $i, j = 0, 2$.

In the second step, if $(r, \boldsymbol{\alpha}, \boldsymbol{\beta})$ are known, the values of $(s, \boldsymbol{\xi}, \boldsymbol{\eta})$ are determined using the reduced rank regression analysis, regressing $\hat{\boldsymbol{\alpha}}'_{\perp} \Delta^2 \mathbf{y}_t$ on $\hat{\boldsymbol{\beta}}'_{\perp} \Delta \mathbf{y}_{t-1}$ corrected for $\Delta^2 \mathbf{y}_{t-1}, \ldots, \Delta^2 \mathbf{y}_{t-p+2}$, D_t, and $\hat{\boldsymbol{\beta}}' \Delta \mathbf{y}_{t-1}$.

The reduced rank regression analysis reduces to the solution of an eigenvalue problem for the equation

$$|\rho M_{\boldsymbol{\beta}_\perp \boldsymbol{\beta}_\perp . \boldsymbol{\beta}} - M_{\boldsymbol{\beta}_\perp \boldsymbol{\alpha}_\perp . \boldsymbol{\beta}} M^{-1}_{\boldsymbol{\alpha}_\perp \boldsymbol{\alpha}_\perp . \boldsymbol{\beta}} M_{\boldsymbol{\alpha}_\perp \boldsymbol{\beta}_\perp . \boldsymbol{\beta}}| = 0$$

where

$$\begin{aligned}
M_{\boldsymbol{\beta}_\perp \boldsymbol{\beta}_\perp . \boldsymbol{\beta}} &= \boldsymbol{\beta}'_\perp (M_{11} - M_{11}\boldsymbol{\beta}(\boldsymbol{\beta}' M_{11} \boldsymbol{\beta})^{-1} \boldsymbol{\beta}' M_{11}) \boldsymbol{\beta}_\perp \\
M'_{\boldsymbol{\beta}_\perp \boldsymbol{\alpha}_\perp . \boldsymbol{\beta}} &= M_{\boldsymbol{\alpha}_\perp \boldsymbol{\beta}_\perp . \boldsymbol{\beta}} = \bar{\boldsymbol{\alpha}}'_\perp (M_{01} - M_{01}\boldsymbol{\beta}(\boldsymbol{\beta}' M_{11} \boldsymbol{\beta})^{-1} \boldsymbol{\beta}' M_{11}) \boldsymbol{\beta}_\perp \\
M_{\boldsymbol{\alpha}_\perp \boldsymbol{\alpha}_\perp . \boldsymbol{\beta}} &= \bar{\boldsymbol{\alpha}}'_\perp (M_{00} - M_{01}\boldsymbol{\beta}(\boldsymbol{\beta}' M_{11} \boldsymbol{\beta})^{-1} \boldsymbol{\beta}' M_{10}) \bar{\boldsymbol{\alpha}}_\perp
\end{aligned}$$

where $\bar{\boldsymbol{\alpha}} = \boldsymbol{\alpha}(\boldsymbol{\alpha}' \boldsymbol{\alpha})^{-1}$.

The solution gives eigenvalues $1 > \rho_1 > \cdots > \rho_s > 0$ and eigenvectors $(v_1, \ldots, v_s)$. Then, the ML estimators are

$$\begin{aligned}
\hat{\boldsymbol{\eta}} &= (v_1, \ldots, v_s) \\
\hat{\boldsymbol{\xi}} &= M_{\boldsymbol{\alpha}_\perp \boldsymbol{\beta}_\perp . \boldsymbol{\beta}} \hat{\eta}
\end{aligned}$$

The likelihood ratio test for the reduced rank model $H_{r,s}$ with rank $\le s$ in the model $H_{r,k-r} = H_r^0$ is given by

$$Q_{r,s} = -T \sum_{i=s+1}^{k-r} \log(1 - \rho_i), \quad s = 0, \ldots, k - r - 1$$

The following statements simulate an I(2) process and compute the rank test to test for cointegrated order 2:

```
proc iml;
   alpha  = { 1, 1};    * alphaOrthogonal = { 1, -1};
   beta   = { 1, -0.5}; * betaOrthogonal  = { 1, 2};
   * alphaOrthogonal' * phiStar * betaOrthogonal = 0;
   phiStar = { 1 0, 0 0.5};
   A1 = 2 * I(2) + alpha * beta` - phiStar;
```

```
   A2 = phiStar - I(2);
   phi = A1 // A2;
   sig = I(2);
   /* to simulate the vector time series */
   call varmasim(y,phi) sigma=sig n=200 seed=2;
   cn = {'y1' 'y2'};
   create simul4 from y[colname=cn];
   append from y;
   close;
quit;

proc varmax data=simul4;
   model y1 y2 /noint p=2 cointtest=(johansen=(iorder=2));
run;
```

The last two columns in Figure 42.76 explain the cointegration rank test with integrated order 1. For a specified significance level, such as 5%, the output indicates that the null hypothesis that the series are not cointegrated (H0: $r = 0$) is rejected, because the p-value for this test, shown in the column Pr > Trace of I(1), is less than 0.05. The results also indicate that the null hypothesis that there is a cointegrated relationship with cointegration rank 1 (H0: $r = 1$) cannot be rejected at the 5% significance level, because the p-value for the test statistic, 0.7961, is greater than 0.05. Because of this latter result, the rows in the table that are associated with $r = 1$ are further examined. The test statistic, 0.0257, tests the null hypothesis that the series are cointegrated order 2. The p-value that is associated with this test is 0.8955, which indicates that the null hypothesis cannot be rejected at the 5% significance level.

Figure 42.76 Cointegrated I(2) Test (IORDER= Option)

The VARMAX Procedure

Cointegration Rank Test for I(2)

r\k-r-s	2	1	Trace of I(1)	Pr > Trace of I(1)
0	575.3784	1.1833	215.3011	<.0001
Pr > Trace of I(2)	0.0000	0.3223		
1		0.0257	0.0986	0.7961
Pr > Trace of I(2)		0.8955		

Vector Error Correction Model in ARMA Form

The vector error correction model in ARMA form (the VEC-ARMA model) introduces MA terms and is defined as follows:

$$\Delta \mathbf{y}_t = \boldsymbol{\alpha}\boldsymbol{\beta}'\mathbf{y}_{t-1} + \sum_{i=1}^{p-1} \Phi_i^* \Delta \mathbf{y}_{t-i} + \boldsymbol{\epsilon}_t - \sum_{i=1}^{q} \Theta_i \boldsymbol{\epsilon}_{t-i}$$

The determined terms and the exogenous variables can also be introduced into the model. Similar to the VECM that has only AR terms, the constant term is constrained in the error correction term in Case 2 and the linear trend term is similarly constrained in Case 4.

The model is estimated through the maximum likelihood method. The log likelihood of the model is defined as

$$\ell = -\frac{T}{2}\log|\Sigma| - \frac{1}{2}\sum_{t=1}^{T}\mathbf{e}_t'\Sigma^{-1}\mathbf{e}_t$$

where

$$\mathbf{e}_t = \Delta\mathbf{y}_t - \boldsymbol{\alpha}\boldsymbol{\beta}'\mathbf{y}_{t-1} - \sum_{i=1}^{p-1}\Phi_i^*\Delta\mathbf{y}_{t-i} + \sum_{i=1}^{q}\Theta_i\mathbf{e}_{t-i}$$

conditional on the presample $\{y_0, \ldots, y_{1-p}\}$, and $e_s = 0, s \leq 0$.

You can specify a VEC-ARMA(2,1) model with cointegration rank 2 on the three-dimensional time series by the following statements:

```
model y1-y3 / p=2 q=1;
cointeg rank=2;
```

For more information about modeling the cointegrated VARMA processes, see Lütkepohl (2007, Chapter 14).

Multivariate GARCH Modeling

Stochastic volatility modeling is important in many areas, particularly in finance. To study the volatility of time series, GARCH models are widely used because they provide a good approach to conditional variance modeling.

BEKK Representation

Engle and Kroner (1995) propose a general multivariate GARCH model and call it a BEKK representation. Let $\mathcal{F}(t-1)$ be the sigma field generated by the past values of $\boldsymbol{\epsilon}_t$, and let H_t be the conditional covariance matrix of the k-dimensional random vector $\boldsymbol{\epsilon}_t$. Let H_t be measurable with respect to $\mathcal{F}(t-1)$; then the multivariate GARCH model can be written as

$$\begin{aligned}\boldsymbol{\epsilon}_t|\mathcal{F}(t-1) &\sim N(0, H_t)\\ H_t &= C + \sum_{i=1}^{q}A_i'\boldsymbol{\epsilon}_{t-i}\boldsymbol{\epsilon}_{t-i}'A_i + \sum_{i=1}^{p}G_i'H_{t-i}G_i\end{aligned}$$

where C, A_i and G_i are $k \times k$ parameter matrices.

Consider the bivariate GARCH(1,1) model

$$\begin{aligned}H_t &= \begin{bmatrix} c_{11} & c_{12}\\ c_{12} & c_{22}\end{bmatrix} + \begin{bmatrix} a_{11} & a_{12}\\ a_{21} & a_{22}\end{bmatrix}'\begin{bmatrix} \epsilon_{1,t-1}^2 & \epsilon_{1,t-1}\epsilon_{2,t-1}\\ \epsilon_{2,t-1}\epsilon_{1,t-1} & \epsilon_{2,t-1}^2\end{bmatrix}\begin{bmatrix} a_{11} & a_{12}\\ a_{21} & a_{22}\end{bmatrix}\\ &\quad + \begin{bmatrix} g_{11} & g_{12}\\ g_{21} & g_{22}\end{bmatrix}'H_{t-1}\begin{bmatrix} g_{11} & g_{12}\\ g_{21} & g_{22}\end{bmatrix}\end{aligned}$$

or, representing the univariate model,

$$
\begin{aligned}
h_{11,t} &= c_{11} + a_{11}^2\epsilon_{1,t-1}^2 + 2a_{11}a_{21}\epsilon_{1,t-1}\epsilon_{2,t-1} + a_{21}^2\epsilon_{2,t-1}^2 \\
&\quad + g_{11}^2 h_{11,t-1} + 2g_{11}g_{21}h_{12,t-1} + g_{21}^2 h_{22,t-1} \\
h_{12,t} &= c_{12} + a_{11}a_{12}\epsilon_{1,t-1}^2 + (a_{21}a_{12} + a_{11}a_{22})\epsilon_{1,t-1}\epsilon_{2,t-1} + a_{21}a_{22}\epsilon_{2,t-1}^2 \\
&\quad + g_{11}g_{12}h_{11,t-1} + (g_{21}g_{12} + g_{11}g_{22})h_{12,t-1} + g_{21}g_{22}h_{22,t-1} \\
h_{22,t} &= c_{22} + a_{12}^2\epsilon_{1,t-1}^2 + 2a_{12}a_{22}\epsilon_{1,t-1}\epsilon_{2,t-1} + a_{22}^2\epsilon_{2,t-1}^2 \\
&\quad + g_{12}^2 h_{11,t-1} + 2g_{12}g_{22}h_{12,t-1} + g_{22}^2 h_{22,t-1}
\end{aligned}
$$

For the BEKK representation of the bivariate GARCH(1,1) model, the SAS statements are

```
model y1 y2;
garch q=1 p=1 form=bekk;
```

The multistep forecast of the conditional covariance matrix, $H_{t+h|t}, h = 1, 2, \ldots$, is obtained recursively through the formula

$$
H_{t+h|t} = C + \sum_{i=1}^{h-1} A_i' H_{t+h-i|t} A_i + \sum_{i=h}^{q} A_i' \boldsymbol{\epsilon}_{t+h-i} \boldsymbol{\epsilon}_{t+h-i}' A_i + \sum_{i=1}^{p} G_i' H_{t+h-i|t} G_i
$$

where $H_{s|t} = H_s$ for $s \leq t$.

CCC Representation

Bollerslev (1990) proposes a multivariate GARCH model with time-varying conditional variances and covariances but constant conditional correlations.

The conditional covariance matrix H_t consists of

$$
H_t = D_t S D_t
$$

where D_t is a $k \times k$ stochastic diagonal matrix with element $\sigma_{i,t}$ and S is a $k \times k$ time-invariant correlation matrix with the typical element s_{ij}.

The element of H_t is

$$
h_{ij,t} = s_{ij}\sigma_{i,t}\sigma_{j,t} \quad i, j = 1, \ldots, k
$$

Note that $h_{ii,t} = \sigma_{i,t}^2, i = 1, \ldots, k$.

If you specify CORRCONSTANT=EXPECT, the element s_{ij} of the time-invariant correlation matrix S is

$$
s_{ij} = \frac{1}{T}\sum_{t=1}^{T} \frac{\boldsymbol{\epsilon}_{i,t}}{\sqrt{h_{ii,t}}} \frac{\boldsymbol{\epsilon}_{j,t}}{\sqrt{h_{jj,t}}}
$$

where T is the sample size.

By default, or when you specify SUBFORM=GARCH, $\sigma^2_{i,t}$ follows a univariate GARCH process,

$$\sigma^2_{i,t} = c_i + \sum_{l=1}^{q} a_{ii,l}\epsilon^2_{i,t-l} + \sum_{l=1}^{p} g_{ii,l}\sigma^2_{i,t-1} \quad i = 1, \ldots, k$$

As shown in many empirical studies, positive and negative innovations have different impacts on future volatility. There is a long list of variations of univariate GARCH models that consider the asymmetricity. Four typical variations follow:

- Exponential GARCH (EGARCH) model (Nelson and Cao 1992)
- Quadratic GARCH (QGARCH) model (Engle and Ng 1993)
- Threshold GARCH (TGARCH) model (Glosten, Jaganathan, and Runkle 1993; Zakoian 1994)
- Power GARCH (PGARCH) model (Ding, Granger, and Engle 1993)

For more information about the asymmetric GARCH models, see Engle and Ng (1993). You can choose the type of GARCH model of interest by specifying the SUBFORM= option.

The EGARCH model was proposed by Nelson (1991). Nelson and Cao (1992) argue that the nonnegativity constraints in the GARCH model are too restrictive. The GARCH model, implicitly or explicitly, imposes the nonnegative constraints on the parameters, whereas these parameters have no restrictions in the EGARCH model. In the EGARCH model, the conditional variance is an asymmetric function of lagged disturbances,

$$\ln(\sigma^2_{i,t}) = c_i + \sum_{l=1}^{q} a_{ii,l}\left(b_{ii,l}\frac{\epsilon_{i,t-l}}{\sigma_{i,t-l}} + |\frac{\epsilon_{i,t-l}}{\sigma_{i,t-l}}| - \sqrt{\frac{2}{\pi}}\right) + \sum_{l=1}^{p} g_{ii,l}\ln(\sigma^2_{i,t-l}) \quad i = 1, \ldots, k$$

In the QGARCH model, the lagged errors' centers are shifted from zero to some constant values,

$$\sigma^2_{i,t} = c_i + \sum_{l=1}^{q} a_{ii,l}(\epsilon_{i,t-l} - b_{ii,l})^2 + \sum_{l=1}^{p} g_{ii,l}\sigma^2_{i,t-1} \quad i = 1, \ldots, k$$

In the TGARCH model, each lagged squared error has an extra slope coefficient,

$$\sigma^2_{i,t} = c_i + \sum_{l=1}^{q} (a_{ii,l} + 1_{\epsilon_{i,t-l}<0} b_{ii,l})\epsilon^2_{i,t-l} + \sum_{l=1}^{p} g_{ii,l}\sigma^2_{i,t-1} \quad i = 1, \ldots, k$$

where the indicator function $1_{\epsilon_{i,t}<0}$ is one if $\epsilon_{i,t} < 0$ and zero otherwise.

The PGARCH model not only considers the asymmetric effect but also provides a way to model the long memory property in the volatility,

$$\sigma_{i,t}^{2\lambda_i} = c_i + \sum_{l=1}^{q} a_{ii,l}(|\epsilon_{i,t-l}| - b_{ii,l}\epsilon_{i,t-l})^{2\lambda_i} + \sum_{l=1}^{p} g_{ii,l}\sigma_{i,t-1}^{2\lambda_i} \quad i = 1, \ldots, k$$

where $\lambda_i > 0$ and $|b_{ii,l}| \leq 1, l = 1, ..., q, i = 1, \ldots, k$.

Note that the implemented TGARCH model is also well known as GJR-GARCH (Glosten, Jaganathan, and Runkle 1993), which is similar to the threshold GARCH model proposed by Zakoian (1994) but not exactly the same. In Zakoian's model, the conditional standard deviation is a linear function of the past values of the white noise. Zakoian's model can be regarded as a special case of the PGARCH model when $\lambda_i = 1/2$.

The following formulas are recursively implemented to obtain the multistep forecast of conditional error variance $\sigma_{i,t+h|t}^2, i = 1, \ldots, k$ and $h = 1, 2, \ldots$:

- for the GARCH(p, q) model:

$$\sigma_{i,t+h|t}^2 = c_i + \sum_{l=1}^{h-1} a_{ii,l}\sigma_{i,t+h-l|t}^2 + \sum_{l=h}^{q} a_{ii,l}\epsilon_{i,t+h-l}^2 + \sum_{l=1}^{p} g_{ii,l}\sigma_{i,t+h-1|t}^2$$

- for the EGARCH(p, q) model:

$$\ln(\sigma_{i,t+h|t}^2) = c_i + \sum_{l=h}^{q} a_{ii,l}\left(b_{ii,l}\frac{\epsilon_{i,t+h-l}}{\sigma_{i,t+h-l}} + |\frac{\epsilon_{i,t+h-l}}{\sigma_{i,t+h-l}}| - \sqrt{\frac{2}{\pi}}\right) + \sum_{l=1}^{p} g_{ii,l}\ln(\sigma_{i,t+h-l|t}^2)$$

- for the QGARCH(p, q) model:

$$\begin{aligned}\sigma_{i,t+h|t}^2 = {} & c_i + \sum_{l=1}^{h-1} a_{ii,l}(\sigma_{i,t+h-l|t}^2 + b_{ii,l}^2) + \sum_{l=h}^{q} a_{ii,l}(\epsilon_{i,t+h-l} - b_{ii,l})^2 \\ & + \sum_{l=1}^{p} g_{ii,l}\sigma_{i,t+h-1|t}^2\end{aligned}$$

- for the TGARCH(p, q) model:

$$\begin{aligned}\sigma_{i,t+h|t}^2 = {} & c_i + \sum_{l=1}^{h-1}(a_{ii,l} + b_{ii,l}/2)\sigma_{i,t+h-1|t}^2 + \sum_{l=h}^{q}(a_{ii,l} + 1_{\epsilon_{i,t-l}<0}b_{ii,l})\epsilon_{i,t-l}^2 \\ & + \sum_{l=1}^{p} g_{ii,l}\sigma_{i,t+h-1|t}^2\end{aligned}$$

- for the PGARCH(p, q) model:

$$\begin{aligned}\sigma_{i,t+h|t}^{2\lambda_i} &= c_i + \sum_{l=1}^{h-1} a_{ii,l}((1+b_{ii,l})^{2\lambda_i} + (1-b_{ii,l})^{2\lambda_i})\sigma_{i,t+h-l|t}^{2\lambda_i}/2 \\ &\quad + \sum_{l=h}^{q} a_{ii,l}(|\epsilon_{i,t-l}| - b_{ii,l}\epsilon_{i,t-l})^{2\lambda_i} + \sum_{l=1}^{p} g_{ii,l}\sigma_{i,t+h-1|t}^{2\lambda_i}\end{aligned}$$

In the preceding equations, $\sigma_{i,s|t} = \sigma_{i,s}$ for $s \le t$. Then, the multistep forecast of conditional covariance matrix $H_{t+h|t}, h = 1, 2, \ldots$, is calculated by

$$H_{t+h|t} = D_{t+h|t} S D_{t+h|t}$$

where $D_{t+h|t}$ is the diagonal matrix with element $\sigma_{i,t+h|t}, i = 1, \ldots, k$.

DCC Representation

Engle (2002) proposes a parsimonious parametric multivariate GARCH model that has time-varying conditional covariances and correlations.

The conditional covariance matrix H_t consists of

$$H_t = D_t \Gamma_t D_t$$

where D_t is a $k \times k$ stochastic diagonal matrix with the element $\sigma_{i,t}$ and Γ_t is a $k \times k$ time-varying matrix with the typical element $\rho_{ij,t}$.

The element of H_t is

$$h_{ij,t} = \rho_{ij,t}\sigma_{i,t}\sigma_{j,t} \quad i, j = 1, \ldots, k$$

Note that $h_{ii,t} = \sigma_{i,t}^2, i = 1, \ldots, k$.

As in the CCC GARCH model, you can choose the type of GARCH model of interest by specifying the SUBFORM= option.

In the GARCH model,

$$\sigma_{i,t}^2 = c_i + \sum_{l=1}^{q} a_{ii,l}\epsilon_{i,t-l}^2 + \sum_{l=1}^{p} g_{ii,l}\sigma_{i,t-1}^2 \quad i = 1, \ldots, k$$

In the EGARCH model, the conditional variance is an asymmetric function of lagged disturbances,

$$\ln(\sigma_{i,t}^2) = c_i + \sum_{l=1}^{q} a_{ii,l}\left(b_{ii,l}\frac{\epsilon_{i,t-l}}{\sigma_{i,t-l}} + |\frac{\epsilon_{i,t-l}}{\sigma_{i,t-l}}| - \sqrt{\frac{2}{\pi}}\right) + \sum_{l=1}^{p} g_{ii,l}\ln(\sigma_{i,t-l}^2) \quad i = 1, \ldots, k$$

In the QGARCH model, the lagged errors' centers are shifted from zero to some constant values,

$$\sigma_{i,t}^2 = c_i + \sum_{l=1}^{q} a_{ii,l}(\epsilon_{i,t-l} - b_{ii,l})^2 + \sum_{l=1}^{p} g_{ii,l}\sigma_{i,t-1}^2 \quad i = 1, \ldots, k$$

In the TGARCH model, each lagged squared error has an extra slope coefficient,

$$\sigma_{i,t}^2 = c_i + \sum_{l=1}^{q} (a_{ii,l} + 1_{\epsilon_{i,t-l}<0} b_{ii,l})\epsilon_{i,t-l}^2 + \sum_{l=1}^{p} g_{ii,l}\sigma_{i,t-1}^2 \quad i = 1, \ldots, k$$

where the indicator function $1_{\epsilon_{i,t}<0}$ is one if $\epsilon_{i,t} < 0$ and zero otherwise.

The PGARCH model not only considers the asymmetric effect but also provides another way to model the long memory property in the volatility,

$$\sigma_{i,t}^{2\lambda_i} = c_i + \sum_{l=1}^{q} a_{ii,l}(|\epsilon_{i,t-l}| - b_{ii,l}\epsilon_{i,t-l})^{2\lambda_i} + \sum_{l=1}^{p} g_{ii,l}\sigma_{i,t-1}^{2\lambda_i} \quad i = 1, \ldots, k$$

where $\lambda_i > 0$ and $|b_{ii,l}| \leq 1, l = 1, \ldots, q; i = 1, \ldots, k$.

The conditional correlation estimator $\rho_{ij,t}$ is

$$\begin{aligned} \rho_{ij,t} &= \frac{q_{ij,t}}{\sqrt{q_{ii,t}q_{jj,t}}} \quad i, j = 1, \ldots, k \\ q_{ij,t} &= (1 - \alpha - \beta)s_{ij} + \alpha \frac{\epsilon_{i,t-1}}{\sigma_{i,t-1}} \frac{\epsilon_{j,t-1}}{\sigma_{j,t-1}} + \beta q_{ij,t-1} \end{aligned}$$

where s_{ij} is the element of S, the unconditional correlation matrix.

If you specify CORRCONSTANT=EXPECT, the element s_{ij} of the unconditional correlation matrix S is

$$s_{ij} = \frac{1}{T} \sum_{t=1}^{T} \frac{\epsilon_{i,t}}{\sigma_{i,t}} \frac{\epsilon_{j,t}}{\sigma_{j,t}}$$

where T is the sample size.

As shown in the CCC GARCH models, the following formulas are recursively implemented to obtain the multistep forecast of conditional error variance $\sigma_{i,t+h|t}^2, i = 1, \ldots, k$ and $h = 1, 2, \ldots$:

- for the GARCH(p, q) model:

$$\sigma_{i,t+h|t}^2 = c_i + \sum_{l=1}^{h-1} a_{ii,l}\sigma_{i,t+h-l|t}^2 + \sum_{l=h}^{q} a_{ii,l}\epsilon_{i,t+h-l}^2 + \sum_{l=1}^{p} g_{ii,l}\sigma_{i,t+h-1|t}^2$$

- for the EGARCH(p, q) model:

$$\ln(\sigma^2_{i,t+h|t}) = c_i + \sum_{l=h}^{q} a_{ii,l}\left(b_{ii,l}\frac{\epsilon_{i,t+h-l}}{\sigma_{i,t+h-l}} + |\frac{\epsilon_{i,t+h-l}}{\sigma_{i,t+h-l}}| - \sqrt{\frac{2}{\pi}}\right) + \sum_{l=1}^{p} g_{ii,l}\ln(\sigma^2_{i,t+h-l|t})$$

- for the QGARCH(p, q) model:

$$\begin{aligned}\sigma^2_{i,t+h|t} &= c_i + \sum_{l=1}^{h-1} a_{ii,l}(\sigma^2_{i,t+h-l|t} + b^2_{ii,l}) + \sum_{l=h}^{q} a_{ii,l}(\epsilon_{i,t+h-l} - b_{ii,l})^2 \\ &\quad + \sum_{l=1}^{p} g_{ii,l}\sigma^2_{i,t+h-1|t}\end{aligned}$$

- for the TGARCH(p, q) model:

$$\begin{aligned}\sigma^2_{i,t+h|t} &= c_i + \sum_{l=1}^{h-1}(a_{ii,l} + b_{ii,l}/2)\sigma^2_{i,t+h-1|t} + \sum_{l=h}^{q}(a_{ii,l} + 1_{\epsilon_{i,t-l}<0}b_{ii,l})\epsilon^2_{i,t-l} \\ &\quad + \sum_{l=1}^{p} g_{ii,l}\sigma^2_{i,t+h-1|t}\end{aligned}$$

- for the PGARCH(p, q) model:

$$\begin{aligned}\sigma^{2\lambda_i}_{i,t+h|t} &= c_i + \sum_{l=1}^{h-1} a_{ii,l}((1 + b_{ii,l})^{2\lambda_i} + (1 - b_{ii,l})^{2\lambda_i})\sigma^{2\lambda_i}_{i,t+h-l|t}/2 \\ &\quad + \sum_{l=h}^{q} a_{ii,l}(|\epsilon_{i,t-l}| - b_{ii,l}\epsilon_{i,t-l})^{2\lambda_i} + \sum_{l=1}^{p} g_{ii,l}\sigma^{2\lambda_i}_{i,t+h-1|t}\end{aligned}$$

In the preceding equations, $\sigma_{i,s|t} = \sigma_{i,s}$ for $s \le t$. Then, the multistep forecast of conditional covariance matrix $H_{t+h|t}, h = 1, 2, \ldots$, is calculated by

$$H_{t+h|t} = D_{t+h|t}\Gamma_{t+h|t}D_{t+h|t}$$

where $D_{t+h|t}$ is the diagonal matrix with element $\sigma_{i,t+h|t}, i = 1, \ldots, k$, and $\Gamma_{t+h|t}$ is the matrix with element $\rho_{ij,t+h|t}, i, j = 1, \ldots, k$,

$$\begin{aligned}\rho_{ij,t+h|t} &= \frac{q_{ij,t+h|t}}{\sqrt{q_{ii,t+h|t}q_{jj,t+h|t}}} \\ q_{ij,t+h|t} &= \begin{cases}(1-\alpha-\beta)s_{ij} + \alpha\frac{\epsilon_{i,t}}{\sigma_{i,t}}\frac{\epsilon_{j,t}}{\sigma_{j,t}} + \beta q_{ij,t} & h = 1 \\ (1-\alpha-\beta)s_{ij} + \alpha q_{ij,t+h-1|t} + \beta q_{ij,t+h-1|t} & h > 1\end{cases}\end{aligned}$$

Estimation of GARCH Model

The log-likelihood function of the multivariate GARCH model is written without a constant term as

$$\ell = -\frac{1}{2}\sum_{t=1}^{T}[\log|H_t| + \boldsymbol{\epsilon}_t' H_t^{-1}\boldsymbol{\epsilon}_t]$$

where $\boldsymbol{\epsilon}_t$ is calculated from the first-moment model (that is, the VARMAX model or VEC-ARMA model). The log-likelihood function is maximized by an iterative numerical method such as quasi-Newton optimization. The starting values for the regression parameters are obtained from the least squares estimates. The covariance of $\boldsymbol{\epsilon}_t$ is used as the starting value for the GARCH constant parameters, and the starting values for the other GARCH parameters are either 10^{-6} or 10^{-3}, depending on the GARCH model's representation.

Prediction of Endogenous (Dependent) Variables

In multivariate GARCH models, the optimal (minimum MSE) l-step-ahead forecast of endogenous variables $\mathbf{y}_{t+l|t}$ uses the same formula as shown in the section "Forecasting" on page 2953. However, the exogenous (independent) variables, if present, are always assumed to be nonstochastic (deterministic); that is, to predict the endogenous variables, you must specify the future values of the exogenous variables. The prediction error of the optimal l-step-ahead forecast is $\mathbf{e}_{t+l|t} = \mathbf{y}_{t+l} - \mathbf{y}_{t+l|t} = \sum_{j=0}^{l-1}\Psi_j\boldsymbol{\epsilon}_{t+l-j}$, with zero mean and covariance matrix,

$$\Sigma_t(l) = \mathrm{Cov}(\mathbf{e}_{t+l|t}) = \sum_{j=0}^{l-1}\Psi_j H_{t+l-j|t}\Psi_j'$$

where $H_{t+h|t}, h = 1,\ldots,l$, is the h-step-ahead forecast of the conditional covariance matrix. As emphasized by the subscript t, $\Sigma_t(l)$ is time-dependent. In the OUT= data set, the forecast standard errors and prediction intervals are constructed according to $\Sigma_t(l)$. If you specify the COVPE option, the prediction error covariances that are output in the CovPredictError and CovPredictErrorbyVar ODS tables are based on the time-independent formula

$$\Sigma(l) = \sum_{j=0}^{l-1}\Psi_j\Sigma\Psi_j'$$

where Σ is the unconditional covariance matrix of innovations. The decomposition of the prediction error covariances is also based on $\Sigma(l)$.

Covariance Stationarity

Define the multivariate GARCH process as

$$\mathbf{h}_t = \sum_{i=1}^{\infty} G(B)^{i-1}[\mathbf{c} + A(B)\boldsymbol{\eta}_t]$$

where $\mathbf{h}_t = \text{vec}(H_t)$, $\mathbf{c} = \text{vec}(C_0)$, and $\boldsymbol{\eta}_t = \text{vec}(\boldsymbol{\epsilon}_t \boldsymbol{\epsilon}_t')$. This representation is equivalent to a GARCH(p, q) model by the following algebra:

$$\begin{aligned}
\mathbf{h}_t &= \mathbf{c} + A(B)\boldsymbol{\eta}_t + \sum_{i=2}^{\infty} G(B)^{i-1}[\mathbf{c} + A(B)\boldsymbol{\eta}_t] \\
&= \mathbf{c} + A(B)\boldsymbol{\eta}_t + G(B)\sum_{i=1}^{\infty} G(B)^{i-1}[tmbc + A(B)\boldsymbol{\eta}_t] \\
&= \mathbf{c} + A(B)\boldsymbol{\eta}_t + G(B)\mathbf{h}_t
\end{aligned}$$

Defining $A(B) = \sum_{i=1}^{q}(A_i \otimes A_i)'B^i$ and $G(B) = \sum_{i=1}^{p}(G_i \otimes G_i)'B^i$ gives a BEKK representation.

The necessary and sufficient conditions for covariance stationarity of the multivariate GARCH process are that all the eigenvalues of $A(1) + G(1)$ are less than 1 in modulus.

An Example of a VAR(1)–ARCH(1) Model

The following DATA step simulates a bivariate vector time series to provide test data for the multivariate GARCH model:

```
data garch;
   retain seed 16587;
   esq1 = 0; esq2 = 0;
   ly1 = 0;  ly2 = 0;
   do i = 1 to 1000;
      ht = 6.25 + 0.5*esq1;
      call rannor(seed,ehat);
      e1 = sqrt(ht)*ehat;
      ht = 1.25 + 0.7*esq2;
      call rannor(seed,ehat);
      e2 = sqrt(ht)*ehat;
      y1 = 2 + 1.2*ly1 - 0.5*ly2 + e1;
      y2 = 4 + 0.6*ly1 + 0.3*ly2 + e2;
      if i>500 then output;
      esq1 = e1*e1; esq2 = e2*e2;
      ly1 = y1;  ly2 = y2;
   end;
   keep y1 y2;
run;
```

The following statements fit a VAR(1)–ARCH(1) model to the data. For a VAR-ARCH model, you specify the order of the autoregressive model with the P=1 option in the MODEL statement and the Q=1 option in the GARCH statement. In order to produce the initial and final values of parameters, the TECH=QN option is specified in the NLOPTIONS statement.

```
proc varmax data=garch;
   model y1 y2 / p=1
         print=(roots estimates diagnose);
   garch q=1;
   nloptions tech=qn;
run;
```

Figure 42.77 through Figure 42.81 show the details of this example. Figure 42.77 shows the initial values of parameters.

Figure 42.77 Start Parameter Estimates for the VAR(1)–ARCH(1) Model

The VARMAX Procedure

Optimization Start

Parameter Estimates

N	Parameter	Estimate	Gradient Objective Function
1	CONST1	2.249575	0.000082533
2	CONST2	3.902673	0.000401
3	AR1_1_1	1.231775	0.000105
4	AR1_2_1	0.576890	-0.004811
5	AR1_1_2	-0.528405	0.000617
6	AR1_2_2	0.343714	0.001811
7	GCHC1_1	9.929763	0.151293
8	GCHC1_2	0.193163	-0.014305
9	GCHC2_2	4.063245	0.370333
10	ACH1_1_1	0.001000	-0.667182
11	ACH1_2_1	0	-0.068905
12	ACH1_1_2	0	-0.734486
13	ACH1_2_2	0.001000	-3.127035

Figure 42.78 shows the final parameter estimates.

Figure 42.78 Results of Parameter Estimates for the VAR(1)–ARCH(1) Model

The VARMAX Procedure

Optimization Results

Parameter Estimates

N	Parameter	Estimate	Gradient Objective Function
1	CONST1	2.156865	0.000246
2	CONST2	4.048879	0.000105
3	AR1_1_1	1.224620	-0.001957
4	AR1_2_1	0.609651	0.000173
5	AR1_1_2	-0.534248	-0.000468
6	AR1_2_2	0.302599	-0.000375
7	GCHC1_1	8.238625	-0.000056090
8	GCHC1_2	-0.231183	-0.000021724
9	GCHC2_2	1.565459	0.000110
10	ACH1_1_1	0.374255	-0.000419
11	ACH1_2_1	0.035883	-0.000606
12	ACH1_1_2	0.057461	0.001636
13	ACH1_2_2	0.717897	-0.000149

Figure 42.79 shows the conditional variance by using the BEKK representation of the ARCH(1) model. The ARCH parameters are estimated as follows by the vectorized parameter matrices:

$$\begin{aligned}
\boldsymbol{\epsilon}_t|\mathcal{F}(t-1) &\sim N(0, H_t) \\
H_t &= \begin{bmatrix} 8.23863 & -0.23118 \\ -0.23118 & 1.56546 \end{bmatrix} \\
&+ \begin{bmatrix} 0.37426 & 0.05746 \\ 0.03588 & 0.71790 \end{bmatrix}' \boldsymbol{\epsilon}_{t-1}\boldsymbol{\epsilon}_{t-1}' \begin{bmatrix} 0.37426 & 0.05746 \\ 0.03588 & 0.71790 \end{bmatrix}
\end{aligned}$$

Figure 42.79 ARCH(1) Parameter Estimates for the VAR(1)–ARCH(1) Model

The VARMAX Procedure

Type of Model	VAR(1)-ARCH(1)
Estimation Method	Maximum Likelihood Estimation
Representation Type	BEKK

GARCH Model Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Pr > |t|
GCHC1_1	8.23863	0.72663	11.34	0.0001
GCHC1_2	-0.23118	0.21434	-1.08	0.2813
GCHC2_2	1.56546	0.19407	8.07	0.0001
ACH1_1_1	0.37426	0.07502	4.99	0.0001
ACH1_2_1	0.03588	0.06974	0.51	0.6071
ACH1_1_2	0.05746	0.02597	2.21	0.0274
ACH1_2_2	0.71790	0.06895	10.41	0.0001

Figure 42.80 shows the AR parameter estimates and their significance.

The fitted VAR(1) model with the previous conditional covariance ARCH model is written as follows:

$$\mathbf{y}_t = \begin{bmatrix} 2.15687 \\ 4.04888 \end{bmatrix} + \begin{bmatrix} 1.22462 & -0.53425 \\ 0.60965 & 0.30260 \end{bmatrix} \mathbf{y}_{t-1} + \boldsymbol{\epsilon}_t$$

Figure 42.80 VAR(1) Parameter Estimates for the VAR(1)–ARCH(1) Model

Model Parameter Estimates

Equation	Parameter	Estimate	Standard Error	t Value	Pr > |t|	Variable
y1	**CONST1**	2.15687	0.21717	9.93	0.0001	1
	AR1_1_1	1.22462	0.02542	48.17	0.0001	y1(t-1)
	AR1_1_2	-0.53425	0.02807	-19.03	0.0001	y2(t-1)
y2	**CONST2**	4.04888	0.10663	37.97	0.0001	1
	AR1_2_1	0.60965	0.01216	50.13	0.0001	y1(t-1)
	AR1_2_2	0.30260	0.01491	20.30	0.0001	y2(t-1)

Figure 42.81 shows the roots of the AR and ARCH characteristic polynomials. The eigenvalues have a modulus less than one.

Figure 42.81 Roots for the VAR(1)–ARCH(1) Model

Roots of AR Characteristic Polynomial

Index	Real	Imaginary	Modulus	Radian	Degree
1	0.76361	0.33641	0.8344	0.4150	23.7762
2	0.76361	-0.33641	0.8344	-0.4150	-23.7762

Roots of GARCH Characteristic Polynomial

Index	Real	Imaginary	Modulus	Radian	Degree
1	0.52388	0.00000	0.5239	0.0000	0.0000
2	0.26661	0.00000	0.2666	0.0000	0.0000
3	0.26661	0.00000	0.2666	0.0000	0.0000
4	0.13569	0.00000	0.1357	0.0000	0.0000

Output Data Sets

The VARMAX procedure can create the OUT=, OUTEST=, OUTHT=, and OUTSTAT= data sets. In general, if processing fails, the output is not recorded or is set to missing in the relevant output data set, and appropriate error and/or warning messages are recorded in the log.

OUT= Data Set

The OUT= data set contains the forecast values produced by the OUTPUT statement. The following output variables can be created:

- the BY variables
- the ID variable
- the MODEL statement dependent (endogenous) variables. These variables contain the actual values from the input data set.
- FOR*i*, numeric variables that contain the forecasts. The FOR*i* variables contain the forecasts for the *i*th endogenous variable in the MODEL statement list. Forecasts are one-step-ahead predictions until the end of the data or until the observation specified by the BACK= option. Multistep forecasts can be computed after that point based on the LEAD= option.
- RES*i*, numeric variables that contain the residual for the forecast of the *i*th endogenous variable in the MODEL statement list. For multistep forecast observations, the actual values are missing and the RES*i* variables contain missing values.
- STD*i*, numeric variables that contain the standard deviation for the forecast of the *i*th endogenous variable in the MODEL statement list. The values of the STD*i* variables can be used to construct univariate confidence limits for the corresponding forecasts.

- LCI*i*, numeric variables that contain the lower confidence limits for the corresponding forecasts of the *i*th endogenous variable in the MODEL statement list.
- UCI*i*, numeric variables that contain the upper confidence limits for the corresponding forecasts of the *i*th endogenous variable in the MODEL statement list.

The OUT= data set contains the values shown in Table 42.7 and Table 42.8 for a bivariate case.

Table 42.7 OUT= Data Set

Obs	ID variable	y1	FOR1	RES1	STD1	LCI1	UCI1
1	date	y_{11}	f_{11}	r_{11}	σ_{11}	l_{11}	u_{11}
2	date	y_{12}	f_{12}	r_{12}	σ_{11}	l_{12}	u_{12}
⋮							

Table 42.8 OUT= Data Set Continued

Obs	y2	FOR2	RES2	STD2	LCI2	UCI2
1	y_{21}	f_{21}	r_{21}	σ_{22}	l_{21}	u_{21}
2	y_{22}	f_{22}	r_{22}	σ_{22}	l_{22}	u_{22}
⋮						

Consider the following example:

```
proc varmax data=simul1 noprint;
   id date interval=year;
   model y1 y2 / p=1 noint;
   output out=out lead=5;
run;

proc print data=out(firstobs=98);
run;
```

The output in Figure 42.82 shows part of the results of the OUT= data set for the preceding example.

Figure 42.82 OUT= Data Set

Obs	date	y1	FOR1	RES1	STD1	LCI1	UCI1	y2	FOR2	RES2	STD2	LCI2	UCI2
98	1997	-0.58433	-0.13500	-0.44934	1.13523	-2.36001	2.09002	0.64397	-0.34932	0.99329	1.19096	-2.68357	1.98492
99	1998	-2.07170	-1.00649	-1.06522	1.13523	-3.23150	1.21853	0.35925	-0.07132	0.43057	1.19096	-2.40557	2.26292
100	1999	-3.38342	-2.58612	-0.79730	1.13523	-4.81113	-0.36111	-0.64999	-0.99354	0.34355	1.19096	-3.32779	1.34070
101	2000	.	-3.59212	.	1.13523	-5.81713	-1.36711	.	-2.09873	.	1.19096	-4.43298	0.23551
102	2001	.	-3.09448	.	1.70915	-6.44435	0.25539	.	-2.77050	.	1.47666	-5.66469	0.12369
103	2002	.	-2.17433	.	2.14472	-6.37792	2.02925	.	-2.75724	.	1.74212	-6.17173	0.65725
104	2003	.	-1.11395	.	2.43166	-5.87992	3.65203	.	-2.24943	.	2.01925	-6.20709	1.70823
105	2004	.	-0.14342	.	2.58740	-5.21463	4.92779	.	-1.47460	.	2.25169	-5.88782	2.93863

OUTEST= Data Set

The OUTEST= data set contains estimation results of the fitted model produced by the VARMAX statement. The following output variables can be created:

- BY variables
- NAME, a character variable that contains the name of the endogenous (dependent) variables or the name of the parameters for the covariance of the matrix of the parameter estimates if you specify the OUTCOV option
- TYPE, a character variable that contains the value EST for parameter estimates, the value STD for standard error of parameter estimates, and the value COV for the covariance of the matrix of the parameter estimates if you specify the OUTCOV option
- CONST, a numeric variable that contains the estimates of constant parameters and their standard errors
- SEASON_i, a numeric variable that contains the estimates of seasonal dummy parameters and their standard errors, where $i = 1, \ldots, (nseason - 1)$, and $nseason$ is based on the NSEASON= option
- LTREND, a numeric variable that contains the estimates of linear trend parameters and their standard errors
- QTREND, a numeric variable that contains the estimates of quadratic trend parameters and their standard errors
- XLl_i, numeric variables that contain the estimates of exogenous parameters and their standard errors, where l is the lag lth coefficient matrix and $i = 1, \ldots, r$, where r is the number of exogenous variables
- ARl_i, numeric variables that contain the estimates of autoregressive parameters and their standard errors, where l is the lag lth coefficient matrix and $i = 1, \ldots, k$, where k is the number of endogenous variables
- MAl_i, numeric variables that contain the estimates of moving-average parameters and their standard errors, where l is the lag lth coefficient matrix and $i = 1, \ldots, k$, where k is the number of endogenous variables
- COV_i, numeric variables that contain the estimates of the covariance of innovations parameters when the maximum likelihood method is applied, where $i = 1, \ldots, k$
- DCCAB, a numeric variable that contains the estimates of α or β in the correlation equation for DCC representation and their standard errors
- CCC_i, numeric variables that contain the estimates of the conditional constant correlation parameters for CCC representation, where $i = 2, \ldots, k$
- DCCS_i, numeric variables that contain the estimates of the unconditional correlation parameters for DCC representation, where $i = 2, \ldots, k$
- GCHC_i, numeric variables that contain the estimates of the constant parameters of the covariance matrix and their standard errors, where $i = 1, \ldots, k$ for BEKK representation, k is the number of endogenous variables, and $i = 1$ for CCC and DCC representations

- ACHl_i, numeric variables that contain the estimates of the ARCH parameters of the covariance matrix and their standard errors, where l is the lag lth coefficient matrix and $i = 1, \ldots, k$ for BEKK, CCC, and DCC representations, where k is the number of endogenous variables
- EACHl_i, numeric variables that contain the estimates of the exponential ARCH parameters of the covariance matrix and their standard errors, where l is the lag lth coefficient matrix and $i = 1, \ldots, k$ for CCC and DCC representations, where k is the number of endogenous variables
- PACHl_i, numeric variables that contain the estimates of the power ARCH parameters of the covariance matrix and their standard errors, where l is the lag lth coefficient matrix and $i = 1, \ldots, k$ for CCC and DCC representations, where k is the number of endogenous variables
- QACHl_i, numeric variables that contain the estimates of the quadratic ARCH parameters of the covariance matrix and their standard errors, where l is the lag lth coefficient matrix and $i = 1, \ldots, k$ for CCC and DCC representations, where k is the number of endogenous variables
- TACHl_i, numeric variables that contain the estimates of the threshold ARCH parameters of the covariance matrix and their standard errors, where l is the lag lth coefficient matrix and $i = 1, \ldots, k$ for CCC and DCC representations, where k is the number of endogenous variables
- GCHl_i, numeric variables that contain the estimates of the GARCH parameters of the covariance matrix and their standard errors, where l is the lag lth coefficient matrix and $i = 1, \ldots, k$ for BEKK, CCC, and DCC representations, where k is the number of endogenous variables
- LAMBDA, a numeric variable that contains the estimates of power parameters in the PGARCH model for CCC and DCC representations and their standard errors

The OUTEST= data set contains the values shown in Table 42.9 for a bivariate case.

Table 42.9 OUTEST= Data Set

Obs	NAME	TYPE	CONST	AR1_1	AR1_2	AR2_1	AR2_2
1	y1	EST	δ_1	$\phi_{1,11}$	$\phi_{1,12}$	$\phi_{2,11}$	$\phi_{2,12}$
2		STD	se(δ_1)	se($\phi_{1,11}$)	se($\phi_{1,12}$)	se($\phi_{2,11}$)	se($\phi_{2,12}$)
3	y2	EST	δ_2	$\phi_{1,21}$	$\phi_{1,22}$	$\phi_{2,21}$	$\phi_{2,22}$
4		STD	se(δ_2)	se($\phi_{1,21}$)	se($\phi_{1,22}$)	se($\phi_{2,21}$)	se($\phi_{2,22}$)

Consider the following example:

```
proc varmax data=simul2 outest=est;
   model y1 y2 / p=2 noint noprint;
   cointeg rank=1 normalize=y1;
run;

proc print data=est;
run;
```

The output in Figure 42.83 shows the results of the OUTEST= data set.

Figure 42.83 OUTEST= Data Set

Obs	NAME	TYPE	AR1_1	AR1_2	AR2_1	AR2_2	COV_1	COV_2	ALPHA1	BETA1
1	y1	EST	-0.46680	0.91295	-0.74332	-0.74621	94.7557	4.527	-0.46680	1.00000
2		STD	0.04786	0.09359	0.04526	0.04769	13.5365	10.303	0.04786	.
3	y2	EST	0.10667	-0.20862	0.40493	-0.57157	4.5268	109.570	0.10667	-1.95575
4		STD	0.05146	0.10064	0.04867	0.05128	10.3030	15.653	0.05146	.

OUTHT= Data Set

The OUTHT= data set contains predictions of conditional covariance matrices of innovations of the fitted GARCH model that the GARCH statement produces. The following output variables can be created:

- the BY variables, if BY-group processing is performed
- the ID variable, if the ID statement is specified
- Hi_j, numeric variables that contain the prediction of covariance, where $1 \leq i \leq j \leq k$, where k is the number of dependent variables

The OUTHT= data set contains the values shown in Table 42.10 for a bivariate case.

Table 42.10 OUTHT= Data Set

Obs	H1_1	H1_2	H2_2
1	h111	h121	h221
2	h112	h122	h222
:	:	:	:

The OUTHT= data set has the same number of observations as the OUT= data set. Both the OUTHT= and OUT= data sets include any observations at the beginning of the data set that are skipped because of the DIF=, DIFY=, DIFX=, P=, or XLAG= option and include the predicted observations at the end of the data set, which correspond with the LEAD= specification. If you specify an ID statement together with the OUTHT= and OUT= options, then the values of the ID variable in the two data sets correspond with one another.

Consider the following example of the OUTHT= option:

```
data garch;
   set garch;
   date = intnx( 'month', '01may1972'd, _n_-1 );
   format date yymms.;
run;

proc varmax data=garch;
   id date interval=month;
   model y1 y2 / p=1;
   garch q=1 outht=ht;
   output out=og lead=6;
run;
```

```
proc print data=og(obs=8);
   var date y1 for1 std1 lci1 uci1 y2 for2 std2 lci2 uci2;
run;

proc print data=ht(obs=8);
run;

proc print data=og(firstobs=499);
   var date y1 for1 std1 lci1 uci1 y2 for2 std2 lci2 uci2;
run;

proc print data=ht(firstobs=499);
run;
```

The output in Figure 42.84 and Figure 42.85 shows the first eight observations in the OUT= and OUTHT= data sets, respectively. The first observation is skipped in the GARCH model estimation because of the P=1 option, resulting in the missing values in the first observations in the OUT= and OUTHT= data sets. The output in Figure 42.86 and Figure 42.87 shows the last eight observations in the OUT= and OUTHT= data sets, respectively. In the OUT= data set, the standard deviations of the forecast of dependent variables are time-variant. The last six observations in OUTHT= data set are the multistep forecast of conditional covariance matrices of innovations.

Figure 42.84 First Part of OUT= Data Set

Obs	date	y1	FOR1	STD1	LCI1	UCI1	y2	FOR2	STD2	LCI2	UCI2
1	1972/05	-4.4005	.	.	.	.	1.83794	.	.	.	.
2	1972/06	-8.0533	-4.2140	3.10387	-10.2975	1.86947	1.59720	1.92227	1.92885	-1.85820	5.70274
3	1972/07	-10.8362	-8.5587	3.21511	-14.8602	-2.25720	1.51833	-0.37752	1.33100	-2.98623	2.23118
4	1972/08	-6.0179	-11.9245	2.97553	-17.7564	-6.09254	-1.57445	-2.09795	1.75464	-5.53697	1.34108
5	1972/09	-7.8272	-4.3716	3.63437	-11.4949	2.75160	-0.03774	-0.09637	1.44118	-2.92102	2.72829
6	1972/10	-8.4293	-7.4084	3.14734	-13.5770	-1.23969	-0.40424	-0.73442	1.26093	-3.20580	1.73695
7	1972/11	-7.8156	-7.9499	2.89408	-13.6222	-2.27757	0.20642	-1.21238	1.26383	-3.68944	1.26469
8	1972/12	-8.0182	-7.5245	2.87208	-13.1537	-1.89535	0.43513	-0.65343	1.61823	-3.82511	2.51825

Figure 42.85 First Part of OUTHT= Data Set

Obs	date	h1_1	h1_2	h2_2
1	1972/05	.	.	.
2	1972/06	9.6340	0.14073	3.72045
3	1972/07	10.3369	0.42643	1.77155
4	1972/08	8.8538	-1.19603	3.07876
5	1972/09	13.2086	1.36328	2.07699
6	1972/10	9.9058	-0.02914	1.58995
7	1972/11	8.3757	-0.29722	1.59728
8	1972/12	8.2489	-0.12736	2.61868

Figure 42.86 Last Part of OUT= Data Set

Obs	date	y1	FOR1	STD1	LCI1	UCI1	y2	FOR2	STD2	LCI2	UCI2
499	2013/11	-6.1917	-4.1545	2.88303	-9.8051	1.4962	6.09470	6.33899	1.43651	3.5235	9.1545
500	2013/12	-10.2133	-8.6817	2.97211	-14.5070	-2.8565	2.88544	2.11833	1.28490	-0.4000	4.6367
501	2014/01	.	-11.8921	2.92171	-17.6186	-6.1657	.	-1.30455	1.33400	-3.9191	1.3100
502	2014/02	.	-11.7095	4.83388	-21.1837	-2.2353	.	-3.59592	2.37237	-8.2457	1.0538
503	2014/03	.	-10.2617	6.20050	-22.4145	1.8910	.	-4.17796	3.77457	-11.5760	3.2201
504	2014/04	.	-8.1778	7.02293	-21.9425	5.5869	.	-3.47144	4.98630	-13.2444	6.3015
505	2014/05	.	-6.0032	7.41997	-20.5461	8.5396	.	-1.98718	5.81618	-13.3867	9.4123
506	2014/06	.	-4.1332	7.56318	-18.9567	10.6904	.	-0.21231	6.27549	-12.5120	12.0874

Figure 42.87 Last Part of OUTHT= Data Set

Obs	date	h1_1	h1_2	h2_2
499	2013/11	8.31189	-0.42221	2.06356
500	2013/12	8.83341	-0.00565	1.65098
501	2014/01	8.53639	-0.48367	1.77955
502	2014/02	9.42359	-0.13271	2.47088
503	2014/03	9.55818	-0.00081	2.85906
504	2014/04	9.58107	0.04780	3.07044
505	2014/05	9.58585	0.06690	3.18347
506	2014/06	9.58718	0.07508	3.24331

OUTSTAT= Data Set

The OUTSTAT= data set contains estimation results of the fitted model produced by the VARMAX statement. The following output variables can be created. The subindex i is $1, \ldots, k$, where k is the number of endogenous variables.

- the BY variables
- NAME, a character variable that contains the name of endogenous (dependent) variables
- SIGMA_i, numeric variables that contain the estimate of the innovation covariance matrix
- AICC, a numeric variable that contains the corrected Akaike's information criterion value
- HQC, a numeric variable that contains the Hannan-Quinn's information criterion value
- AIC, a numeric variable that contains the Akaike's information criterion value
- SBC, a numeric variable that contains the Schwarz Bayesian's information criterion value
- FPEC, a numeric variable that contains the final prediction error criterion value
- FValue, a numeric variable that contains the F statistics
- PValue, a numeric variable that contains p-value for the F statistics

If the JOHANSEN= option is specified, the following items are added:

- Eigenvalue, a numeric variable that contains eigenvalues for the cointegration rank test of integrated order 1
- RestrictedEigenvalue, a numeric variable that contains eigenvalues for the cointegration rank test of integrated order 1 when the NOINT option is not specified
- Beta_i, numeric variables that contain long-run effect parameter estimates, $\boldsymbol{\beta}$
- Alpha_i, numeric variables that contain adjustment parameter estimates, $\boldsymbol{\alpha}$

If the JOHANSEN=(IORDER=2) option is specified, the following items are added:

- EValueI2_i, numeric variables that contain eigenvalues for the cointegration rank test of integrated order 2
- EValueI1, a numeric variable that contains eigenvalues for the cointegration rank test of integrated order 1
- Eta_i, numeric variables that contain the parameter estimates in integrated order 2, $\boldsymbol{\eta}$
- Xi_i, numeric variables that contain the parameter estimates in integrated order 2, $\boldsymbol{\xi}$

The OUTSTAT= data set contains the values shown Table 42.11 for a bivariate case.

Table 42.11 OUTSTAT= Data Set

Obs	NAME	SIGMA_1	SIGMA_2	AICC	RSquare	FValue	PValue
1	y1	σ_{11}	σ_{12}	$aicc$	R_1^2	F_1	$prob_1$
2	y2	σ_{21}	σ_{22}	.	R_2^2	F_2	$prob_2$

Obs	EValueI2_1	EValueI2_2	EValueI1	Beta_1	Beta_2
1	e_{11}	e_{12}	e_1	β_{11}	β_{12}
2	e_{21}	.	e_2	β_{21}	β_{21}

Obs	Alpha_1	Alpha_2	Eta_1	Eta_2	Xi_1	Xi_2
1	α_{11}	α_{12}	η_{11}	η_{12}	ξ_{11}	ξ_{12}
2	α_{21}	α_{22}	η_{21}	η_{22}	ξ_{21}	ξ_{22}

Consider the following example:

```
proc varmax data=simul2 outstat=stat;
   model y1 y2 / p=2 noint noprint
                 cointtest=(johansen=(iorder=2));
   cointeg rank=1 normalize=y1;
run;

proc print data=stat;
run;
```

The output in Figure 42.88 shows the results of the OUTSTAT= data set.

Figure 42.88 OUTSTAT= Data Set

Obs	NAME	SIGMA_1	SIGMA_2	AICC	HQC	AIC	SBC	FPEC	RSquare	FValue	PValue
1	y1	94.7557	4.527	9.37221	9.43236	9.36834	9.52661	11712.14	0.93900	482.308	6.1637E-57
2	y2	4.5268	109.570	.	.	.	.	.	0.93912	483.334	5.6124E-57

Obs	EValuel2_1	EValuel2_2	EValuel1	Beta_1	Beta_2	Alpha_1	Alpha_2	Eta_1	Eta_2	Xi_1	Xi_2
1	0.98486	0.95079	0.50864	1.00000	1.00000	-0.46680	0.007937	-0.012307	0.027030	54.1606	-52.3144
2	0.81451	.	0.01108	-1.95575	-1.33622	0.10667	0.033530	0.015555	0.023086	-79.4240	-18.3308

Printed Output

The default printed output produced by the VARMAX procedure is described in the following list:

- descriptive statistics, which include the number of observations used, the names of the variables, their means and standard deviations (STD), their minimums and maximums, the differencing operations used, and the labels of the variables
- a type of model to fit the data and an estimation method
- a table of parameter estimates that shows the following for each parameter: the variable name for the left-hand side of equation, the parameter name, the parameter estimate, the approximate standard error, t value, the approximate probability ($Pr > |t|$), and the variable name for the right-hand side of equations in terms of each parameter
- the innovation covariance matrix
- the information criteria

If PRINT=ESTIMATES is specified, the VARMAX procedure prints the following list with the default printed output:

- the estimates of the constant vector (or seasonal constant matrix), the trend vector, the coefficient matrices of the distributed lags, the AR coefficient matrices, and the MA coefficient matrices
- the ALPHA and BETA parameter estimates for the error correction model
- the schematic representation of parameter estimates

If PRINT=DIAGNOSE is specified, the VARMAX procedure prints the following list with the default printed output:

- the cross-covariance and cross-correlation matrices of the residuals

- the tables of test statistics for the hypothesis that the residuals of the model are white noise:
  - Durbin-Watson (DW) statistics
  - F test for autoregressive conditional heteroscedastic (ARCH) disturbances
  - F test for AR disturbance
  - Jarque-Bera normality test
  - Portmanteau test

ODS Table Names

The VARMAX procedure assigns a name to each table that it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in Table 42.12.

Table 42.12 ODS Tables Produced in the VARMAX Procedure

ODS Table Name	Description	Option
ODS Tables Created by the MODEL Statement		
AccumImpulse	Accumulated impulse response matrices	IMPULSE=(ACCUM) IMPULSE=(ALL)
AccumImpulsebyVar	Accumulated impulse response by variable	IMPULSE=(ACCUM) IMPULSE=(ALL)
AccumImpulseX	Accumulated transfer function matrices	IMPULSX=(ACCUM) IMPULSX=(ALL)
AccumImpulseXbyVar	Accumulated transfer function by variable	IMPULSX=(ACCUM) IMPULSX=(ALL)
Alpha	$\boldsymbol{\alpha}$ coefficients	JOHANSEN=
AlphaInECM	$\boldsymbol{\alpha}$ coefficients when rank=r	PRINT=(ESTIMATES) with ECM=
AlphaOnDrift	$\boldsymbol{\alpha}$ coefficients under the restriction of a deterministic term	JOHANSEN=
AlphaBetaInECM	$\Pi = \boldsymbol{\alpha}\boldsymbol{\beta}'$ coefficients when rank=r	PRINT=(ESTIMATES) with ECM=
ANOVA	Univariate model diagnostic checks for the residuals	PRINT=DIAGNOSE
ARCoef	AR coefficients	PRINT=(ESTIMATES) with P=
ARRoots	Roots of AR characteristic polynomial	ROOTS with P=
Beta	$\boldsymbol{\beta}$ coefficients	JOHANSEN=
BetaInECM	$b\beta$ coefficients when rank=r	PRINT=(ESTIMATES) with ECM=
BetaOnDrift	$\boldsymbol{\beta}$ coefficients under the restriction of a deterministic term	JOHANSEN=

Table 42.12 *continued*

ODS Table Name	Description	Option
CCCCorrConstant	Constant correlation matrix in the CCC GARCH model	CORRCONSTANT=EXPECT with FORM=CCC
Constant	Constant estimates	without NOINT
CorrB	Correlations of parameter estimates	CORRB
CorrResiduals	Correlations of residuals	PRINT=DIAGNOSE
CorrResidualsbyVar	Correlations of residuals by variable	PRINT=DIAGNOSE
CorrResidualsGraph	Schematic representation of correlations of residuals	PRINT=DIAGNOSE
CorrXGraph	Schematic representation of sample correlations of independent series	CORRX
CorrYGraph	Schematic representation of sample correlations of dependent series	CORRY
CorrXLags	Correlations of independent series	CORRX
CorrXbyVar	Correlations of independent series by variable	CORRX
CorrYLags	Correlations of dependent series	CORRY
CorrYbyVar	Correlations of dependent series by variable	CORRY
CovarianceParameterEstimates	Covariance parameter estimates	METHOD=ML without the PRIOR= option, or GARCH statement
CovB	Covariances of parameter estimates	COVB
CovInnovation	Covariances of the innovations	Default
CovPredictError	Covariance matrices of the prediction error	COVPE
CovPredictErrorbyVar	Covariances of the prediction error by variable	COVPE
CovResiduals	Covariances of residuals	PRINT=DIAGNOSE
CovResidualsbyVar	Covariances of residuals by variable	PRINT=DIAGNOSE
CovXLags	Covariances of independent series	COVX
CovXbyVar	Covariances of independent series by variable	COVX
CovYLags	Covariances of dependent series	COVY
CovYbyVar	Covariances of dependent series by variable	COVY
DCCCorrConstant	Unconditional correlation matrix in the DCC GARCH model	CORRCONSTANT=EXPECT with FORM=DCC
DecomposeCovPredictError	Decomposition of the prediction error covariances	DECOMPOSE
DecomposeCovPredictErrorbyVar	Decomposition of the prediction error covariances by variable	DECOMPOSE
DFTest	Dickey-Fuller test	DFTEST
DiagnostAR	Test the AR disturbance for the residuals	PRINT=DIAGNOSE

Table 42.12 *continued*

ODS Table Name	Description	Option
DiagnostWN	Test the ARCH disturbance and normality for the residuals	PRINT=DIAGNOSE
DynamicARCoef	AR coefficients of the dynamic model	DYNAMIC
DynamicConstant	Constant estimates of the dynamic model	DYNAMIC
DynamicCovInnovation	Covariances of the innovations of the dynamic model	DYNAMIC
DynamicLinearTrend	Linear trend estimates of the dynamic model	DYNAMIC
DynamicMACoef	MA coefficients of the dynamic model	DYNAMIC
DynamicSConstant	Seasonal constant estimates of the dynamic model	DYNAMIC
DynamicParameterEstimates	Parameter estimates table of the dynamic model	DYNAMIC
DynamicParameterGraph	Schematic representation of the parameters of the dynamic model	DYNAMIC
DynamicQuadTrend	Quadratic trend estimates of the dynamic model	DYNAMIC
DynamicSeasonGraph	Schematic representation of the seasonal dummies of the dynamic model	DYNAMIC
DynamicXLagCoef	Dependent coefficients of the dynamic model	DYNAMIC
Hypothesis	Hypothesis of different deterministic terms in cointegration rank test	JOHANSEN=
HypothesisTest	Test hypothesis of different deterministic terms in cointegration rank test	JOHANSEN=
EigenvalueI2	Eigenvalues in integrated order 2	JOHANSEN= (IORDER=2)
Eta	η coefficients	JOHANSEN= (IORDER=2)
InfiniteARRepresent	Infinite order ar representation	IARR
InfoCriteria	Information criteria	default
LinearTrend	Linear trend estimates	TREND=
LogLikelihood	Log likelihood	default
MACoef	MA coefficients	Q=
MARoots	Roots of MA characteristic polynomial	ROOTS with Q=
MaxTest	Cointegration rank test using the maximum eigenvalue	JOHANSEN= (TYPE=MAX)
Minic	Tentative order selection	MINIC or MINIC=
ModelType	Type of model	default
NObs	Number of observations	default
OrthoImpulse	Orthogonalized impulse response matrices	IMPULSE=(ORTH) IMPULSE=(ALL)
OrthoImpulsebyVar	Orthogonalized impulse response by variable	IMPULSE=(ORTH) IMPULSE=(ALL)

Table 42.12 *continued*

ODS Table Name	Description	Option
ParameterEstimates	Parameter estimates table	default
ParameterGraph	Schematic representation of the parameters	PRINT=ESTIMATES
PartialAR	Partial autoregression matrices	PARCOEF
PartialARGraph	Schematic representation of partial autoregression	PARCOEF
PartialCanCorr	Partial canonical correlation analysis	PCANCORR
PartialCorr	Partial cross-correlation matrices	PCORR
PartialCorrbyVar	Partial cross-correlations by variable	PCORR
PartialCorrGraph	Schematic representation of partial cross-correlations	PCORR
PortmanteauTest	Chi-square test table for residual cross-correlations	PRINT=DIAGNOSE
ProportionCovPredictError	Proportions of prediction error covariance decomposition	DECOMPOSE
ProportionCovPredictErrorbyVar	Proportions of prediction error covariance decomposition by variable	DECOMPOSE
RankTestI2	Cointegration rank test in integrated order 2	JOHANSEN= (IORDER=2)
RestrictMaxTest	Cointegration rank test using the maximum eigenvalue under the restriction of a deterministic term	JOHANSEN= (TYPE=MAX) without NOINT
RestrictTraceTest	Cointegration rank test using the trace under the restriction of a deterministic term	JOHANSEN= (TYPE=TRACE) without NOINT
QuadTrend	Quadratic trend estimates	TREND=QUAD
SeasonGraph	Schematic representation of the seasonal dummies	PRINT=ESTIMATES with NSEASON=
SConstant	Seasonal constant estimates	NSEASON=
SimpleImpulse	Impulse response matrices	IMPULSE=(SIMPLE) IMPULSE=(ALL)
SimpleImpulsebyVar	Impulse response by variable	IMPULSE=(SIMPLE) IMPULSE=(ALL)
SimpleImpulseX	Impulse response matrices of transfer function	IMPULSX=(SIMPLE) IMPULSX=(ALL)
SimpleImpulseXbyVar	Impulse response of transfer function by variable	IMPULSX=(SIMPLE) IMPULSX=(ALL)
Summary	Simple summary statistics	default
SWTest	Common trends test	SW=
TraceTest	Cointegration rank test using the trace	JOHANSEN= (TYPE=TRACE)
Xi	ξ coefficient matrix	JOHANSEN= (IORDER=2)
XLagCoef	Dependent coefficients	XLAG=

Table 42.12 *continued*

ODS Table Name	Description	Option
YWEstimates	Yule-Walker estimates	YW
ODS Tables Created by the GARCH Statement		
ARCHCoef	ARCH coefficients	Q=
GARCHCoef	GARCH coefficients	P=
GARCHConstant	GARCH constant estimates	PRINT=ESTIMATES
GARCHParameter-Estimates	GARCH parameter estimates table	default
GARCHParameter-Graph	Schematic representation of the garch parameters	PRINT=ESTIMATES
GARCHRoots	Roots of GARCH characteristic polynomial	ROOTS
ODS Tables Created by the COINTEG Statement or the ECM option in the MODEL statement		
AlphaAndBetaParameterEstimaters	parameter estimates of $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{\beta}_0$, and $\boldsymbol{\beta}_1$	default
AlphaInECM	$\boldsymbol{\alpha}$ coefficients when rank=r	PRINT=ESTIMATES
AlphaBetaInECM	$\Pi = \boldsymbol{\alpha}\boldsymbol{\beta}'$ coefficients when rank=r	PRINT=ESTIMATES
AlphaOnAlpha	$\boldsymbol{\alpha}$ coefficients under the restriction of α	J=
AlphaOnBeta	$\boldsymbol{\alpha}$ coefficients under the restriction of β	H=
AlphaTestResults	Hypothesis testing of $\boldsymbol{\beta}$	J=
BetaInECM	$\boldsymbol{\beta}$ coefficients when rank=r	PRINT=ESTIMATES
BetaOnBeta	$\boldsymbol{\beta}$ coefficients under the restriction of $\boldsymbol{\beta}$	H=
BetaOnAlpha	$\boldsymbol{\beta}$ coefficients under the restriction of $\boldsymbol{\alpha}$	J=
BetaTestResults	Hypothesis testing of β	H=
GrangerRepresent	Coefficient of Granger representation	PRINT=ESTIMATES
HMatrix	Restriction matrix for $\boldsymbol{\beta}$	H=
JMatrix	Restriction matrix for $\boldsymbol{\alpha}$	J=
WeakExogeneity	Testing weak exogeneity of each dependent variable with respect to BETA	EXOGENEITY
ODS Tables Created by the CAUSAL Statement		
CausalityTest	Granger causality test	default
GroupVars	Two groups of variables	default
ODS Tables Created by the RESTRICT Statement		
Restrict	Restriction table	default

Table 42.12 *continued*

ODS Table Name	Description	Option
ODS Tables Created by the TEST Statement		
Test	Wald test	default
ODS Tables Created by the OUTPUT Statement		
Forecasts	Forecasts table	without NOPRINT

Note that the ODS table names suffixed by "byVar" can be obtained with the PRINTFORM=UNIVARIATE option.

ODS Graphics

This section describes the use of ODS for creating statistical graphs with the VARMAX procedure.

When ODS GRAPHICS are in effect, the VARMAX procedure produces a variety of plots for each dependent variable.

The plots available are as follows:

- The procedure displays the following plots for each dependent variable in the MODEL statement with the PLOT= option in the VARMAX statement:
  - impulse response function
  - impulse response of the transfer function
  - time series and predicted series
  - prediction errors
  - distribution of the prediction errors
  - normal quantile of the prediction errors
  - ACF of the prediction errors
  - PACF of the prediction errors
  - IACF of the prediction errors
  - log scaled white noise test of the prediction errors
- The procedure displays forecast plots for each dependent variable in the OUTPUT statement with the PLOT= option in the VARMAX statement.

ODS Graph Names

The VARMAX procedure assigns a name to each graph it creates by using ODS. You can use these names to reference the graphs when using ODS. The names are listed in Table 42.13.

Table 42.13 ODS Graphics Produced in the VARMAX Procedure

ODS Table Name	Plot Description	Statement
ErrorACFPlot	Autocorrelation function of prediction errors	MODEL
ErrorIACFPlot	Inverse autocorrelation function of prediction errors	MODEL
ErrorPACFPlot	Partial autocorrelation function of prediction errors	MODEL
ErrorDiagnosticsPanel	Diagnostics of prediction errors	MODEL
ErrorNormalityPanel	Histogram and Q-Q plot of prediction errors	MODEL
ErrorDistribution	Distribution of prediction errors	MODEL
ErrorQQPlot	Q-Q plot of prediction errors	MODEL
ErrorWhiteNoisePlot	White noise test of prediction errors	MODEL
ErrorPlot	Prediction errors	MODEL
ModelPlot	Time series and predicted series	MODEL
AccumulatedIRFPanel	Accumulated impulse response function	MODEL
AccumulatedIRFXPanel	Accumulated impulse response of transfer function	MODEL
OrthogonalIRFPanel	Orthogonalized impulse response function	MODEL
SimpleIRFPanel	Simple impulse response function	MODEL
SimpleIRFXPanel	Simple impulse response of transfer function	MODEL
ModelForecastsPlot	Time series and forecasts	OUTPUT
ForecastsOnlyPlot	Forecasts	OUTPUT

Computational Issues

Computational Method

The VARMAX procedure uses numerous linear algebra routines and frequently uses the sweep operator (Goodnight 1979) and the Cholesky root (Golub and Van Loan 1983).

In addition, the VARMAX procedure uses the nonlinear optimization (NLO) subsystem to perform nonlinear optimization tasks for the maximum likelihood estimation. The optimization requires intensive computation.

Convergence Problems

For some data sets, the computation algorithm can fail to converge. Nonconvergence can result from a number of causes, including flat or ridged likelihood surfaces and ill-conditioned data.

If you experience convergence problems, the following points might be helpful:

- Data that contain extreme values can affect results in PROC VARMAX. Rescaling the data can improve stability.
- Changing the TECH=, MAXITER=, and MAXFUNC= options in the NLOPTIONS statement can improve the stability of the optimization process.
- Specifying a different model that might fit the data more closely and might improve convergence.

Memory

Let T be the length of each series, k be the number of dependent variables, p be the order of autoregressive terms, and q be the order of moving-average terms. The number of parameters to estimate for a VARMA(p,q) model is

$$k + (p+q)k^2 + k * (k+1)/2$$

As k increases, the number of parameters to estimate increases very quickly. Furthermore the memory requirement for VARMA(p,q) quadratically increases as k and T increase.

For a VARMAX(p,q,s) model and GARCH-type multivariate conditional heteroscedasticity models, the number of parameters to estimate and the memory requirements are considerable.

Computing Time

PROC VARMAX is computationally intensive, and execution times can be long. Extensive CPU time is often required to compute the maximum likelihood estimates.

Examples: VARMAX Procedure

Example 42.1: Analysis of United States Economic Variables

Consider the following four-dimensional system of US economic variables. Quarterly data for the years 1954 to 1987 are used (Lütkepohl 1993, Table E.3.).

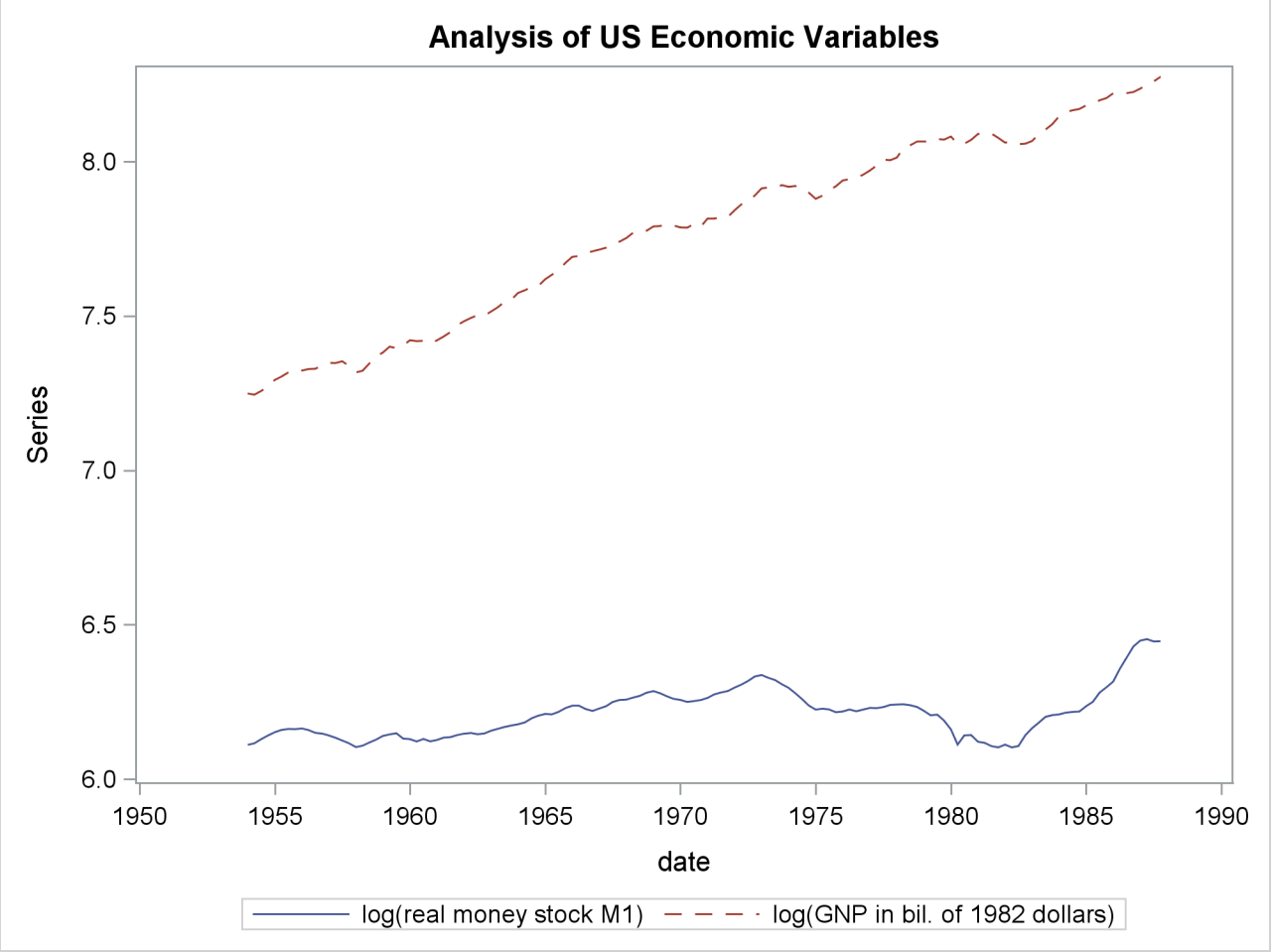
```
title 'Analysis of US Economic Variables';
data us_money;
   date=intnx( 'qtr', '01jan54'd, _n_-1 );
   format date yyq. ;
   input y1 y2 y3 y4 @@;
   y1=log(y1);
   y2=log(y2);
   label y1='log(real money stock M1)'
         y2='log(GNP in bil. of 1982 dollars)'
         y3='Discount rate on 91-day T-bills'
         y4='Yield on 20-year Treasury bonds';
datalines;
450.9 1406.8  0.010800000 0.026133333
453.0 1401.2 0.0081333333 0.025233333
459.1 1418.0 0.0087000000 0.024900000

   ... more lines ...
```

The following statements plot the series:

```
proc sgplot data=us_money;
   series x=date y=y1 / lineattrs=(pattern=solid);
   series x=date y=y2 / lineattrs=(pattern=dash);
   yaxis label="Series";
run;
```

Output 42.1.1 shows the plot of the variables $y1$ and $y2$.

Output 42.1.1 Plot of Data

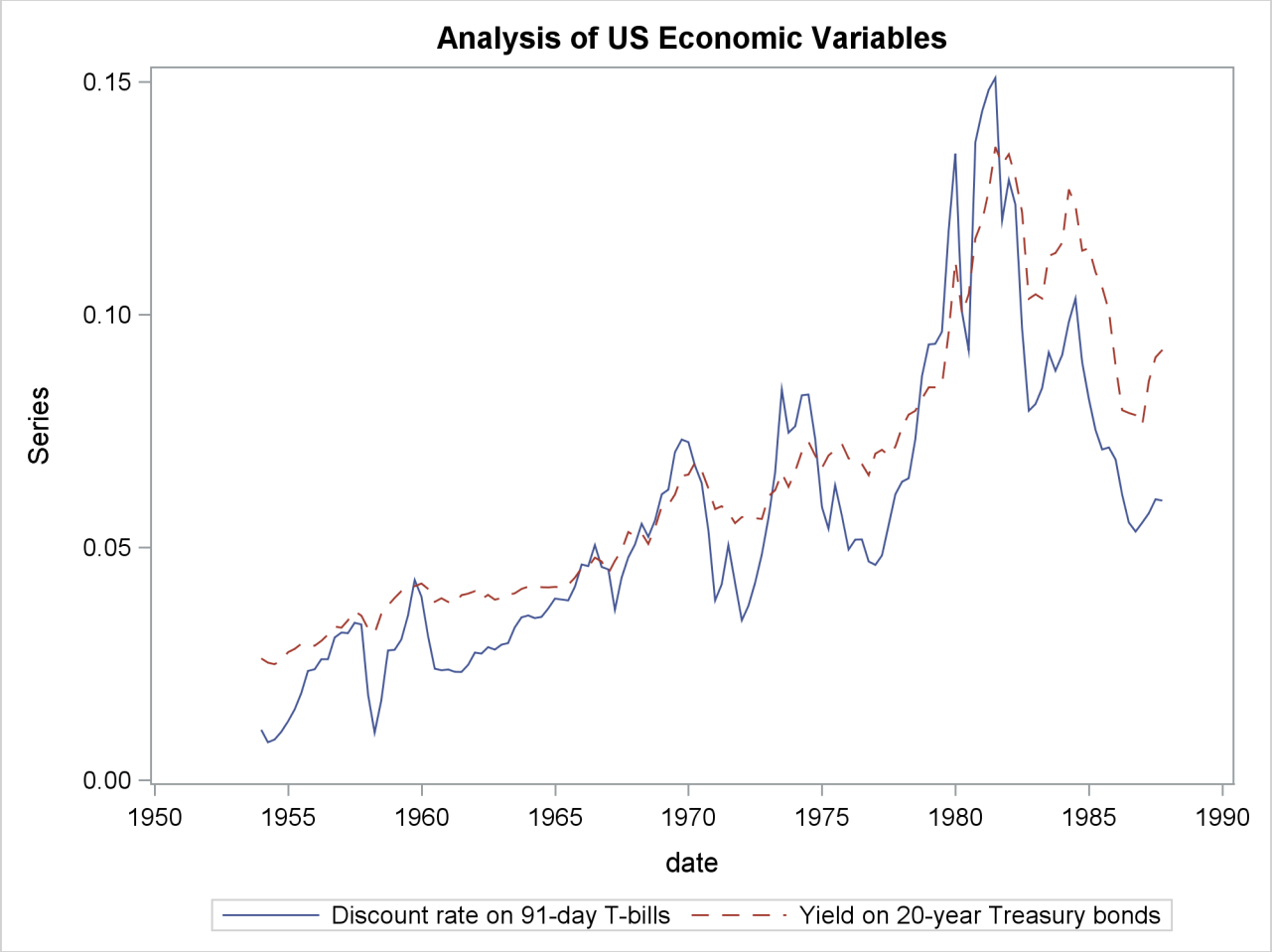


The following statements plot the variables $y3$ and $y4$:

```
proc sgplot data=us_money;
   series x=date y=y3 / lineattrs=(pattern=solid);
   series x=date y=y4 / lineattrs=(pattern=dash);
   yaxis label="Series";
run;
```

Output 42.1.2 shows the plot of the variables $y3$ and $y4$.

Output 42.1.2 Plot of Data



The following statements perform the Dickey-Fuller test for stationarity, the Johansen cointegrated test integrated order 2, and the exogeneity test. The VECM(2) is fit to the data.

```
proc varmax data=us_money;
   id date interval=qtr;
   model y1-y4 / p=2 lagmax=6 dftest
                 print=(iarr(3) estimates diagnose)
                 cointtest=(johansen=(iorder=2));
   cointeg rank=1 normalize=y1 exogeneity;
run;
```

From the outputs shown in Output 42.1.5, you can see that the series has unit roots and is cointegrated in rank 1 with integrated order 1. The fitted VECM(2) is given as

$$\Delta \mathbf{y}_t = \begin{pmatrix} 0.0408 \\ 0.0860 \\ 0.0052 \\ -0.0144 \end{pmatrix} + \begin{pmatrix} -0.0140 & 0.0065 & -0.2026 & 0.1306 \\ -0.0281 & 0.0131 & -0.4080 & 0.2630 \\ -0.0022 & 0.0010 & -0.0312 & 0.0201 \\ 0.0051 & -0.0024 & 0.0741 & -0.0477 \end{pmatrix} \mathbf{y}_{t-1} + \begin{pmatrix} 0.3460 & 0.0913 & -0.3535 & -0.9690 \\ 0.0994 & 0.0379 & 0.2390 & 0.2866 \\ 0.1812 & 0.0786 & 0.0223 & 0.4051 \\ 0.0322 & 0.0496 & -0.0329 & 0.1857 \end{pmatrix} \Delta \mathbf{y}_{t-1} + \boldsymbol{\epsilon}_t$$

The Δ prefixed to a variable name implies differencing.

Output 42.1.3 through Output 42.1.16 show the details. Output 42.1.3 shows the descriptive statistics.

Output 42.1.3 Descriptive Statistics

Analysis of US Economic Variables

The VARMAX Procedure

Number of Observations	136
Number of Pairwise Missing	0

Simple Summary Statistics

Variable	Type	N	Mean	Standard Deviation	Min	Max	Label
y1	**Dependent**	136	6.21295	0.07924	6.10278	6.45331	log(real money stock M1)
y2	**Dependent**	136	7.77890	0.30110	7.24508	8.27461	log(GNP in bil. of 1982 dollars)
y3	**Dependent**	136	0.05608	0.03109	0.00813	0.15087	Discount rate on 91-day T-bills
y4	**Dependent**	136	0.06458	0.02927	0.02490	0.13600	Yield on 20-year Treasury bonds

Output 42.1.4 shows the output for Dickey-Fuller tests for the nonstationarity of each series. The null hypothesis is that there exists a unit root. All series have a unit root.

Output 42.1.4 Unit Root Tests

Unit Root Test					
Variable	**Type**	**Rho**	**Pr < Rho**	**Tau**	**Pr < Tau**
y1	**Zero Mean**	0.05	0.6934	1.14	0.9343
	Single Mean	-2.97	0.6572	-0.76	0.8260
	Trend	-5.91	0.7454	-1.34	0.8725
y2	**Zero Mean**	0.13	0.7124	5.14	0.9999
	Single Mean	-0.43	0.9309	-0.79	0.8176
	Trend	-9.21	0.4787	-2.16	0.5063
y3	**Zero Mean**	-1.28	0.4255	-0.69	0.4182
	Single Mean	-8.86	0.1700	-2.27	0.1842
	Trend	-18.97	0.0742	-2.86	0.1803
y4	**Zero Mean**	0.40	0.7803	0.45	0.8100
	Single Mean	-2.79	0.6790	-1.29	0.6328
	Trend	-12.12	0.2923	-2.33	0.4170

The Johansen cointegration rank test shows whether the series is integrated order either 1 or 2 as shown in Output 42.1.5. The last two columns in Output 42.1.5 explain the cointegration rank test with integrated order 1. The results indicate that there is a cointegrated relationship with cointegration rank 1 with respect to the 0.05 significance level because the test statistic for the null hypothesis H0: $r = 0$ is 55.9633 and its corresponding p-value is 0.0072, less than 0.05 (indicating that H0: $r = 0$ should be rejected), and the test statistic for the null hypothesis H0: $r = 1$ is 20.6542 and its corresponding p-value is 0.3775, greater than 0.05 (indicating that H0: $r = 1$ cannot be rejected). Now, look at the row associated with $r = 1$. All p-values of the tests for the null hypothesis that the series are integrated order 2 are zeros, less than 0.05 significance level (indicating that the null hypothesis should be rejected).

Output 42.1.5 Cointegration Rank Test

Cointegration Rank Test for I(2)						
r\k-r-s	**4**	**3**	**2**	**1**	**Trace of I(1)**	**Pr > Trace of I(1)**
0	384.6090	214.3790	107.9378	37.0252	55.9633	0.0072
Pr > Trace of I(2)	0.0000	0.0000	0.0000	0.0000		
1		219.6239	89.2151	27.3261	20.6542	0.3775
Pr > Trace of I(2)		0.0000	0.0000	0.0000		
2			73.6178	22.1328	2.6477	0.9803
Pr > Trace of I(2)			0.0000	0.0000		
3				38.2943	0.0149	0.9031
Pr > Trace of I(2)				0.0000		

Output 42.1.6 shows the estimates of the long-run parameter, $\boldsymbol{\beta}$, and the adjustment coefficient, $\boldsymbol{\alpha}$.

Output 42.1.6 Cointegration Rank Test, Continued

Beta				
Variable	**1**	**2**	**3**	**4**
y1	1.00000	1.00000	1.00000	1.00000
y2	-0.46458	-0.63174	-0.69996	-0.16140
y3	14.51619	-1.29864	1.37007	-0.61806
y4	-9.35520	7.53672	2.47901	1.43731

Alpha				
Variable	**1**	**2**	**3**	**4**
y1	-0.01396	0.01396	-0.01119	0.00008
y2	-0.02811	-0.02739	-0.00032	0.00076
y3	-0.00215	-0.04967	-0.00183	-0.00072
y4	0.00510	-0.02514	-0.00220	0.00016

Output 42.1.7 shows the estimates $\boldsymbol{\eta}$ and $\boldsymbol{\xi}$.

Output 42.1.7 Cointegration Rank Test, Continued

Eta				
Variable	**1**	**2**	**3**	**4**
y1	52.74907	41.74502	-20.80403	55.77415
y2	-49.10609	-9.40081	98.87199	22.56416
y3	68.29674	-144.83173	-27.35953	15.51142
y4	121.25932	271.80496	85.85156	-130.11599

Xi				
Variable	**1**	**2**	**3**	**4**
y1	-0.00842	-0.00052	-0.00208	-0.00250
y2	0.00141	0.00213	-0.00736	-0.00058
y3	-0.00445	0.00541	-0.00150	0.00310
y4	-0.00211	-0.00064	-0.00130	0.00197

Output 42.1.8 shows that the VECM(2) is fit to the data. The RANK=1 option in the COINTEG statement produces the estimates of the long-run parameter, $\boldsymbol{\beta}$, and the adjustment coefficient, $\boldsymbol{\alpha}$.

Output 42.1.8 Parameter Estimates

Analysis of US Economic Variables

The VARMAX Procedure

Type of Model	VECM(2)
Estimation Method	Maximum Likelihood Estimation
Cointegrated Rank	1

Beta	
Variable	**1**
y1	1.00000
y2	-0.46458
y3	14.51619
y4	-9.35520

Alpha	
Variable	**1**
y1	-0.01396
y2	-0.02811
y3	-0.00215
y4	0.00510

Output 42.1.9 shows the parameter estimates in terms of the constant, the lag 1 coefficients (y_{t-1}) that are contained in the $\alpha\beta'$ estimates, and the coefficients that are associated with the lag 1 first differences (Δy_{t-1}).

Output 42.1.9 Parameter Estimates, Continued

Constant	
Variable	**Constant**
y1	0.04076
y2	0.08595
y3	0.00518
y4	-0.01438

Parameter Alpha * Beta' Estimates				
Variable	**y1**	**y2**	**y3**	**y4**
y1	-0.01396	0.00648	-0.20263	0.13059
y2	-0.02811	0.01306	-0.40799	0.26294
y3	-0.00215	0.00100	-0.03121	0.02011
y4	0.00510	-0.00237	0.07407	-0.04774

AR Coefficients of Differenced Lag					
DIF Lag	**Variable**	**y1**	**y2**	**y3**	**y4**
1	**y1**	0.34603	0.09131	-0.35351	-0.96895
	y2	0.09936	0.03791	0.23900	0.28661
	y3	0.18118	0.07859	0.02234	0.40508
	y4	0.03222	0.04961	-0.03292	0.18568

Output 42.1.10 through Output 42.1.12 show the parameter estimates and their significance.

Output 42.1.10 Parameter Estimates, Continued

Model Parameter Estimates

Equation	Parameter	Estimate	Standard Error	t Value	Pr > |t|	Variable
D_y1	CONST1	0.04076	0.01418	2.87	0.0048	1
	AR1_1_1	-0.01396	0.00495	-2.82	0.0056	y1(t-1)
	AR1_1_2	0.00648	0.00230	2.82	0.0056	y2(t-1)
	AR1_1_3	-0.20263	0.07191	-2.82	0.0056	y3(t-1)
	AR1_1_4	0.13059	0.04634	2.82	0.0056	y4(t-1)
	AR2_1_1	0.34603	0.06414	5.39	<.0001	D_y1(t-1)
	AR2_1_2	0.09131	0.07334	1.25	0.2154	D_y2(t-1)
	AR2_1_3	-0.35351	0.11024	-3.21	0.0017	D_y3(t-1)
	AR2_1_4	-0.96895	0.20737	-4.67	<.0001	D_y4(t-1)
D_y2	CONST2	0.08595	0.01679	5.12	<.0001	1
	AR1_2_1	-0.02811	0.00586	-4.79	<.0001	y1(t-1)
	AR1_2_2	0.01306	0.00272	4.79	<.0001	y2(t-1)
	AR1_2_3	-0.40799	0.08514	-4.79	<.0001	y3(t-1)
	AR1_2_4	0.26294	0.05487	4.79	<.0001	y4(t-1)
	AR2_2_1	0.09936	0.07594	1.31	0.1932	D_y1(t-1)
	AR2_2_2	0.03791	0.08683	0.44	0.6632	D_y2(t-1)
	AR2_2_3	0.23900	0.13052	1.83	0.0695	D_y3(t-1)
	AR2_2_4	0.28661	0.24552	1.17	0.2453	D_y4(t-1)
D_y3	CONST3	0.00518	0.01608	0.32	0.7476	1
	AR1_3_1	-0.00215	0.00562	-0.38	0.7024	y1(t-1)
	AR1_3_2	0.00100	0.00261	0.38	0.7024	y2(t-1)
	AR1_3_3	-0.03121	0.08151	-0.38	0.7024	y3(t-1)
	AR1_3_4	0.02011	0.05253	0.38	0.7024	y4(t-1)
	AR2_3_1	0.18118	0.07271	2.49	0.0140	D_y1(t-1)
	AR2_3_2	0.07859	0.08313	0.95	0.3463	D_y2(t-1)
	AR2_3_3	0.02234	0.12496	0.18	0.8584	D_y3(t-1)
	AR2_3_4	0.40508	0.23506	1.72	0.0873	D_y4(t-1)
D_y4	CONST4	-0.01438	0.00803	-1.79	0.0758	1
	AR1_4_1	0.00510	0.00281	1.82	0.0713	y1(t-1)
	AR1_4_2	-0.00237	0.00130	-1.82	0.0713	y2(t-1)
	AR1_4_3	0.07407	0.04072	1.82	0.0713	y3(t-1)
	AR1_4_4	-0.04774	0.02624	-1.82	0.0713	y4(t-1)
	AR2_4_1	0.03222	0.03632	0.89	0.3768	D_y1(t-1)
	AR2_4_2	0.04961	0.04153	1.19	0.2345	D_y2(t-1)
	AR2_4_3	-0.03292	0.06243	-0.53	0.5990	D_y3(t-1)
	AR2_4_4	0.18568	0.11744	1.58	0.1164	D_y4(t-1)

Output 42.1.11 Parameter Estimates, Continued

Alpha and Beta Parameter Estimates

Equation	Parameter	Estimate	Standard Error	t Value	Pr > |t|	Variable
D_y1	ALPHA1_1	-0.01396	0.00495	-2.82	0.0056	Beta[,1]'*_DEP_(t-1)
	BETA1_1	1.00000				y1(t-1)
D_y2	ALPHA2_1	-0.02811	0.00586	-4.79	<.0001	Beta[,1]'*_DEP_(t-1)
	BETA2_1	-0.46458				y2(t-1)
D_y3	ALPHA3_1	-0.00215	0.00562	-0.38	0.7024	Beta[,1]'*_DEP_(t-1)
	BETA3_1	14.51619				y3(t-1)
D_y4	ALPHA4_1	0.00510	0.00281	1.82	0.0713	Beta[,1]'*_DEP_(t-1)
	BETA4_1	-9.35520				y4(t-1)

Output 42.1.12 Parameter Estimates, Continued

Covariance Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Pr > |t|
COV1_1	0.00005	0.00001	8.19	<.0001
COV1_2	0.00001	0.00001	2.78	0.0062
COV2_2	0.00007	0.00001	8.19	<.0001
COV1_3	-0.00001	0.00001	-1.60	0.1118
COV2_3	0.00002	0.00001	2.71	0.0077
COV3_3	0.00007	0.00001	8.19	<.0001
COV1_4	-0.00000	0.00000	-1.31	0.1936
COV2_4	0.00001	0.00000	3.29	0.0013
COV3_4	0.00002	0.00000	6.67	<.0001
COV4_4	0.00002	0.00000	8.19	<.0001

Output 42.1.13 shows the innovation covariance matrix estimates, the log-likelihood, the various information criteria results, and the tests for white noise residuals. According to the Portmanteau test results, the residuals have significant correlations at lag 2 and 3, indicating that a VECM(3) model might be a better fit than the VECM(2) model.

Output 42.1.13 Diagnostic Checks

Covariances of Innovations				
Variable	**y1**	**y2**	**y3**	**y4**
y1	0.00005	0.00001	-0.00001	-0.00000
y2	0.00001	0.00007	0.00002	0.00001
y3	-0.00001	0.00002	0.00007	0.00002
y4	-0.00000	0.00001	0.00002	0.00002

Log-likelihood	2479.23

Information Criteria	
AICC	-4859
HQC	-4844.07
AIC	-4886.46
SBC	-4782.14
FPEC	2.23E-18

Schematic Representation of Cross Correlations of Residuals							
Variable/Lag	**0**	**1**	**2**	**3**	**4**	**5**	**6**
y1	++..	....	++..	....	+...	..−−	....
y2	++++	....	....	....	....	....	....
y3	.+++	....	+.−.	..++	−...	....	....
y4	.+++	....	....	..+.	....	....	....
+ is > 2*std error, - is < -2*std error, . is between							

Portmanteau Test for Cross Correlations of Residuals			
Up To Lag	**DF**	**Chi-Square**	**Pr > ChiSq**
3	16	53.90	<.0001
4	32	74.03	<.0001
5	48	103.08	<.0001
6	64	116.94	<.0001

Output 42.1.14 describes how well each univariate equation fits the data. The residuals for $y3$ and $y4$ differ from normality. Except for the residuals for $y3$, there are no AR effects on other residuals. Except for the residuals for $y4$, there are no ARCH effects on other residuals.

Output 42.1.14 Diagnostic Checks, Continued

Univariate Model ANOVA Diagnostics

Variable	R-Square	Standard Deviation	F Value	Pr > F
y1	0.6754	0.00712	32.51	<.0001
y2	0.3070	0.00843	6.92	<.0001
y3	0.1328	0.00807	2.39	0.0196
y4	0.0831	0.00403	1.42	0.1963

Univariate Model White Noise Diagnostics

		Normality		ARCH	
Variable	Durbin Watson	Chi-Square	Pr > ChiSq	F Value	Pr > F
y1	2.13418	7.19	0.0275	1.62	0.2053
y2	2.04003	1.20	0.5483	1.23	0.2697
y3	1.86892	253.76	<.0001	1.78	0.1847
y4	1.98440	105.21	<.0001	21.01	<.0001

Univariate Model AR Diagnostics

	AR1		AR2		AR3		AR4	
Variable	F Value	Pr > F	F Value	Pr > F	F Value	Pr > F	F Value	Pr > F
y1	0.68	0.4126	2.98	0.0542	2.01	0.1154	2.48	0.0473
y2	0.05	0.8185	0.12	0.8842	0.41	0.7453	0.30	0.8762
y3	0.56	0.4547	2.86	0.0610	4.83	0.0032	3.71	0.0069
y4	0.01	0.9340	0.16	0.8559	1.21	0.3103	0.95	0.4358

The PRINT=(IARR) option provides the VAR(2) representation in Output 42.1.15.

Output 42.1.15 Infinite Order AR Representation

Infinite Order AR Representation

Lag	Variable	y1	y2	y3	y4
1	y1	1.33208	0.09780	-0.55614	-0.83836
	y2	0.07125	1.05096	-0.16899	0.54955
	y3	0.17903	0.07959	0.99113	0.42520
	y4	0.03732	0.04724	0.04116	1.13795
2	y1	-0.34603	-0.09131	0.35351	0.96895
	y2	-0.09936	-0.03791	-0.23900	-0.28661
	y3	-0.18118	-0.07859	-0.02234	-0.40508
	y4	-0.03222	-0.04961	0.03292	-0.18568
3	y1	0.00000	0.00000	0.00000	0.00000
	y2	0.00000	0.00000	0.00000	0.00000
	y3	0.00000	0.00000	0.00000	0.00000
	y4	0.00000	0.00000	0.00000	0.00000

Output 42.1.16 shows whether each variable is the weak exogeneity of other variables. The variable $y1$ is not the weak exogeneity of other variables, $y2$, $y3$, and $y4$; the variable $y2$ is not the weak exogeneity of other variables, $y1$, $y3$, and $y4$; the variable $y3$ and $y4$ are the weak exogeneity of other variables.

Output 42.1.16 Weak Exogeneity Test

Testing Weak Exogeneity of Each Variables			
Variable	**DF**	**Chi-Square**	**Pr > ChiSq**
y1	1	6.55	0.0105
y2	1	12.54	0.0004
y3	1	0.09	0.7695
y4	1	1.81	0.1786

Example 42.2: Analysis of German Economic Variables

This example considers a three-dimensional VAR(2) model. The model contains the logarithms of a quarterly, seasonally adjusted West German fixed investment, disposable income, and consumption expenditures. The data used are in Lütkepohl (1993, Table E.1).

```
title 'Analysis of German Economic Variables';
data west;
   date = intnx( 'qtr', '01jan60'd, _n_-1 );
   format date yyq. ;
   input y1 y2 y3 @@;
   y1 = log(y1);
   y2 = log(y2);
   y3 = log(y3);
   label y1 = 'logarithm of investment'
         y2 = 'logarithm of income'
         y3 = 'logarithm of consumption';
datalines;
180  451  415 179  465  421 185  485  434 192  493  448
211  509  459 202  520  458 207  521  479 214  540  487

   ... more lines ...


data use;
   set west;
   where  date < '01jan79'd;
   keep date y1 y2 y3;
run;

proc varmax data=use;
   id date interval=qtr;
   model y1-y3 / p=2 dify=(1)
                 print=(decompose(6) impulse=(stderr) estimates diagnose)
                 printform=both lagmax=3;
   causal group1=(y1) group2=(y2 y3);
   output lead=5;
run;
```

First, the differenced data is modeled as a VAR(2) with the following result:

$$\Delta \mathbf{y}_t = \begin{pmatrix} -0.01672 \\ 0.01577 \\ 0.01293 \end{pmatrix} + \begin{pmatrix} -0.31963 & 0.14599 & 0.96122 \\ 0.04393 & -0.15273 & 0.28850 \\ -0.00242 & 0.22481 & -0.26397 \end{pmatrix} \Delta \mathbf{y}_{t-1} + \begin{pmatrix} -0.16055 & 0.11460 & 0.93439 \\ 0.05003 & 0.01917 & -0.01020 \\ 0.03388 & 0.35491 & -0.02223 \end{pmatrix} \Delta \mathbf{y}_{t-2} + \boldsymbol{\epsilon}_t$$

The parameter estimates AR1_1_2, AR1_1_3, AR2_1_2, and AR2_1_3 are insignificant, and the VARX model is fitted in the next step.

The detailed output is shown in Output 42.2.1 through Output 42.2.8.

Output 42.2.1 shows the descriptive statistics.

Output 42.2.1 Descriptive Statistics

Analysis of German Economic Variables

The VARMAX Procedure

Number of Observations	75
Number of Pairwise Missing	0
Observation(s) eliminated by differencing	1

Simple Summary Statistics

Variable	Type	N	Mean	Standard Deviation	Min	Max	Difference	Label
y1	**Dependent**	75	0.01811	0.04680	-0.14018	0.19358	1	logarithm of investment
y2	**Dependent**	75	0.02071	0.01208	-0.02888	0.05023	1	logarithm of income
y3	**Dependent**	75	0.01987	0.01040	-0.01300	0.04483	1	logarithm of consumption

Output 42.2.2 shows that a VAR(2) model is fit to the data.

Output 42.2.2 Parameter Estimates

Analysis of German Economic Variables

The VARMAX Procedure

Type of Model	VAR(2)
Estimation Method	Least Squares Estimation

Constant	
Variable	**Constant**
y1	-0.01672
y2	0.01577
y3	0.01293

		AR		
Lag	**Variable**	**y1**	**y2**	**y3**
1	**y1**	-0.31963	0.14599	0.96122
	y2	0.04393	-0.15273	0.28850
	y3	-0.00242	0.22481	-0.26397
2	**y1**	-0.16055	0.11460	0.93439
	y2	0.05003	0.01917	-0.01020
	y3	0.03388	0.35491	-0.02223

Output 42.2.3 shows the parameter estimates and their significance.

Output 42.2.3 Parameter Estimates, Continued

Schematic Representation

Variable/Lag	C	AR1	AR2
y1	.	-..	...
y2	+	...	...
y3	+	.+.	.+.

+ is > 2*std error, - is < -2*std error, . is between, * is N/A

Model Parameter Estimates

Equation	Parameter	Estimate	Standard Error	t Value	Pr > |t|	Variable
y1	CONST1	-0.01672	0.01723	-0.97	0.3352	1
	AR1_1_1	-0.31963	0.12546	-2.55	0.0132	y1(t-1)
	AR1_1_2	0.14599	0.54567	0.27	0.7899	y2(t-1)
	AR1_1_3	0.96122	0.66431	1.45	0.1526	y3(t-1)
	AR2_1_1	-0.16055	0.12491	-1.29	0.2032	y1(t-2)
	AR2_1_2	0.11460	0.53457	0.21	0.8309	y2(t-2)
	AR2_1_3	0.93439	0.66510	1.40	0.1647	y3(t-2)
y2	CONST2	0.01577	0.00437	3.60	0.0006	1
	AR1_2_1	0.04393	0.03186	1.38	0.1726	y1(t-1)
	AR1_2_2	-0.15273	0.13857	-1.10	0.2744	y2(t-1)
	AR1_2_3	0.28850	0.16870	1.71	0.0919	y3(t-1)
	AR2_2_1	0.05003	0.03172	1.58	0.1195	y1(t-2)
	AR2_2_2	0.01917	0.13575	0.14	0.8882	y2(t-2)
	AR2_2_3	-0.01020	0.16890	-0.06	0.9520	y3(t-2)
y3	CONST3	0.01293	0.00353	3.67	0.0005	1
	AR1_3_1	-0.00242	0.02568	-0.09	0.9251	y1(t-1)
	AR1_3_2	0.22481	0.11168	2.01	0.0482	y2(t-1)
	AR1_3_3	-0.26397	0.13596	-1.94	0.0565	y3(t-1)
	AR2_3_1	0.03388	0.02556	1.33	0.1896	y1(t-2)
	AR2_3_2	0.35491	0.10941	3.24	0.0019	y2(t-2)
	AR2_3_3	-0.02223	0.13612	-0.16	0.8708	y3(t-2)

Output 42.2.4 shows the innovation covariance matrix estimates, the various information criteria results, and the tests for white noise residuals. The residuals are uncorrelated except at lag 3 for $y2$ variable.

Output 42.2.4 Diagnostic Checks

Covariances of Innovations

Variable	y1	y2	y3
y1	0.00213	0.00007	0.00012
y2	0.00007	0.00014	0.00006
y3	0.00012	0.00006	0.00009

Information Criteria	
AICC	-1527.51
HQC	-1536.46
AIC	-1561.11
SBC	-1499.27
FPEC	2.18E-11

Cross Correlations of Residuals

Lag	Variable	y1	y2	y3
0	y1	1.00000	0.13242	0.28275
	y2	0.13242	1.00000	0.55526
	y3	0.28275	0.55526	1.00000
1	y1	0.01461	-0.00666	-0.02394
	y2	-0.01125	-0.00167	-0.04515
	y3	-0.00993	-0.06780	-0.09593
2	y1	0.07253	-0.00226	-0.01621
	y2	-0.08096	-0.01066	-0.02047
	y3	-0.02660	-0.01392	-0.02263
3	y1	0.09915	0.04484	0.05243
	y2	-0.00289	0.14059	0.25984
	y3	-0.03364	0.05374	0.05644

Schematic Representation of Cross Correlations of Residuals

Variable/Lag	0	1	2	3
y1	+.+	...	...	...
y2	.++	...	...	..+
y3	+++	...	...	...

+ is > 2*std error, - is < -2*std error, . is between

Portmanteau Test for Cross Correlations of Residuals

Up To Lag	DF	Chi-Square	Pr > ChiSq
3	9	9.69	0.3766

Output 42.2.5 describes how well each univariate equation fits the data. The residuals are off from the normality, but have no AR effects. The residuals for $y1$ variable have the ARCH effect.

Output 42.2.5 Diagnostic Checks Continued

Univariate Model ANOVA Diagnostics

Variable	R-Square	Standard Deviation	F Value	Pr > F
y1	0.1286	0.04615	1.62	0.1547
y2	0.1142	0.01172	1.42	0.2210
y3	0.2513	0.00944	3.69	0.0032

Univariate Model White Noise Diagnostics

		Normality		ARCH	
Variable	Durbin Watson	Chi-Square	Pr > ChiSq	F Value	Pr > F
y1	1.96269	10.22	0.0060	12.39	0.0008
y2	1.98145	11.98	0.0025	0.38	0.5386
y3	2.14583	34.25	<.0001	0.10	0.7480

Univariate Model AR Diagnostics

	AR1		AR2		AR3		AR4	
Variable	F Value	Pr > F	F Value	Pr > F	F Value	Pr > F	F Value	Pr > F
y1	0.01	0.9029	0.19	0.8291	0.39	0.7624	1.39	0.2481
y2	0.00	0.9883	0.00	0.9961	0.46	0.7097	0.34	0.8486
y3	0.68	0.4129	0.38	0.6861	0.30	0.8245	0.21	0.9320

Output 42.2.6 is the output in a matrix format associated with the PRINT=(IMPULSE=) option for the impulse response function and standard errors. The $y3$ variable in the first row is an impulse variable. The $y1$ variable in the first column is a response variable. The numbers, 0.96122, 0.41555, –0.40789 at lag 1 to 3 are decreasing.

Output 42.2.6 Impulse Response Function

Simple Impulse Response by Variable

Variable Response\Impulse	Lag	y1	y2	y3
y1	**1**	-0.31963	0.14599	0.96122
	STD	0.12546	0.54567	0.66431
	2	-0.05430	0.26174	0.41555
	STD	0.12919	0.54728	0.66311
	3	0.11904	0.35283	-0.40789
	STD	0.08362	0.38489	0.47867
y2	**1**	0.04393	-0.15273	0.28850
	STD	0.03186	0.13857	0.16870
	2	0.02858	0.11377	-0.08820
	STD	0.03184	0.13425	0.16250
	3	-0.00884	0.07147	0.11977
	STD	0.01583	0.07914	0.09462
y3	**1**	-0.00242	0.22481	-0.26397
	STD	0.02568	0.11168	0.13596
	2	0.04517	0.26088	0.10998
	STD	0.02563	0.10820	0.13101
	3	-0.00055	-0.09818	0.09096
	STD	0.01646	0.07823	0.10280

The proportions of decomposition of the prediction error covariances of three variables are given in Output 42.2.7. If you see the $y3$ variable in the first column, then the output explains that about 64.713% of the one-step-ahead prediction error covariances of the variable y_{3t} is accounted for by its own innovations, about 7.995% is accounted for by y_{1t} innovations, and about 27.292% is accounted for by y_{2t} innovations.

Output 42.2.7 Proportions of Prediction Error Covariance Decomposition

Proportions of Prediction Error Covariances by Variable

Variable	Lead	y1	y2	y3
y1	**1**	1.00000	0.00000	0.00000
	2	0.95996	0.01751	0.02253
	3	0.94565	0.02802	0.02633
	4	0.94079	0.02936	0.02985
	5	0.93846	0.03018	0.03136
	6	0.93831	0.03025	0.03145
y2	**1**	0.01754	0.98246	0.00000
	2	0.06025	0.90747	0.03228
	3	0.06959	0.89576	0.03465
	4	0.06831	0.89232	0.03937
	5	0.06850	0.89212	0.03938
	6	0.06924	0.89141	0.03935
y3	**1**	0.07995	0.27292	0.64713
	2	0.07725	0.27385	0.64890
	3	0.12973	0.33364	0.53663
	4	0.12870	0.33499	0.53631
	5	0.12859	0.33924	0.53217
	6	0.12852	0.33963	0.53185

The table in Output 42.2.8 gives forecasts and their prediction error covariances.

Output 42.2.8 Forecasts

Forecasts

Variable	Obs	Time	Forecast	Standard Error	95% Confidence Limits	
y1	**77**	**1979:1**	6.54027	0.04615	6.44982	6.63072
	78	**1979:2**	6.55105	0.05825	6.43688	6.66522
	79	**1979:3**	6.57217	0.06883	6.43725	6.70708
	80	**1979:4**	6.58452	0.08021	6.42732	6.74173
	81	**1980:1**	6.60193	0.09117	6.42324	6.78063
y2	**77**	**1979:1**	7.68473	0.01172	7.66176	7.70770
	78	**1979:2**	7.70508	0.01691	7.67193	7.73822
	79	**1979:3**	7.72206	0.02156	7.67980	7.76431
	80	**1979:4**	7.74266	0.02615	7.69140	7.79392
	81	**1980:1**	7.76240	0.03005	7.70350	7.82130
y3	**77**	**1979:1**	7.54024	0.00944	7.52172	7.55875
	78	**1979:2**	7.55489	0.01282	7.52977	7.58001
	79	**1979:3**	7.57472	0.01808	7.53928	7.61015
	80	**1979:4**	7.59344	0.02205	7.55022	7.63666
	81	**1980:1**	7.61232	0.02578	7.56179	7.66286

Output 42.2.9 shows that you cannot reject Granger noncausality from $(y2, y3)$ to $y1$ using the 0.05 significance level.

Output 42.2.9 Granger Causality Tests

Granger-Causality Wald Test			
Test	DF	Chi-Square	Pr > ChiSq
1	4	6.37	0.1734

Test 1:	Group 1 Variables:	y1
	Group 2 Variables:	y2 y3

The following SAS statements show that the variable $y1$ is the exogenous variable and fit the VARX(2,1) model to the data.

```
proc varmax data=use;
   id date interval=qtr;
   model y2 y3 = y1 / p=2 dify=(1) difx=(1) xlag=1 lagmax=3
                      print=(estimates diagnose);
run;
```

The fitted VARX(2,1) model is written as

$$\begin{pmatrix} \Delta y_{2t} \\ \Delta y_{3t} \end{pmatrix} = \begin{pmatrix} 0.01542 \\ 0.01319 \end{pmatrix} + \begin{pmatrix} 0.02520 \\ 0.05130 \end{pmatrix} \Delta y_{1t} + \begin{pmatrix} 0.03870 \\ 0.00363 \end{pmatrix} \Delta y_{1,t-1} + \begin{pmatrix} -0.12258 & 0.25811 \\ 0.24367 & -0.31809 \end{pmatrix} \begin{pmatrix} \Delta y_{2,t-1} \\ \Delta y_{3,t-1} \end{pmatrix} + \begin{pmatrix} 0.01651 & 0.03498 \\ 0.34921 & -0.01664 \end{pmatrix} \begin{pmatrix} \Delta y_{2,t-2} \\ \Delta y_{3,t-2} \end{pmatrix} + \begin{pmatrix} \epsilon_{1t} \\ \epsilon_{2t} \end{pmatrix}$$

The detailed output is shown in Output 42.2.10 through Output 42.2.13.

Output 42.2.10 shows the parameter estimates in terms of the constant, the current and the lag one coefficients of the exogenous variable, and the lag two coefficients of the dependent variables.

Output 42.2.10 Parameter Estimates

Analysis of German Economic Variables

The VARMAX Procedure

Type of Model	VARX(2,1)
Estimation Method	Least Squares Estimation

Constant

Variable	Constant
y2	0.01542
y3	0.01319

XLag

Lag	Variable	y1
0	**y2**	0.02520
	y3	0.05130
1	**y2**	0.03870
	y3	0.00363

AR

Lag	Variable	y2	y3
1	**y2**	-0.12258	0.25811
	y3	0.24367	-0.31809
2	**y2**	0.01651	0.03498
	y3	0.34921	-0.01664

Output 42.2.11 shows the parameter estimates and their significance.

Output 42.2.11 Parameter Estimates, Continued

Model Parameter Estimates

Equation	Parameter	Estimate	Standard Error	t Value	Pr > |t|	Variable
y2	**CONST1**	0.01542	0.00443	3.48	0.0009	1
	XL0_1_1	0.02520	0.03130	0.81	0.4237	y1(t)
	XL1_1_1	0.03870	0.03252	1.19	0.2383	y1(t-1)
	AR1_1_1	-0.12258	0.13903	-0.88	0.3811	y2(t-1)
	AR1_1_2	0.25811	0.17370	1.49	0.1421	y3(t-1)
	AR2_1_1	0.01651	0.13766	0.12	0.9049	y2(t-2)
	AR2_1_2	0.03498	0.16783	0.21	0.8356	y3(t-2)
y3	**CONST2**	0.01319	0.00346	3.81	0.0003	1
	XL0_2_1	0.05130	0.02441	2.10	0.0394	y1(t)
	XL1_2_1	0.00363	0.02536	0.14	0.8868	y1(t-1)
	AR1_2_1	0.24367	0.10842	2.25	0.0280	y2(t-1)
	AR1_2_2	-0.31809	0.13546	-2.35	0.0219	y3(t-1)
	AR2_2_1	0.34921	0.10736	3.25	0.0018	y2(t-2)
	AR2_2_2	-0.01664	0.13088	-0.13	0.8992	y3(t-2)

Output 42.2.12 shows the innovation covariance matrix estimates, the various information criteria results, and the tests for white noise residuals. The residuals is uncorrelated except at lag 3 for $y2$ variable.

Output 42.2.12 Diagnostic Checks

Covariances of Innovations		
Variable	**y2**	**y3**
y2	0.00014	0.00006
y3	0.00006	0.00009

Information Criteria	
AICC	-1182.33
HQC	-1177.94
AIC	-1193.46
SBC	-1154.52
FPEC	9.91E-9

Cross Correlations of Residuals			
Lag	**Variable**	**y2**	**y3**
0	**y2**	1.00000	0.56462
	y3	0.56462	1.00000
1	**y2**	-0.02312	-0.05927
	y3	-0.07056	-0.09145
2	**y2**	-0.02849	-0.05262
	y3	-0.05804	-0.08567
3	**y2**	0.16071	0.29588
	y3	0.10882	0.13002

Schematic Representation of Cross Correlations of Residuals				
Variable/Lag	**0**	**1**	**2**	**3**
y2	++	. .	. .	. +
y3	++	. .	. .	. .
+ is > 2*std error, - is < -2*std error, . is between				

Portmanteau Test for Cross Correlations of Residuals			
Up To Lag	**DF**	**Chi-Square**	**Pr > ChiSq**
3	4	8.38	0.0787

Output 42.2.13 describes how well each univariate equation fits the data. The residuals are off from the normality, but have no ARCH and AR effects.

Output 42.2.13 Diagnostic Checks Continued

Univariate Model ANOVA Diagnostics

Variable	R-Square	Standard Deviation	F Value	Pr > F
y2	0.0897	0.01188	1.08	0.3809
y3	0.2796	0.00926	4.27	0.0011

Univariate Model White Noise Diagnostics

Variable	Durbin Watson	Normality Chi-Square	Normality Pr > ChiSq	ARCH F Value	ARCH Pr > F
y2	2.02413	14.54	0.0007	0.49	0.4842
y3	2.13414	32.27	<.0001	0.08	0.7782

Univariate Model AR Diagnostics

Variable	AR1 F Value	AR1 Pr > F	AR2 F Value	AR2 Pr > F	AR3 F Value	AR3 Pr > F	AR4 F Value	AR4 Pr > F
y2	0.04	0.8448	0.04	0.9570	0.62	0.6029	0.42	0.7914
y3	0.62	0.4343	0.62	0.5383	0.72	0.5452	0.36	0.8379

Example 42.3: Analysis of Restricted Cointegrated Systems

The structural relationships between economic time series have been of interest for decades. Because of the cointegration, the vector error correction model (VECM), introduced by Engle and Granger (1987), is one of the most important tools for performing such analysis. Although there exist analytical solutions for a nonrestricted VECM and some restricted VECMs in special forms, the estimation of a generally restricted VECM relies on numerical methods. This section illustrates how to use the RESTRICT (or BOUND) and TEST statements, together with the COINTEG statement, to estimate the restricted VECM and perform the statistical tests. For more information about this topic, see Boswijk and Doornik (2004) and references therein.

The data are simulated based on the VECM

$$
\begin{aligned}
\Delta \mathbf{y}_t &= \boldsymbol{\alpha}\boldsymbol{\beta}'\mathbf{y}_{t-1} + \Phi_1^*\Delta\mathbf{y}_{t-1} + \Theta_0^*\mathbf{x}_t + \boldsymbol{\epsilon}_t \\
&= \begin{bmatrix} 0.01 & -0.02 \\ -0.03 & 0.04 \\ 0.05 & -0.06 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \mathbf{y}_{t-1} \\
&\quad + \begin{bmatrix} -0.01 & 0.03 & 0.05 & -0.02 \\ 0.02 & -0.04 & 0.06 & 0.03 \\ 0 & 0 & 0.10 & 0 \\ 0 & 0 & 0 & 0.04 \end{bmatrix} \Delta\mathbf{y}_{t-1} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \mathbf{x}_t + \boldsymbol{\epsilon}_t, \\
\boldsymbol{\epsilon}_t &\sim \text{i.i.d. } N(0, \Sigma), \Sigma = I_4
\end{aligned}
$$

where I_4 is the 4×4 identity matrix.

The following statements implement the simulation:

```
title 'Analysis of Restricted Cointegrated Systems';
proc iml;
   alpha = {0.01 -0.02, -0.03 0.04, 0.05 -0.06, 0 0};
   beta = {1 0, 0 1, -1 0, 0 -1};
   phiStar = {-0.01  0.03 0.05 -0.02,
               0.02 -0.04 0.06  0.03,
                  0     0 0.10     0,
                  0     0    0  0.04};
   Pi = alpha * beta` ;
   A1 = I(4) + Pi+ phiStar;
   A2 = -phiStar;
   phi = A1 // A2;
   sig = I(4);

   /* to simulate the vector time series */
   T = 600;
   myseed = 2;
   call varmasim(y,phi) sigma=sig n=T seed=myseed;
   x = J(T,1,0);
   do i = 1 to T;
      x[i] = normal(myseed);
   end;
   y = y || x;

   cn = {'y1' 'y2' 'y3' 'y4' 'x'};
   create simul5 from y[colname=cn];
   append from y;
   close;
quit;
```

Weak Exogeneity Tests

This example shows different methods for checking weak exogeneity.

The first method uses the EXOGENEITY option in the following statements, and the test results are shown in Output 42.3.1.

```
/* Method 1 -- To use the EXOGENEITY option */
ods output LogLikelihood = tbl_ll_g;
proc varmax data=simul5;
   model y1 y2 y3 y4 = x / noint p=2;
   cointeg rank=2 exogeneity;
run;
```

Output 42.3.1 Test Weak Exogeneity with the EXOGENEITY Option

Analysis of Restricted Cointegrated Systems

The VARMAX Procedure

Testing Weak Exogeneity of Each Variables

Variable	DF	Chi-Square	Pr > ChiSq
y1	2	102.96	<.0001
y2	2	116.12	<.0001
y3	2	200.80	<.0001
y4	2	3.99	0.1357

The second method uses the RESTRICT statement and then the likelihood ratio (LR) test in the following statements. The results are shown in Output 42.3.2. In theory, the first and second methods should have exactly same statistics and p-values because they implement the same LR tests. However, because of the difference between the analytical solution and the numerical solution for the restricted VECM, the statistics are a little different, although for the 0.05 significance level they lead to the same correct conclusion: the variable $y1$ is not the weak exogeneity of variables $y2$, $y3$, and $y4$; the variable $y2$ is not the weak exogeneity of variables $y1$, $y3$, and $y4$; the variable $y3$ is not the weak exogeneity of variables $y1$, $y2$, and $y4$; the variable $y4$ is the weak exogeneity of variables $y1$, $y2$, and $y3$.

```
/* Method 2 -- Use the RESTRICT statement and implement LR test */
%macro LRTestForVECM();
   %do i = 1 %to 4;
      ods output LogLikelihood = tbl_ll_r1_&i.;
      proc varmax data=simul5;
         model y1 y2 y3 y4 = x / noint p=2;
         cointeg rank=2;
         restrict alpha(&i.,1:2) = 0;
      run;
   %end;
   proc iml;
      use tbl_ll_g;
      read all var {nValue1} into ll_g;
      close;
      %do i = 1 %to 4;
         use tbl_ll_r1_&i.;
         read all var {nValue1} into ll_r_&i.;
         close;
      %end;
      DF = J(4,1,2);
      ll_r = ll_r_1 // ll_r_2 // ll_r_3 // ll_r_4;
      Stat = -2*(ll_r - ll_g);
      pValue = 1-cdf("CHISQUARE", Stat, DF);
      Test =  {"H0: Alpha(1,)=0"} // {"H0: Alpha(2,)=0"}
           // {"H0: Alpha(3,)=0"} // {"H0: Alpha(4,)=0"};
      print  Test DF Stat pValue;
   quit;
%mend;
%LRTestForVECM();
```

Output 42.3.2 Test Weak Exogeneity with the RESTRICT Statement and LR Tests

Analysis of Restricted Cointegrated Systems

Test	DF	Stat	pValue
H0: Alpha(1,)=0	2	109.05157	0
H0: Alpha(2,)=0	2	124.56535	0
H0: Alpha(3,)=0	2	238.35505	0
H0: Alpha(4,)=0	2	5.0877699	0.0785606

The third method uses the TEST statement, which implements the Wald tests. Asymptotically, the Wald test has the same distribution as the LR test.

```
/* Method 3 -- To use the TEST statement and the Wald test */
proc varmax data=simul5;
   model y1 y2 y3 y4 = x / noint p=2;
   cointeg rank=2;
   test alpha(1,1:2) = 0;
   test alpha(2,1:2) = 0;
   test alpha(3,1:2) = 0;
   test alpha(4,1:2) = 0;
run;
```

Based on the test results shown in Output 42.3.3, the same correct conclusion can be obtained at the 0.05 significance level: the variable $y1$ is not the weak exogeneity of variables $y2$, $y3$, and $y4$; the variable $y2$ is not the weak exogeneity of variables $y1$, $y3$, and $y4$; the variable $y3$ is not the weak exogeneity of variables $y1$, $y2$, and $y4$; the variable $y4$ is the weak exogeneity of variables $y1$, $y2$, and $y3$.

Output 42.3.3 Test Weak Exogeneity with the TEST Statement, Wald Tests

Analysis of Restricted Cointegrated Systems

The VARMAX Procedure

Testing of the Parameters			
Test	DF	Chi-Square	Pr > ChiSq
1	2	113.27	<.0001
2	2	129.15	<.0001
3	2	245.21	<.0001
4	2	4.81	0.0903

Identification

This example shows how important it is to identify $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ when applying the Wald test on $\boldsymbol{\alpha}$. First, in the following statements, there are no constraints on $\boldsymbol{\beta}$:

```
proc varmax data=simul5;
   model y1 y2 y3 y4 = x / noint p=2;
   cointeg rank=2;
   test alpha(1,2) = alpha(2,2) + alpha(3,2);
run;
```

As shown in Output 42.3.4, based on the test results, the null hypothesis H0: $\boldsymbol{\alpha}[1,2] = \boldsymbol{\alpha}[2,2] + \boldsymbol{\alpha}[3,2]$ should be rejected at the 0.05 significance level, although the true parameter values for the data generating process indicate that H0 is correct.

Output 42.3.4 Importance of Identifying $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ in the Wald Test

Analysis of Restricted Cointegrated Systems

The VARMAX Procedure

Testing of the Parameters			
Test	DF	Chi-Square	Pr > ChiSq
1	1	21.44	<.0001

In the following statements, r^2 constraints are now imposed on $\boldsymbol{\beta}$, where r is the cointegration rank:

```
proc varmax data=simul5;
   model y1 y2 y3 y4 = x / noint p=2;
   cointeg rank=2;
   restrict beta(3:4,1:2) = -I(2);
   test alpha(1,2) = alpha(2,2) + alpha(3,2);
run;
```

As shown in Output 42.3.5, the null hypothesis cannot be rejected at 0.05 significance level; that is to say, the correct conclusion is achieved.

Output 42.3.5 Importance of Identifying $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ in the Wald Test, Continued

Analysis of Restricted Cointegrated Systems

The VARMAX Procedure

Testing of the Parameters			
Test	DF	Chi-Square	Pr > ChiSq
1	1	0.25	0.6157

Besides $\boldsymbol{\alpha}$, other short-run parameters in a VECM can also be tested by using the TEST statement. Because short-run parameters other than $\boldsymbol{\alpha}$ are identified in a VECM, it is not necessary to impose additional constraints on $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. The following statements test the null hypothesis H0: $\Phi_1^* = 0$:

```
proc varmax data=simul5;
   model y1 y2 y3 y4 = x / noint p=2;
   cointeg rank=2;
   test ar(2);
run;
```

According to the results shown in Output 42.3.6, the null hypothesis should be rejected at the 0.05 significance level.

Output 42.3.6 Wald Tests for Short-Run Parameters

Analysis of Restricted Cointegrated Systems

The VARMAX Procedure

Testing of the Parameters			
Test	DF	Chi-Square	Pr > ChiSq
1	16	32.79	0.0079

The following statements test the null hypothesis H0: $\Theta_0^* = 0$:

```
proc varmax data=simul5;
   model y1 y2 y3 y4 = x / noint p=2;
   cointeg rank=2;
   test xl;
run;
```

According to the results shown in Output 42.3.7, the null hypothesis cannot be rejected at the 0.05 significance level.

Output 42.3.7 Wald Tests for Short-Run Parameters, Continued

Analysis of Restricted Cointegrated Systems

The VARMAX Procedure

Testing of the Parameters			
Test	DF	Chi-Square	Pr > ChiSq
1	4	6.01	0.1982

Besides the parameters that are estimated in a VECM, you can also use the TEST statement on $\Pi(= \boldsymbol{\alpha}\boldsymbol{\beta}')$, and $\boldsymbol{\delta}_0$ or $\boldsymbol{\delta}_1$ for Case 2 or 4 when the constant or linear trend, respectively, is restricted in the error correction term. However, keep in mind that the covariance matrix for these parameter estimates is singular when the cointegration rank is less than the number of dependent variables; hence, you might not get the results for some tests.

```
proc varmax data=simul5;
   model y1 y2 y3 y4 = x / noint p=2;
   cointeg rank=2;
   test ar(1,4,1:4);
   test ar(1,4,{1 3});
run;
```

As shown in Output 42.3.8, the first test on H0: $\Pi[4,] = 0$ cannot be calculated, whereas the second test on H0: $\Pi[4,1] = \Pi[4,3] = 0$ can be.

Output 42.3.8 Wald Tests for Π

Analysis of Restricted Cointegrated Systems

The VARMAX Procedure

Testing of the Parameters			
Test	DF	Chi-Square	Pr > ChiSq
1	4		
2	2	4.81	0.0903

Tests for Long-Run Parameter

This example focuses on testing the relationships on the long-run parameter $\boldsymbol{\beta}$. Here, only the following specific form of hypothesis is discussed,

$$\text{H0}: \boldsymbol{\beta} = (\mathbf{H}, \boldsymbol{\phi})$$

where $\mathbf{H}$ is a known $k \times r_1$ matrix, $\boldsymbol{\phi}$ is a freely varying $k \times (r - r_1)$ parameter matrix, k is the number of dependent variables, r is the cointegration rank, and $0 \leq r_1 \leq r$. Other forms of hypothesis—for example, H0: $\boldsymbol{\beta} = (\mathbf{H}_1\boldsymbol{\phi}_1, \ldots, \mathbf{H}_r\boldsymbol{\phi}_r)$ or H0: $\mathbf{H}\text{vec}(\boldsymbol{\beta}) = \mathbf{h}$—are omitted, although they can also be implemented in the same logic. The following statements test the null hypothesis that $(1\ 0\ -1\ 0)'$ is in the cointegrating space that is spanned by $\boldsymbol{\beta}$:

```
/* Use the RESTRICT statement and LR test for restrictions on Beta. */
/* H0: Beta = [ H, phi ] where H is known and phi is free */
ods output LogLikelihood = tbl_ll_r2;
proc varmax data=simul5;
   model y1 y2 y3 y4 = x / noint p=2;
   cointeg rank=2;
   restrict beta(,1) = {1, 0, -1, 0};
   nloptions tech=qn maxit=5000;
run;

proc iml;
   use tbl_ll_g;
   read all var {nValue1} into ll_g;
   close;
   use tbl_ll_r2;
   read all var {nValue1} into ll_r;
   close;
   DF = (4-2)*1; /* DF = (k-r)*r_1 */
   Stat = -2*(ll_r - ll_g);
   pValue = 1-cdf("CHISQUARE", stat, df);
   Test = "H0: Beta[1,1:4] = {1 0 -1 0}'";
   print  Test DF Stat pValue;
quit;
```

According to the result shown in Output 42.3.9, the null hypothesis cannot be rejected at the 0.05 significance level.

Output 42.3.9 LR Tests on Long-Run Parameter β

Analysis of Restricted Cointegrated Systems

Test	DF	Stat	pValue
H0: Beta[1,1:4] = {1 0 -1 0}'	2	3.8112302	0.1487311

The following statements test the null hypothesis that the cointegrating space is spanned by $(1\ 0\ -1\ 0, 0\ 1\ 0\ -1)'$:

```
/* H0: Beta = H, where H is the true Beta for DGP */
ods output LogLikelihood = tbl_ll_r3;
proc varmax data=simul5;
   model y1 y2 y3 y4 = x / noint p=2;
   cointeg rank=2;
   restrict beta = I(2) // (-I(2));
   nloptions tech=qn maxit=5000;
run;

proc iml;
   use tbl_ll_g;
   read all var {nValue1} into ll_g;
   close;
   use tbl_ll_r3;
   read all var {nValue1} into ll_r;
   close;
   DF = (4-2)*2; /* DF = (k-r)*r_1 */
   Stat = -2*(ll_r - ll_g);
   pValue = 1-cdf("CHISQUARE", stat, df);
   Test = "H0: Beta = {1 0, 0 1, -1 0, 0 -1}";
   print  Test DF Stat pValue;
quit;
```

According to the result shown in Output 42.3.10, the null hypothesis cannot be rejected at the 0.05 significance level.

Output 42.3.10 LR Tests on Long-Run Parameter β, Continued

Analysis of Restricted Cointegrated Systems

Test	DF	Stat	pValue
H0: Beta = {1 0, 0 1, -1 0, 0 -1}	4	1.5854572	0.8114035

The following statements test the null hypothesis that the cointegrating space is spanned by $(1\ 0\ 1\ 0, 0\ 1\ 0\ 1)'$, the orthogonal matrix to the true β for the data generating process:

```
/* H0: Beta = H, where H is the matrix orthogonal
       to the true Beta for DGP */
ods output LogLikelihood = tbl_ll_r4;
proc varmax data=simul5;
   model y1 y2 y3 y4 = x / noint p=2;
   cointeg rank=2;
   restrict beta = {1 0, 0 1, 1 0, 0 1};
   nloptions tech=qn maxit=5000;
run;

proc iml;
   use tbl_ll_g;
   read all var {nValue1} into ll_g;
   close;
   use tbl_ll_r4;
   read all var {nValue1} into ll_r;
   close;
   DF = (4-2)*2; /* DF = (k-r)*r_1 */
   Stat = -2*(ll_r - ll_g);
   pValue = 1-cdf("CHISQUARE", stat, df);
   Test = "H0: Beta = {1 0, 0 1, 1 0, 0 1}";
   print  Test DF Stat pValue;
quit;
```

According to the result shown in Output 42.3.11, the null hypothesis should be rejected at the 0.05 significance level.

Output 42.3.11 LR Tests on Long-Run Parameter β, Continued

Analysis of Restricted Cointegrated Systems

Test	DF	Stat	pValue
H0: Beta = {1 0, 0 1, 1 0, 0 1}	4	224.99155	0

For the VECM, the BOUND statement can be regarded as an alias of the RESTRICT statement; that is, you can directly replace any RESTRICT statement with a BOUND statement and get the same result. The linear inequality constraints in the restricted cointegrated systems are not discussed in this section, although they are also supported in the BOUND and RESTRICT statements. For more information, see the sections "BOUND Statement" on page 2896 and "RESTRICT Statement" on page 2922.

Obtaining the numerical solution for the restricted VECM is not an easy task in most cases. You might need to use the INITIAL and NLOPTIONS statements to tune the process. For more information, see the sections "INITIAL Statement" on page 2904 and "NLOPTIONS Statement" on page 2921.

Example 42.4: Analysis of Euro Foreign Exchange Reference Rates

This example illustrates how to use and select the VARMA-GARCH model for exchange rates, a general type of financial data. As shown in much of the literature, the financial variables are cross-correlated and autocorrelated not only on first moments, but also on second moments. The VARMA-GARCH model and the vector error correction GARCH model are often used to catch the stylized fact.

The data, downloaded from European Central Bank website (https://www.ecb.europa.eu), consist of four pairs of daily foreign exchange reference rates: the euro and the Australian dollar (AUD), the euro and the British pound sterling (GBP), the euro and the Japanese yen (JPY), and the euro and the US dollar (USD). The full sample covers the period from January 4, 1999, to February 12, 2015 (4,127 days). In the following statements, the series are logarithmically transformed, and the returns (in percentage) are calculated:

```
title 'Analysis of Euro Foreign Exchange Reference Rates';
data eurofxrr;
   input date : MMDDYY10. aud gbp jpy usd;
   label aud='The euro and the Australian dollar'
         usd='The euro and the U.S. dollar'
         jpy='The euro and the Japanese yen'
         gbp='The euro and the British pound sterling';
   logAUD = log(AUD); logGBP = log(GBP);
   logJPY = log(JPY); logUSD = log(USD);
   rAUD = (logAUD - lag(logAUD))*100;
   rGBP = (logGBP - lag(logGBP))*100;
   rJPY = (logJPY - lag(logJPY))*100;
   rUSD = (logUSD - lag(logUSD))*100;
datalines;
01/04/1999    1.9100    0.71110    133.73    1.1789
01/05/1999    1.8944    0.71220    130.96    1.1790
01/06/1999    1.8820    0.70760    131.42    1.1743
01/07/1999    1.8474    0.70585    129.43    1.1632

   ... more lines ...

02/10/2015    1.4522    0.74200    134.67    1.1297
02/11/2015    1.4606    0.73960    135.50    1.1314
02/12/2015    1.4761    0.73760    135.72    1.1328
;
```

Although it is well known that unit roots exist in the exchange rate series and they are not cointegrated, you can use the following statements to verify:

```
/*--- Unit Roots and Cointegration in Log Exchange Rates ---*/

proc varmax data=eurofxrr;
   model logAUD logGBP logJPY logUSD / p=2 dftest cointtest;
run;
```

According to the results of the Dickey-Fuller unit root tests shown in Output 42.4.1, the null hypothesis that there is a unit root in each series cannot be rejected at the 5% significance level. The results of the Johansen cointegration rank trace tests shown in Output 42.4.2 confirm that there is no cointegration between series because the null hypothesis that the cointegration rank is 0, in both unrestricted and restricted cases, cannot be rejected at the 5% significance level. Because there is no cointegration, you do not need to consider vector error correction models; otherwise, the final selected model might be a vector error correction GARCH model, instead of a VARMA-GARCH model.

Output 42.4.1 Dickey-Fuller Unit Root Tests

Analysis of Euro Foreign Exchange Reference Rates

The VARMAX Procedure

Unit Root Test					
Variable	Type	Rho	Pr < Rho	Tau	Pr < Tau
logAUD	Zero Mean	-1.05	0.4644	-1.08	0.2549
	Single Mean	-9.44	0.1549	-2.31	0.1683
	Trend	-13.85	0.2287	-2.63	0.2657
logGBP	Zero Mean	-0.57	0.5554	-0.59	0.4630
	Single Mean	-3.23	0.6297	-1.27	0.6445
	Trend	-11.11	0.3666	-2.27	0.4502
logJPY	Zero Mean	0.00	0.6836	0.02	0.6894
	Single Mean	-6.11	0.3394	-1.73	0.4140
	Trend	-6.56	0.7000	-1.83	0.6901
logUSD	Zero Mean	-1.46	0.4014	-0.88	0.3346
	Single Mean	-3.29	0.6216	-1.27	0.6471
	Trend	-5.76	0.7638	-1.47	0.8394

Output 42.4.2 Johansen Cointegration Rank Trace Tests

Cointegration Rank Test Using Trace						
H0: Rank=r	H1: Rank>r	Eigenvalue	Trace	Pr > Trace	Drift in ECM	Drift in Process
0	0	0.0059	36.6836	0.3601	Constant	Linear
1	1	0.0018	12.1427	0.9269		
2	2	0.0008	4.7724	0.8319		
3	3	0.0003	1.3036	0.2532		

Cointegration Rank Test Using Trace Under Restriction						
H0: Rank=r	H1: Rank>r	Eigenvalue	Trace	Pr > Trace	Drift in ECM	Drift in Process
0	0	0.0060	37.1246	0.6151	Constant	Constant
1	1	0.0018	12.1792	0.9921		
2	2	0.0008	4.7941	0.9855		
3	3	0.0003	1.3041	0.9066		

Before modeling returns, you can test whether unit roots still exist in the differenced data with the following statement:

```
/*--- Unit Roots in Returns and Model Specification ---*/
proc varmax data=eurofxrr;
   model rAUD rGBP rJPY rUSD / p=2 dftest;
   test const; test ar(1); test ar(2);
run;
```

Output 42.4.3 shows that there is no unit root in each differenced series.

Output 42.4.3 Dickey-Fuller Unit Root Tests

Analysis of Euro Foreign Exchange Reference Rates

The VARMAX Procedure

Unit Root Test					
Variable	Type	Rho	Pr < Rho	Tau	Pr < Tau
rAUD	Zero Mean	-4242.7	0.0001	-46.04	<.0001
	Single Mean	-4243.7	0.0001	-46.04	<.0001
	Trend	-4244.2	0.0001	-46.04	<.0001
rGBP	Zero Mean	-4358.4	0.0001	-46.67	<.0001
	Single Mean	-4358.4	0.0001	-46.67	<.0001
	Trend	-4358.5	0.0001	-46.66	<.0001
rJPY	Zero Mean	-4181.4	0.0001	-45.72	<.0001
	Single Mean	-4181.4	0.0001	-45.72	<.0001
	Trend	-4181.9	0.0001	-45.72	<.0001
rUSD	Zero Mean	-4306.8	0.0001	-46.40	<.0001
	Single Mean	-4306.8	0.0001	-46.39	<.0001
	Trend	-4307.4	0.0001	-46.39	<.0001

The preceding statements also test whether the constant and each of two lags of AR terms are 0. The test results are shown in Output 42.4.4.

Output 42.4.4 Tests on Constant and AR Terms

Testing of the Parameters			
Test	DF	Chi-Square	Pr > ChiSq
1	4	0.46	0.9776
2	16	59.42	<.0001
3	16	15.67	0.4759

The null hypothesis that the constant term is 0 and the null hypothesis that the second lag AR term is 0 are both accepted at the 5% significance level. However, the null hypothesis that the first lag AR term is 0 is rejected at the 5% significance level. In the remaining model selection process, only the first lag AR term is considered.

The following statements estimate a zero-mean VAR(1) model and also print some diagnostic results:

```
/*--- VAR Model ---*/

proc varmax data=eurofxrr;
   model rAUD rGBP rJPY rUSD / noint p=1 print=(diagnose);
run;
```

Output 42.4.5 shows the information criteria for the estimated zero-mean VAR(1) model. In this example, AICC is used as the criterion for model selection: the smaller the AICC, the better the model.

Output 42.4.5 Information Criteria for the VAR Model

Analysis of Euro Foreign Exchange Reference Rates

The VARMAX Procedure

Information Criteria	
AICC	-1745.29
HQC	-1687.44
AIC	-1745.64
SBC	-1581.19
FPEC	0.011938

Diagnostics are printed because the PRINT=(DIAGNOSE) option is specified. As shown in Output 42.4.6, the null hypotheses that there is no ARCH effect in each series are all rejected at the 5% significance level.

Output 42.4.6 Tests on ARCH Effects

Univariate Model White Noise Diagnostics

		Normality		ARCH	
Variable	Durbin Watson	Chi-Square	Pr > ChiSq	F Value	Pr > F
rAUD	1.99811	8277.31	<.0001	217.35	<.0001
rGBP	1.99601	2537.71	<.0001	315.25	<.0001
rJPY	2.00007	2456.22	<.0001	149.75	<.0001
rUSD	1.99959	1398.54	<.0001	157.85	<.0001

To find the right GARCH model, you can start with the VAR(1)-CCC-GARCH(1,1) model (which is usually the fastest one to be estimated) as in the following statement:

```
/*--- VAR CCC GARCH Model ---*/

proc varmax data=eurofxrr;
   model rAUD rGBP rJPY rUSD / noint p=1;
   garch p=1 q=1 form=ccc;
run;
```

Compared to the AICC for the zero-mean VAR(1) model (shown in Output 42.4.5), the AICC for VAR(1)-CCC-GARCH(1,1) model, as shown in Output 42.4.7, dramatically decreases, which means that the ARCH effects do play an important role and should be modeled.

Output 42.4.7 Information Criteria for VAR CCC GARCH Model

Analysis of Euro Foreign Exchange Reference Rates

The VARMAX Procedure

Information Criteria	
AICC	-4646.77
HQC	-4571.24
AIC	-4647.35
SBC	-4432.31
FPEC	0.011966

As indicated by its name, a basic assumption of CCC GARCH model is that the conditional correlation is time-invarying, which might not be true. The following statements estimate a BEKK GARCH model to see whether modeling the conditional correlation could improve the model performance:

```
/*---  VAR BEKK GARCH Model ---*/

proc varmax data=eurofxrr outest=oediagbekk;
   model rAUD rGBP rJPY rUSD / noint p=1;
   garch p=1 q=1 form=bekk;
run;
```

As shown in Output 42.4.8, the AICC for VAR BEKK GARCH model does get smaller than the AICC for the CCC GARCH model (shown in Output 42.4.7). The smaller AICC implies that the assumption of the CCC GARCH model might be inaccurate.

Output 42.4.8 Information Criteria for VAR BEKK GARCH Model

Analysis of Euro Foreign Exchange Reference Rates

The VARMAX Procedure

Information Criteria	
AICC	-5667.7
HQC	-5539.55
AIC	-5669.38
SBC	-5302.54
FPEC	0.011979

One shortcoming of the BEKK GARCH model is that it has too many parameters. In practice, especially for a large number of dependent variables, the scalar BEKK GARCH model and the diagonal BEKK GARCH model are often applied, as shown in the following statements. In the RESTRICT statement, matrix operations are used; using matrix operations is much more concise than restricting tens of ARCH and GARCH parameters one by one.

```
/*--- VAR Scalar BEKK GARCH Model ---*/

proc varmax data=eurofxrr outest=oediagbekk;
   model rAUD rGBP rJPY rUSD / noint p=1;
   garch p=1 q=1 form=bekk;
   restrict ach(1)=ach(1,1,1)*I(4), gch(1)=gch(1,1,1)*I(4);
run;

/*--- VAR Diagonal BEKK GARCH Model ---*/

proc varmax data=eurofxrr outest=oediagbekk;
   model rAUD rGBP rJPY rUSD / noint p=1;
   garch p=1 q=1 form=bekk;
   restrict ach(1)=ach(1)#I(4), gch(1)=gch(1)#I(4);
run;
```

The AICCs for the scalar and diagonal BEKK GARCH models are shown in Output 42.4.9 and Output 42.4.10, respectively, and both of them are larger than the AICC for the BEKK GARCH model (shown in Output 42.4.8). Hence, so far, the VAR BEKK GARCH model is the best.

Output 42.4.9 Information Criteria for VAR Scalar BEKK GARCH Model

Analysis of Euro Foreign Exchange Reference Rates

The VARMAX Procedure

Information Criteria	
AICC	-5615.11
HQC	-5552.83
AIC	-5615.51
SBC	-5438.41
FPEC	0.011974

Output 42.4.10 Information Criteria for VAR Diagonal BEKK GARCH Model

Analysis of Euro Foreign Exchange Reference Rates

The VARMAX Procedure

Information Criteria	
AICC	-5630.31
HQC	-5554.78
AIC	-5630.89
SBC	-5415.85
FPEC	0.011978

Another type of multivariate GARCH model that is suitable for modeling the time-varying conditional correlation is the dynamic conditional correlation (DCC) GARCH model, as indicated by its name. The following statements estimate the DCC GARCH model:

```
/*--- VAR DCC GARCH Model ---*/

proc varmax data=eurofxrr;
   model rAUD rGBP rJPY rUSD / noint p=1;
   garch p=1 q=1 form=dcc;
run;
```

As shown in Output 42.4.11, the AICC for the VAR DCC GARCH model is smaller than the AICC for the VAR BEKK GARCH model (shown in Output 42.4.8), implying that the best model should be in the class of DCC GARCH models.

Output 42.4.11 Information Criteria for VAR DCC GARCH Model

Analysis of Euro Foreign Exchange Reference Rates

The VARMAX Procedure

Information Criteria	
AICC	-5689.43
HQC	-5609.5
AIC	-5690.08
SBC	-5462.39
FPEC	0.011973

Could the DCC GARCH model be more parsimonious? The following statements use the sample correlation matrix of the standardized residuals (saving six parameters) to calculate the unconditional correlation matrix in the DCC GARCH model:

```
/*--- Parsimonious VAR DCC GARCH Model ---*/

proc varmax data=eurofxrr;
   model rAUD rGBP rJPY rUSD / noint p=1;
   garch p=1 q=1 form=dcc corrconst=expect;
run;
```

The AICC of the parsimonious VAR DCC GARCH model, as shown in Output 42.4.12, becomes a little smaller. Hence, the best model so far is the parsimonious VAR DCC GARCH model.

Output 42.4.12 Information Criteria for the Parsimonious VAR DCC GARCH Model

Analysis of Euro Foreign Exchange Reference Rates

The VARMAX Procedure

Information Criteria	
AICC	-5694.89
HQC	-5628.19
AIC	-5695.35
SBC	-5505.6
FPEC	0.011973

Another way to refine the model is to try different subforms of GARCH models for each series. The following statements estimate the VAR DCC EGARCH model and produce Output 42.4.13:

```
/*--- VAR DCC EGARCH Model ---*/

proc varmax data=eurofxrr;
   model rAUD rGBP rJPY rUSD / noint p=1;
   garch p=1 q=1 form=dcc subform=egarch corrconst=expect;
   nloptions maxit=5000 pall;
run;
```

The following statements estimate the VAR DCC PGARCH model and produce Output 42.4.14:

```
/*--- VAR DCC PGARCH Model ---*/

proc varmax data=eurofxrr;
   model rAUD rGBP rJPY rUSD / noint p=1;
   garch p=1 q=1 form=dcc subform=pgarch corrconst=expect;
   nloptions maxit=5000 pall;
run;
```

The following statements estimate the VAR DCC QGARCH model and produce Output 42.4.15:

```
/*--- VAR DCC QGARCH Model ---*/

proc varmax data=eurofxrr;
   model rAUD rGBP rJPY rUSD / noint p=1;
   garch p=1 q=1 form=dcc subform=qgarch corrconst=expect;
   nloptions maxit=5000 pall;
run;
```

The following statements estimate the VAR DCC TGARCH model and produce Output 42.4.16:

```
/*--- VAR DCC TGARCH Model ---*/

proc varmax data=eurofxrr;
   model rAUD rGBP rJPY rUSD / noint p=1;
   garch p=1 q=1 form=dcc subform=tgarch corrconst=expect;
   nloptions maxit=5000 pall;
run;
```

Comparing the AICCs shown in Output 42.4.13 through Output 42.4.16, you find that the AICC for the VAR DCC PGARCH model is the smallest. Hence, the best model becomes the zero-mean VAR(1)-DCC-PGARCH(1,1) model, whose unconditional correlation matrix is estimated by the sample correlation matrix of the standardized residuals.

Output 42.4.13 Information Criteria for the Parsimonious VAR DCC EGARCH Model

Analysis of Euro Foreign Exchange Reference Rates

The VARMAX Procedure

Information Criteria	
AICC	-5704.33
HQC	-5628.81
AIC	-5704.92
SBC	-5489.87
FPEC	0.011982

Output 42.4.14 Information Criteria for the Parsimonious VAR DCC PGARCH Model

Analysis of Euro Foreign Exchange Reference Rates

The VARMAX Procedure

Information Criteria	
AICC	-5724.44
HQC	-5640.1
AIC	-5725.16
SBC	-5484.82
FPEC	0.011974

Output 42.4.15 Information Criteria for the Parsimonious VAR DCC QGARCH Model

Analysis of Euro Foreign Exchange Reference Rates

The VARMAX Procedure

Information Criteria	
AICC	-5696.97
HQC	-5621.44
AIC	-5697.55
SBC	-5482.51
FPEC	0.011972

Output 42.4.16 Information Criteria for the Parsimonious VAR DCC TGARCH Model

Analysis of Euro Foreign Exchange Reference Rates

The VARMAX Procedure

Information Criteria	
AICC	-5705.59
HQC	-5630.06
AIC	-5706.17
SBC	-5491.13
FPEC	0.011973

Output 42.4.17 shows that most of the AR parameter estimates in the VAR DCC PGARCH model are not significant.

Output 42.4.17 AR Parameter Estimates for the Parsimonious VAR DCC PGARCH Model

Model Parameter Estimates

Equation	Parameter	Estimate	Standard Error	t Value	Pr > |t|	Variable
rAUD	AR1_1_1	0.05720	0.01790	3.19	0.0014	rAUD(t-1)
	AR1_1_2	0.00036	0.02396	0.02	0.9879	rGBP(t-1)
	AR1_1_3	-0.02307	0.01619	-1.42	0.1543	rJPY(t-1)
	AR1_1_4	0.02007	0.02020	0.99	0.3206	rUSD(t-1)
rGBP	AR1_2_1	0.02684	0.01147	2.34	0.0193	rAUD(t-1)
	AR1_2_2	0.04514	0.01880	2.40	0.0164	rGBP(t-1)
	AR1_2_3	-0.00466	0.01138	-0.41	0.6824	rJPY(t-1)
	AR1_2_4	-0.04649	0.01474	-3.15	0.0016	rUSD(t-1)
rJPY	AR1_3_1	0.05600	0.01846	3.03	0.0024	rAUD(t-1)
	AR1_3_2	-0.05010	0.02697	-1.86	0.0633	rGBP(t-1)
	AR1_3_3	-0.00183	0.01894	-0.10	0.9229	rJPY(t-1)
	AR1_3_4	-0.00841	0.02226	-0.38	0.7055	rUSD(t-1)
rUSD	AR1_4_1	0.03856	0.01513	2.55	0.0108	rAUD(t-1)
	AR1_4_2	0.00552	0.02290	0.24	0.8096	rGBP(t-1)
	AR1_4_3	0.00086	0.01477	0.06	0.9536	rJPY(t-1)
	AR1_4_4	-0.03203	0.02011	-1.59	0.1113	rUSD(t-1)

The following statements test the significance of some parameter estimates:

```
/*--- Significance Of Some Parameter Estimates ---*/

proc varmax data=eurofxrr;
   model rAUD rGBP rJPY rUSD / noint p=1;
   garch p=1 q=1 form=dcc subform=pgarch corrconst=expect;
   nloptions maxit=5000 pall;
   test ar(1, 1, 2:4), ar(1, 2, 3), ar(1, 3, 3:4), ar(1, 4, 2:4);
run;
```

As shown in Output 42.4.18, the null hypothesis that all nine of the parameters in the TEST statement are 0 cannot be rejected at the 5% significance level.

Output 42.4.18 Test on Significance of Some Parameter Estimates

Analysis of Euro Foreign Exchange Reference Rates

The VARMAX Procedure

Testing of the Parameters			
Test	DF	Chi-Square	Pr > ChiSq
1	9	7.37	0.5989

The following statements estimate the VAR DCC PGARCH model without those insignificant parameters:

```
/*--- VAR DCC PGARCH Model w/o Insignificant Parameters ---*/

proc varmax data=eurofxrr;
   model rAUD rGBP rJPY rUSD / noint p=1;
   garch p=1 q=1 form=dcc subform=pgarch corrconst=expect;
   nloptions maxit=5000 pall;
   restrict ar(1, 1, 2:4), ar(1, 2, 3), ar(1, 3, 3:4), ar(1, 4, 2:4);
run;
```

As shown in Output 42.4.19, the AICC does improve and decrease. Further refining the model is possible but beyond the scope of this example. Hence, the best model, according to the AICC, is the zero-mean VAR(1)-DCC-PGARCH(1,1) model without insignificant AR parameters, and its unconditional correlation matrix is estimated by the sample correlation matrix of the standardized residuals.

Output 42.4.19 Information Criteria for the VAR DCC PGARCH Model without Insignificant Parameters

Analysis of Euro Foreign Exchange Reference Rates

The VARMAX Procedure

Information Criteria	
AICC	-5735.05
HQC	-5670.56
AIC	-5735.48
SBC	-5552.06
FPEC	0.011996

This example focuses only on using the information criterion to distinguish models. In practice, the forecast performance of the model might be more important. The VARMAX procedure supports multistep forecasting in both VARMAX-GARCH models and vector error correction GARCH models. Hence, although it is not covered in this example, you can also use the VARMAX procedure and a criterion based on out-of-sample forecast to perform model selection.

Example 42.5: Numerous Examples

The following are examples of syntax for model fitting:

```
/* Data 'a' Generated Process */
proc iml;
   sig = {1.0  0.5, 0.5  1.25};
   phi = {1.2 -0.5, 0.6  0.3};
   call varmasim(y,phi) sigma = sig n = 100 seed = 46859;
   cn = {'y1' 'y2'};
   create a from y[colname=cn];
   append from y;
run;;

/* when the series has a linear trend */
proc varmax data=a;
   model y1 y2 / p=1 trend=linear;
run;

/* Fit subset of AR order 1 and 3 */
proc varmax data=a;
   model y1 y2 / p=(1,3);
run;

/* Check if the series is nonstationary */
proc varmax data=a;
   model y1 y2 / p=1 dftest print=(roots);
run;

/* Fit VAR(1) in differencing */
proc varmax data=a;
   model y1 y2 / p=1 print=(roots) dify=(1);
run;

/* Fit VAR(1) in seasonal differencing */
proc varmax data=a;
   model y1 y2 / p=1 dify=(4) lagmax=5;
run;

/* Fit VAR(1) in both regular and seasonal differencing */
proc varmax data=a;
   model y1 y2 / p=1 dify=(1,4) lagmax=5;
run;

/* Fit VAR(1) in different differencing */
proc varmax data=a;
   model y1 y2 / p=1 dif=(y1(1,4) y2(1)) lagmax=5;
run;

/* Options related to prediction */
proc varmax data=a;
   model y1 y2 / p=1 lagmax=3
                 print=(impulse covpe(5) decompose(5));
```

```
run;

/* Options related to tentative order selection */
proc varmax data=a;
   model y1 y2 / p=1 lagmax=5 minic
                 print=(parcoef pcancorr pcorr);
run;

/* Automatic selection of the AR order */
proc varmax data=a;
   model y1 y2 / minic=(type=aic p=5);
run;

/* Compare results of LS and Yule-Walker Estimators */
proc varmax data=a;
   model y1 y2 / p=1 print=(yw);
run;

/* BVAR(1) of the nonstationary series y1 and y2 */
proc varmax data=a;
   model y1 y2 / p=1
      prior=(lambda=1 theta=0.2 ivar);
run;

/* BVAR(1) of the nonstationary series y1 */
proc varmax data=a;
   model y1 y2 / p=1
      prior=(lambda=0.1 theta=0.15 ivar=(y1));
run;

/* Data 'b' Generated Process */
proc iml;
   sig = { 0.5  0.14 -0.08 -0.03,  0.14 0.71 0.16 0.1,
          -0.08 0.16  0.65  0.23, -0.03 0.1  0.23 0.16};
   sig = sig * 0.0001;
   phi = {1.2 -0.5 0.  0.1,  0.6 0.3 -0.2  0.5,
          0.4  0. -0.2 0.1, -1.0 0.2  0.7 -0.2};
   call varmasim(y,phi) sigma = sig n = 100 seed = 32567;
   cn = {'y1' 'y2' 'y3' 'y4'};
   create b from y[colname=cn];
   append from y;
quit;

/* Cointegration Rank Test using Trace statistics */
proc varmax data=b;
   model y1-y4 / p=2 lagmax=4 cointtest;
run;

/* Cointegration Rank Test using Max statistics */
proc varmax data=b;
   model y1-y4 / p=2 lagmax=4 cointtest=(johansen=(type=max));
run;

/* Common Trends Test using Filter(Differencing) statistics */
```

```
proc varmax data=b;
   model y1-y4 / p=2 lagmax=4 cointtest=(sw);
run;

/* Common Trends Test using Filter(Residual) statistics */
proc varmax data=b;
   model y1-y4 / p=2 lagmax=4 cointtest=(sw=(type=filtres lag=1));
run;

/* Common Trends Test using Kernel statistics */
proc varmax data=b;
   model y1-y4 / p=2 lagmax=4 cointtest=(sw=(type=kernel lag=1));
run;

/* Cointegration Rank Test for I(2) */
proc varmax data=b;
   model y1-y4 / p=2 lagmax=4 cointtest=(johansen=(iorder=2));
run;

/* Fit VECM(2) with rank=3 */
proc varmax data=b;
   model y1-y4 / p=2 lagmax=4 print=(roots iarr);
   cointeg rank=3 normalize=y1;
run;

/* Weak Exogenous Testing for each variable */
proc varmax data=b outstat=bbb;
   model y1-y4 / p=2 lagmax=4;
   cointeg rank=3 exogeneity normalize=y1;
run;

/* Hypotheses Testing for long-run and adjustment parameter */
proc varmax data=b outstat=bbb;
   model y1-y4 / p=2 lagmax=4;
   cointeg rank=3 normalize=y1
      h=(1 0 0, 0 1 0, -1 0 0, 0 0 1)
      j=(1 0 0, 0 1 0, 0 0 1, 0 0 0);
run;

/* ordinary regression model */
proc varmax data=grunfeld;
   model y1 y2 = x1-x3;
run;

/* Ordinary regression model with subset lagged terms */
proc varmax data=grunfeld;
   model y1 y2 = x1 / xlag=(1,3);
run;

/* VARX(1,1) with no current time Exogenous Variables */
proc varmax data=grunfeld;
   model y1 y2 = x1 / p=1 xlag=1 nocurrentx;
run;
```

```
/* VARX(1,1) with different Exogenous Variables */
proc varmax data=grunfeld;
   model y1 = x3, y2 = x1 x2 / p=1 xlag=1;
run;

/* VARX(1,2) in difference with current Exogenous Variables */
proc varmax data=grunfeld;
   model y1 y2 = x1 / p=1 xlag=2 difx=(1) dify=(1);
run;
```

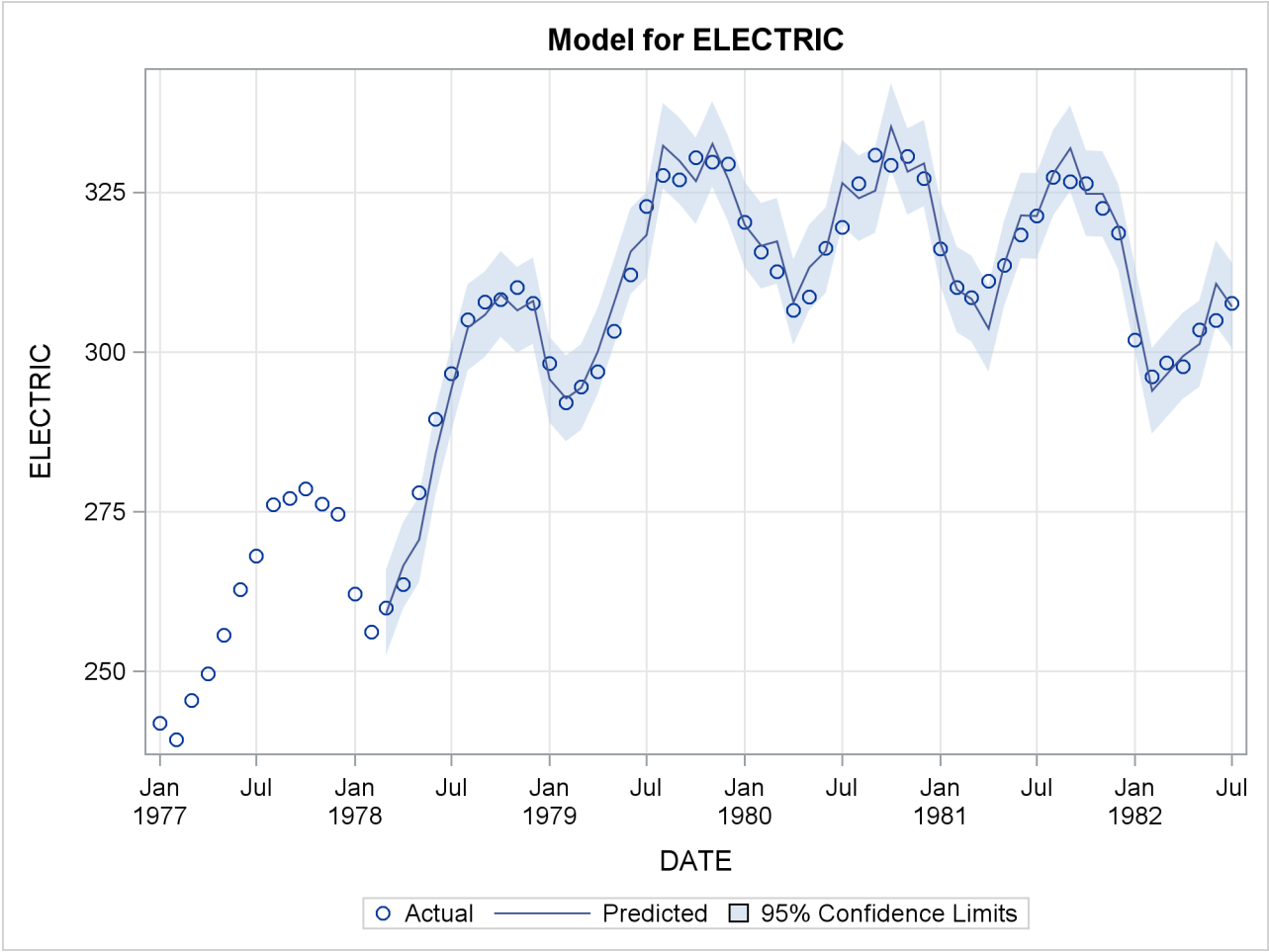
Example 42.6: Illustration of ODS Graphics

This example illustrates the use of ODS Graphics. For information about the graphics available in the VARMAX procedure, see the section "ODS Graphics" on page 3031.

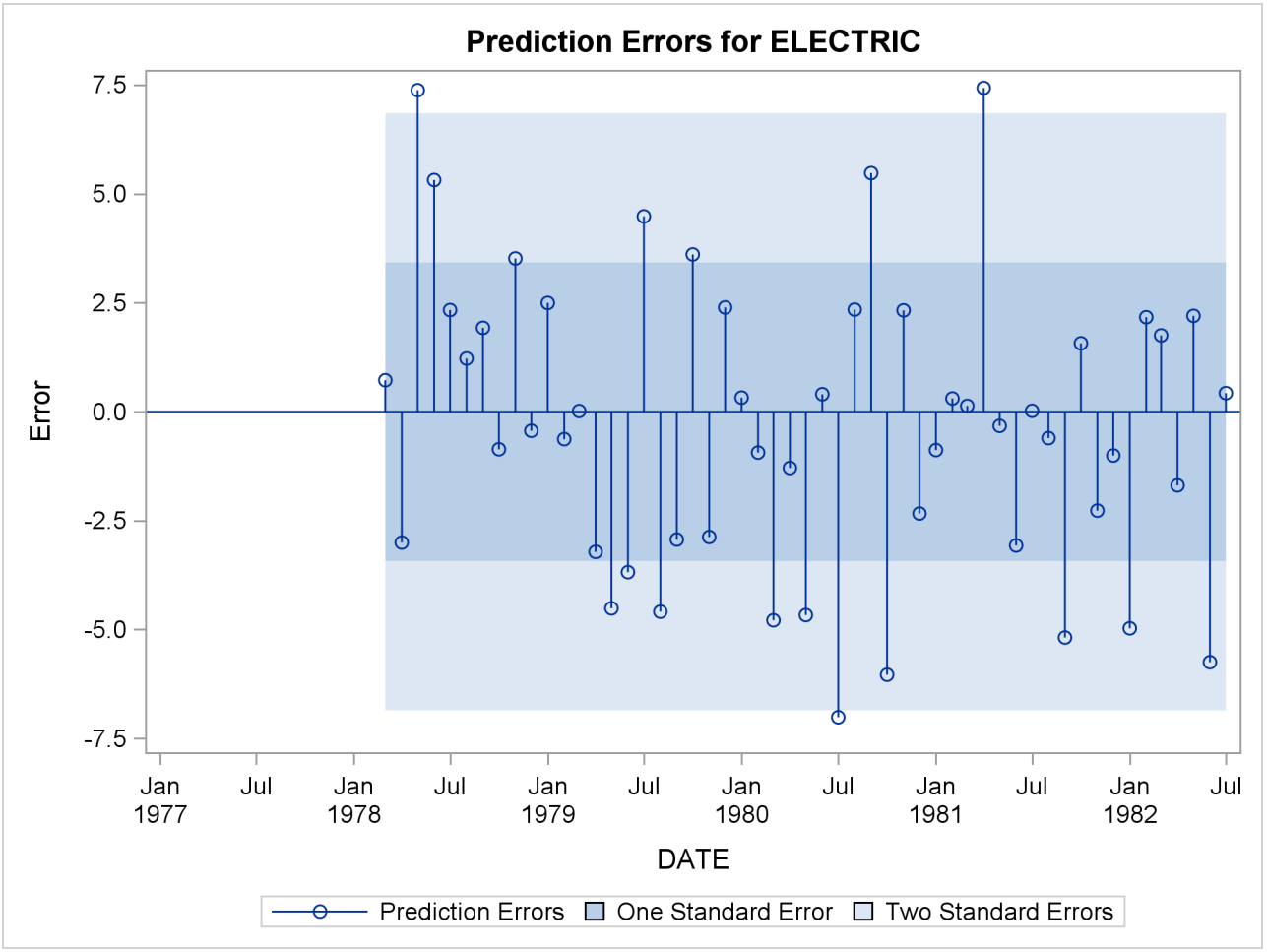
The following statements use the SASHELP.WORKERS data set to study the time series of electrical workers and its interaction with the series of masonry workers. The series and predict plots, the residual plot, and the forecast plot are created in Output 42.6.1 through Output 42.6.3. These are a selection of the plots created by the VARMAX procedure.

```
title "Illustration of ODS Graphics";
proc varmax data=sashelp.workers plot(unpack)=(residual model forecasts);
   id date interval=month;
   model electric masonry / dify=(1,12) noint p=1;
   output lead=12;
run;
```

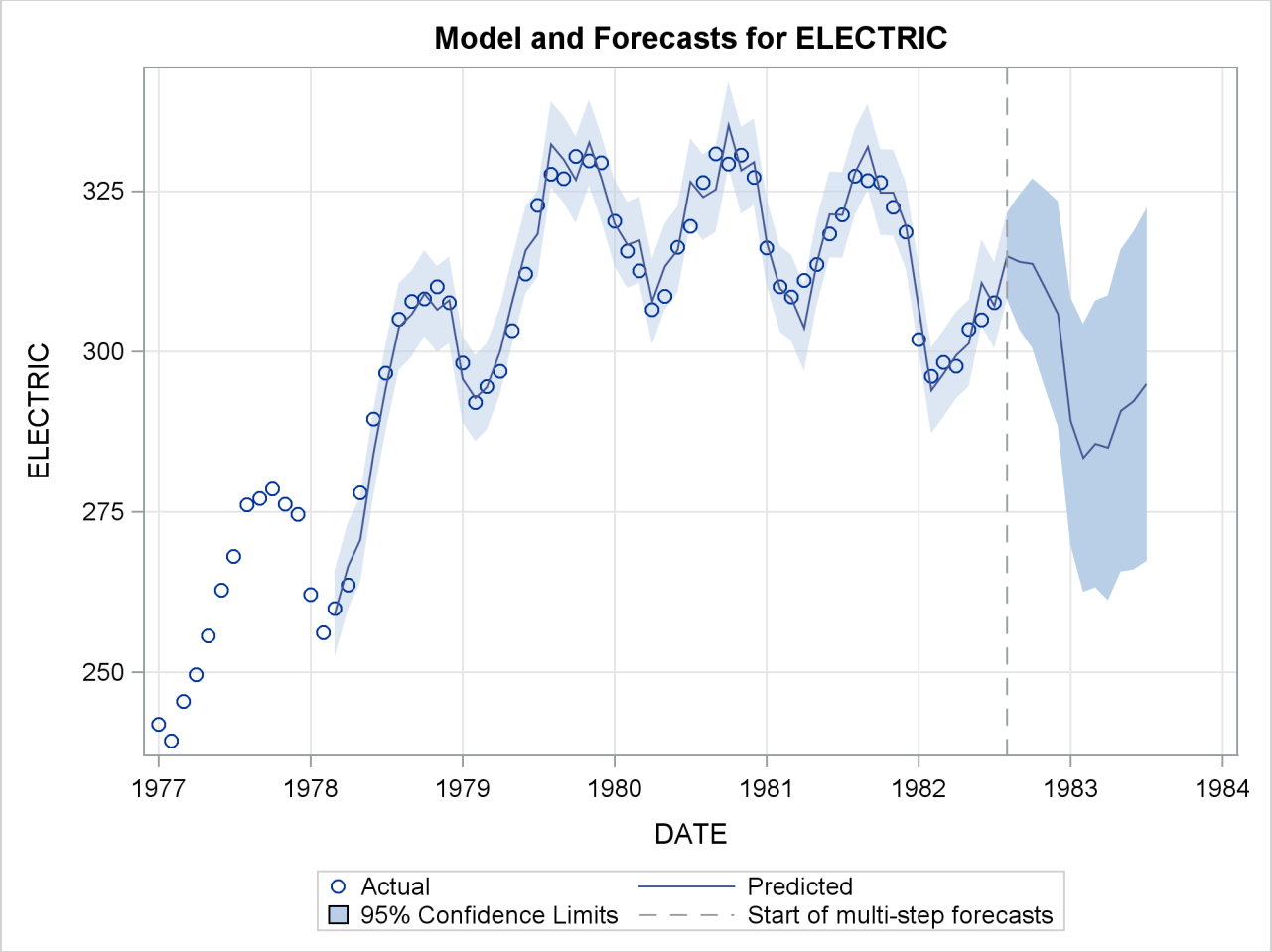
Output 42.6.1 Series and Predicted Series Plots



Output 42.6.2 Residual Plot



Output 42.6.3 Series and Forecast Plots



References

Anderson, T. W. (1951). "Estimating Linear Restrictions on Regression Coefficients for Multivariate Normal Distributions." *Annals of Mathematical Statistics* 22:327–351.

Ansley, C. F., and Newbold, P. (1979). "Multivariate Partial Autocorrelations." In *Proceedings of the Business and Economic Statistics Section*, 349–353. Washington, DC: American Statistical Association.

Bollerslev, T. (1990). "Modeling the Coherence in Short-Run Nominal Exchange Rates: A Multivariate Generalized ARCH Model." *Review of Economics and Statistics* 72:498–505.

Boswijk, H. P., and Doornik, J. A. (2004). "Identifying, Estimating and Testing Restricted Cointegrated Systems: An Overview." *Statistica Neerlandica* 58:440–465.

Ding, Z., Granger, C. W. J., and Engle, R. F. (1993). "A Long Memory Property of Stock Market Returns and a New Model." *Journal of Empirical Finance* 1:83–106.

Engle, R. F. (2002). "Dynamic Conditional Correlation: A Simple Class of Multivariate Generalized Autoregressive Conditional Heteroskedasticity Models." *Journal of Business and Economic Statistics* 20:339–350.

Engle, R. F., and Granger, C. W. J. (1987). "Co-integration and Error Correction: Representation, Estimation, and Testing." *Econometrica* 55:251–276.

Engle, R. F., and Kroner, K. F. (1995). "Multivariate Simultaneous Generalized ARCH." *Econometric Theory* 11:122–150.

Engle, R. F., and Ng, V. K. (1993). "Measuring and Testing the Impact of News on Volatility." *Journal of Finance* 48:1749–1778.

Glosten, L., Jaganathan, R., and Runkle, D. (1993). "Relationship between the Expected Value and Volatility of the Nominal Excess Returns on Stocks." *Journal of Finance* 48:1779–1802.

Golub, G. H., and Van Loan, C. F. (1983). *Matrix Computations*. Baltimore: Johns Hopkins University Press.

Goodnight, J. H. (1979). "A Tutorial on the Sweep Operator." *American Statistician* 33:149–158.

Hosking, J. R. M. (1980). "The Multivariate Portmanteau Statistic." *Journal of the American Statistical Association* 75:602–608.

Johansen, S. (1988). "Statistical Analysis of Cointegration Vectors." *Journal of Economic Dynamics and Control* 12:231–254.

Johansen, S. (1995a). *Likelihood-Based Inference in Cointegrated Vector Autoregressive Models*. New York: Oxford University Press.

Johansen, S. (1995b). "A Statistical Analysis of Cointegration for I(2) Variables." *Econometric Theory* 11:25–59.

Johansen, S., and Juselius, K. (1990). "Maximum Likelihood Estimation and Inference on Cointegration: With Applications to the Demand for Money." *Oxford Bulletin of Economics and Statistics* 52:169–210.

Koreisha, S., and Pukkila, T. (1989). "Fast Linear Estimation Methods for Vector Autoregressive Moving Average Models." *Journal of Time Series Analysis* 10:325–339.

Litterman, R. B. (1986). "Forecasting with Bayesian Vector Autoregressions: Five Years of Experience." *Journal of Business and Economic Statistics* 4:25–38.

Lütkepohl, H. (1993). *Introduction to Multiple Time Series Analysis*. 2nd ed. Berlin: Springer-Verlag.

Lütkepohl, H. (2007). *New Introduction to Multiple Time Series Analysis*. Berlin: Springer.

Nelson, D. B. (1991). "Conditional Heteroskedasticity in Asset Returns: A New Approach." *Econometrica* 59:347–370.

Nelson, D. B., and Cao, C. Q. (1992). "Inequality Constraints in the Univariate GARCH Model." *Journal of Business and Economic Statistics* 10:229–235.

Osterwald-Lenum, M. (1992). "A Note with Quantiles of the Asymptotic Distribution of the Maximum Likelihood Cointegration Rank Test Statistics." *Oxford Bulletin of Economics and Statistics* 54:461–472.

Pringle, R. M., and Rayner, A. A. (1971). *Generalized Inverse Matrices with Applications to Statistics*. New York: Hafner Publishing.

Quinn, B. G. (1980). "Order Determination for a Multivariate Autoregression." *Journal of the Royal Statistical Society, Series B* 42:182–185.

Reinsel, G. C. (1997). *Elements of Multivariate Time Series Analysis*. 2nd ed. New York: Springer-Verlag.

Spliid, H. (1983). "A Fast Estimation for the Vector Autoregressive Moving Average Models with Exogenous Variables." *Journal of the American Statistical Association* 78:843–849.

Stock, J. H., and Watson, M. W. (1988). "Testing for Common Trends." *Journal of the American Statistical Association* 83:1097–1107.

Zakoian, J. M. (1994). "Threshold Heteroscedastic Models." *Journal of Economic Dynamics and Control* 18:931–955.

Chapter 43
The X11 Procedure

Contents

Overview: X11 Procedure **3086**
Getting Started: X11 Procedure **3086**
Basic Seasonal Adjustment 3087
X-11-ARIMA 3090
Syntax: X11 Procedure **3091**
Functional Summary 3092
PROC X11 Statement 3093
ARIMA Statement 3094
BY Statement 3097
ID Statement 3097
MACURVES Statement 3097
MONTHLY Statement 3098
OUTPUT Statement 3101
PDWEIGHTS Statement 3102
QUARTERLY Statement 3103
SSPAN Statement 3105
TABLES Statement 3106
VAR Statement 3106
Details: X11 Procedure **3106**
Historical Development of X-11 3106
Implementation of the X-11 Seasonal Adjustment Method 3108
Computational Details for Sliding Spans Analysis 3111
Data Requirements 3114
Missing Values 3114
Prior Daily Weights and Trading-Day Regression 3114
Adjustment for Prior Factors 3115
The YRAHEADOUT Option 3116
Effect of Backcast and Forecast Length 3116
Details of Model Selection 3117
OUT= Data Set 3119
The OUTSPAN= Data Set 3120
OUTSTB= Data Set 3120
OUTTDR= Data Set 3121
Printed Output 3122
ODS Table Names 3132
Examples: X11 Procedure **3137**

Example 43.1: Component Estimation—Monthly Data 3137
Example 43.2: Components Estimation—Quarterly Data 3141
Example 43.3: Outlier Detection and Removal 3143
References **3145**

Overview: X11 Procedure

The X11 procedure, an adaptation of the U.S. Bureau of the Census X-11 Seasonal Adjustment program, seasonally adjusts monthly or quarterly time series. The procedure makes additive or multiplicative adjustments and creates an output data set containing the adjusted time series and intermediate calculations.

The X11 procedure also provides the X-11-ARIMA method developed by Statistics Canada. This method fits an ARIMA model to the original series, then uses the model forecast to extend the original series. This extended series is then seasonally adjusted by the standard X-11 seasonal adjustment method. The extension of the series improves the estimation of the seasonal factors and reduces revisions to the seasonally adjusted series as new data become available.

The X11 procedure incorporates sliding spans analysis. This type of analysis provides a diagnostic for determining the suitability of seasonal adjustment for an economic series.

Seasonal adjustment of a series is based on the assumption that seasonal fluctuations can be measured in the original series, $O_t, t = 1, \ldots, n$, and separated from trend cycle, trading-day, and irregular fluctuations. The seasonal component of this time series, S_t, is defined as the intrayear variation that is repeated constantly or in an evolving fashion from year to year. The trend cycle component, C_t, includes variation due to the long-term trend, the business cycle, and other long-term cyclical factors. The trading-day component, D_t, is the variation that can be attributed to the composition of the calendar. The irregular component, I_t, is the residual variation. Many economic time series are related in a multiplicative fashion ($O_t = S_t C_t D_t I_t$). A seasonally adjusted time series, $C_t I_t$, consists of only the trend cycle and irregular components.

Getting Started: X11 Procedure

The most common use of the X11 procedure is to produce a seasonally adjusted series. Eliminating the seasonal component from an economic series facilitates comparison among consecutive months or quarters. A plot of the seasonally adjusted series is often more informative about trends or location in a business cycle than a plot of the unadjusted series.

The following example shows how to use PROC X11 to produce a seasonally adjusted series, $C_t I_t$, from an original series $O_t = S_t C_t D_t I_t$.

In the multiplicative model, the trend cycle component C_t keeps the same scale as the original series O_t, while S_t, D_t, and I_t vary around 1.0. In all printed tables and in the output data set, these latter components are expressed as percentages, and thus will vary around 100.0 (in the additive case, they vary around 0.0).

The naming convention used in PROC X11 for the tables follows the original U.S. Bureau of the Census X-11 Seasonal Adjustment program specification (Shiskin, Young, and Musgrave 1967). Also, see the section "Printed Output" on page 3122. This convention is outlined in Figure 43.1.

The tables corresponding to parts A – C are intermediate calculations. The final estimates of the individual components are found in the D tables: table D10 contains the final seasonal factors, table D12 contains the final trend cycle, and table D13 contains the final irregular series. If you are primarily interested in seasonally adjusting a series without consideration of intermediate calculations or diagnostics, you only need to look at table D11, the final seasonally adjusted series.

For further details about the X-11-ARIMA tables, see Ladiray and Quenneville (2001).

Basic Seasonal Adjustment

Suppose you have monthly retail sales data starting in September 1978 in a SAS data set named SALES. At this point you do not suspect that any calendar effects are present, and there are no prior adjustments that need to be made to the data.

In this simplest case, you need only specify the DATE= variable in the MONTHLY statement, which associates a SAS date value to each observation. To see the results of the seasonal adjustment, you must request table D11, the final seasonally adjusted series, in a TABLES statement.

```
data sales;
   input sales @@;
   date = intnx( 'month', '01sep1978'd, _n_-1 );
   format date monyy7.;
datalines;
112 118 132 129 121 135 148 148 136 119 104 118

   ... more lines ...


/*--- X-11 ARIMA ---*/
proc x11 data=sales;
   monthly date=date;
   var sales;
   tables d11;
run;
```

Figure 43.1 Basic Seasonal Adjustment

The X11 Procedure

Seasonal Adjustment of - sales

X-11 Seasonal Adjustment Program
U. S. Bureau of the Census
Economic Research and Analysis Division
November 1, 1968

The X-11 program is divided into seven major parts.
Part Description
A. Prior adjustments, if any
B. Preliminary estimates of irregular component weights and regression trading day factors
C. Final estimates of above
D. Final estimates of seasonal, trend-cycle and irregular components
E. Analytical tables
F. Summary measures
G. Charts

Series - sales
Period covered - 9/1978 to 8/1990
Type of run: multiplicative seasonal adjustment.
Selected Tables or Charts.
Sigma limits for graduating extreme values are 1.5 and 2.5
Irregular values outside of 2.5-sigma limits are excluded from trading day regression

Figure 43.2 Basic Seasonal Adjustment

D11 Final Seasonally Adjusted Series

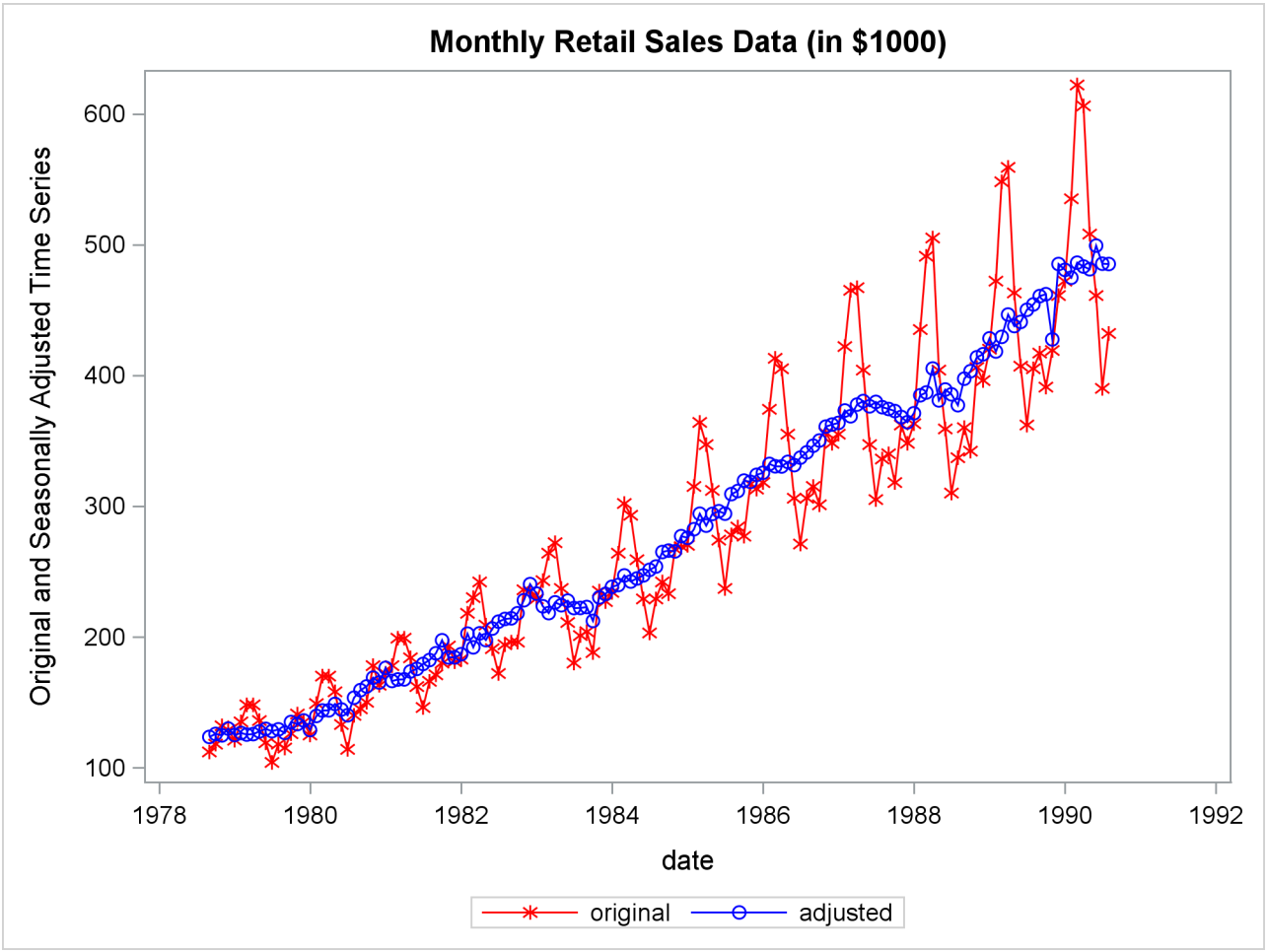
Year	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	Total
1978	.	.	.	.	.	.	.	.	123.507	125.776	124.735	129.870	503.887
1979	124.935	126.533	125.282	125.650	127.754	129.648	127.880	129.285	126.562	134.905	133.356	136.117	1547.91
1980	128.734	139.542	143.726	143.854	148.723	144.530	140.120	153.475	159.281	162.128	168.848	165.159	1798.12
1981	176.329	166.264	167.433	167.509	173.573	175.541	179.301	182.254	187.448	197.431	184.341	184.304	2141.73
1982	186.747	202.467	192.024	202.761	197.548	206.344	211.690	213.691	214.204	218.060	228.035	240.347	2513.92
1983	233.109	223.345	218.179	226.389	224.249	227.700	222.045	222.127	222.835	212.227	230.187	232.827	2695.22
1984	238.261	239.698	246.958	242.349	244.665	247.005	251.247	253.805	264.924	266.004	265.366	277.025	3037.31
1985	275.766	282.316	294.169	285.034	294.034	296.114	294.196	309.162	311.539	319.518	318.564	323.921	3604.33
1986	325.471	332.228	330.401	330.282	333.792	331.349	337.095	341.127	346.173	350.183	360.792	362.333	4081.23
1987	363.592	373.118	368.670	377.650	380.316	376.297	379.668	375.607	374.257	372.672	368.135	364.150	4474.13
1988	370.966	384.743	386.833	405.209	380.840	389.132	385.479	377.147	397.404	403.156	413.843	416.142	4710.89
1989	428.276	418.236	429.409	446.467	437.639	440.832	450.103	454.176	460.601	462.029	427.499	485.113	5340.38
1990	480.631	474.669	486.137	483.140	481.111	499.169	485.370	485.103	.	.	.	.	3875.33
Avg	277.735	280.263	282.435	286.358	285.354	288.638	288.683	291.413	265.728	268.674	268.642	276.442	

Total: 40324 Mean: 280.03 S.D.: 111.31

You can compare the original series, table B1, and the final seasonally adjusted series, table D11, by plotting them together. These tables are requested and named in the OUTPUT statement.

```
title 'Monthly Retail Sales Data (in $1000)';

proc x11 data=sales noprint;
   monthly date=date;
   var sales;
   output out=out b1=sales d11=adjusted;
run;

proc sgplot data=out;
   series x=date y=sales    / markers
                              markerattrs=(color=red symbol='asterisk')
                              lineattrs=(color=red)
                              legendlabel="original" ;
   series x=date y=adjusted / markers
                              markerattrs=(color=blue symbol='circle')
                              lineattrs=(color=blue)
                              legendlabel="adjusted" ;
   yaxis label='Original and Seasonally Adjusted Time Series';
run;
```

Figure 43.3 Plot of Original and Seasonally Adjusted Data



X-11-ARIMA

An inherent problem with the X-11 method is the revision of the seasonal factor estimates as new data become available. The X-11 method uses a set of centered moving averages to estimate the seasonal components. These moving averages apply symmetric weights to all observations except those at the beginning and end of the series, where asymmetric weights have to be applied. These asymmetric weights can cause poor estimates of the seasonal factors, which then can cause large revisions when new data become available.

While large revisions to seasonally adjusted values are not common, they can happen. When they do happen, it undermines the credibility of the X-11 seasonal adjustment method.

A method to address this problem was developed at Statistics Canada (Dagum 1980, 1982a). This method, known as X-11-ARIMA, applies an ARIMA model to the original data (after adjustments, if any) to forecast the series one or more years. This extended series is then seasonally adjusted, allowing symmetric weights to be applied to the end of the original data. This method was tested against a large number of Canadian economic series and was found to greatly reduce the amount of revisions as new data were added.

The X-11-ARIMA method is available in PROC X11 through the use of the ARIMA statement. The ARIMA statement extends the original series either with a user-specified ARIMA model or by an automatic selection process in which the best model from a set of five predefined ARIMA models is used.

The following example illustrates the use of the ARIMA statement. The ARIMA statement does not contain a user-specified model, so the best model is chosen by the automatic selection process. Forecasts from this best model are then used to extend the original series by one year. The following partial listing shows parameter estimates and model diagnostics for the ARIMA model chosen by the automatic selection process.

```
proc x11 data=sales;
   monthly date=date;
   var sales;
   arima;
run;
```

Figure 43.4 X-11-ARIMA Model Selection

Monthly Retail Sales Data (in $1000)

The X11 Procedure

Seasonal Adjustment of - sales

Conditional Least Squares Estimation

Parameter	Estimate	Approx. Std Error	t Value	Lag
MU	0.0001728	0.0009596	0.18	0
MA1,1	0.3739984	0.0893427	4.19	1
MA1,2	0.0231478	0.0892154	0.26	2
MA2,1	0.5727914	0.0790835	7.24	12

Figure 43.4 *continued*

Conditional Least Squares Estimation		
Variance Estimate =	0.0014313	
Std Error Estimate =	0.0378326	
AIC =	-482.2412	*
SBC =	-470.7404	*
Number of Residuals=	131	
*** Does not include log determinant**		

Criteria Summary for Model 2: (0,1,2)(0,1,1)s, Log Transform

Box-Ljung Chi-square: 22.03 with 21 df Prob= 0.40
(Criteria prob > 0.05)
Test for over-differencing: sum of MA parameters = 0.57
(must be < 0.90)
MAPE - Last Three Years: 2.84 (Must be < 15.00 %)
- Last Year: 3.04
- Next to Last Year: 1.96
- Third from Last Year: 3.51

Table D11 (final seasonally adjusted series) is now constructed using symmetric weights on observations at the end of the actual data. This should result in better estimates of the seasonal factors and, thus, smaller revisions in Table D11 as more data become available.

Syntax: X11 Procedure

The X11 procedure uses the following statements:

PROC X11 *options* ;
ARIMA *options* ;
BY *variables* ;
ID *variables* ;
MACURVES *option* ;
MONTHLY *options* ;
OUTPUT *OUT=dataset options* ;
PDWEIGHTS *option* ;
QUARTERLY *options* ;
SSPAN *options* ;
TABLES *tablenames* ;
VAR *variables* ;

Either the MONTHLY or QUARTERLY statement must be specified, depending on the type of time series data you have. The PDWEIGHTS and MACURVES statements can be used only with the MONTHLY statement. The TABLES statement controls the printing of tables, while the OUTPUT statement controls the creation of the OUT= data set.

Functional Summary

The statements and options controlling the X11 procedures are summarized in the following table.

Description	Statement	Option
Data Set Options		
specify input data set	PROC X11	DATA=
write the trading-day regression results to an output data set	PROC X11	OUTTDR=
write the stable seasonality test results to an output data set	PROC X11	OUTSTB=
write table values to an output data set	OUTPUT	OUT=
add extrapolated values to the output data set	PROC X11	OUTEX
add year ahead estimates to the output data set	PROC X11	YRAHEADOUT
write the sliding spans analysis results to an output data set	PROC X11	OUTSPAN=
Printing Control Options		
suppress all printed output	PROC X11	NOPRINT
suppress all printed ARIMA output	ARIMA	NOPRINT
print all ARIMA output	ARIMA	PRINTALL
print selected tables and charts	TABLES	
print selected groups of tables	MONTHLY	PRINTOUT=
	QUARTERLY	PRINTOUT=
print selected groups of charts	MONTHLY	CHARTS=
	QUARTERLY	CHARTS=
print preliminary tables associated with ARIMA processing	ARIMA	PRINTFP
specify number of decimals for printed tables	MONTHLY	NDEC=
	QUARTERLY	NDEC=
suppress all printed SSPAN output	SSPAN	NOPRINT
print all SSPAN output	SSPAN	PRINTALL
Date Information Options		
specify a SAS date variable	MONTHLY	DATE=
	QUARTERLY	DATE=
specify the beginning date	MONTHLY	START=
	QUARTERLY	START=
specify the ending date	MONTHLY	END=
	QUARTERLY	END=
specify beginning year for trading-day regression	MONTHLY	TDCOMPUTE=

Description	Statement	Option
Declaring the Role of Variables		
specify BY-group processing	BY	
specify the variables to be seasonally adjusted	VAR	
specify identifying variables	ID	
specify the prior monthly factor	MONTHLY	PMFACTOR=
Controlling the Table Computations		
use additive adjustment	MONTHLY	ADDITIVE
	QUARTERLY	ADDITIVE
specify seasonal factor moving average length	MACURVES	
specify the extreme value limit for trading-day regression	MONTHLY	EXCLUDE=
specify the lower bound for extreme irregulars	MONTHLY	FULLWEIGHT=
	QUARTERLY	FULLWEIGHT=
specify the upper bound for extreme irregulars	MONTHLY	ZEROWEIGHT=
	QUARTERLY	ZEROWEIGHT=
include the length-of-month in trading-day regression	MONTHLY	LENGTH
specify trading-day regression action	MONTHLY	TDREGR=
compute summary measure only	MONTHLY	SUMMARY
	QUARTERLY	SUMMARY
modify extreme irregulars prior to trend	MONTHLY	TRENDADJ
cycle estimation	QUARTERLY	TRENDADJ
specify moving average length in trend	MONTHLY	TRENDMA=
cycle estimation	QUARTERLY	TRENDMA=
specify weights for prior trading-day factors	PDWEIGHTS	

PROC X11 Statement

PROC X11 *options* **;**

The following options can appear in the PROC X11 statement:

DATA= *SAS-data-set*
specifies the input SAS data set used. If it is omitted, the most recently created SAS data set is used.

OUTEXTRAP
adds the extra observations used in ARIMA processing to the output data set.

When ARIMA forecasting/backcasting is requested, extra observations are appended to the ends of the series, and the calculations are carried out on this extended series. The appended observations are not normally written to the OUT= data set. However, if OUTEXTRAP is specified, these extra observations are written to the output data set. If a DATE= variable is specified in the MONTHLY/QUARTERLY statement, the date variable is extrapolated to identify forecasts/backcasts. The OUTEXTRAP option can be abbreviated as OUTEX.

NOPRINT
suppresses any printed output. The NOPRINT option overrides any PRINTOUT=, CHARTS=, or TABLES statement and any output associated with the ARIMA statement.

OUTSPAN= *SAS-data-set*
specifies the output data set to store the sliding spans analysis results. Tables A1, C18, D10, and D11 for each span are written to this data set. See the section "The OUTSPAN= Data Set" on page 3120 for details.

OUTSTB= *SAS-data-set*
specifies the output data set to store the stable seasonality test results (table D8). All the information in the analysis of variance table associated with the stable seasonality test is contained in the variables written to this data set. See the section "OUTSTB= Data Set" on page 3120 for details.

OUTTDR= *SAS-data-set*
specifies the output data set to store the trading-day regression results (tables B15 and C15). All the information in the analysis of variance table associated with the trading-day regression is contained in the variables written to this data set. This option is valid only when TDREGR=PRINT, TEST, or ADJUST is specified in the MONTHLY statement. See the section "OUTTDR= Data Set" on page 3121 for details.

YRAHEADOUT
adds one-year-ahead forecast values to the output data set for tables C16, C18, and D10. The original purpose of this option was to avoid recomputation of the seasonal adjustment factors when new data became available. While computing costs were an important factor when the X-11 method was developed, this is no longer the case and this option is obsolete. See the section "The YRAHEADOUT Option" on page 3116 for details.

ARIMA Statement

ARIMA *options* **;**

The ARIMA statement applies the X-11-ARIMA method to the series specified in the VAR statement. This method uses an ARIMA model estimated from the original data to extend the series one or more years. The ARIMA statement options control the ARIMA model used and the estimation, forecasting, and printing of this model.

There are two ways of obtaining an ARIMA model to extend the series. A model can be given explicitly with the MODEL= and TRANSFORM= options. Alternatively, the best-fitting model from a set of five predefined models is found automatically whenever the MODEL= option is absent. See the section "Details of Model Selection" on page 3117 for details.

BACKCAST= *n*
specifies the number of years to backcast the series. The default is BACKCAST= 0. See the section "Effect of Backcast and Forecast Length" on page 3116 for details.

CHICR= *value*

specifies the criteria for the significance level for the Box-Ljung chi-square test for lack of fit when testing the five predefined models. The default is CHICR= 0.05. The CHICR= option values must be between 0.01 and 0.90. The hypothesis being tested is that of model adequacy. Nonrejection of the hypothesis is evidence for an adequate model. Making the CHICR= value smaller makes it easier to accept the model. See the section "Criteria Details" on page 3118 for further details on the CHICR= option.

CONVERGE= *value*

specifies the convergence criterion for the estimation of an ARIMA model. The default value is 0.001. The CONVERGE= value must be positive.

FORECAST= *n*

specifies the number of years to forecast the series. The default is FORECAST= 1. See the section "Effect of Backcast and Forecast Length" on page 3116 for details.

MAPECR= *value*

specifies the criteria for the mean absolute percent error (MAPE) when testing the five predefined models. A small MAPE value is evidence for an adequate model; a large MAPE value results in the model being rejected. The MAPECR= value is the boundary for acceptance/rejection. Thus a larger MAPECR= value would make it easier for a model to pass the criteria. The default is MAPECR= 15. The MAPECR= option values must be between 1 and 100. See the section "Criteria Details" on page 3118 for further details on the MAPECR= option.

MAXITER= *n*

specifies the maximum number of iterations in the estimation process. MAXITER must be between 1 and 60; the default value is 15.

METHOD= CLS

METHOD= ULS

METHOD= ML

specifies the estimation method. ML requests maximum likelihood, ULS requests unconditional least squares, and CLS requests conditional least squares. METHOD=CLS is the default. The maximum likelihood estimates are more expensive to compute than the conditional least squares estimates. In some cases, however, they can be preferable. For further information on the estimation methods, see "Estimation Details" on page 272 in Chapter 8, "The ARIMA Procedure."

MODEL= (P=*n1* **Q=***n2* **SP=***n3* **SQ=***n4* **DIF=***n5* **SDIF=***n6* **< NOINT > < CENTER >)**

specifies the ARIMA model. The AR and MA orders are given by P=*n1* and Q=*n2*, respectively, while the seasonal AR and MA orders are given by SP=*n3* and SQ=*n4*, respectively. The lag corresponding to seasonality is determined by the MONTHLY or QUARTERLY statement. Similarly, differencing and seasonal differencing are given by DIF=*n5* and SDIF=*n6*, respectively.

For example

```
arima model=( p=2 q=1 sp=1 dif=1 sdif=1 );
```

specifies a (2,1,1)(1,1,0)*s* model, where *s*, the seasonality, is either 12 (monthly) or 4 (quarterly). More examples of the MODEL= syntax are given in the section "Details of Model Selection" on page 3117.

NOINT
suppresses the fitting of a constant (or intercept) parameter in the model. (That is, the parameter μ is omitted.)

CENTER
centers each time series by subtracting its sample mean. The analysis is done on the centered data. Later, when forecasts are generated, the mean is added back. Note that centering is done after differencing. The CENTER option is normally used in conjunction with the NOCONSTANT option of the ESTIMATE statement.

For example, to fit an AR(1) model on the centered data without an intercept, use the following ARIMA statement:

```
arima model=( p=1 center noint );
```

NOPRINT
suppresses the normal printout generated by the ARIMA statement. Note that the effect of specifying the NOPRINT option in the ARIMA statement is different from the effect of specifying the NOPRINT in the PROC X11 statement, since the former only affects ARIMA output.

OVDIFCR= *value*
specifies the criteria for the over-differencing test when testing the five predefined models. When the MA parameters in one of these models sum to a number close to 1.0, this is an indication of over-parameterization and the model is rejected. The OVDIFCR= value is the boundary for this rejection; values greater than this value fail the over-differencing test. A larger OVDIFCR= value would make it easier for a model to pass the criteria. The default is OVDIFCR= 0.90. The OVDIFCR= option values must be between 0.80 and 0.99. See the section "Criteria Details" on page 3118 for further details on the OVDIFCR= option.

PRINTALL
provides the same output as the default printing for all models fit and, in addition, prints an estimation summary and chi-square statistics for each model fit. See "Printed Output" on page 3122 for details.

PRINTFP
prints the results for the initial pass of X11 made to exclude trading-day effects. This option has an effect only when the TDREGR= option specifies ADJUST, TEST, or PRINT. In these cases, an initial pass of the standard X11 method is required to get rid of calendar effects before doing any ARIMA estimation. Usually this first pass is not of interest, and by default no tables are printed. However, specifying PRINTFP in the ARIMA statement causes any tables printed in the final pass to also be printed for this initial pass.

TRANSFORM= (LOG) | LOG

TRANSFORM= (*constant ** power* **)**
The ARIMA statement in PROC X11 allows certain transformations on the series before estimation. The specified transformation is applied only to a user-specified model. If TRANSFORM= is specified and the MODEL= option is not specified, the transformation request is ignored and a warning is printed.

The LOG transformation requests that the natural log of the series be used for estimation. The resulting forecast values are transformed back to the original scale.

A general power transformation of the form $X_t \rightarrow (X_t + a)^b$ is obtained by specifying

```
transform= ( a ** b )
```

If the constant a is not specified, it is assumed to be zero. The specified ARIMA model is then estimated using the transformed series. The resulting forecast values are transformed back to the original scale.

BY Statement

BY *variables* **;**

A BY statement can be used with PROC X11 to obtain separate analyses on observations in groups defined by the BY variables. When a BY statement appears, the procedure expects the input DATA= data set to be sorted in order of the BY variables.

ID Statement

ID *variables* **;**

If you are creating an output data set, use the ID statement to put values of the ID variables, in addition to the table values, into the output data set. The ID statement has no effect when an output data set is not created. If the DATE= variable is specified in the MONTHLY or QUARTERLY statement, this variable is included automatically in the OUTPUT data set. If no DATE= variable is specified, the variable _DATE_ is added.

The date variable (or _DATE_) values outside the range of the actual data (from ARIMA forecasting or backcasting, or from YRAHEADOUT) are extrapolated, while all other ID variables are missing.

MACURVES Statement

MACURVES *month=option . . .* **;**

The MACURVES statement specifies the length of the moving-average curves for estimating the seasonal factors for any month. This statement can be used only with monthly time series data.

The *month=option* specifications consist of the month name (or the first three letters of the month name), an equal sign, and one of the following option values:

’3’	specifies a three-term moving average for the month
’3X3’	specifies a three-by-three moving average
’3X5’	specifies a three-by-five moving average
’3X9’	specifies a three-by-nine moving average
STABLE	specifies a stable seasonal factor (average of all values for the month)

For example, the statement

```
macurves jan='3' feb='3x3' march='3x5' april='3x9';
```

uses a three-term moving average to estimate seasonal factors for January, a 3×3 (a three-term moving average of a three-term moving average) for February, a 3×5 (a three-term moving average of a five-term moving average) for March, and a 3×9 (a three-term moving average of a nine-term moving average) for April.

The numeric values used for the weights of the various moving averages and a discussion of the derivation of these weights are given in Shiskin, Young, and Musgrave (1967). A general discussion of moving average weights is given in Dagum (1985).

If the specification for a month is omitted, the X11 procedure uses a three-by-three moving average for the first estimate of each iteration and a three-by-five average for the second estimate.

MONTHLY Statement

MONTHLY *options* **;**

The MONTHLY statement must be used when the input data to PROC X11 are a monthly time series. The MONTHLY statement specifies options that determine the computations performed by PROC X11 and what is included in its output. Either the DATE= or START= option must be used.

The following options can appear in the MONTHLY statement.

ADDITIVE
performs additive adjustments. If the ADDITIVE option is omitted, PROC X11 performs multiplicative adjustments.

CHARTS= STANDARD

CHARTS= FULL

CHARTS= NONE
specifies the charts produced by the procedure. The default is CHARTS=STANDARD, which specifies 12 monthly seasonal charts and a trend cycle chart. If you specify CHARTS=FULL (or CHARTS=ALL), the procedure prints additional charts of irregular and seasonal factors. To print no charts, specify CHARTS=NONE.

The TABLES statement can also be used to specify particular monthly charts to be printed. If no CHARTS= option is given, and a TABLES statement is given, the TABLES statement overrides the default value of CHARTS=STANDARD; that is, no charts (or tables) are printed except those specified in the TABLES statement. However, if both the CHARTS= option and a TABLES statement are given, the charts corresponding to the CHARTS= option and those requested by the TABLES statement are printed.

For example, suppose you wanted only charts G1, the final seasonally adjusted series and trend cycle, and G4, the final irregular and final modified irregular series. You would specify the following statements:

```
monthly date=date;
tables g1 g4;
```

DATE= *variable*

specifies a variable that gives the date for each observation. The starting and ending dates are obtained from the first and last values of the DATE= variable, which must contain SAS date values. The procedure checks values of the DATE= variable to ensure that the input observations are sequenced correctly. This variable is automatically added to the OUTPUT= data set if one is requested and extrapolated if necessary. If the DATE= option is not specified, the START= option must be specified.

The DATE= option and the START= and END= options can be used in combination to subset a series for processing. For example, suppose you have 12 years of monthly data (144 observations, no missing values) beginning in January 1970 and ending in December 1981, and you wanted to seasonally adjust only six years beginning in January 1974. Specifying

```
monthly date=date start=jan1974 end=dec1979;
```

would seasonally adjust only this subset of the data. If instead you wanted to adjust the last eight years of data, only the START= option is needed:

```
monthly date=date start=jan1974;
```

END= *mmmyyyy*

specifies that only the part of the input series ending with the month and year given be adjusted (for example, END=DEC1970). See the DATE=*variable* option for using the START= and END= options to subset a series for processing.

EXCLUDE= *value*

excludes from the trading-day regression any irregular values that are more than *value* standard deviations from the mean. The EXCLUDE=value must be between 0.1 and 9.9, with the default value being 2.5.

FULLWEIGHT= *value*

assigns weights to irregular values based on their distance from the mean in standard deviation units. The weights are used for estimating seasonal and trend cycle components. Irregular values less than the FULLWEIGHT= *value* (in standard deviation units) are assigned full weights of 1, values that fall between the ZEROWEIGHT= and FULLWEIGHT= limits are assigned weights linearly graduated between 0 and 1, and values greater than the ZEROWEIGHT= limit are assigned a weight of 0.

For example, if ZEROWEIGHT=2 and FULLWEIGHT=1, a value 1.3 standard deviations from the mean would be assigned a graduated weight. The FULLWEIGHT=value must be between 0.1 and 9.9 but must be less than the ZEROWEIGHT=value. The default is FULLWEIGHT=1.5.

LENGTH

includes length-of-month allowance in computing trading-day factors. If this option is omitted, length-of-month allowances are included with the seasonal factors.

NDEC= *n*

specifies the number of decimal places shown in the printed tables in the listing. This option has no effect on the precision of the variable values in the output data set.

PMFACTOR= *variable*

specifies a variable containing the prior monthly factors. Use this option if you have previous knowledge of monthly adjustment factors. The PMFACTOR= option can be used to make the following adjustments:

- adjust the level of all or part of a series with discontinuities
- adjust for the influence of holidays that fall on different dates from year to year, such as the effect of Easter on certain retail sales
- adjust for unreasonable weather influence on series, such as housing starts
- adjust for changing starting dates of fiscal years (for budget series) or model years (for automobiles)
- adjust for temporary dislocating events, such as strikes

See the section "Prior Daily Weights and Trading-Day Regression" on page 3114 for details and examples using the PMFACTOR= option.

PRINTOUT= STANDARD | LONG | FULL | NONE

specifies the tables to be printed by the procedure. If the PRINTOUT=STANDARD option is specified, between 17 and 27 tables are printed, depending on the other options that are specified. PRINTOUT=LONG prints between 27 and 39 tables, and PRINTOUT=FULL prints between 44 and 59 tables. Specifying PRINTOUT=NONE results in no tables being printed; however, charts are still printed. The default is PRINTOUT=STANDARD.

The TABLES statement can also be used to specify particular monthly tables to be printed. If no PRINTOUT= option is specified, and a TABLES statement is given, the TABLES statement overrides the default value of PRINTOUT=STANDARD; that is, no tables (or charts) are printed except those given in the TABLES statement. However, if both the PRINTOUT= option and a TABLES statement are specified, the tables corresponding to the PRINTOUT= option and those requested by the TABLES statement are printed.

START= *mmmyyyy*

adjusts only the part of the input series starting with the specified month and year. When the DATE= option is not used, the START= option gives the year and month of the first input observation — for example, START=JAN1966. START= must be specified if DATE= is not given. If START= is specified (and no DATE= option is given), and an OUT= data set is requested, a variable named _DATE_ is added to the data set, giving the date value for each observation. See the DATE= *variable* option for using the START= and END= options to subset a series.

SUMMARY

specifies that the data are already seasonally adjusted and the procedure is to produce summary measures. If the SUMMARY option is omitted, the X11 procedure performs seasonal adjustment of the input data before calculating summary measures.

TDCOMPUTE= *year*

uses the part of the input series beginning with January of the specified year to derive trading-day weights. If this option is omitted, the entire series is used.

TDREGR= NONE | PRINT | ADJUST | TEST

specifies the treatment of trading-day regression. TDREG=NONE omits the computation of the trading-day regression. TDREG=PRINT computes and prints the trading-day regressions but does not adjust the series. TDREG=ADJUST computes and prints the trading-day regression and adjusts the irregular components to obtain preliminary weights. TDREG=TEST adjusts the final series if the trading-day regression estimates explain significant variation on the basis of an F test (or residual trading-day variation if prior weights are used). The default is TDREGR=NONE.

See the section "Prior Daily Weights and Trading-Day Regression" on page 3114 for details and examples using the TDREGR= option.

If ARIMA processing is requested, any value of TDREGR other than the default TDREGR=NONE will cause PROC X11 to perform an initial pass (see the "Details: X11 Procedure" on page 3106 section and the PRINTFP option).

The significance level reported in Table C15 should be viewed with caution. The dependent variable in the trading-day regression is the irregular component formed by an averaging operation. This induces a correlation in the dependent variable and hence in the residuals from which the F test is computed. Hence the distribution of the trading-day regression F statistics differs from an exact F; see Cleveland and Devlin (1980) for details.

TRENDADJ

modifies extreme irregular values prior to computing the trend cycle estimates in the first iteration. If the TRENDADJ option is omitted, the trend cycle is computed without modifications for extremes.

TRENDMA= 9 | 13 | 23

specifies the number of terms in the moving average to be used by the procedure in estimating the variable trend cycle component. The value of the TRENDMA= option must be 9, 13, or 23. If the TRENDMA= option is omitted, the procedure selects an appropriate moving average. For information about the number of terms in the moving average, see Shiskin, Young, and Musgrave (1967).

ZEROWEIGHT= *value*

assigns weights to irregular values based on their distance from the mean in standard deviation units. The weights are used for estimating seasonal and trend cycle components. Irregular values beyond the standard deviation limit specified in the ZEROWEIGHT= option are assigned zero weights. Values that fall between the two limits (ZEROWEIGHT= and FULLWEIGHT=) are assigned weights linearly graduated between 0 and 1. For example, if ZEROWEIGHT=2 and FULLWEIGHT=1, a value 1.3 standard deviations from the mean would be assigned a graduated weight. The ZEROWEIGHT=value must be between 0.1 and 9.9 but must be greater than the FULLWEIGHT=value. The default is ZEROWEIGHT=2.5.

The ZEROWEIGHT option can be used in conjunction with the FULLWEIGHT= option to adjust outliers from a monthly or quarterly series. See Example 43.3 later in this chapter for an illustration of this use.

OUTPUT Statement

OUTPUT OUT= *SAS-data-set tablename=var1 var2 . . .* **;**

The OUTPUT statement creates an output data set containing specified tables. The data set is named by the OUT= option.

OUT= *SAS-data-set*

If OUT= is omitted, the SAS System names the new data set by using the DATA*n* convention.

For each table to be included in the output data set, write the X11 table identification keyword, an equal sign, and a list of new variable names:

tablename = var1 var2 ...

The *tablename* keywords that can be used in the OUTPUT statement are listed in the section "Printed Output" on page 3122. The following is an example of a VAR statement and an OUTPUT statement:

```
var z1 z2 z3;
output out=out_x11  b1=s  d11=w x y;
```

The variable s contains the table B1 values for the variable z1, while the table D11 values for variables z1, z2, and z3 are contained in variables w, x, and y, respectively. As this example shows, the list of variables following a *tablename*= keyword can be shorter than the VAR variable list.

In addition to the variables named by *tablename* =*var1 var2* . . . , the ID variables, and BY variables, the output data set contains a date identifier variable. If the DATE= option is given in the MONTHLY or QUARTERLY statement, the DATE= variable is the date identifier. If no DATE= option is given, a variable named _DATE_ is the date identifier.

PDWEIGHTS Statement

PDWEIGHTS *day=w . . .* **;**

The PDWEIGHTS statement can be used to specify one to seven daily weights. The statement can only be used with monthly series that are seasonally adjusted using the multiplicative model. These weights are used to compute prior trading-day factors, which are then used to adjust the original series prior to the seasonal adjustment process. Only relative weights are needed; the X11 procedure adjusts the weights so that they sum to 7.0. The weights can also be corrected by the procedure on the basis of estimates of trading-day variation from the input data.

See the section "Prior Daily Weights and Trading-Day Regression" on page 3114 for details and examples using the PDWEIGHTS statement.

Each *day*=*w* option specifies a weight (*w*) for the named day. The *day* can be any day, Sunday through Saturday. The *day* keyword can be the full spelling of the day, or the three-letter abbreviation. For example, SATURDAY=1.0 and SAT=1.0 are both valid. The weights *w* must be a numeric value between 0.0 and 10.0.

The following is an example of a PDWEIGHTS statement:

```
pdweights sun=.2 mon=.9 tue=1 wed=1 thu=1 fri=.8 sat=.3;
```

Any number of days can be specified with one PDWEIGHTS statement. The default weight value for any day that is not specified is 0. If you do not use a PDWEIGHTS statement, the program computes daily weights if TDREGR=ADJUST is specified. See Shiskin, Young, and Musgrave (1967) for details.

QUARTERLY Statement

QUARTERLY *options* **;**

The QUARTERLY statement must be used when the input data are quarterly time series. This statement includes options that determine the computations performed by the procedure and what is in the printed output. The DATE= option or the START= option must be used.

The following options can appear in the QUARTERLY statement.

ADDITIVE
performs additive adjustments. If this option is omitted, the procedure performs multiplicative adjustments.

CHARTS= STANDARD

CHARTS= FULL

CHARTS= NONE
specifies the charts to be produced by the procedure. The default value is CHARTS=STANDARD, which specifies four quarterly seasonal charts and a trend cycle chart. If you specify CHARTS=FULL (or CHARTS=ALL), the procedure prints additional charts of irregular and seasonal factors. To print no charts, specify CHARTS=NONE. The TABLES statement can also be used to specify particular charts to be printed. The presence of a TABLES statement overrides the default value of CHARTS=STANDARD; that is, if a TABLES statement is specified, and no CHARTS=option is specified, no charts (nor tables) are printed except those given in the TABLES statement. However, if both the CHARTS= option and a TABLES statement are given, the charts corresponding to the CHARTS= option and those requested by the TABLES statement are printed.

For example, suppose you wanted only charts G1, the final seasonally adjusted series and trend cycle, and G4, the final irregular and final modified irregular series. This is accomplished by specifying the following statements:

```
quarterly date=date;
tables g1 g4;
```

DATE= *variable*
specifies a variable that gives the date for each observation. The starting and ending dates are obtained from the first and last values of the DATE= variable, which must contain SAS date values. The procedure checks values of the DATE= variable to ensure that the input observations are sequenced correctly. This variable is automatically added to the OUTPUT= data set if one is requested, and extrapolated if necessary. If the DATE= option is not specified, the START= option must be specified.

The DATE= option and the START= and END= options can be used in combination to subset a series for processing. For example, suppose you have a series with 10 years of quarterly data (40 observations, no missing values) beginning in '1970Q1' and ending in '1979Q4', and you want to seasonally adjust only four years beginning in '1974Q1' and ending in '1977Q4'. Specifying

```
quarterly date=variable start='1974q1' end='1977q4';
```

seasonally adjusts only this subset of the data. If instead you wanted to adjust the last six years of data, only the START= option is needed:

```
quarterly date=variable start='1974q1';
```

END= *'yyyyQq'*
specifies that only the part of the input series ending with the quarter and year given be adjusted (for example, END='1973Q4'). The specification must be enclosed in quotes and *q* must be 1, 2, 3, or 4. See the DATE= *variable* option for using the START= and END= options to subset a series.

FULLWEIGHT= *value*
assigns weights to irregular values based on their distance from the mean in standard deviation units. The weights are used for estimating seasonal and trend cycle components. Irregular values less than the FULLWEIGHT= value (in standard deviation units) are assigned full weights of 1, values that fall between the ZEROWEIGHT= and FULLWEIGHT= limits are assigned weights linearly graduated between 0 and 1, and values greater than the ZEROWEIGHT= limit are assigned a weight of 0.

For example, if ZEROWEIGHT=2 and FULLWEIGHT=1, a value 1.3 standard deviations from the mean would be assigned a graduated weight. The default is FULLWEIGHT=1.5.

NDEC= *n*
specifies the number of decimal places shown on the output tables. This option has no effect on the precision of the variables in the output data set.

PRINTOUT= STANDARD

PRINTOUT= LONG

PRINTOUT= FULL

PRINTOUT= NONE
specifies the tables to print. If PRINTOUT=STANDARD is specified, between 17 and 27 tables are printed, depending on the other options that are specified. PRINTOUT=LONG prints between 27 and 39 tables, and PRINTOUT=FULL prints between 44 and 59 tables. Specifying PRINTOUT=NONE results in no tables being printed. The default is PRINTOUT=STANDARD.

The TABLES statement can also specify particular quarterly tables to be printed. If no PRINTOUT= is given, and a TABLES statement is given, the TABLES statement overrides the default value of PRINTOUT=STANDARD; that is, no tables (or charts) are printed except those given in the TABLES statement. However, if both the PRINTOUT= option and a TABLES statement are given, the tables corresponding to the PRINTOUT= option and those requested by the TABLES statement are printed.

START= *'yyyyQq'*
adjusts only the part of the input series starting with the quarter and year given. When the DATE= option is not used, the START= option gives the year and quarter of the first input observation (for example, START='1967Q1'). The specification must be enclosed in quotes, and *q* must be 1, 2, 3, or 4. START= must be specified if the DATE= option is not given. If START= is specified (and no DATE= is given), and an OUTPUT= data set is requested, a variable named _DATE_ is added to the data set, giving the date value for a given observation. See the DATE= option for using the START= and END= options to subset a series.

SUMMARY
specifies that the input is already seasonally adjusted and that the procedure is to produce summary measures. If this option is omitted, the procedure performs seasonal adjustment of the input data before calculating summary measures.

TRENDADJ
modifies extreme irregular values prior to computing the trend cycle estimates. If this option is omitted, the trend cycle is computed without modification for extremes.

ZEROWEIGHT= *value*
assigns weights to irregular values based on their distance from the mean in standard deviation units. The weights are used for estimating seasonal and trend cycle components. Irregular values beyond the standard deviation limit specified in the ZEROWEIGHT= option are assigned zero weights. Values that fall between the two limits (ZEROWEIGHT= and FULLWEIGHT=) are assigned weights linearly graduated between 0 and 1. For example, if ZEROWEIGHT=2 and FULLWEIGHT=1, a value 1.3 standard deviations from the mean would be assigned a graduated weight. The default is ZEROWEIGHT=2.5.

The ZEROWEIGHT option can be used in conjunction with the FULLWEIGHT= option to adjust outliers from a monthly or quarterly series. See Example 43.3 later in this chapter for an illustration of this use.

SSPAN Statement

SSPAN *options* **;**

The SSPAN statement applies sliding spans analysis to determine the suitability of seasonal adjustment for an economic series.

The following options can appear in the SSPAN statement:

NDEC= *n*
specifies the number of decimal places shown on selected sliding span reports. This option has no effect on the precision of the variables values in the OUTSPAN output data set.

CUTOFF= *value*
gives the percentage value for determining an excessive difference within a span for the seasonal factors, the seasonally adjusted series, and month-to-month and year-to-year differences in the seasonally adjusted series. The default value is 3.0. The use of the CUTOFF=value in determining the maximum percent difference (MPD) is described in the section "Computational Details for Sliding Spans Analysis" on page 3111. Caution should be used in changing the default CUTOFF=value. The empirical threshold ranges found by the U.S. Census Bureau no longer apply when value is changed.

TDCUTOFF= *value*
gives the percentage value for determining an excessive difference within a span for the trading-day factors. The default value is 2.0. The use of the TDCUTOFF=value in determining the maximum percent difference (MPD) is described in the section "Computational Details for Sliding Spans Analysis" on page 3111. Caution should be used in changing the default TDCUTOFF=value. The empirical threshold ranges found by the U.S. Census Bureau no longer apply when the value is changed.

NOPRINT
suppresses all sliding span reports. See "Computational Details for Sliding Spans Analysis" on page 3111 for more details on sliding span reports.

PRINT
prints the summary sliding span reports S 0 through S 6.E.

PRINTALL
prints the summary sliding spans report S 0 through S 6.E, along with detail reports S 7.A through S 7.E.

TABLES Statement

TABLES *tablenames* **;**

The TABLES statement prints the tables specified in addition to the tables that are printed as a result of the PRINTOUT= option in the MONTHLY or QUARTERLY statement. Table names are listed in Table 43.4 later in this chapter.

To print only selected tables, omit the PRINTOUT= option in the MONTHLY or QUARTERLY statement and list the tables to be printed in the TABLES statement. For example, to print only the final seasonal factors and final seasonally adjusted series, use the statement

```
tables d10 d11;
```

VAR Statement

VAR *variables* **;**

The VAR statement is used to specify the variables in the input data set that are to be analyzed by the procedure. Only numeric variables can be specified. If the VAR statement is omitted, all numeric variables are analyzed except those appearing in a BY or ID statement or the variable named in the DATE= option in the MONTHLY or QUARTERLY statement.

Details: X11 Procedure

Historical Development of X-11

This section briefly describes the historical development of the standard X-11 seasonal adjustment method and the later development of the X-11-ARIMA method. Most of the following discussion is based on a comprehensive article by Bell and Hillmer (1984), which describes the history of X-11 and the justification of using seasonal adjustment methods, such as X-11, given the current availability of time series software. For further discussions about statistical problems associated with the X-11 method, see Ghysels (1990).

Seasonal adjustment methods began to be developed in the 1920s and 1930s, before there were suitable analytic models available and before electronic computing devices were in existence. The lack of any suitable model led to methods that worked the same for any series — that is, methods that were not model-based and

that could be applied to any series. Experience with economic series had shown that a given mathematical form could adequately represent a time series only for a fixed length; as more data were added, the model became inadequate. This suggested an approach that used moving averages. For further analysis of the properties of X-11 moving averages, see Cleveland and Tiao (1976).

The basic method was to break up an economic time series into long-term trend, long-term cyclical movements, seasonal movements, and irregular fluctuations.

Early investigators found that it was not possible to uniquely decompose the trend and cycle components. Thus, these two were grouped together; the resulting component is usually referred to as the "trend cycle component."

It was also found that estimating seasonal components in the presence of trend produced biased estimates of the seasonal components, but, at the same time, estimating trend in the presence of seasonality was difficult. This eventually lead to the iterative approach used in the X-11 method.

Two other problems were encountered by early investigators. First, some economic series appear to have changing or evolving seasonality. Secondly, moving averages were very sensitive to extreme values. The estimation method used in the X-11 method allows for evolving seasonal components. For the second problem, the X-11 method uses repeated adjustment of extreme values.

All of these problems encountered in the early investigation of seasonal adjustment methods suggested the use of moving averages in estimating components. Even with the use of moving averages instead of a model-based method, massive amounts of hand calculations were required. Only a small number of series could be adjusted, and little experimentation could be done to evaluate variations on the method.

With the advent of electronic computing in the 1950s, work on seasonal adjustment methods proceeded rapidly. These methods still used the framework previously described; variants of these basic methods could now be easily tested against a large number of series.

Much of the work was done by Julian Shiskin and others at the U.S. Bureau of the Census beginning in 1954 and culminating after a number of variants into the *X-11 Variant of the Census Method II Seasonal Adjustment Program*, which PROC X11 implements.

References for this work during this period include Shiskin and Eisenpress (1957), Shiskin (1958), and Marris (1961). The authoritative documentation for the X-11 Variant is in Shiskin, Young, and Musgrave (1967). This document is not equivalent to a program specification; however, the FORTRAN code that implements the X-11 Variant is in the public domain. A less detailed description of the X-11 Variant is given in US Bureau of the Census (1969).

Development of the X-11-ARIMA Method

The X-11 method uses symmetric moving averages in estimating the various components. At the end of the series, however, these symmetric weights cannot be applied. Either asymmetric weights have to be used, or some method of extending the series must be found.

While various methods of extending a series have been proposed, the most important method to date has been the X-11-ARIMA method developed at Statistics Canada. This method uses Box-Jenkins ARIMA models to extend the series.

The Time Series Research and Analysis Division of Statistics Canada investigated 174 Canadian economic series and found five ARIMA models out of twelve that fit the majority of series well and reduced revisions for the most recent months. References that give details of various aspects of the X-11-ARIMA methodology

include Dagum (1980, 1982a, c, 1983, 1988), Laniel (1985), Lothian and Morry (1978a), and Huot et al. (1986).

Differences between X11ARIMA/88 and PROC X11

The original implementation of the X-11-ARIMA method was by Statistics Canada in 1980 (Dagum 1980), with later changes and enhancements made in 1988 (Dagum 1988). The calculations performed by PROC X11 differ from those in X11ARIMA/88, which will result in differences in the final component estimates provided by these implementations.

There are three areas where Statistics Canada made changes to the original X-11 seasonal adjustment method in developing X11ARIMA/80 (Monsell 1984). These are (a) selection of extreme values, (b) replacement of extreme values, and (c) generation of seasonal and trend cycle weights.

These changes have not been implemented in the current version of PROC X11. Thus the procedure produces results identical to those from previous versions of PROC X11 in the absence of an ARIMA statement.

Additional differences can result from the ARIMA estimation. X11ARIMA/88 uses conditional least squares (CLS), while CLS, unconditional least squares (ULS) and maximum likelihood (ML) are all available in PROC X11 by using the METHOD= option in the ARIMA statement. Generally, parameters estimates will differ for the different methods.

Implementation of the X-11 Seasonal Adjustment Method

The following steps describe the analysis of a monthly time series using multiplicative seasonal adjustment. Additional steps used by the X-11-ARIMA method are also indicated. Equivalent descriptions apply for an additive model if you replace *divide* with *subtract* where applicable.

In the multiplicative adjustment, the original series O_t is assumed to be of the form

$$O_t = C_t S_t I_t P_t D_t$$

where C_t is the trend cycle component, S_t is the seasonal component, I_t is the irregular component, P_t is the prior monthly factors component, and D_t is the trading-day component.

The trading-day component can be further factored as

$$D_t = D_{r,t} D_{tr,t},$$

where $D_{tr,t}$ are the trading-day factors derived from the prior daily weights, and $D_{r,t}$ are the residual trading-day factors estimated from the trading-day regression. For further information about estimating trading day variation, see Young (1965).

Additional Steps When Using the X-11-ARIMA Method

The X-11-ARIMA method consists of extending a given series by an ARIMA model and applying the usual X-11 seasonal adjustment method to this extended series. Thus in the simplest case in which there are no prior factors or calendar effects in the series, the ARIMA model selection, estimation, and forecasting are performed first, and the resulting extended series goes through the standard X-11 steps described in the next section.

If prior factor or calendar effects are present, they must be eliminated from the series before the ARIMA estimation is done because these effects are not stochastic.

Prior factors, if present, are removed first. Calendar effects represented by prior daily weights are then removed. If there are no further calendar effects, the adjusted series is extended by the ARIMA model, and this extended series goes through the standard X-11 steps without repeating the removal of prior factors and calendar effects from prior daily weights.

If further calendar effects are present, a trading-day regression must be performed. In this case it is necessary to go through an initial pass of the X-11 steps to obtain a final trading-day adjustment. In this initial pass, the series, adjusted for prior factors and prior daily weights, goes through the standard X-11 steps. At the conclusion of these steps, a final series adjusted for prior factors and all calendar effects is available. This adjusted series is then extended by the ARIMA model, and this extended series goes through the standard X-11 steps again, without repeating the removal of prior factors and calendar effects from prior daily weights and trading-day regression.

The Standard X-11 Seasonal Adjustment Method

The standard X-11 seasonal adjustment method consists of the following steps. These steps are applied to the original data or the original data extended by an ARIMA model.

1. In step 1, the data are read, ignoring missing values until the first nonmissing value is found. If prior monthly factors are present, the procedure reads prior monthly P_t factors and divides them into the original series to obtain $O_t/P_t = C_t S_t I_t D_{tr,t} D_{r,t}$.

   Seven daily weights can be specified to develop monthly factors to adjust the series for trading-day variation, $D_{tr,t}$; these factors are then divided into the original or prior adjusted series to obtain $C_t S_t I_t D_{r,t}$.

2. In steps 2, 3, and 4, three iterations are performed, each of which provides estimates of the seasonal S_t, trading-day $D_{r,t}$, trend cycle C_t, and irregular components I_t. Each iteration refines estimates of the extreme values in the irregular components. After extreme values are identified and modified, final estimates of the seasonal component, seasonally adjusted series, trend cycle, and irregular components are produced. Step 2 consists of three substeps:

   a) During the first iteration, a centered, 12-term moving average is applied to the original series O_t to provide a preliminary estimate $\hat{C}_t$ of the trend cycle curve C_t. This moving average combines 13 (a 2-term moving average of a 12-term moving average) consecutive monthly values, removing the S_t and I_t. Next, it obtains a preliminary estimate $\widehat{S_t I_t}$ by

   $$\widehat{S_t I_t} = \frac{O_t}{\hat{C}_t}$$

   b) A moving average is then applied to the $\widehat{S_t I_t}$ to obtain an estimate $\hat{S}_t$ of the seasonal factors. $\widehat{S_t I_t}$ is then divided by this estimate to obtain an estimate $\hat{I}_t$ of the irregular component. Next, a moving standard deviation is calculated from the irregular component and is used in assigning a weight to each monthly value for measuring its degree of extremeness. These weights are used to modify extreme values in $\widehat{S_t I_t}$. New seasonal factors are estimated by applying a moving average to the modified value of $\widehat{S_t I_t}$. A preliminary seasonally adjusted series is obtained by dividing the original series by these new seasonal factors. A second estimate of the trend cycle is obtained by applying a weighted moving average to this seasonally adjusted series.

c) The same process is used to obtain second estimates of the seasonally adjusted series and improved estimates of the irregular component. This irregular component is again modified for extreme values and then used to provide estimates of trading-day factors and refined weights for the identification of extreme values.

3. Using the same computations, a second iteration is performed on the original series that has been adjusted by the trading-day factors and irregular weights developed in the first iteration. The second iteration produces final estimates of the trading-day factors and irregular weights.

4. A third and final iteration is performed using the original series that has been adjusted for trading-day factors and irregular weights computed during the second iteration. During the third iteration, PROC X11 develops final estimates of seasonal factors, the seasonally adjusted series, the trend cycle, and the irregular components. The procedure computes summary measures of variation and produces a moving average of the final adjusted series.

Sliding Spans Analysis

The motivation for sliding spans analysis is to answer the question, When is a economic series unsuitable for seasonal adjustment? There have been a number of past attempts to answer this question: stable seasonality F test; moving seasonality F test, Q statistics, and others.

Sliding spans analysis attempts to quantify the stability of the seasonal adjustment process, and hence quantify the suitability of seasonal adjustment for a given series.

It is based on a very simple idea: for a stable series, deleting a small number of observations should not result in greatly different component estimates compared with the original, full series. Conversely, if deleting a small number of observations results in drastically different estimates, the series is unstable. For example, a drastic difference in the seasonal factors (Table D10) might result from a dominating irregular component or sudden changes in the seasonally component. When the seasonal component estimates of a series is unstable in this manner, they have little meaning and the series is likely to be unsuitable for seasonal adjustment.

Sliding spans analysis, developed at the Statistical Research Division of the U.S. Census Bureau (Findley et al. 1990; Findley and Monsell 1986), performs a repeated seasonal adjustment on subsets or spans of the full series. In particular, an initial span of the data, typically eight years in length, is seasonally adjusted, and the Tables C18, the trading-day factors (if trading-day regression performed), D10, the seasonal factors, and D11, the seasonally adjusted series are retained for further processing. Next, one year of data is deleted from the beginning of the initial span and one year of data is added. This new span is seasonally adjusted as before, with the same tables retained. This process continues until the end of the data is reached. The beginning and ending dates of the spans are such that the last observation in the original data is also the last observation in the last span. This is discussed in more detail in the following paragraphs.

The following notation for the components or differences computed in the sliding spans analysis follows Findley et al. (1990). The meaning for the symbol $X_t(k)$ is component X in month (or quarter) t, computed from data in the kth span. These components are now defined.

- Seasonal Factors (Table D10): $S_t(k)$
- Trading-Day Factors (Table C18): $TD_t(k)$
- Seasonally Adjusted Data (Table D11): $SA_t(k)$

- Month-to-Month Changes in the Seasonally Adjusted Data: $MM_t(k)$
- Year-to-Year Changes in the Seasonally Adjusted Data: $YY_t(k)$

The key measure is the maximum percent difference across spans. For example, consider a series that begins in January 1972, ends in December 1984, and has four spans, each of length 8 years (see Figure 1 in Findley et al. (1990), p. 346). Consider $S_t(k)$ the seasonal factor (Table D10) for month t for span k, and let N_t denote the number of spans containing month t; that is,

$$N_t = \{k : \ span\ k\ contains\ month\ t\}$$

In the middle years of the series there is overlap of all four spans, and N_t will be 4. The last year of the series will have only one span, while the beginning can have 1 or 0 spans depending on the original length.

Since we are interested in how much the seasonal factors vary for a given month across the spans, a natural quantity to consider is

$$max_{k\epsilon N_t} S_t(k) - min_{k\epsilon N_t} S_t(k)$$

In the case of the multiplicative model, it is useful to compute a percentage difference; define the maximum percentage difference (MPD) at time t as

$$MPD_t = \frac{max_{k\epsilon N_t} S_t(k) - min_{k\epsilon N_t} S_t(k)}{min_{k\epsilon N_t} S_t(k)}$$

The seasonal factor for month t is then unreliable if MPD_t is large. While no exact significance level can be computed for this statistic, empirical levels have been established by considering over 500 economic series (Findley et al. 1990; Findley and Monsell 1986). For these series it was found that for four spans, stable series typically had less than 15% of the MPD values exceeding 3.0%, while in marginally stable series, between 15% and 25% of the MPD values exceeded 3.0%. A series in which 25% or more of the MPD values exceeded 3.0% is almost always unstable.

While these empirical values cannot be considered an exact significance level, they provide a useful empirical basis for deciding if a series is suitable for seasonal adjustment. These percentage values are shifted down when fewer than four spans are used.

Computational Details for Sliding Spans Analysis

Length and Number of Spans

The algorithm for determining the length and number of spans for a given series was developed at the U.S. Bureau of the Census, Statistical Research Division. A summary of this algorithm is as follows.

First, an initial length based on the MACURVE *month=option* specification is determined, and then the maximum number of spans possible using this length is determined. If this maximum number exceeds four, set the number of spans to four. If this maximum number is one or zero, there are not enough observations to perform the sliding spans analysis. In this case a note is written to the log and the sliding spans analysis is skipped for this variable.

If the maximum number of spans is two or three, the actual number of spans used is set equal to this maximum. Finally, the length is adjusted so that the spans begin in January (or the first quarter) of the beginning year of the span.

The remainder of this section gives the computation formulas for the maximum percentage difference (MPD) calculations along with the threshold regions.

Seasonal Factors (Table D10)

For the additive model, the MPD is defined as

$$max_{k\epsilon N_t} S_t(k) - min_{k\epsilon N_t} S_t(k)$$

For the multiplicative model, the MPD is

$$MPD_t = \frac{max_{k\epsilon N_t} S_t(k) - min_{k\epsilon N_t} S_t(k)}{min_{k\epsilon N_t} S_t(k)}$$

A series for which less than 15% of the MPD values of D10 exceed 3.0% is stable; between 15% and 25% is marginally stable; and greater than 25% is unstable. Span reports S 2.A through S 2.C give the various breakdowns for the number of times the MPD exceeded these levels.

Trading Day Factor (Table C18)

For the additive model, the MPD is defined as

$$max_{k\epsilon N_t} TD_t(k) - min_{k\epsilon N_t} TD_t(k)$$

For the multiplicative model, the MPD is

$$MPD_t = \frac{max_{k\epsilon N_t} TD_t(k) - min_{k\epsilon N_t} TD_t(k)}{min_{k\epsilon N_t} TD_t(k)}$$

The U.S. Census Bureau currently gives no recommendation concerning MPD thresholds for the trading-day factors. Span reports S 3.A through S 3.C give the various breakdowns for MPD thresholds. When TDREGR=NONE is specified, no trading-day computations are done, and this table is skipped.

Seasonally Adjusted Data (Table D11)

For the additive model, the MPD is defined as

$$max_{k\epsilon N_t} SA_t(k) - min_{k\epsilon N_t} SA_t(k)$$

For the multiplicative model, the MPD is

$$MPD_t = \frac{max_{k\epsilon N_t} SA_t(k) - min_{k\epsilon N_t} SA_t(k)}{min_{k\epsilon N_t} SA_t(k)}$$

A series for which less than 15% of the MPD values of D11 exceed 3.0% is stable; between 15% and 25% is marginally stable; and greater than 25% is unstable. Span reports S 4.A through S 4.C give the various breakdowns for the number of times the MPD exceeded these levels.

Month-to-Month Changes in the Seasonally Adjusted Data

Some additional notation is needed for the month-to-month and year-to-year differences. Define $N1_t$ as

$$N1_t = \{k : \ span\ k\ contains\ month\ t\ and\ t-1\}$$

For the additive model, the month-to-month change for span k is defined as

$$MM_t(k) = SA_t - SA_{t-1}$$

while for the multiplicative model

$$MM_t(k) = \frac{SA_t - SA_{t-1}}{SA_{t-1}}$$

Since this quantity is already in percentage form, the MPD for both the additive and multiplicative model is defined as

$$MPD_t = max_{k \epsilon N1_t} MM_t(k) - min_{k \epsilon N1_t} MM_t(k)$$

The current recommendation of the U.S. Census Bureau is that if 35% or more of the MPD values of the month-to-month differences of D11 exceed 3.0%, then the series is usually not stable; 40% exceeding this level clearly marks an unstable series. Span reports S 5.A.1 through S 5.C give the various breakdowns for the number of times the MPD exceeds these levels.

Year-to-Year Changes in the Seasonally Adjusted Data

First define $N12_t$ as

$$N12_t = \{k : \ span\ k\ contains\ month\ t\ and\ t-12\}$$

(Appropriate changes in notation for a quarterly series are obvious.)

For the additive model, the month-to-month change for span k is defined as

$$YY_t(k) = SA_t - SA_{t-12}$$

while for the multiplicative model

$$YY_t(k) = \frac{SA_t - SA_{t-12}}{SA_{t-12}}$$

Since this quantity is already in percentage form, the MPD for both the additive and multiplicative model is defined as

$$MPD_t = max_{k \epsilon N1_t} YY_t(k) - min_{k \epsilon N1_t} YY_t(k)$$

The current recommendation of the U.S. Census Bureau is that if 10% or more of the MPD values of the month-to-month differences of D11 exceed 3.0%, then the series is usually not stable. Span reports S 6.A through S 6.C give the various breakdowns for the number of times the MPD exceeds these levels.

Data Requirements

The input data set must contain either quarterly or monthly time series, and the data must be in chronological order. For the standard X-11 method, there must be at least three years of observations (12 for quarterly time series or 36 for monthly) in the input data sets or in each BY group in the input data set if a BY statement is used.

For the X-11-ARIMA method, there must be at least five years of observations (20 for quarterly time series or 60 for monthly) in the input data sets or in each BY group in the input data set if a BY statement is used.

Missing Values

Missing values at the beginning of a series to be adjusted are skipped. Processing starts with the first nonmissing value and continues until the end of the series or until another missing value is found.

Missing values are not allowed for the DATE= variable. The procedure terminates if missing values are found for this variable.

Missing values found in the PMFACTOR= variable are replaced by 100 for the multiplicative model (default) and by 0 for the additive model.

Missing values can occur in the output data set. If the time series specified in the OUTPUT statement is not computed by the procedure, the values of the corresponding variable are missing. If the time series specified in the OUTPUT statement is a moving average, the values of the corresponding variable are missing for the first *n* and last *n* observations, where *n* depends on the length of the moving average. Additionally, if the time series specified is an irregular component modified for extremes, only the modified values are given, and the remaining values are missing.

Prior Daily Weights and Trading-Day Regression

Suppose that a detailed examination of retail sales at ZXY Company indicates that certain days of the week have higher amounts of sales. In particular, Thursday, Friday, and Saturday have approximately twice the amount of sales as Monday, Tuesday, and Wednesday, and no sales occur on Sunday. This means that months with five Saturdays would have higher amounts of sales than months with only four Saturdays.

This phenomenon is called a calendar effect; it can be handled in PROC X11 by using the PDWEIGHTS (prior daily weights) statement or the TDREGR=option (trading-day regression). The PDWEIGHTS statement and the TDREGR=option can be used separately or together.

If the relative weights are known (as in the preceding) it is appropriate to use the PDWEIGHTS statement. If further residual calendar variation is present, TDREGR=ADJUST should also be used. If you know that a calendar effect is present, but know nothing about the relative weights, use TDREGR=ADJUST without a PDWEIGHTS statement.

In this example, it is assumed that the calendar variation is due to both prior daily weights and residual variation. Thus both a PDWEIGHTS statement and TDREGR=ADJUST are specified.

Note that only the relative weights are needed; in the actual computations, PROC X11 normalizes the weights to sum to 7.0. If a day of the week is not present in the PDWEIGHTS statement, it is given a value of zero. Thus "sun=0" is not needed.

```
proc x11 data=sales;
   monthly date=date tdregr=adjust;
   var sales;
   tables a1 a4 b15 b16 C14 C15 c18 d11;
   pdweights mon=1 tue=1 wed=1 thu=2 fri=2 sat=2;
   output out=x11out a1=a1 a4=a4 b1=b1 c14=c14
                     c16=c16 c18=c18 d11=d11;
run;
```

Tables of interest include A1, A4, B15, B16, C14, C15, C18, and D11. Table A4 contains the adjustment factors derived from the prior daily weights; Table C14 contains the extreme irregular values excluded from trading-day regression; Table C15 contains the trading-day-regression results; Table C16 contains the monthly factors derived from the trading-day regression; and Table C18 contains the final trading-day factors derived from the combined daily weights. Finally, Table D11 contains the final seasonally adjusted series.

Adjustment for Prior Factors

Suppose now that a strike at ZXY Company during July and August of 1988 caused sales to decrease an estimated 50%. Since this is a one-time event with a known cause, it is appropriate to prior adjust the data to reflect the effects of the strike. This is done in PROC X11 through the use of PMFACTOR=*varname* (prior monthly factor) in the MONTHLY statement.

In the following example, the PMFACTOR variable is named PMF. Since the estimate of the decrease in sales is 50%, PMF has a value of 50.0 for the observations corresponding to July and August 1988, and a value of 100.0 for the remaining observations.

This prior adjustment on SALES is performed by replacing SALES with the calculated value (SALES/PMF) * 100.0. A value of 100.0 for PMF leaves SALES unchanged, while a value of 50.0 for PMF doubles SALES. This value is the estimate of what SALES would have been without the strike. The following example shows how this prior adjustment is accomplished.

```
data sales2;
   set sales;
   if '01jul1988'd <= date <= '01aug1988'd then pmf = 50;
   else pmf = 100;
run;

proc x11 data=sales2;
   monthly date=date pmfactor=pmf;
   var sales;
   tables a1 a2 a3 d11;
   output out=x11out a1=a1 a2=a2 a3=a3 d11=d11;
run;
```

Table A2 contains the prior monthly factors (the values of PMF), and Table A3 contains the prior adjusted series.

The YRAHEADOUT Option

For monthly data, the YRAHEADOUT option affects only Tables C16 (regression trading-day adjustment factors), C18 (trading-day factors from combined daily weights), and D10 (seasonal factors). For quarterly data, only Table D10 is affected. Variables for all other tables have missing values for the forecast observations. The forecast values for a table are included only if that table is specified in the OUTPUT statement.

Tables C16 and C18 are calendar effects that are extrapolated by calendar composition. These factors are independent of the data once trading-day weights have been calculated. Table D10 is extrapolated by a linear combination of past values. If N is the total number of nonmissing observations for the analysis variable, this linear combination is given by

$$D10_t = \frac{1}{2}(3 \times D10_{t-12} - D10_{t-24}), \quad t = N+1, .., N+12$$

If the input data are monthly time series, 12 extra observations are added to the end of the output data set. (If a BY statement is used, 12 extra observations are added to the end of each BY group.) If the input data are a quarterly time series, four extra observations are added to the end of the output data set. (If a BY statement is used, four extra observations are added to each BY group.)

The DATE= variable (or _DATE_) is extrapolated for the extra observations generated by the YRAHEADOUT option, while all other ID variables will have missing values.

If ARIMA processing is requested, and if both the OUTEXTRAP and YRAHEADOUT options are specified in the PROC X11 statement, an additional 12 (or 4) observations are added to the end of output data set for monthly (or quarterly) data after the ARIMA forecasts, using the same linear combination of past values as before.

Effect of Backcast and Forecast Length

Based on a number of empirical studies (Dagum 1982a, b, c; Dagum and Laniel 1987), one year of forecasts minimize revisions when new data become available. Two and three years of forecasts show only small gains.

Backcasting improves seasonal adjustment but introduces permanent revisions at the beginning of the series and also at the end for series of length 8, 9, or 10 years. For series shorter than 7 years, the advantages of backcasting outweigh the disadvantages (Dagum 1988).

Other studies (Pierce 1980; Bobbit and Otto 1990; Buszuwski 1987) suggest "full forecasting"— that is, using enough forecasts to allow symmetric weights for the seasonal moving averages for the most current data. For example, if a 3×9 seasonal moving average was specified for one or more months by using the MACURVES statement, five years of forecasts would be required. This is because the seasonal moving averages are performed on calendar months separately, and the 3×9 is an 11-term centered moving average, requiring five observations before and after the current observation. Thus

```
macurves dec='3x9';
```

would require five additional December values to compute the seasonal moving average.

Details of Model Selection

If an ARIMA statement is present but no MODEL= is given, PROC X11 estimates and forecasts five predefined models and selects the best. This section describes the details of the selection criteria and the selection process.

The five predefined models used by PROC X11 are the same as those used by X11ARIMA/88 from Statistics Canada. These particular models, shown in Table 43.2, were chosen on the basis of testing a large number of economics series (Dagum 1988) and should provide reasonable forecasts for most economic series.

Table 43.2 Five Predefined Models

Model #	Specification	Multiplicative	Additive
1	(0,1,1)(0,1,1)s	log transform	no transform
2	(0,1,2)(0,1,1)s	log transform	no transform
3	(2,1,0)(0,1,1)s	log transform	no transform
4	(0,2,2)(0,1,1)s	log transform	no transform
5	(2,1,2)(0,1,1)s	no transform	no transform

The selection process proceeds as follows. The five models are estimated and one-step-ahead forecasts are produced in the order shown in Table 43.2. As each model is estimated, the following three criteria are checked:

- The mean absolute percent error (MAPE) for the last three years of the series must be less than 15%.
- The significance probability for the Box-Ljung chi-square for up to lag 24 for monthly (8 for quarterly) must greater than 0.05.
- The over-differencing criteria must not exceed 0.9.

The descriptions of these three criteria are given in the section "Criteria Details" on page 3118. The default values for these criteria are those used by X11ARIMA/88 from Statistics Canada; these defaults can be changed by the MAPECR=, CHICR=, and OVDIFCR= options.

A model that fails any one of these three criteria is excluded from further consideration. In addition, if the ARIMA estimation fails for a given model, a warning is issued, and the model is excluded. The final set of all models considered consists of those that pass all three criteria and are estimated successfully. From this set, the model with the smallest MAPE for the last three years is chosen.

If all five models fail, ARIMA processing is skipped for the variable being processed, and the standard X-11 seasonal adjustment is performed. A note is written to the log with this information.

The chosen model is then used to forecast the series one or more years (determined by the FORECAST= option in the ARIMA statement). These forecasts are appended to the original data (or the prior and calendar-adjusted data).

If a BACKCAST= option is specified, the chosen model form is used, but the parameters are reestimated using the reversed series. Using these parameters, the reversed series is forecast for the number of years specified by the BACKCAST= option. These forecasts are then reversed and appended to the beginning of the original series, or the prior and calendar-adjusted series, to produce the backcasts.

Note that the final selection rule (the smallest MAPE using the last three years) emphasizes the quality of the forecasts at the end of the series. This is consistent with the purpose of the X-11-ARIMA methodology, which is to improve the estimates of seasonal factors and thus minimize revisions to recent past data as new data become available.

Criteria Details

Mean Absolute Percent Error (MAPE)

For the MAPE criteria testing, only the last three years of the original series (or prior and calendar adjusted series) is used in computing the MAPE.

Let y_t, $t = 1,..,n$, be the last three years of the series, and denote its one-step-ahead forecast by $\hat{y}_t$, where $n = 36$ for a monthly series and $n = 12$ for a quarterly series.

With this notation, the MAPE criteria are computed as

$$MAPE = \frac{100}{n} \sum_{t=1}^{n} \frac{|y_t - \hat{y}_t|}{|y_t|}$$

Box-Ljung Chi-Square

The Box-Ljung chi-square is a lack-of-fit test based on the model residuals. This test statistic is computed using the Ljung-Box formula

$$\chi^2_m = n(n+2) \sum_{k=1}^{m} \frac{r_k^2}{(n-k)}$$

where n is the number of residuals that can be computed for the time series, and

$$r_k = \frac{\sum_{t=1}^{n-k} a_t a_{t+k}}{\sum_{t=1}^{n} a_t^2}$$

where the a_t's are the residual sequence. This formula has been suggested by Ljung and Box (1978) as yielding a better fit to the asymptotic chi-square distribution. Some simulation studies of the finite sample properties of this statistic are given by Davies, Triggs, and Newbold (1977) and by Ljung and Box (1978).

For monthly series, $m = 24$, while for quarterly series, $m = 8$.

Over-Differencing Test

From Table 43.2 you can see that all models have a single seasonal MA factor and at most two nonseasonal MA factors. Also, all models have seasonal and nonseasonal differencing. Consider model 2 applied to a monthly series y_t with $E(y_t) = \mu$:

$$(1 - B^1)(1 - B^{12})(y_t - \mu) = (1 - \theta_1 B - \theta_2 B^2)(1 - \theta_3 B^{12})a_t$$

If $\theta_3 = 1.0$, then the factors $(1 - \theta_3 B^{12})$ and $(1 - B^{12})$ will cancel, resulting in a lower-order model.

Similarly, if $\theta_1 + \theta_2 = 1.0$,

$$(1 - \theta_1 B - \theta_2 B^2) = (1 - B)(1 - \alpha B)$$

for some $\alpha \neq 0.0$. Again, this results in cancellation and a lower-order model.

Since the parameters are not exact, it is not reasonable to require that

$$\theta_3 < 1.0 \text{ and } \theta_1 + \theta_2 < 1.0$$

Instead, an approximate test is performed by requiring that

$$\theta_3 \leq 0.9 \text{ and } \theta_1 + \theta_2 \leq 0.9$$

The default value of 0.9 can be changed by the OVDIFCR= option. Similar reasoning applies to the other models.

ARIMA Statement Options for the Five Predefined Models

Table 43.3 lists the five predefined models and gives the equivalent MODEL= parameters in a PROC X11 ARIMA statement.

In all models except the fifth, a log transformation is performed before the ARIMA estimation for the multiplicative case; no transformation is performed for the additive case. For the fifth model, no transformation is done for either case.

The multiplicative case is assumed in the following table. The indicated seasonality *s* in the specification is either 12 (monthly) or 4 (quarterly). The MODEL statement assumes a monthly series.

Table 43.3 ARIMA Statements Options for Predefined Models

Model	ARIMA Statement Options
(0,1,1)(0,1,1)s	MODEL=(Q=1 SQ=1 DIF=1 SDIF=1) TRANSFORM=LOG
(0,1,2)(0,1,1)s	MODEL=(Q=2 SQ=1 DIF=1 SDIF=1) TRANSFORM=LOG
(2,1,0)(0,1,1)s	MODEL=(P=2 SQ=1 DIF=1 SDIF=1) TRANSFORM=LOG
(0,2,2)(0,1,1)s	MODEL=(Q=2 SQ=1 DIF=2 SDIF=1) TRANSFORM=LOG
(2,1,2)(0,1,1)s	MODEL=(P=2 Q=2 SQ=1 DIF=1 SDIF=1)

OUT= Data Set

The OUT= data set specified in the OUTPUT statement contains the BY variables, if any; the ID variables, if any; and the DATE= variable if the DATE= option is given, or _DATE_ if the DATE= option is not specified.

In addition, the variables specified by the option

tablename =var1 var2 . . . varn

are placed in the OUT= data set. A list of tables available for monthly and quarterly series is given later, in Table 43.4.

The OUTSPAN= Data Set

The OUTSPAN= option is specified in the PROC statement, and writes the sliding spans results to the specified output data set. The OUTSPAN= data set contains the following variables:

- A1, a numeric variable that is a copy of the original series truncated to the current span. Note that overlapping spans will contain identical values for this variable.
- C18, a numeric variable that contains the trading-day factors for the seasonal adjustment for the current span
- D10, a numeric variable that contains the seasonal factors for the seasonal adjustment for the current span
- D11, a numeric variable that contains the seasonally adjusted series for the current span
- DATE, a numeric variable that contains the date within the current span
- SPAN, a numeric variable that contains the current span. The first span is the earliest span—that is the one with the earliest starting date.
- VARNAME, a character variable containing the name of each variable in the VAR list. A separate sliding spans analysis is performed on each variable in the VAR list.

OUTSTB= Data Set

The output data set produced by the OUTSTB= option of the PROC X11 statement contains the information in the analysis of variance on table D8 (Final Unmodified S-I Ratios). This analysis of variance, following table D8 in the printed output, tests for stable seasonality (Shiskin, Young, and Musgrave 1967, Appendix A). These data contain the following variables:

- VARNAME, a character variable containing the name of each variable in the VAR list
- TABLE, a character variable specifying the table from which the analysis of variance is performed. When ARIMA processing is requested, and two passes of X11 are required (when TDREGR=PRINT, TEST, or ADJUST), Table D8 and the stable seasonality test are computed twice: once in the initial pass, then again in the final pass. Both of these computations are put in the OUTSTB data set and are identified by D18.1 and D18.2, respectively.
- SOURCE, a character variable corresponding to the "source" column in the analysis of variance table following Table D8
- SS, a numeric variable containing the sum of squares associated with the corresponding source term
- DF, a numeric variable containing the degrees of freedom associated with the corresponding source term

- MS, a numeric variable containing the mean square associated with the corresponding source term. MS is missing for the source term "Total"
- F, a numeric variable containing the *F* statistic for the "Between" source term. F is missing for all other source terms.
- PROBF, a numeric variable containing the significance level for the *F* statistic. PROBF is missing for the source terms "Total" and "Error."

OUTTDR= Data Set

The trading-day regression results (tables B15 and C15) are written to the OUTTDR= data set, which contains the following variables:

- VARNAME, a character variable containing the name of the VAR variable being processed
- TABLE, a character variable containing the name of the table. It can have only the value B15 (Preliminary Trading-Day Regression) or C15 (Final Trading-Day Regression).
- _TYPE_, a character variable whose value distinguishes the three distinct table format types. These types are (a) the regression, (b) the listing of the standard error associated with length-of-month, and (c) the analysis of variance. The first seven observations in the OUTTDR data set correspond to the regression on days of the week; thus the _TYPE_ variable is given the value "REGRESS" (day-of-week regression coefficient). The next four observations correspond to 31-, 30-, 29-, and 28-day months and are given the value _TYPE_=LOM_STD (length-of-month standard errors). Finally, the last three observations correspond to the analysis of variance table, and _TYPE_=ANOVA.
- PARM, a character variable, further identifying the nature of the observation. PARM is set to blank for the three _TYPE_=ANOVA observations.
- SOURCE, a character variable containing the source in the regression. This variable is missing for all _TYPE_=REGRESS and LOM_STD.
- CWGT, a numeric variable containing the combined trading-day weight (prior weight + weight found from regression). The variable is missing for all _TYPE_=LOM_STD and _TYPE_=ANOVA.
- PRWGT, a numeric variable containing the prior weight. The prior weight is 1.0 if PDWEIGHTS are not specified. This variable is missing for all _TYPE_=LOM_STD and _TYPE_=ANOVA.
- COEFF, a numeric variable containing the calculated regression coefficient for the given day. This variable is missing for all _TYPE_=LOM_STD and _TYPE_=ANOVA.
- STDERR, a numeric variable containing the standard errors. For observations with _TYPE_=REGRESS, this is the standard error corresponding to the regression coefficient. For observations with _TYPE_=LOM_STD, this is standard error for the corresponding length-of-month. This variable is missing for all _TYPE_=ANOVA.
- T1, a numeric variable containing the *t* statistic corresponding to the test that the combined weight is different from the prior weight. This variable is missing for all _TYPE_=LOM_STD and _TYPE_=ANOVA.

- T2, a numeric variable containing the *t* statistic corresponding to the test that the combined weight is different from 1.0. This variable is missing for all _TYPE_=LOM_STD and _TYPE_=ANOVA.
- PROBT1, a numeric variable containing the significance level for *t* statistic T1. The variable is missing for all _TYPE_=LOM_STD and _TYPE_=ANOVA.
- PROBT2, a numeric variable containing the significance level for *t* statistic T2. The variable is missing for all _TYPE_=LOM_STD and _TYPE_=ANOVA.
- SS, a numeric variable containing the sum of squares associated with the corresponding source term. This variable is missing for all _TYPE_=REGRESS and LOM_STD.
- DF, a numeric variable containing the degrees of freedom associated with the corresponding source term. This variable is missing for all _TYPE_=REGRESS and LOM_STD.
- MS, a numeric variable containing the mean square associated with the corresponding source term. This variable is missing for the source term 'Total' and for all _TYPE_=REGRESS and LOM_STD.
- F, a numeric variable containing the *F* statistic for the 'Regression' source term. The variable is missing for the source terms 'Total' and 'Error', and for all _TYPE_=REGRESS and LOM_STD.
- PROBF, a numeric variable containing the significance level for the *F* statistic. This variable is missing for the source term 'Total' and 'Error' and for all _TYPE_=REGRESS and LOM_STD.

Printed Output

The output from PROC X11, both printed tables and the series written to the OUT= data set, depends on whether the data are monthly or quarterly. For the printed tables, the output depends further on the value of the PRINTOUT= option and the TABLE statement, along with other options specified.

The printed output is organized into tables identified by a part letter and a sequence number within the part. The seven major parts of the X11 procedure are as follows:

A	prior adjustments (optional)
B	preliminary estimates of irregular component weights and regression trading-day factors
C	final estimates of irregular component weights and regression trading-day factors
D	final estimates of seasonal, trend cycle, and irregular components
E	analytical tables
F	summary measures
G	charts

Table 43.4 describes the individual tables and charts. Most tables apply both to quarterly and monthly series. Those that apply only to a monthly time series are indicated by an "M" in the notes section, while "P" indicates the table is not a time series, and is only printed, not output to the OUT= data set.

Table 43.4 Table Names and Descriptions

Table	Description	Notes
A1	original series	M
A2	prior monthly adjustment factors	M
A3	original series adjusted for prior monthly factors	M
A4	prior trading-day adjustments	M
A5	prior adjusted or original series	M
A13	ARIMA forecasts	
A14	ARIMA backcasts	
A15	prior adjusted or original series extended by ARIMA backcasts and forecasts	
B1	prior adjusted or original series	
B2	trend cycle	
B3	unmodified seasonal-irregular (S-I) ratios	
B4	replacement values for extreme S-I ratios	
B5	seasonal factors	
B6	seasonally adjusted series	
B7	trend cycle	
B8	unmodified S-I ratios	
B9	replacement values for extreme S-I ratios	
B10	seasonal factors	
B11	seasonally adjusted series	
B13	irregular series	
B14	extreme irregular values excluded from trading-day regression	M
B15	preliminary trading-day regression	M,P
B16	trading-day adjustment factors	M
B17	preliminary weights for irregular components	
B18	trading-day factors derived from combined daily weights	M
B19	original series adjusted for trading-day and prior variation	M
C1	original series modified by preliminary weights and adjusted for trading-day and prior variation	
C2	trend cycle	
C4	modified S-I ratios	
C5	seasonal factors	
C6	seasonally adjusted series	
C7	trend cycle	
C9	modified S-I ratios	
C10	seasonal factors	
C11	seasonally adjusted series	
C13	irregular series	
C14	extreme irregular values excluded from trading-day regression	M
C15	final trading-day regression	M,P
C16	final trading-day adjustment factors derived from regression coefficients	M
C17	final weight for irregular components	

Table 43.4 *continued*

Table	Description	Notes
C18	final trading-day factors derived from combined daily weights	M
C19	original series adjusted for trading-day and prior variation	M
D1	original series modified for final weights and adjusted for trading-day and prior variation	
D2	trend cycle	
D4	modified S-I ratios	
D5	seasonal factors	
D6	seasonally adjusted series	
D7	trend cycle	
D8	final unmodified S-I ratios	
D9	final replacement values for extreme S-I ratios	
D10	final seasonal factors	
D11	final seasonally adjusted series	
D12	final trend cycle	
D13	final irregular series	
E1	original series with outliers replaced	
E2	modified seasonally adjusted series	
E3	modified irregular series	
E4	ratios of annual totals	P
E5	percent changes in original series	
E6	percent changes in final seasonally adjusted series	
F1	MCD moving average	
F2	summary measures	P
G1	chart of final seasonally adjusted series and trend cycle	P
G2	chart of S-I ratios with extremes, S-I ratios without extremes, and final seasonal factors	P
G3	chart of S-I ratios with extremes, S-I ratios without extremes, and final seasonal factors in calendar order	P
G4	chart of final irregular and final modified irregular series	P

The PRINTOUT= Option

The PRINTOUT= option controls printing for groups of tables. See the "TABLES Statement" on page 3106 for details on specifying individual tables. The following list gives the tables printed for each value of the PRINTOUT= option:

STANDARD (26 tables)	A1–A4, B1, C13–C19, D8–D13, E1–E6, F1, F2
LONG (40 tables)	A1–A5, A13–A15, B1, B2, B7, B10, B13–B15, C1, C7, C10, C13–C19, D1, D7–D11, D13, E1–E6, F1, F2
FULL (62 tables)	A1–A5, A13–A15, B1–B11, B13–B19, C1–C11, C13–C19, D1, D2, D4–D12, E1–E6, F1, F2

The actual number of tables printed depends on the options and statements specified. If a table is not computed, it is not printed. For example, if TDREGR=NONE is specified, none of the tables associated with the trading-day are printed.

The CHARTS= Option

Of the four charts listed in Table 43.4, G1 and G2 are printed by default (CHARTS=STANDARD). Charts G3 and G4 are printed when CHARTS=FULL is specified. See the "TABLES Statement" on page 3106 for details on specifying individual charts.

Stable, Moving, and Combined Seasonality Tests on the Final Unmodified SI Ratios (Table D8)

PROC X11 displays four tests used to identify stable seasonality and moving seasonality and to measure identifiable seasonality. These tests are displayed after Table D8. They are "Stable Seasonality Test," "Moving Seasonality Test," "Nonparametric Test for the Presence of Seasonality Assuming Stability," and "Summary of Results and Combined Test for the Presence of Identifiable Seasonality." The motivation, interpretation, and statistical details of all these tests are now given.

Motivation

The seasonal component of this time series, S_t, is defined as the intrayear variation that is repeated constantly (stable) or in an evolving fashion from year to year (moving seasonality). If the increase in the seasonal factors from year to year is too large, then the seasonal factors will introduce distortion into the model. It is important to determine if seasonality is identifiable without distorting the series.

To determine if stable seasonality is present in a series, PROC X11 computes a one-way analysis of variance by using the seasons (months or quarters) as the factor on the Final Unmodified SI Ratios (Table D8). This is the appropriate table to use because the removal of the trend cycle is equivalent to detrending. PROC X11 prints this test, labeled "Stable Seasonality Test," immediately after the Table D8.

The X11 seasonal adjustment method tests for moving seasonality. Moving seasonality can be a source of distortion when seasonal factors are used in the model. PROC X11 computes and prints a test for moving seasonality. The test is a two-way analysis of variance that uses months (or quarters) and years. As in the "Stable Seasonality Test," this analysis of variance is performed on the Final Unmodified SI Ratios (Table D8). PROC X11 prints this test, labeled "Moving Seasonality Test," after the "Stable Seasonality Test."

PROC X11 next computes a nonparametric Kruskal-Wallis chi-squared test for stable seasonality, "Nonparametric Test for the Presence of Seasonality Assuming Stability." The Kruskal-Wallis test is performed on the ranks of the Final Unmodified SI Ratios (Table D8). For further details about the Kruskal-Wallis test, see Lehmann and D'Abrera (2006, pp. 204–210).

The results of the preceding three tests are combined into a joint test to measure identifiable seasonality, "Summary of Results and Combined Test for the Presence of Identifiable Seasonality." This test combines the two F tests previously described, along with the Kruskal-Wallis chi-squared test for stable seasonality, to determine "identifiable" seasonality. This test is printed after "Nonparametric Test for the Presence of Seasonality Assuming Stability."

Interpretation and Statistical Details

The "Stable Seasonality Test" is a one-way analysis of variance on the "Final Unmodified SI Ratios" with seasons (months or quarters) as the factor.

To determine whether stable seasonality is present in a series, PROC X11 computes a one-way analysis of variance by using the seasons (months or quarters) as the factor on the Final Unmodified SI Ratios (Table D8). This is the appropriate table to use because the removal of the trend cycle is similar to detrending.

A large F statistic and a small significance level are evidence that a significant amount of variation in the SI-ratios is due to months or quarters, which in turn is evidence of seasonality; the null hypothesis of no month/quarter effect is rejected.

Conversely, a small F statistic and a large significance level (close to 1.0) are evidence that variation due to month or quarter could be due to random error, and the null hypothesis of no month/quarter effect is not rejected. The interpretation and utility of seasonal adjustment are problematic under such conditions.

The F test for moving seasonality is performed by a two-way analysis of variance. The two factors are seasons (months or quarters) and years. The years effect is tested separately; the null hypothesis is no effect due to years after accounting for variation due to months or quarters. For further details about the moving seasonality test, see Lothian (1984a, b, 1978) and Higginson (1975).

The significance level reported in both the moving and stable seasonality tests are only approximate. Table D8, the Final Unmodified SI Ratios, is constructed from an averaging operation that induces a correlation in the residuals from which the F test is computed. Hence the computed F statistic differs from an exact F statistic; see Cleveland and Devlin (1980) for details.

The test for identifiable seasonality is performed by combining the F tests for stable and moving seasonality, along with a Kruskal-Wallis test for stable seasonality. The following description is based on Lothian and Morry (1978b); other details can be found in Dagum (1988, 1983).

Let F_s and F_m denote the F value for the stable and moving seasonality tests, respectively. The combined test is performed as shown in Table 43.5 and as follows:

1. If the null hypothesis of no stable seasonality is not rejected at the 0.10% significance level ($P_S \geq 0.001$), then the series is considered to be nonseasonal. PROC X11 returns the conclusion, "Identifiable Seasonality Not Present."

2. If the null hypothesis in step 1 is rejected, then PROC X11 computes the following quantities:

$$T_1 = \frac{7}{F_s}$$

$$T_2 = \frac{3F_m}{F_s}$$

   Let T denote the simple average of T_1 and T_2:

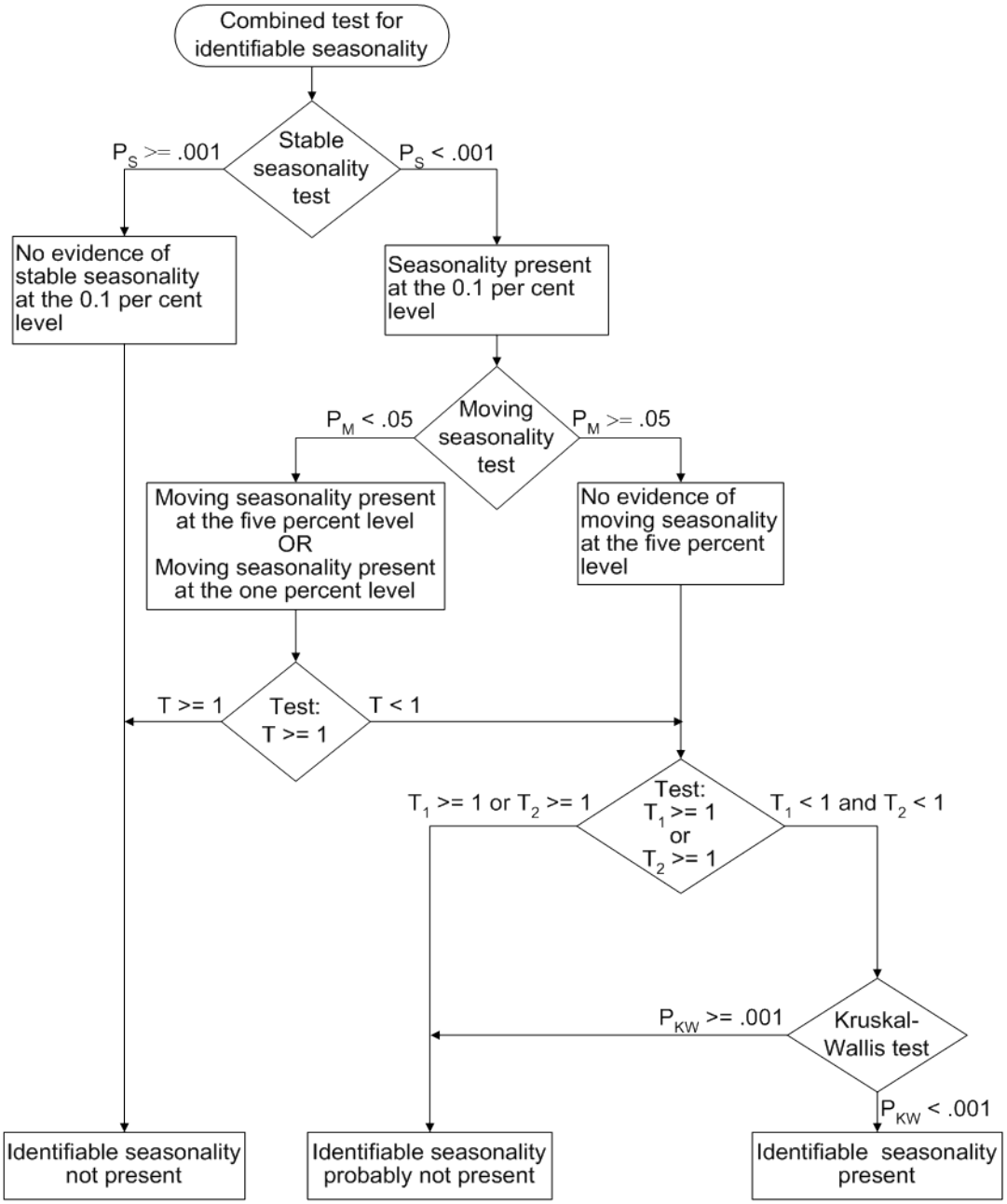
$$T = \frac{(T_1 + T_2)}{2}$$

   If the null hypothesis of no moving seasonality is rejected at the 5.0% significance level ($P_M < 0.05$) and if $T \geq 1.0$, the null hypothesis of identifiable seasonality *not* present is not rejected and PROC X11 returns the conclusion, "Identifiable Seasonality Not Present."

3. If the null hypothesis of identifiable seasonality *not* present has not been accepted, but $T_1 \geq 1.0$, $T_2 \geq 1.0$, or the Kruskal-Wallis chi-squared test fails to reject at the 0.10% significance level ($P_{KW} \geq 0.001$), then PROC X11 returns the conclusion "Identifiable Seasonality Probably Not Present."

4. If the null hypotheses of no stable seasonality associated with the F_S and Kruskal-Wallis chi-squared tests are rejected and if none of the combined measures described in steps 2 and 3 fail, then the null hypothesis of identifiable seasonality *not* present is rejected and PROC X11 returns the conclusion "Identifiable Seasonality Present."

Figure 43.5 Combined Seasonality Test Flowchart



Tables Written to the OUT= Data Set

All tables that are time series can be written to the OUT= data set. However, depending on the specified options and statements, not all tables are computed. When a table is not computed, but is requested in the OUTPUT statement, the resulting variable has all missing values.

For example, if the PMFACTOR= option is not specified, Table A2 is not computed, and requesting this table in the OUTPUT statement results in the corresponding variable having all missing values.

The trading-day regression results, Tables B15 and C15, although not written to the OUT= data set, can be written to an output data set; see the OUTTDR= option for details.

Printed Output Generated by Sliding Spans Analysis

Table S 0.A

Table S 0.A gives the variable name, the length and number of spans, and the beginning and ending dates of each span.

Table S 0.B

Table S 0.B gives the summary of the two F tests performed during the standard X11 seasonal adjustments for stable and moving seasonality on Table D8, the final SI ratios. These tests are described in the section "Printed Output" on page 3122.

Table S 1.A

Table S 1.A gives the range analysis of seasonal factors. This includes the means for each month (or quarter) within a span, the maximum percentage difference across spans for each month, and the average. The minimum and maximum within a span are also indicated.

For example, for a monthly series and an analysis with four spans, the January row would contain a column for each span, with the value representing the average seasonal factor (Table D10) over all January calendar months occurring within the span. Beside each span column is a character column with either a MIN, MAX, or blank value, indicating which calendar month had the minimum and maximum value over that span.

Denote the average over the jth calendar month in span $k, k = 1, .., 4$, by $\bar{S}_j(k)$; then the maximum percent difference (MPD) for month j is defined by

$$MPD_j = \frac{max_{k=1,..,4}\bar{S}_j(k) - min_{k=1,..,4}\bar{S}_j(k)}{min_{k=1,..,4}\bar{S}_j(k)}$$

The last numeric column of Table S 1.A is the average value over all spans for each calendar month, with the minimum and maximum row flagged as in the span columns.

Table S 1.B

Table S 1.B gives a summary of range measures for each span. The first column, Range Means, is calculated by computing the maximum and minimum over all months or quarters in a span, then taking the difference. The next column is the range ratio means, which is simply the ratio of the previously described maximum and minimum. The next two columns are the minimum and maximum seasonal factors over the entire span, while the range sf column is the difference of these. Finally, the last column is the ratio of the Max SF and Min SF columns.

Breakdown Tables

Table S 2.A.1 begins the breakdown analysis for the various series considered in the sliding spans analysis. The key concept here is the MPD described above in the section "Table S 1.A" on page 3128 and in the section "Computational Details for Sliding Spans Analysis" on page 3111. For a month or quarter that appears in two or more spans, the maximum percentage difference is computed and tested against a cutoff level. If it exceeds this cutoff, it is counted as an instance of exceeding the level. It is of interest to see if such instances fall disproportionately in certain months and years. Tables S 2.A.1 through S 6.A.3 display this breakdown for all series considered.

Table S 2.A.1

Table S 2.A.1 gives the monthly (quarterly) breakdown for the seasonal factors (table D10). The first column identifies the month or quarter. The next column is the number of times the MPD for D10 exceeded 3.0%, followed by the total count. The last is the average maximum percentage difference for the corresponding month or quarter.

Table S 2.A.2

Table S 2.A.2 gives the same information as Table S 2.A.1, but on a yearly basis.

Table S 2.A.3

The description of Table S 2.A.3 requires the definition of "Sign Change" and "Turning Point."

First, some motivation. Recall that for a highly stable series, adding or deleting a small number of observations should not affect the estimation of the various components of a seasonal adjustment procedure.

Consider Table D10, the seasonal factors in a sliding spans analysis that uses four spans. For a given observation t, looking across the four spans, we can easily pick out large differences if they occur. More subtle differences can occur when estimates go from above to below (or vice versa) a base level. In the case of multiplicative model, the seasonal factors have a base level of 100.0. So it is useful to enumerate those instances where both a large change occurs (an MPD value exceeding 3.0%) and a change of sign (with respect to the base) occur.

Let B denote the base value (which in general depends on the component being considered and the model type, multiplicative or additive). If, for span 1, $S_t(1)$ is below B (i.e., $S_t(1) - B$ is negative) and for some subsequent span k, $S_t(k)$ is above B (i.e., $S_t(k) - B$ is positive), then a positive "Change in Sign" has occurred at observation t. Similarly, if, for span 1, $S_t(1)$ is above B, and for some subsequent span k, $S_t(k)$ is below B, then a negative "Change in Sign" has occurred. Both cases, positive or negative, constitute a "Change in Sign"; the actual direction is indicated in tables S 7.A through S 7.E, which are described below.

Another behavior of interest occurs when component estimates increase then decrease (or vice versa) across spans for a given observation. Using the preceding example, the seasonal factors at observation t could first increase, then decrease across the four spans.

This behavior, combined with an MPD exceeding the level, is of interest in questions of stability.

Again, consider Table D10, the seasonal factors in a sliding spans analysis that uses four spans. For a given observation t (containing at least three spans), note the level of D10 for the first span. Continue across the spans until a difference of 1.0% or greater occurs (or no more spans are left), noting whether the difference is up or down. If the difference is up, continue until a difference of 1.0% or greater occurs downward (or no more spans are left). If such an up-down combination occurs, the observation is counted as an up-down turning point. A similar description occurs for a down-up turning point. Tables S 7.A through S 7.E, described

below, show the occurrence of turning points, indicating whether up-down or down-up. Note that it requires at least three spans to test for a turning point. Hence Tables S 2.A.3 through S 6.A.3 show a reduced number in the "Turning Point" row for the "Total Tested" column, and in Tables S 7.A through S 7.E, the turning points symbols can occur only where three or more spans overlap.

With these descriptions of sign change and turning point, we now describe Table S 2.A.3. The first column gives the type or category, the second column gives the total number of observations falling into the category, the third column gives the total number tested, and the last column gives the percentage for the number found in the category.

The first category (row) of the table is for flagged observations—that is, those observations where the MPD exceeded the appropriate cutoff level (3.0% is default for the seasonal factors). The second category is for level changes, while the third category is for turning points. The fourth category is for flagged sign changes—that is, for those observations that are sign changes, how many are also flagged. Note the total tested column for this category equals the number found for sign change, reflecting the definition of the fourth category.

The fifth column is for flagged turning points—that is, for those observations that are turning points, how many are also flagged.

The footnote to Table S 2.A.3 gives the U.S. Census Bureau recommendation for thresholds, as described in the section "Computational Details for Sliding Spans Analysis" on page 3111.

Table S 2.B

Table S 2.B gives the histogram of flagged for seasonal factors (Table D10) using the appropriate cutoff value (default 3.0%). This table looks at the spread of the number of times the MPD exceeded the corresponding level. The range is divided up into four intervals: 3.0%–4.0%, 4.0%–5.0%, 5.0%–6.0%, and greater than 6.0%. The first column shows the symbol used in Table S 7.A, the second column gives the range in interval notation, and the last column gives the number found in the corresponding interval. Note that the sum of the last column should agree with the "Number Found" column of the "Flagged MPD" row in Table S 2.A.3.

Table S 2.C

Table S 2.C gives selected percentiles for the MPD for the seasonal factors (Table D10).

Tables S 3.A.1 through S 3.A.3

These table relate to the trading-day factors (Table C18) and follow the same format as Tables S 2.A.1 through S 2.A.3. The only difference between these tables and Tables S 2.A.1 through S 2.A.3 is the default cutoff value of 2.0% instead of the 3.0% used for the seasonal factors.

Tables S 3.B, S 3.C

These tables, applied to the trading-day factors (Table C18), are the same format as Tables S 2.B through S 2.C. The default cutoff value is different, with corresponding differences in the intervals in S 3.B.

Tables S 4.A.1 through S 4.A.3

These tables relate to the seasonally adjusted series (Table D11) and follow the same format as Tables S 2.A.1 through S 2.A.3. The same default cutoff value of 3.0% is used.

Tables S 4.B, S 4.C

These tables, applied to the seasonally adjusted series (Table D11), are the same format as tables S 2.B through S 2.C.

Tables S 5.A.1 through S 5.A.3

These table relate to the month-to-month (or quarter-to-quarter) differences in the seasonally adjusted series, and follow the same format as Tables S 2.A.1 through S 2.A.3. The same default cutoff value of 3.0% is used.

Tables S 5.B, S 5.C

These tables, applied to the month-to-month (or quarter-to-quarter) differences in the seasonally adjusted series, are the same format as tables S 2.B through S 2.C. The same default cutoff value of 3.0% is used.

Tables S 6.A.1 through S 6.A.3

These table relate to the year-to-year differences in the seasonally adjusted series, and follow the same format as Tables S 2.A.1 through S 2.A.3. The same default cutoff value of 3.0% is used.

Tables S 6.B, S 6.C

These tables, applied to the year-to-year differences in the seasonally adjusted series, are the same format as tables S 2.B through S 2.C. The same default cutoff value of 3.0% is used.

Table S 7.A

Table S 7.A gives the entire listing of the seasonal factors (Table D10) for each span. The first column gives the date for each observation included in the spans. Note that the dates do not cover the entire original data set. Only those observations included in one or more spans are listed.

The next *N* columns (where *N* is the number of spans) are the individual spans starting at the earliest span. The span columns are labeled by their beginning and ending dates.

Following the last span is the "Sign Change" column. As explained in the description of Table S 2.A.3, a sign change occurs at a given observation when the seasonal factor estimates go from above to below, or below to above, a base level. For the seasonal factors, 100.0 is the base level for the multiplicative model, 0.0 for the additive model. A blank value indicates no sign change, a "U" indicates a movement "upward" from the base level and a "D" indicates a movement "downward" from the base level.

The next column is the "Turning Point" column. As explained in the description of Table S 2.A.3, a turning point occurs when there is an upward then downward movement, or downward then upward movement, of sufficient magnitude. A blank value indicates no turning point, a "U-D" indicates a movement "upward then downward," and a "D-U" indicates a movement "downward then upward."

The next column is the maximum percentage difference (MPD). This quantity, described in the section "Computational Details for Sliding Spans Analysis" on page 3111, is the main computation for sliding spans analysis. A measure of how extreme the MPD value is given in the last column, the "Level of Excess" column. The symbols used and their meaning are described in Table S 2.A.3. If a given observation has exceeded the cutoff, the level of excess column is blank.

Table S 7.B

Table S 7.B gives the entire listing of the trading-day factors (Table C18) for each span. The format of this table is exactly like that of Table S 7.A.

Table S 7.C

Table S 7.C gives the entire listing of the seasonally adjusted data (Table D11) for each span. The format of this table is exactly like that of Table S 7.A except for the "Sign Change" column, which is not printed. The seasonally adjusted data have the same units as the original data; there is no natural base level as in the case of a percentage. Hence the sign change is not appropriate for D11.

Table S 7.D

Table S 7.D gives the entire listing of the month-to-month (or quarter-to-quarter) changes in seasonally adjusted data for each span. The format of this table is exactly like that of Table S 7.A.

Table S 7.E

Table S 7.E gives the entire listing of the year-to-year changes in seasonally adjusted data for each span. The format of this table is exactly like that of Table S 7.A.

Printed Output from the ARIMA Statement

The information printed by default for the ARIMA model includes the parameter estimates, their approximate standard errors, *t* ratios, and variances, the standard deviation of the error term, and the AIC and SBC statistics for the model. In addition, a criteria summary for the chosen model is given that shows the values for each of the three test criteria and the corresponding critical values.

If the PRINTALL option is specified, a summary of the nonlinear estimation optimization and a table of Box-Ljung statistics is also produced. If the automatic model selection is used, this information is printed for each of the five predefined models. Finally, a model selection summary is printed, showing the final model chosen.

ODS Table Names

PROC X11 assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in the following table.

NOTE: For monthly and quarterly tables, use the ODS names MonthlyTables and QuarterlyTables; For brevity, only the MonthlyTables are listed here; the QuarterlyTables are simply duplicates. Printing of individual tables can be specified by using the TABLES table_name, which is not listed here. Printing groups of tables is specified in the MONTHLY and QUARTERLY statements by specifying the option PRINTOUT=NONE|STANDARD|LONG|FULL. The default is PRINTOUT=STANDARD.

Table 43.5 ODS Tables Produced in PROC X11

ODS Table Name	Description	Option
ODS Tables Created by the MONTHLY and QUARTERLY Statements		
Preface	X11 Seasonal Adjustment Program information giving credits, dates, and so on	always printed unless NOPRINT
A1	Table A1: original series	
A2	Table A2: prior monthly	
A3	Table A3: original series adjusted for prior monthly factors	
A4	Table A4: prior trading day adjustment factors with and without length of month adjustment	
A5	Table A5: original series adjusted for priors	
B1	Table B1: original series or original series adjusted for priors	
B2	Table B2: trend cycle—centered nn-term moving average	
B3	Table B3: unmodified SI ratios	
B4	Table B4: replacement values for extreme SI ratios	
B5	Table B5: seasonal factors	
B6	Table B6: seasonally adjusted series	
B7	Table B7: trend cycle—Henderson curve	
B8	Table B8: unmodified SI ratios	
B9	Table B9: replacement values for extreme SI ratios	
B10	Table B10: seasonal factors	
B11	Table B11: seasonally adjusted series	
B13	Table B13: irregular series	
B15	Table B15: preliminary trading day regression	
B16	Table B16: trading day adjustment factors derived from regression	
B17	Table B17: preliminary weights for irregular component	
B18	Table B18: trading day adjustment factors from combined weights	
B19	Table B19: original series adjusted for preliminary combined trading day weights	
C1	Table C1: original series adjusted for preliminary weights	
C2	Table C2: trend cycle—centered nn-term moving average	

Table 43.5 *continued*

ODS Table Name	Description	Option
C4	Table C4: modified SI ratios	
C5	Table C5: seasonal factors	
C6	Table C6: seasonally adjusted series	
C7	Table C7 trend cycle—Henderson curve	
C9	Table C9: modified SI ratios	
C10	Table C10: seasonal factors	
C11	Table C11: seasonally adjusted series	
C13	Table C13: irregular series	
C15	Table C15: final trading day regression	
C16	Table C16: trading day adjustment factors derived from regression	
C17	Table C17: final weights for irregular component	
C18	Table C18: trading day adjustment factors from combined weights	
C19	Table C19: original series adjusted for final combined trading day weights	
D1	Table D1: original series adjusted for final weights nn-term moving average	
D4	Table D4: modified SI ratios	
D5	Table D5: seasonal factors	
D6	Table D6: seasonally adjusted series	
D7	Table D7: trend cycle—Henderson curve	
D8	Table D8: final unmodified SI ratios	
D10	Table D10: final seasonal factors	
D11	Table D11: final seasonally adjusted series	
D12	Table D12: final trend cycle—Henderson curve	
D13	Table D13: final irregular series	
E1	Table E1: original series modified for extremes	
E2	Table E2: modified seasonally adjusted series	
E3	Table E3: modified irregular series	
E5	Table E5: month-to-month changes in original series	
E6	Table E6: month-to-month changes in final seasonally adjusted series	
F1	Table F1: MCD moving average	
A13	Table A13: ARIMA forecasts	ARIMA statement
A14	Table A14: ARIMA backcasts	ARIMA statement
A15	Table A15: ARIMA extrapolation	ARIMA statement

Table 43.5 *continued*

ODS Table Name	Description	Option
B14	Table B14: irregular values excluded from trading day regression	
C14	Table C14: irregular values excluded from trading day regression	
D9	Table D9: final replacement values	
PriorDailyWgts	adjusted prior daily weights	
TDR_0	final/preliminary trading day regression, part 1	MONTHLY only, TDREGR=ADJUST, TEST
TDR_1	final/preliminary trading day regression, part 2	MONTHLY only, TDREGR=ADJUST, TEST
StandErrors	standard errors of trading day adjustment factors	MONTHLY only, TDREGR=ADJUST, TEST
D9A	year-to-year change in irregular and seasonal components and moving seasonality ratio	
StableSeasTest	stable seasonality test	
StableSeasFTest	moving seasonality test	
KruskalWallisTest	nonparametric test for the presence of seasonality assuming stability	
CombinedSeasonalityTest	summary of results and combined test for the presence of identifiable seasonality	
f2a	F2 summary measures, part 1	
f2b	F2 summary measures, part 2	
f2c	F2 summary measures, part 3	
f2d	I/C ratio for monthly/quarterly span	
f2f	average % change with regard to sign and standard deviation over span	
E4	differences or ratios of annual totals for original and adjusted series	
ChartG1	chart G1	
ChartG2	chart G2	

Table 43.5 *continued*

ODS Table Name	Description	Option
ODS Tables Created by the ARIMA Statement		
CriteriaSummary	criteria summary	ARIMA statement
ConvergeSummary	convergence summary	
ArimaEst	ARIMA estimation results, part 1	
ArimaEst2	ARIMA estimation results, part 2	
Model_Summary	model summary	
Ljung_BoxQ	table of Ljung-Box Q statistics	
A13	Table A13: ARIMA forecasts	
A14	Table A14: ARIMA backcasts	
A15	Table A15: ARIMA extrapolation	
ODS Tables Created by the SSPAN Statement		
SPR0A_1	S 0.A sliding spans analysis, number, length of spans	default printing
SpanDates	S 0.A sliding spans analysis: dates of spans	
SPR0B	S 0.B summary of F tests for stable and moving seasonality	
SPR1_1	S 1.A range analysis of seasonal factors	
SPR1_b	S 1.B summary of range measures	
SPRXA	2XA.1 breakdown of differences by month or quarter	
SPRXB_2	S X.B histogram of flagged observations	
SPRXA_2	S X.A.2 breakdown of differences by year	
MpdStats	S X.C: statistics for maximum percentage differences	
S_X_A_3	S 2.X.3 breakdown summary of flagged observations	
SPR7_X	S 7.X sliding spans analysis	PRINTALL

Examples: X11 Procedure

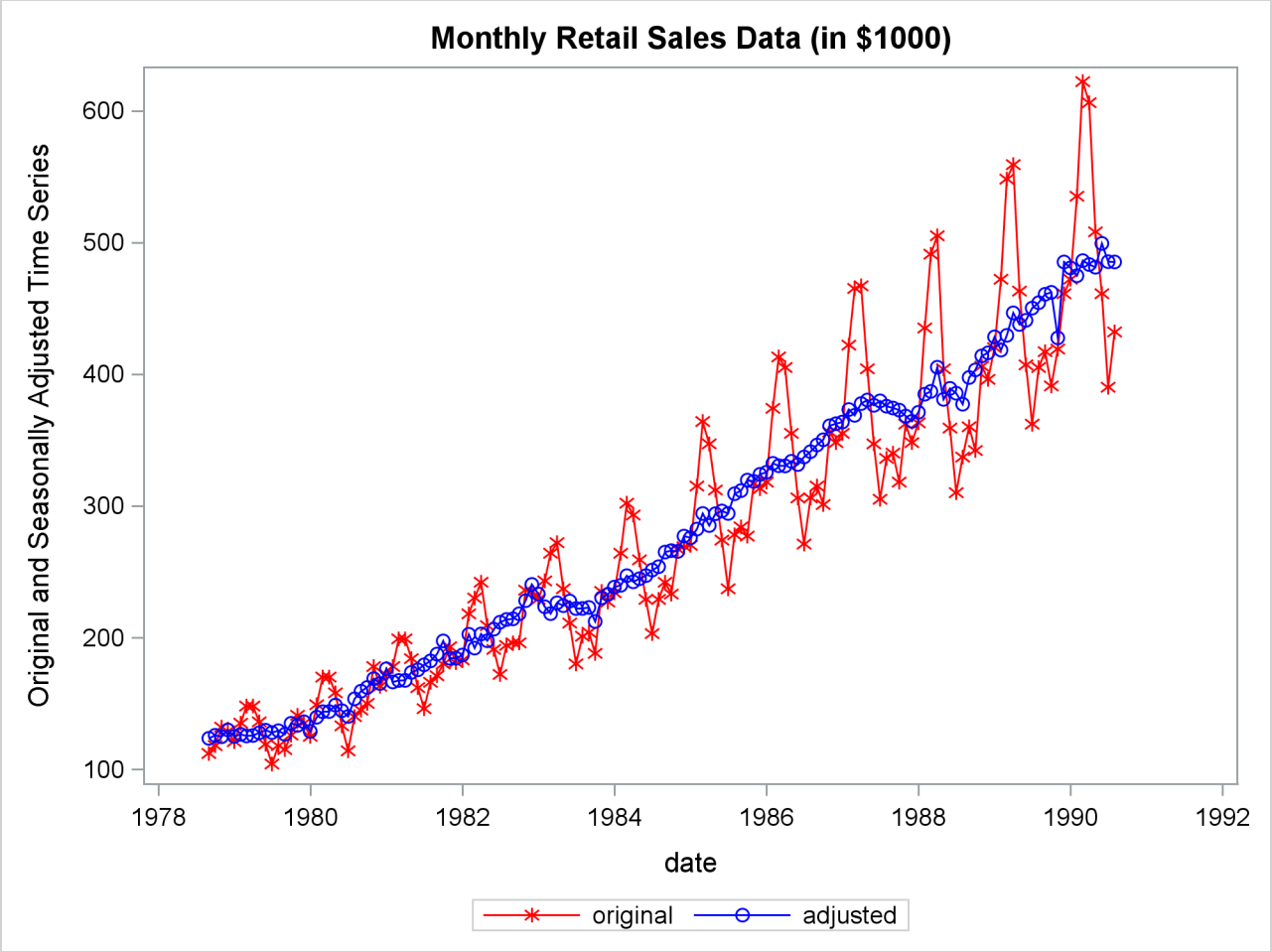
Example 43.1: Component Estimation—Monthly Data

This example computes and plots the final estimates of the individual components for a monthly series. In the first plot (Output 43.1.1), an overlaid plot of the original and seasonally adjusted data is produced. The trend in the data is more evident in the seasonally adjusted data than in the original data. This trend is even more clear in Output 43.1.3, the plot of Table D12, the trend cycle. Note that both the seasonal factors and the irregular factors vary around 100, while the trend cycle and the seasonally adjusted data are in the scale of the original data.

From Output 43.1.2 the seasonal component appears to be slowly increasing, while no apparent pattern exists for the irregular series in Output 43.1.4.

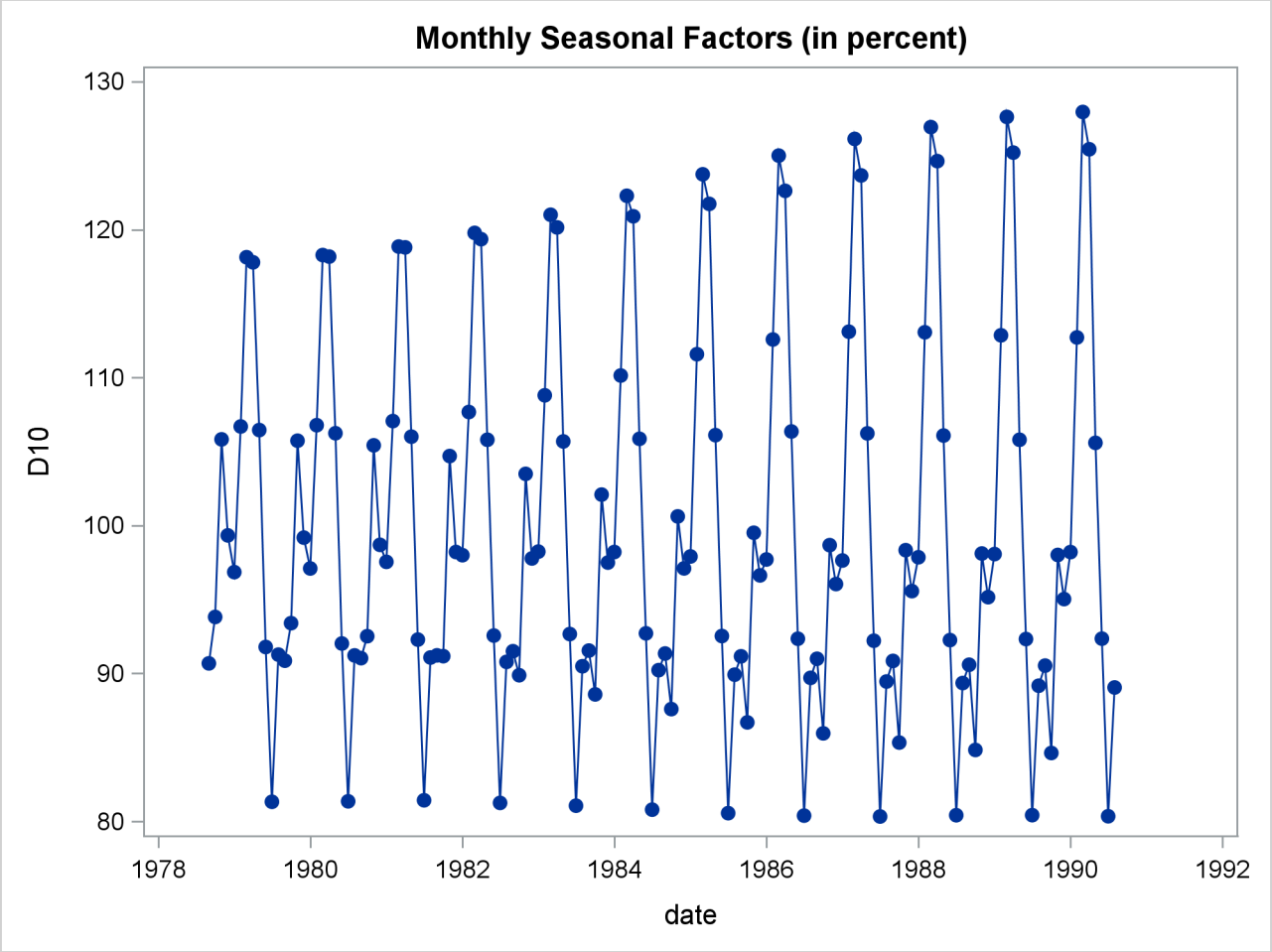
```
data sales;
   input sales @@;
   date = intnx( 'month', '01sep1978'd, _n_-1 );
   format date monyy7.;
datalines;
112 118 132 129 121 135 148 148 136 119 104 118

   ... more lines ...


proc x11 data=sales noprint;
   monthly date=date;
   var sales;
   tables b1 d11;
   output out=out b1=series d10=d10 d11=d11
                  d12=d12 d13=d13;
run;

title 'Monthly Retail Sales Data (in $1000)';
proc sgplot data=out;
   series x=date y=series / markers
                           markerattrs=(color=red symbol='asterisk')
                           lineattrs=(color=red)
                           legendlabel="original" ;
   series x=date y=d11    / markers
                           markerattrs=(color=blue symbol='circle')
                           lineattrs=(color=blue)
                           legendlabel="adjusted" ;
   yaxis label='Original and Seasonally Adjusted Time Series';
run;
```

Output 43.1.1 Plots of Original and Seasonally Adjusted Data

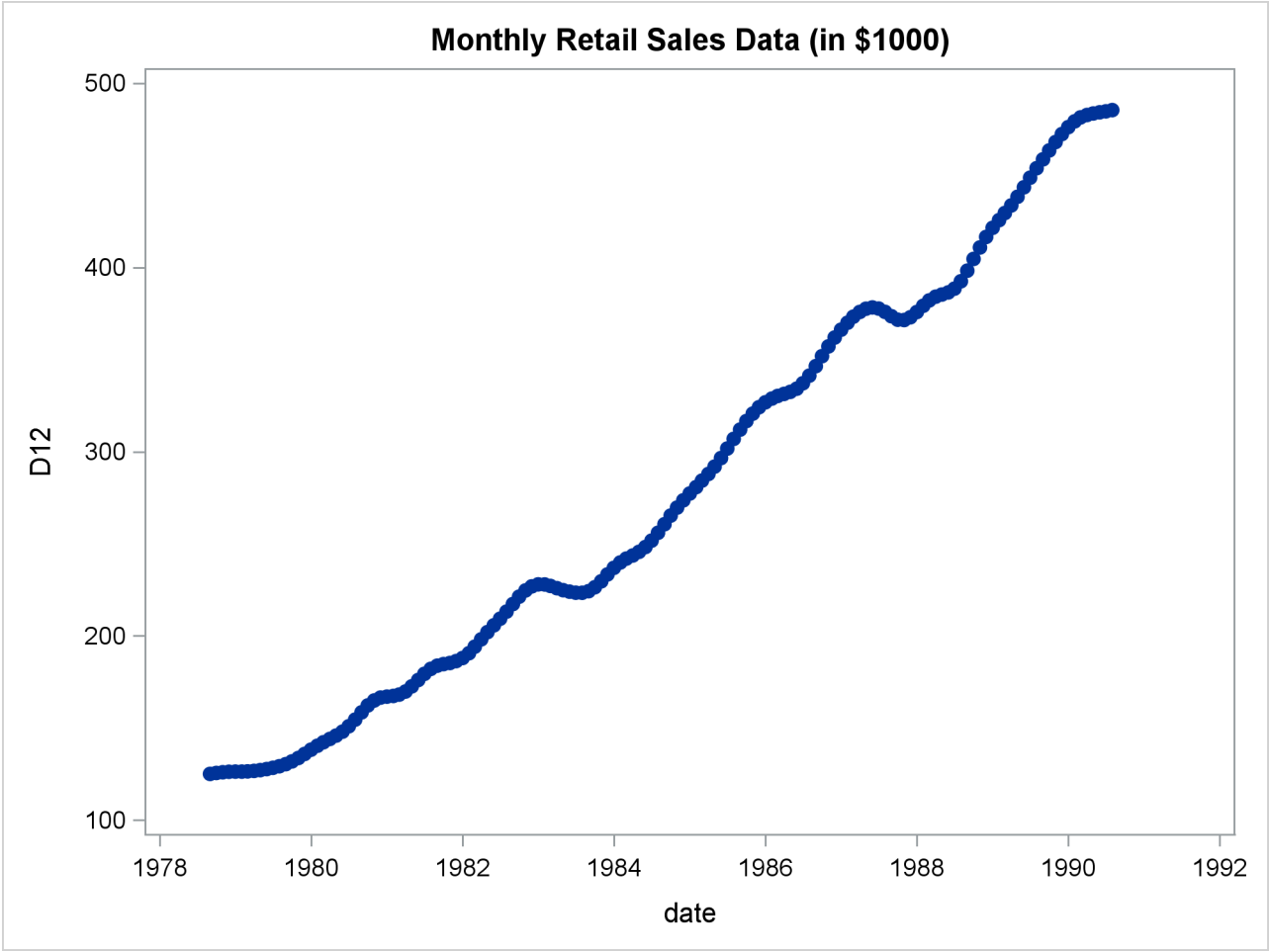



```
title 'Monthly Seasonal Factors (in percent)';
proc sgplot data=out;
   series x=date y=d10 / markers markerattrs=(symbol=CircleFilled) ;
run;

title 'Monthly Retail Sales Data (in $1000)';
proc sgplot data=out;
   series x=date y=d12 / markers markerattrs=(symbol=CircleFilled) ;
run;

title 'Monthly Irregular Factors (in percent)';
proc sgplot data=out;
   series x=date y=d13 / markers markerattrs=(symbol=CircleFilled) ;
run;
```

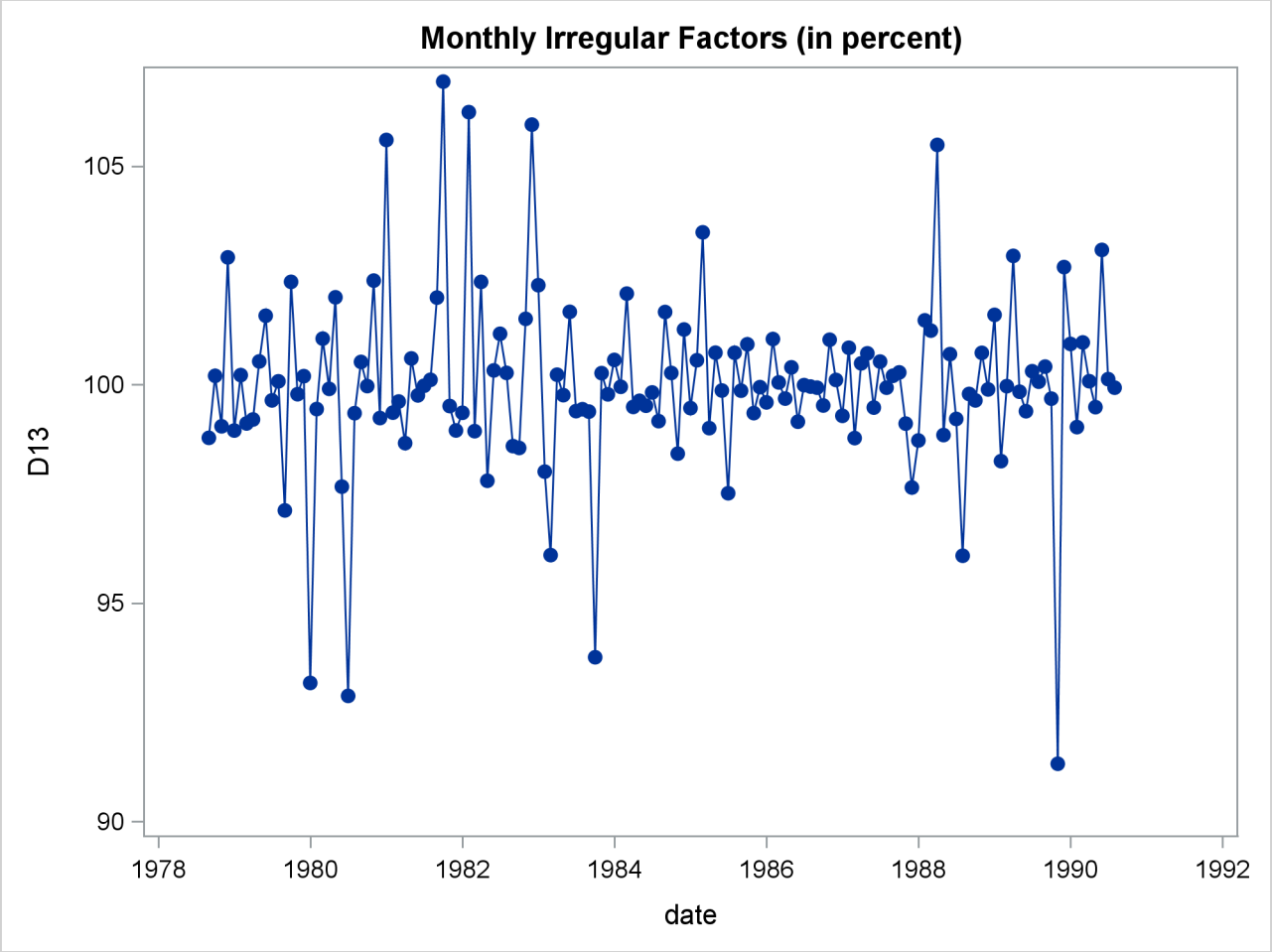
Output 43.1.2 Plot of D10, the Final Seasonal Factors



Output 43.1.3 Plot of D12, the Final Trend Cycle



Output 43.1.4 Plot of D13, the Final Irregular Series



Example 43.2: Components Estimation—Quarterly Data

This example is similar to Example 43.1, except quarterly data are used. Tables B1, the original series, and D11, the final seasonally adjusted series, are printed by the TABLES statement. The OUTPUT statement writes the listed tables to an output data set.

```
data quarter;
   input date yyq6. +1 fy35rr 5.2;
   format date yyq6.;
datalines;
1971Q1 6.59

   ... more lines ...
```

```
title 'Monthly Retail Sales Data (in $1000)';
proc x11 data=quarter;
   var fy35rr;
   quarterly date=date;
   tables b1 d11;
   output out=out b1=b1 d10=d10 d11=d11 d12=d12 d13=d13;
run;
```

Output 43.2.1 X11 Procedure Quarterly Example

Monthly Retail Sales Data (in $1000)

The X11 Procedure

Seasonal Adjustment of - fy35rr

X-11 Seasonal Adjustment Program
U. S. Bureau of the Census
Economic Research and Analysis Division
November 1, 1968

The X-11 program is divided into seven major parts.

Part Description

A. Prior adjustments, if any
B. Preliminary estimates of irregular component weights and regression trading day factors
C. Final estimates of above
D. Final estimates of seasonal, trend-cycle and irregular components
E. Analytical tables
F. Summary measures
G. Charts

Series - fy35rr
Period covered - 1st Quarter 1971 to 4th Quarter 1976

B1 Original Series

Year	1st	2nd	3rd	4th	Total
1971	6.590	6.010	6.510	6.180	25.290
1972	5.520	5.590	5.840	6.330	23.280
1973	6.520	7.350	9.240	10.080	33.190
1974	9.910	11.150	12.400	11.640	45.100
1975	9.940	8.160	8.220	8.290	34.610
1976	7.540	7.440	7.800	7.280	30.060
Avg	7.670	7.617	8.335	8.300	

Total: 191.53 Mean: 7.9804 S.D.: 1.9424

Output 43.2.2 X11 Procedure Quarterly Example, Table D11

D11 Final Seasonally Adjusted Series					
Year	1st	2nd	3rd	4th	Total
1971	6.877	6.272	6.222	5.956	25.326
1972	5.762	5.836	5.583	6.089	23.271
1973	6.820	7.669	8.840	9.681	33.009
1974	10.370	11.655	11.855	11.160	45.040
1975	10.418	8.534	7.853	7.947	34.752
1976	7.901	7.793	7.444	6.979	30.116
Avg	8.025	7.960	7.966	7.969	

Total: 191.51 Mean: 7.9797 S.D.: 1.9059

Example 43.3: Outlier Detection and Removal

PROC X11 can be used to detect and replace outliers in the irregular component of a monthly or quarterly series.

The weighting scheme used in measuring the "extremeness" of the irregulars is developed iteratively; thus the statistical properties of the outlier adjustment method are unknown.

In this example, the data are simulated by generating a trend plus a random error. Two periods in the series were made "extreme" by multiplying one generated value by 2.0 and another by 0.10. The additive model is appropriate based on the way the data were generated. Note that the trend in the generated data was modeled automatically by the trend cycle component estimation.

The detection of outliers is accomplished by considering Table D9, the final replacement values for extreme S-I ratios. This table indicates which observations had irregular component values more than FULLWEIGHT= standard deviation units from 0.0 (1.0 for the multiplicative model). The default value of the FULLWEIGHT= option is 1.5; a larger value would result in fewer observations being declared extreme.

In this example, FULLWEIGHT=3.0 is used to isolate the extreme inflated and deflated values generated in the DATA step. The value of ZEROWEIGHT= must be greater than FULLWEIGHT; it is given a value of 3.5.

A plot of the original and modified series, Output 43.3.2, shows that the deviation from the trend line for the modified series is greatly reduced compared with the original series.

```
data a;
   retain seed 99831;
   do kk = 1 to 48;
      x = kk + 100 + rannor( seed );
      date = intnx( 'month', '01jan1970'd, kk-1 );
      if kk = 20 then x = 2 * x;
      else if kk = 30 then x = x / 10;
      output;
      end;
run;
```

```
proc x11 data=a;
   monthly date=date additive
           fullweight=3.0 zeroweight=3.5;
   var x;
   table d9;
   output out=b b1=original e1=e1;
run;

proc sgplot data=b;
   series x=date y=original / markers
                              markerattrs=(color=red symbol='asterisk')
                              lineattrs=(color=red)
                              legendlabel="unmodified" ;
   series x=date y=e1       / markers
                              markerattrs=(color=blue symbol='circle')
                              lineattrs=(color=blue)
                              legendlabel="modified" ;
   yaxis label='Original and Outlier Adjusted Time Series';
run;
```

Output 43.3.1 Detection of Extreme Irregulars

Monthly Retail Sales Data (in $1000)

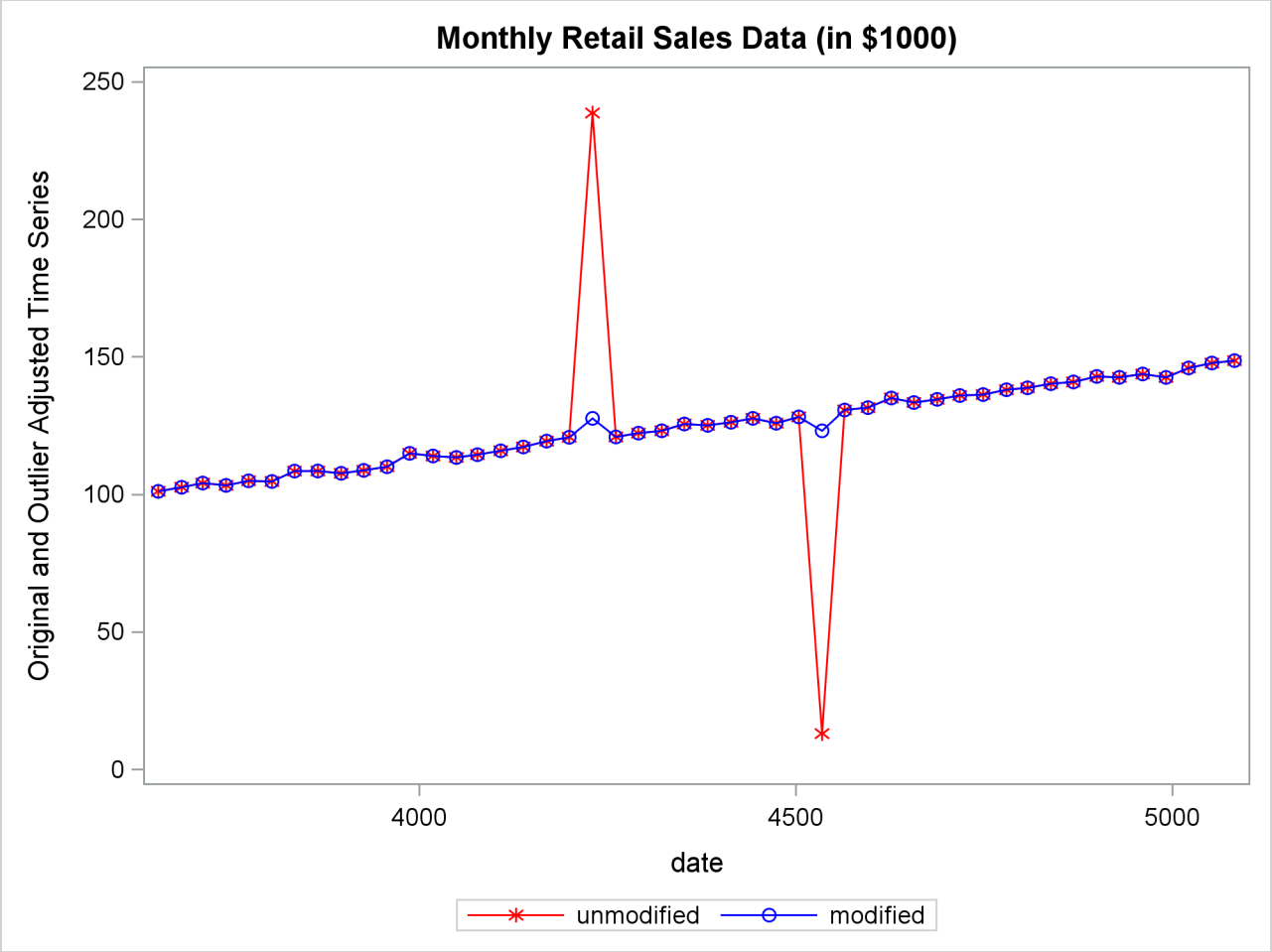
The X11 Procedure

Seasonal Adjustment of - x

D9 Final Replacement Values for Extreme SI Ratios

Year	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
1970	.	.	.	.	.	.	.	.	.	.	.	.
1971	.	.	.	.	.	.	.	11.180	.	.	.	.
1972	.	.	.	.	.	-10.671	.	.	.	.	.	.
1973	.	.	.	.	.	.	.	.	.	.	.	.

Output 43.3.2 Plot of Modified and Unmodified Values



References

Bell, W. R., and Hillmer, S. C. (1984). "Issues Involved with the Seasonal Adjustment of Economic Time Series." *Journal of Business and Economic Statistics* 2:291–320.

Bobbit, L. G., and Otto, M. C. (1990). "Effects of Forecasts on the Revisions of Seasonally Adjusted Data Using the X-11 Adjustment Procedure." *Proceedings of the Business and Economic Statistics Section of the American Statistical Association* 449–453.

Buszuwski, J. A. (1987). "Alternative ARIMA Forecasting Horizons When Seasonally Adjusting Producer Price Data with X-11-ARIMA." In *Proceedings of the Business and Economic Statistics Section*, 488–493. Alexandria, VA: American Statistical Association.

Cleveland, W. P., and Tiao, G. C. (1976). "Decomposition of Seasonal Time Series: A Model for the Census X-11 Program." *Journal of the American Statistical Association* 71:581–587.

Cleveland, W. S., and Devlin, S. J. (1980). "Calendar Effects in Monthly Time Series: Detection by Spectrum Analysis and Graphical Methods." *Journal of the American Statistical Association* 75:487–496.

Dagum, E. B. (1980). *The X-11-ARIMA Seasonal Adjustment Method*. Ottawa: Statistics Canada.

Dagum, E. B. (1982a). "The Effects of Asymmetric Filters on Seasonal Factor Revision." *Journal of the American Statistical Association* 77:732–738.

Dagum, E. B. (1982b). "Revisions of Seasonally Adjusted Data Due to Filter Changes." In *Proceedings of the American Statistical Association, Business and Economic Section*, 39–45. Alexandria, VA: American Statistical Association.

Dagum, E. B. (1982c). "Revisions of Time Varying Seasonal Filters." *Journal of Forecasting* 1:173–187.

Dagum, E. B. (1983). *The X-11-ARIMA Seasonal Adjustment Method*. Technical Report 12-564E, Statistics Canada, Ottawa.

Dagum, E. B. (1985). "Moving Averages." In *Encyclopedia of Statistical Sciences*, vol. 5, edited by S. Kotz, N. L. Johnson, and C. B. Read. New York: John Wiley & Sons.

Dagum, E. B. (1988). *The X-11-ARIMA/88 Seasonal Adjustment Method: Foundations and User's Manual*. Ottawa: Statistics Canada.

Dagum, E. B., and Laniel, N. (1987). "Revisions of Trend Cycle Estimators of Moving Average Seasonal Adjustment Method." *Journal of Business and Economic Statistics* 5:177–189.

Davies, N., Triggs, C. M., and Newbold, P. (1977). "Significance Levels of the Box-Pierce Portmanteau Statistic in Finite Samples." *Biometrika* 64:517–522.

Findley, D. F., and Monsell, B. C. (1986). "New Techniques for Determining If a Time Series Can Be Seasonally Adjusted Reliably, and Their Application to U.S. Foreign Trade Series." In *Regional Econometric Modeling*, edited by M. R. Perryman, and J. R. Schmidt, 195–228. Amsterdam: Kluwer-Nijhoff.

Findley, D. F., Monsell, B. C., Shulman, H. B., and Pugh, M. G. (1990). "Sliding Spans Diagnostics for Seasonal and Related Adjustments." *Journal of the American Statistical Association* 85:345–355.

Ghysels, E. (1990). "Unit Root Tests and the Statistical Pitfalls of Seasonal Adjustment: The Case of U.S. Post War Real GNP." *Journal of Business and Economic Statistics* 8:145–152.

Higginson, J. (1975). *An F Test for the Presence of Moving Seasonality When Using Census Method II-X-II Variant*. StatCan Staff Paper STC2102E, Seasonal Adjustment and Time Series Analysis Staff, Statistics Canada, Ottawa.

Huot, G., Chui, L., Higginson, J., and Gait, N. (1986). "Analysis of Revisions in the Seasonal Adjustment of Data Using X11ARIMA Model-Based Filters." *International Journal of Forecasting* 2:217–229.

Ladiray, D., and Quenneville, B. (2001). *Seasonal Adjustment with the X-11 Method*. New York: Springer-Verlag.

Laniel, N. (1985). "Design Criteria for the 13-Term Henderson End-Weights." Working paper, Methodology Branch, Statistics Canada, Ottawa.

Lehmann, E. L., and D'Abrera, H. J. M. (2006). *Nonparametrics: Statistical Methods Based on Ranks*. Rev. ed. New York: Springer Science & Business Media.

Ljung, G. M., and Box, G. E. P. (1978). "On a Measure of Lack of Fit in Time Series Models." *Biometrika* 65:297–303.

Lothian, J. (1978). *The Identification and Treatment of Moving Seasonality in the X-11 Seasonal Adjustment Method*. StatCan Staff Paper STC0803E, Seasonal Adjustment and Time Series Analysis Staff, Statistics Canada, Ottawa.

Lothian, J. (1984a). *The Identification and Treatment of Moving Seasonality in the X-11-ARIMA Seasonal Adjustment Method*. StatCan Staff Paper, Seasonal Adjustment and Time Series Analysis Staff, Statistics Canada, Ottawa.

Lothian, J. (1984b). "The Identification and Treatment of Moving Seasonality in X-11-ARIMA." In *Proceedings of the Business and Economic Statistics Section*, 166–171. Alexandria, VA: American Statistical Association.

Lothian, J., and Morry, M. (1978a). *Selection of Models for the Automated X-11-ARIMA Seasonal Adjustment Program*. StatCan Staff Paper STC1789, Seasonal Adjustment and Time Series Analysis Staff, Statistics Canada, Ottawa.

Lothian, J., and Morry, M. (1978b). *A Test for the Presence of Identifiable Seasonality When Using the X-11-ARIMA Program*. StatCan Staff Paper STC2118, Seasonal Adjustment and Time Series Analysis Staff, Statistics Canada, Ottawa.

Marris, S. N. (1961). "The Treatment of Moving Seasonality in Census Method II." In *Seasonal Adjustment on Electronic Computers*, 257–309. Paris: Organisation for Economic Co-operation and Development.

Monsell, B. C. (1984). *The Substantive Changes in the X-11 Procedure of X-11-ARIMA*. SRD Research Report Census/SRD/RR-84/10, Statistical Research Division, US Bureau of the Census.

Pierce, D. A. (1980). "Data Revisions with Moving Average Seasonal Adjustment Procedures." *Journal of Econometrics* 14:95–114.

Shiskin, J. (1958). "Decomposition of Economic Time Series." *Science* 128:1539–1546.

Shiskin, J., and Eisenpress, H. (1957). "Seasonal Adjustment by Electronic Computer Methods." *Journal of the American Statistical Association* 52.

Shiskin, J., Young, A. H., and Musgrave, J. C. (1967). *The X-11 Variant of the Census Method II Seasonal Adjustment Program*. Technical Report 15, US Department of Commerce, Bureau of the Census.

US Bureau of the Census (1969). *X-11 Information for the User*. Washington, DC: US Government Printing Office.

Young, A. H. (1965). *Estimating Trading Day Variation in Monthly Economic Time Series*. Technical Report 12, US Department of Commerce, Bureau of the Census, Washington, DC.

Chapter 44
The X12 Procedure

Contents

Overview: X12 Procedure **3149**
References **3149**

Overview: X12 Procedure

The X12 procedure is an adaptation of the US Bureau of the Census X-12-ARIMA seasonal adjustment program (US Bureau of the Census 2010). The X-12-ARIMA program was developed by the Time Series Staff of the Statistical Research Division, US Census Bureau. The X-12-ARIMA seasonal adjustment program contains components developed from Statistics Canada's X-11-ARIMA program. The X-12-ARIMA automatic modeling method is based on the work of Gómez and Maravall (1997a, b).

The Time Series Staff of the Statistical Research Division, US Census Bureau, has recently developed a new program, X-13ARIMA-SEATS (US Bureau of the Census 2013). This program incorporates the X-12-ARIMA functionality along with the SEATS functionality that was developed by Gómez and Maravall (1997a, b). The X12 procedure includes improvements on X-12-ARIMA methods that are incorporated into the X-13ARIMA-SEATS program.

Because the US Census Bureau has focused its new development on the X-13ARIMA-SEATS program, a new X13 procedure has been developed to incorporate the X-13ARIMA-SEATS method.

NOTE: The functionality previously available in the X12 procedure is included in the new X13 procedure. You can specify either of the following with the same results:

```
proc x12 ...

proc x13 ...
```

For documentation of the PROC X12 syntax and a description of its details, see Chapter 45, "The X13 Procedure."

References

Gómez, V., and Maravall, A. (1997a). *Guide for Using the Programs TRAMO and SEATS, Beta Version*. Madrid: Banco de España.

Gómez, V., and Maravall, A. (1997b). *Programs TRAMO and SEATS: Instructions for the User, Beta Version*. Madrid: Banco de España.

US Bureau of the Census (2010). *X-12-ARIMA Seasonal Adjustment Program, Version 0.3*. US Bureau of the Census, Washington, DC. http://www.census.gov/srd/www/x12a/.

US Bureau of the Census (2013). *X-13ARIMA-SEATS Reference Manual, Version 1.1*. US Bureau of the Census, Washington, DC. http://www.census.gov/ts/x13as/docX13ASHTML.pdf.

Chapter 45
The X13 Procedure

Contents

Overview: X13 Procedure **3152**
Getting Started: X13 Procedure **3153**
Basic Seasonal Adjustment 3154
Syntax: X13 Procedure **3157**
Functional Summary 3158
PROC X13 Statement 3162
ADJUST Statement 3169
ARIMA Statement 3169
AUTOMDL Statement 3170
BY Statement 3174
CHECK Statement 3174
ESTIMATE Statement 3175
EVENT Statement 3177
FORECAST Statement 3179
ID Statement 3180
IDENTIFY Statement 3180
INPUT Statement 3182
OUTLIER Statement 3183
OUTPUT Statement 3186
PICKMDL Statement (Experimental) 3187
REGRESSION Statement 3188
SEATSDECOMP Statement (Experimental) 3195
TABLES Statement 3197
TRANSFORM Statement 3197
USERDEFINED Statement 3199
VAR Statement 3199
X11 Statement 3199
Details: X13 Procedure **3204**
Data Requirements 3204
Missing Values 3204
SAS Predefined Events 3205
User-Defined Regression Variables 3208
Combined Test for the Presence of Identifiable Seasonality 3209
Computations 3212
PICKMDL Model Selection 3212
SEATS Decomposition 3213

Displayed Output, ODS Table Names, and OUTPUT Tablename Keywords 3213
Table D 8.B 3217
Using Auxiliary Variables to Subset Output Data Sets 3217
ODS Graphics 3218
OUT= Data Set 3220
SEATSDECOMP OUT= Data Set 3221
Special Data Sets 3222
Examples: X13 Procedure **3227**
Example 45.1: ARIMA Model Identification 3227
Example 45.2: Model Estimation 3231
Example 45.3: Seasonal Adjustment 3233
Example 45.4: RegARIMA Automatic Model Selection 3236
Example 45.5: Automatic Outlier Detection 3242
Example 45.6: User-Defined Regressors 3248
Example 45.7: MDLINFOIN= and MDLINFOOUT= Data Sets 3255
Example 45.8: Setting Regression Parameters 3258
Example 45.9: Creating an MDLINFO= Data Set for Use with the PICKMDL Statement 3265
Example 45.10: Illustration of ODS Graphics 3271
Example 45.11: AUXDATA= Data Set 3272
References **3274**

Overview: X13 Procedure

The X13 procedure is an adaptation of the US Bureau of the Census X-13ARIMA-SEATS seasonal adjustment program (US Bureau of the Census 2013c). The X-13ARIMA-SEATS program was developed by the Time Series Staff of the Statistical Research Division, US Census Bureau, by incorporating the SEATS method into the X-12-ARIMA seasonal adjustment program. The X-12-ARIMA seasonal adjustment program contains components developed from Statistics Canada's X-11-ARIMA program (US Bureau of the Census 2010). The X-12-ARIMA automatic modeling method and the SEATS method are based on the work of Gómez and Maravall (1997a, b).

The new X-13ARIMA-SEATS program incorporates the X-12-ARIMA functionality. It also incorporates improvements on X-12-ARIMA methods. Because the X-12-ARIMA methods and improvements are available in X-13ARIMA-SEATS, the new X13 procedure and the existing X12 procedure use the same X-13ARIMA-SEATS methodology, and PROC X12 and PROC X13 are aliases for the same procedure.

The version of PROC X13 documented here was produced by converting the US Census Bureau's FORTRAN code to the SAS development language and adding typical SAS procedure syntax. This conversion work was performed by SAS and resulted in the X13 procedure. Although several features were added during the conversion, credit for the statistical aspects and general methodology of the X13 procedure belongs to the US Census Bureau.

The X13 procedure seasonally adjusts monthly or quarterly time series. The procedure makes additive or multiplicative adjustments and creates an output data set that contains the adjusted time series and intermediate calculations.

The X-13ARIMA-SEATS program includes the X-12-ARIMA program, which combines the capabilities of the X-11 program (Shiskin, Young, and Musgrave 1967) and the X-11-ARIMA/88 program (Dagum 1988) and also introduces some new features (Findley et al. 1998). One of the main enhancements in the X-12-ARIMA program involves the use of a regARIMA model, a regression model with ARIMA (autoregressive integrated moving average) errors. Thus, the X-12-ARIMA program contains methods developed by both the US Census Bureau and Statistics Canada. In addition, the X-12-ARIMA automatic modeling routine is based on the TRAMO (time series regression with ARIMA noise, missing values, and outliers) method (Gómez and Maravall 1997a, b). The four major components of the X-12-ARIMA program are regARIMA modeling, model diagnostics, seasonal adjustment that uses enhanced X-11 methodology, and post-adjustment diagnostics. Statistics Canada's X-11 method fits an ARIMA model to the original series, and then uses the model forecasts to extend the original series. This extended series is then seasonally adjusted by the standard X-11 seasonal adjustment method. The extension of the series improves the estimation of the seasonal factors and reduces revisions to the seasonally adjusted series as new data become available.

Seasonal adjustment of a series is based on the assumption that seasonal fluctuations can be measured in the original series, $O_t, t = 1, \ldots, n$, and separated from trend cycle, trading day, and irregular fluctuations. The seasonal component of this time series, S_t, is defined as the intrayear variation that is repeated consistently or evolves slowly from year to year (Hillmer and Tiao 1982). The trend cycle component, C_t, includes variation that is attributed to the long-term trend, the business cycle, and other long-term cyclical factors. The trading day component, D_t, is the variation that can be attributed to the composition of the calendar. The irregular component, I_t, is the residual variation. Many economic time series are related in a multiplicative fashion ($O_t = S_t C_t D_t I_t$). Other economic series are related in an additive fashion ($O_t = S_t + C_t + D_t + I_t$). A seasonally adjusted time series, $C_t I_t$ or $C_t + I_t$, consists of only the trend cycle and irregular components. For more information about the X-11 seasonal adjustment method, see Ladiray and Quenneville (2001).

Graphics are now available with the X13 procedure. For more information, see the section "ODS Graphics" on page 3218.

Getting Started: X13 Procedure

The most common use of the X13 procedure is to produce a seasonally adjusted series. Eliminating the seasonal component from an economic series facilitates comparison among consecutive months or quarters. A plot of the seasonally adjusted series is often more informative about trends or location in a business cycle than a plot of the unadjusted series.

The following example shows how to use PROC X13 to produce a seasonally adjusted series, $C_t I_t$, from an original series $O_t = S_t C_t D_t I_t$.

In the multiplicative model, the trend cycle component C_t keeps the same scale as the original series O_t, while S_t, D_t, and I_t vary around 1.0. In all displayed tables, these latter components are expressed as percentages and thus vary around 100.0 (in the additive case, they vary around 0.0). However, in the output data set, the data displayed as percentages in the displayed output are expressed as the decimal equivalent and thus vary around 1.0 in the multiplicative case.

The naming convention used in PROC X13 for the tables follows the convention used in the Census Bureau's X-13ARIMA-SEATS program; see *X-13ARIMA-SEATS Reference Manual* (US Bureau of the Census 2013c), *X-13ARIMA-SEATS Quick Reference for DOS* (US Bureau of the Census 2013a), and *X-13ARIMA-SEATS Quick Reference for UNIX/Linux* (US Bureau of the Census 2013b). Also see the section "Displayed Output, ODS Table Names, and OUTPUT Tablename Keywords" on page 3213. The table names are outlined in Table 45.14.

The tables that correspond to parts A through C are intermediate calculations. The final estimates of the individual components are found in the D tables: Table D10 contains the final seasonal factors, Table D12 contains the final trend cycle, and Table D13 contains the final irregular series. If you are primarily interested in seasonally adjusting a series without consideration of intermediate calculations or diagnostics, you need to look only at Table D11, the final seasonally adjusted series. Tables in part E contain information about extreme values and changes in the original and seasonally adjusted series. The tables in part F are seasonal adjustment quality measures. Spectral analysis is performed in part G. For further information about the tables produced by the X11 statement, see Ladiray and Quenneville (2001).

Basic Seasonal Adjustment

Suppose that you have monthly retail sales data starting in September 1978 in a SAS data set named SALES. At this point, you do not suspect that any calendar effects are present, and there are no prior adjustments that need to be made to the data.

In this simplest case, you need only specify the DATE= variable in the PROC X13 statement and request seasonal adjustment in the X11 statement as shown in the following statements:

```
data sales;
   set sashelp.air;
   sales = air;
   date = intnx( 'month', '01sep78'd, _n_-1 );
   format date monyy.;
run;

proc x13 data=sales date=date;
   var sales;
   x11;
   ods select d11;
run;
```

The results of the seasonal adjustment are in Table D11 (the final seasonally adjusted series) in the displayed output shown in Figure 45.1.

Figure 45.1 Basic Seasonal Adjustment

The X13 Procedure

Table D 11: Final Seasonally Adjusted Data
For Variable sales

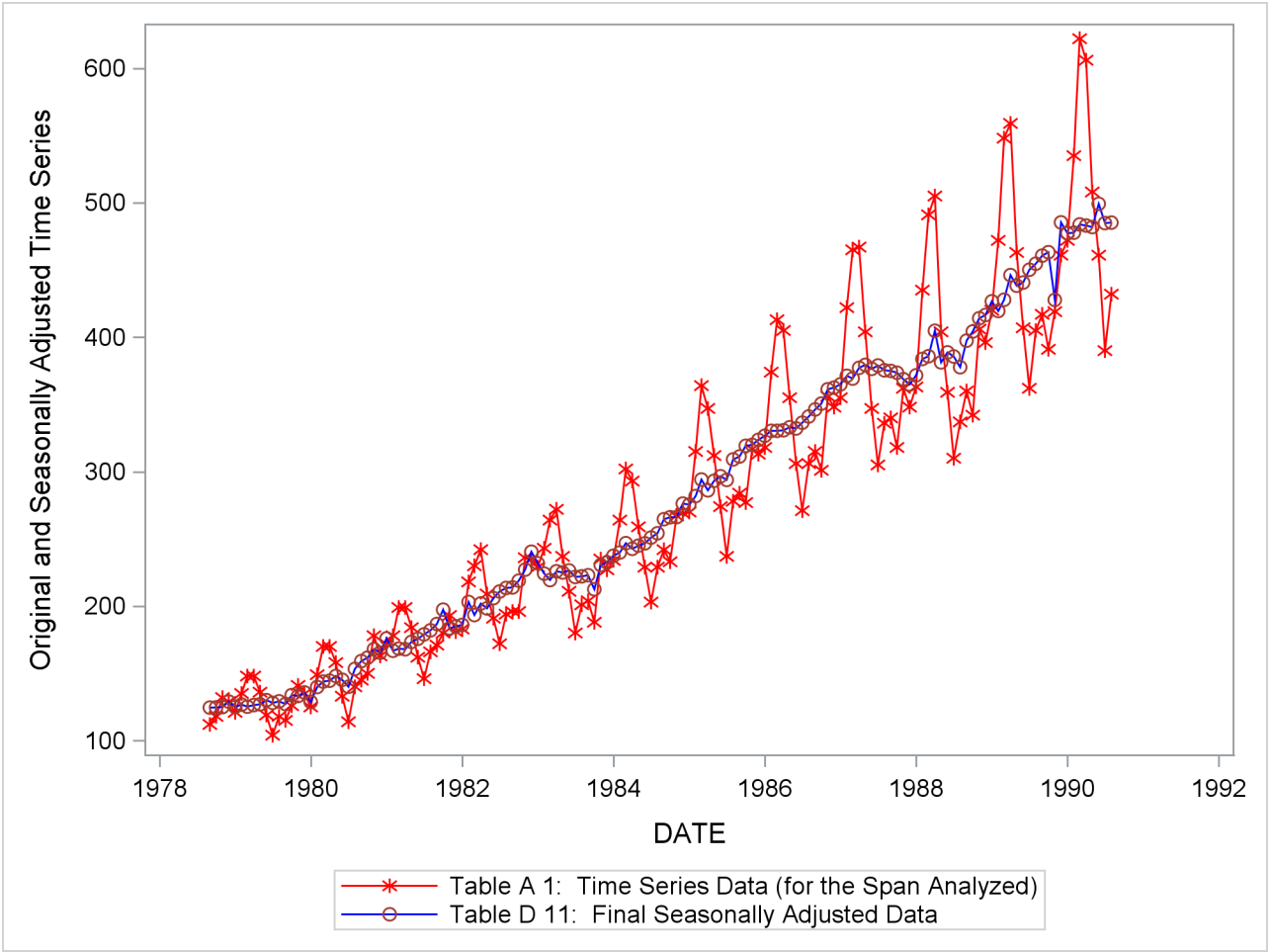
Year	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	Total
1978	.	.	.	.	.	.	.	.	124.560	124.649	124.920	129.002	503.131
1979	125.087	126.759	125.252	126.415	127.012	130.041	128.056	129.165	127.182	133.847	133.199	135.847	1547.86
1980	128.767	139.839	143.883	144.576	148.048	145.170	140.021	153.322	159.128	161.614	167.996	165.388	1797.75
1981	175.984	166.805	168.380	167.913	173.429	175.711	179.012	182.017	186.737	197.367	183.443	184.907	2141.71
1982	186.080	203.099	193.386	201.988	198.322	205.983	210.898	213.516	213.897	218.902	227.172	240.453	2513.69
1983	231.839	224.165	219.411	225.907	225.015	226.535	221.680	222.177	222.959	212.531	230.552	232.565	2695.33
1984	237.477	239.870	246.835	242.642	244.982	246.732	251.023	254.210	264.670	266.120	266.217	276.251	3037.03
1985	275.485	281.826	294.144	286.114	293.192	296.601	293.861	309.102	311.275	319.239	319.936	323.663	3604.44
1986	326.693	330.341	330.383	330.792	333.037	332.134	336.444	341.017	346.256	350.609	361.283	362.519	4081.51
1987	364.951	371.274	369.238	377.242	379.413	376.451	378.930	375.392	374.940	373.612	368.753	364.885	4475.08
1988	371.618	383.842	385.849	404.810	381.270	388.689	385.661	377.706	397.438	404.247	414.084	416.486	4711.70
1989	426.716	419.491	427.869	446.161	438.317	440.639	450.193	454.638	460.644	463.209	427.728	485.386	5340.99
1990	477.259	477.753	483.841	483.056	481.902	499.200	484.893	485.245	.	.	.	.	3873.15
Avg	277.330	280.422	282.373	286.468	285.328	288.657	288.389	291.459	265.807	268.829	268.774	276.446	

Total: 40323 Mean: 280.02 S.D.: 111.31
Min: 124.56 Max: 499.2

You can compare the original series (Table A1) and the final seasonally adjusted series (Table D11) by plotting them together as shown in Figure 45.2. These tables are requested in the OUTPUT statement and are written to the OUT= data set. Note that the default variable name used in the output data set is the input variable name followed by an underscore and the corresponding table name.

```
proc x13 data=sales date=date noprint;
   var sales;
   x11;
   output out=out a1 d11;
run;

proc sgplot data=out;
   series x=date y=sales_A1 / name = "A1" markers
                              markerattrs=(color=red symbol='asterisk')
                              lineattrs=(color=red);
   series x=date y=sales_D11 / name= "D11" markers
                               markerattrs=(symbol='circle')
                               lineattrs=(color=blue);
   yaxis label='Original and Seasonally Adjusted Time Series';
run;
```

Figure 45.2 Plot of Original and Seasonally Adjusted Data



Syntax: X13 Procedure

The X13 procedure uses the following statements:

PROC X13 *options* ;
VAR *variables* ;
BY *variables* ;
ID *variables* ;
EVENT *variables* < / *options* > ;
USERDEFINED *variables* ;
TRANSFORM *options* ;
ADJUST *option* ;
IDENTIFY *options* ;
PICKMDL *options* ;
AUTOMDL *options* ;
OUTLIER *options* ;
REGRESSION *options* ;
INPUT *variables* < / *options* > ;
ARIMA *option* ;
ESTIMATE *options* ;
X11 *options* ;
FORECAST *options* ;
CHECK *options* ;
SEATSDECOMP OUT= *SAS-data-set* < *options* > ;
OUTPUT OUT= *SAS-data-set* < **YEARSEAS** > *tablename1 tablename2 . . .* ;
TABLES *tablename1 tablename2 . . . options* ;

The statements used by PROC X13 perform basically the same function as the Census Bureau's X-13ARIMA-SEATS specs (specifications). *Specs* are used in X-13ARIMA-SEATS to control the computations and output. The PROC X13 statement performs some of the same functions as the Series spec in the Census Bureau's X-13ARIMA-SEATS software. The ADJUST statement performs some of the same functions as the Transform spec. The ARIMA, AUTOMDL, CHECK, ESTIMATE, FORECAST, IDENTIFY, OUTLIER, PICKMDL, REGRESSION, TRANSFORM, and X11 statements are designed to perform the same functions as the corresponding X-13ARIMA-SEATS specs, although full compatibility is not yet available. The Census Bureau documentation *X-13ARIMA-SEATS Reference Manual* (US Bureau of the Census 2009) provides added insight to the functionality of these statements. The SEATSDECOMP statement is an experimental statement to provide a SEATS (signal extraction in ARIMA time series) seasonal decomposition for the B1 series that uses the same ARIMA model as is used to model the series. For more information about SEATS, see Gómez and Maravall (1997a, b).

Functional Summary

Table 45.1 summarizes the statements and options that control the X13 procedure.

Table 45.1 X13 Syntax Summary

Description	Statement	Option
Data Set Options		
Specifies the auxiliary data set	PROC X13	AUXDATA=
Specifies the input data set	PROC X13	DATA=
Specifies the user-defined event definition data set	PROC X13	INEVENT=
Specifies regression and ARIMA information	PROC X13	MDLINFOIN=
Outputs regression and ARIMA information	PROC X13	MDLINFOOUT=
Writes summary statistics to an output data set	PROC X13	OUTSTAT=
Writes table values to an output data set	OUTPUT	OUT=
Appends forecasts to the OUTPUT OUT= data set	X11 or FORECAST	OUTFORECAST
Prefixes backcasts to the OUTPUT OUT= data set	FORECAST	OUTBACKCAST
Display Control Options		
Suppresses all displayed output	PROC X13	NOPRINT
Specifies the plots to be displayed	PROC X13	PLOTS=
Specifies the type of spectral plot to be displayed	PROC X13	PERIODOGRAM
Specifies the series for spectral analysis	PROC X13	SPECTRUMSERIES=
Displays automatic model information	AUTOMDL	PRINT=
Specifies the number of lags in regARIMA model residuals ACF and PACF tables and plots	CHECK	MAXLAG=
Displays regARIMA model residuals information	CHECK	PRINT=
Displays the iterations history	ESTIMATE	ITPRINT
Displays information about restarted iterations	ESTIMATE	PRINTERR
Specifies the differencing used in the ARIMA model identification ACF and PACF tables and plots	IDENTIFY	DIFF=
Specifies the seasonal differencing used in the ARIMA model identification ACF and PACF tables and plots	IDENTIFY	SDIFF=
Specifies the number of lags in ARIMA model identification ACF and PACF tables and plots	IDENTIFY	MAXLAG=
Displays regression model parameter estimates	IDENTIFY	PRINTREG

Table 45.1 *continued*

Description	Statement	Option
Requests tables that are not displayed by default	TABLES	
Specifies that the summary line not be displayed	TABLES	NOSUM
Date Information Options		
Specifies the date variable	PROC X13	DATE=
Specifies the date of the first observation	PROC X13	START=
Specifies the beginning or ending date or both of the subset	PROC X13	SPAN=
Specifies the interval of the time series	PROC X13	INTERVAL=
Specifies the interval of the time series	PROC X13	SEASONS=
Declaring the Role of Variables		
Specifies BY-group processing	BY	
Specifies identifying variables	ID	
Specifies the variables to be seasonally adjusted	VAR	
Specifies the user-defined variables that are available for regression	USERDEFINED	
Controlling the Table Computations		
Suppresses trimming of leading and trailing missing values (if they exist)	PROC X13	NOTRIMMISS
Transforms or prior-adjusts the series	TRANSFORM	FUNCTION=
Transforms or prior-adjusts the series	TRANSFORM	POWER=
Adjusts the series by using a predefined adjustment variable	ADJUST	PREDEFINED=
Specifies the likelihood function to be used for estimating AR and MA parameters	ESTIMATE	EXACT=
Specifies the maximum number of iterations for estimating AR and MA parameters	ESTIMATE	MAXITER
Specifies the convergence tolerance for nonlinear estimation	ESTIMATE	TOL=
Specifies size of forecast confidence limits	FORECAST	ALPHA=
Specifies the number of backcasts by which to extend the series for seasonal adjustment	FORECAST	NBACKCAST=
Specifies the number of forecasts by which to extend the series for seasonal adjustment	FORECAST	LEAD=
Specifies that one-step-ahead forecasts be computed	FORECAST	OUT1STEP
Specifying Outlier Detection Options		
Specifies automatic outlier detection	OUTLIER	
Specifies the span for outlier detection	OUTLIER	SPAN=

Table 45.1 *continued*

Description	Statement	Option
Specifies the outlier types to be detected	OUTLIER	TYPE=
Specifies the critical values for outlier detection	OUTLIER	CV=
Specifies the critical values for AO outlier detection	OUTLIER	AOCV=
Specifies the critical values for LS outlier detection	OUTLIER	LSCV=
Specifies the critical values for TC outlier detection	OUTLIER	TCCV=
Specifies the alpha value for outlier detection	OUTLIER	ALPHA=
Specifies the method for calculating the critical value for outlier detection based on the alpha value	OUTLIER	CVMETHOD=
Specifies the number of level-shift outliers to consider for forming a temporary level-shift	OUTLIER	LSRUN=
Specifies the rate of decay for temporary change outliers	OUTLIER	TCRATE=
Specifies the method of adding outliers at each iteration	OUTLIER	METHOD=
Specifies the difference in critical values for almost outliers	OUTLIER	ALMOST=
Specifying the Regression Model		
Specifies regression variables to be selected using an AIC-based test	REGRESSION	AICTEST=
Specifies predefined regression variables	REGRESSION	PREDEFINED=
Specifies user-defined regression variables	REGRESSION	USERVAR=
Specifies user-defined regression variables	INPUT	
Specifies user defined event regression variables	EVENT	
Specifies the method used to calculate the means for the Easter regression variable	REGRESSION	EASTERMEANS=
Specifies which types of regression effects are not to be removed before seasonal adjustment	REGRESSION	NOAPPLY=
Specifying the ARIMA Model		
Uses the X-13ARIMA-SEATS TRAMO-based method to choose a model	AUTOMDL	
Chooses a regARIMA model from a set that you specify	PICKMDL	
Specifies the ARIMA part of the model	ARIMA	MODEL=

Table 45.1 *continued*

Description	Statement	Option
Specifying Automatic Model Detection Options		
Specifies the maximum orders of ARMA polynomials	AUTOMDL	MAXORDER=
Specifies the maximum orders of differencing	AUTOMDL	MAXDIFF=
Specifies the estimation method for identifying difference orders	AUTOMDL	DIFFID=
Specifies the maximum number of iterations for exact likelihood for DIFFID=EXACTFIRST	AUTOMDL	DIFFIDITER=
Specifies the fixed orders of differencing	AUTOMDL	DIFFORDER=
Suppresses fitting of a constant parameter	AUTOMDL	NOINT
Specifies the preference for balanced models	AUTOMDL	BALANCED
Specifies Hannan-Rissanen initial estimation	AUTOMDL	HRINITIAL
Specifies default model acceptance based on Ljung-Box Q	AUTOMDL	ACCEPTDEFAULT
Specifies the acceptance value for Ljung-Box Q	AUTOMDL	LJUNGBOXLIMIT=
Specifies the percentage by which to reduce the outlier critical value	AUTOMDL	REDUCECV=
Specifies the critical value for ARMA coefficients	AUTOMDL	ARMACV=
Model Diagnostics		
Examines the regARIMA model residuals	CHECK	
Specifying Seasonal Adjustment Options		
Specifies seasonal adjustment	X11	
Specifies the mode of seasonal adjustment decomposition	X11	MODE=
Specifies the seasonal filter	X11	SEASONALMA=
Specifies the sigma limits	X11	SIGMALIM=
Specifies the Henderson trend filter	X11	TRENDMA=
Specifies the D11 calculation method	X11	TYPE=
Specifies the adjustment factors to remove from final seasonally adjusted series	X11	FINAL=
Specifies a method for reconciling the seasonally adjusted series to the original series	X11	FORCE=
Specifies that SEATS seasonal decomposition be output to a data set	SEATSDECOMP	OUT=

PROC X13 Statement

PROC X13 *options* **;**

The PROC X13 statement provides information about the time series to be processed by PROC X13. Either the DATE= or the START= option must be specified. If both options are specified, then a syntax error results and the X13 procedure is not executed.

The original series is displayed in Table A1. If there are missing values in the original series and a regARIMA model is specified or automatically selected, then Table MV1 is displayed. Table MV1 contains the original series with missing values replaced by the predicted values from the fitted model. If outliers are identified and Table A19 is added in the TABLES statement, then the outlier adjusted series is displayed in Table A19. Table B1 is displayed when the original data are altered (for example, through an ARIMA model estimation, prior adjustment factor, or regression) or the series is extended with forecasts.

Although the X-13ARIMA-SEATS method handles missing values, there are some restrictions. In order for PROC X13 to process the series, no month or quarter can contain missing values for all years. For instance, if the third quarter contained only missing values for all years, then processing is skipped for that series. In addition, if more than half the values for a month or a quarter are missing, then a warning message is displayed in the log file, and other errors might occur later in processing. If a series contains many missing values, other methods of missing value replacement should be considered prior to seasonally adjusting the series.

You can specify the following *options* in the PROC X13 statement:

AUXDATA=*SAS-data-set*
specifies an auxiliary input data set that contains user-defined variables, which are specified in the INPUT statement, the USERVAR= option in the REGRESSION statement, or the USERDEFINED statement. The AUXDATA= data set can also contain the date variable, which is specified in the DATE= option in the PROC X13 statement. If the date variable is present, then the date variable is used to align the observations in the auxiliary data set to the observations in the series that is being processed. The date values must be sorted in ascending order with no gaps or duplications, and the interval must match the interval of the series. If the date variable is not present or valid, then observations in the auxiliary data set are matched by observation number to the series that is being processed. The auxiliary data set does not support BY-group processing. The variables in the auxiliary data set are applied to all BY groups, where the dates of the BY group correspond to the dates of the auxiliary data set. Example 45.11 shows the use of the AUXDATA= data set.

DATA=*SAS-data-set*
specifies the input SAS data set to use. If this option is omitted, the most recently created SAS data set is used.

DATE=*variable*

DATEVAR=*variable*
specifies a variable that gives the date for each observation. Unless specified in the SPAN= option, the starting and ending dates are obtained from the first and last values of the BY group for the DATE= variable, which must contain SAS date or datetime values. The procedure checks values of the DATE= variable to ensure that the input observations are sequenced correctly in ascending order. If the INTERVAL= option or the SEASONS= option is specified, the values of the date variable must be consistent with the specified seasonality or interval. If neither the INTERVAL= option nor the

SEASONS= option is specified, then the procedure tries to determine the type of data from the values of the date variable. This variable is automatically added to the OUT= data set if a data set is requested in an OUTPUT statement, and the date values for the variable are extrapolated if necessary. If the DATE= option is not specified, the START= option must be specified.

INEVENT=*SAS-data-set*

specifies the input data set that defines any user-defined event variables. This option can be omitted if events are not specified or if only SAS predefined events are specified in an EVENT statement. For more information about the format of this data set, see the section "INEVENT= Data Set" on page 3224.

INTERVAL=*interval*

specifies the frequency of the input time series. If the input data consist of quarterly observations, then INTERVAL=QTR should be used. If the input data consist of monthly observations, then INTERVAL=MONTH should be used. If the INTERVAL= option is not specified and SEASONS=4, then INTERVAL=QTR is assumed; likewise, SEASONS=12 implies INTERVAL=MONTH. If both the INTERVAL= option and the SEASONS= option are specified, the values should not be conflicting. If neither the INTERVAL= option nor the SEASONS= option is specified and the START= option is specified, then the data are assumed to be monthly. If a date variable is specified using the DATE= option, it is not necessary to specify the INTERVAL= option or the SEASONS= option; however, if specified, the values of the INTERVAL= option or the SEASONS= option should not be in conflict with the values of the date variable. See Chapter 5, "Date Intervals, Formats, and Functions," for more details about intervals.

MDLINFOIN=*SAS-data-set*

specifies an optional input data set that contains model information that can replace the information contained in the TRANSFORM, REGRESSION, ARIMA, and AUTOMDL statements. The MDLINFOIN= data set can contain BY-group and series names. It is useful for providing specific information about each series to be seasonally adjusted. See the section "MDLINFOIN= and MDLINFOOUT= Data Sets" on page 3222 for details.

MDLINFOOUT=*SAS-data-set*

specifies the optional output data set that contains the transformation, regression, and ARIMA information related to each seasonally adjusted series. The data set is sorted by the BY-group variables, if any, and by series names. The MDLINFOOUT= data set can be used as input for the MDLINFOIN= option. See the section "MDLINFOIN= and MDLINFOOUT= Data Sets" on page 3222 for details.

NOPRINT

suppresses any printed output.

NOTRIMMISS

suppresses the default, by which leading and trailing missing values are trimmed from each variable listed (or implied) in the VAR statement. If you specify the NOTRIMMISS option, PROC X13 treats leading and trailing missing values in the same manner as it treats embedded missing values. For information about the treatment of embedded missing values, see the section "Missing Values" on page 3204. Missing values are not supported in the regression variables that you specify in the REGRESSION, INPUT, or USERDEFINED statement; therefore, leading and trailing missing values are always trimmed from user-defined regressors even if you specify NOTRIMMISS.

OUTSTAT=*SAS-data-set*
specifies an optional output data set which contains the summary statistics that related to each seasonally adjusted series. The data set is sorted by the BY-group variables, if any, and by series names. See the section "OUTSTAT= Data Set" on page 3225 for details.

PERIODOGRAM
specifies that the PERIODOGRAM rather than the spectrum of the series be plotted in the G tables and plots. If PERIODOGRAM is not specified, then the spectrum is plotted in the G tables.

PLOTS< (*global-plot-options***) > < = ***plot-request* < *(options)* > **>**

PLOTS< (*global-plot-options***) > < = (***plot-request* < *(options)* > **< ...** *plot-request* < *(options)* > **>) >**
controls the plots that are produced through ODS Graphics. When you specify only one plot request, you can omit the parentheses around the plot request.

Following are some examples of the PLOTS= option:

```
plots=none
plots=all
plots=residual(none)
plots(only)=(series(acf pacf) residual(hist))
```

ODS Graphics must be enabled before you request plots. For example:

```
ods graphics on;

proc x13 data=sales date=date;
   var sales;
   identify diff=(0,1) sdiff=(0,1);
run;
```

Since no specific plot is requested in this program, the default plots associated with the PROC X13 and IDENTIFY statements are produced.

For general information about ODS Graphics, see Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*). If you have enabled ODS Graphics but do not specify any specific plot request, then the default plots that are associated with each of the PROC X13 statements used in the program are produced. Line printer plots are suppressed when ODS Graphics is enabled.

If NONE is specified in an option, then no plots are produced for that option. If ALL is specified without NONE in an option, then all plots are produced for that option.

Global Plot Options:
The *global-plot-options* apply to all relevant plots that are generated by the X13 procedure. The following *global-plot-option* is supported:

ONLY
suppresses the default plots. Only the plots specifically requested are produced.

Specific Plot Options:
The following list describes the specific plots and their *options*:

ALL
produces all plots that are appropriate for the particular analysis.

NONE
suppresses all plots.

ADJUSTED(< *sa-plot-options* **>)**
SA(< *sa-plot-options* **>)**
produces plots of the seasonally adjusted series that results from the decomposition specified in the X11 statement. The SPECTRUM plot is produced by default.

The following *sa-plot-options* are available:

ALL
produces all seasonally adjusted plots.

NONE
suppresses all seasonally adjusted plots.

SPECTRUM
produces the spectral plot of Table G1. Table G1 is calculated based on the modified seasonally adjusted series (Table E2). The data is first-differenced and transformed as specified in the TRANSFORM statement. By default, the type of spectral estimate used to calculate the spectral plot is the spectrum. If the PERIODOGRAM option is specified in the PROC X13 statement, then the periodogram of the series is used to calculate the spectral plot.

FORECAST(< *forecast-plot-options* **>)**
produces the regARIMA model forecast plots if the FORECAST statement is specified. The FORECAST plot is produced by default. The following *forecast-plot-options* are available:

ALL
produces all the forecast plots that are appropriate for the particular analysis.

FORECAST
plots the actual time series and its one-step-ahead forecast over the historical period, and plots the forecast and its confidence bands over the forecast horizon. The OUT1STEP option must be specified in the FORECAST statement in order for the X13 procedure to calculate the one-step-ahead forecasts.

FORECASTONLY
plots the forecast and its confidence bands over the forecast horizon only.

MODELFORECASTS
plots the one-step-ahead model forecast and its confidence bands in the historical period; plots the forecast and its confidence bands over the forecast horizon. The OUT1STEP option must be specified in the FORECAST statement in order for the X13 procedure to calculate the one-step-ahead forecasts.

MODELS
plots the one-step-ahead model forecast and its confidence bands in the historical period. The OUT1STEP option must be specified in the FORECAST statement in order for the X13 procedure to calculate the one-step-ahead forecasts.

NONE
suppresses all the forecast plots.

TRANSFORECAST
plots the transformed time series and its one-step-ahead forecast over the historical period; plots the forecast and its confidence bands over the forecast horizon. The OUT1STEP option must be specified in the FORECAST statement in order for the X13 procedure to calculate the one-step-ahead forecasts. The TRANSFORECAST plot is available only if the data have been transformed using the TRANSFORM statement.

TRANSFORECASTONLY
plots the forecast of the transformed series and its confidence bands over the forecast horizon only. The TRANSFORECASTONLY plot is available only if the data have been transformed using the TRANSFORM statement.

TRANSMODELFORECASTS
plots the one-step-ahead model forecast of the transformed series and its confidence bands in the historical period; plots the forecast and its confidence bands over the forecast horizon. The OUT1STEP option must be specified in the FORECAST statement in order for the X13 procedure to calculate the one-step-ahead forecasts. The TRANSMODELFORECASTS plot is available only if the data have been transformed using the TRANSFORM statement.

TRANSMODELS
plots the one-step-ahead model forecast of the transformed series and its confidence bands in the historical period. The OUT1STEP option must be specified in the FORECAST statement in order for the X13 procedure to calculate the one-step-ahead forecasts. The TRANSMODELS plot is available only if the data have been transformed using the TRANSFORM statement.

IRREGULAR(< *ic-plot-options* **>)**

IC(< *ic-plot-options* **>)**
produces plots of the irregular series that results from the decomposition specified in the X11 statement. The SPECTRUM plot is produced by default.

The following *ic-plot-options* are available:

ALL
produces all irregular plots.

NONE
suppresses all irregular plots.

SPECTRUM
produces the spectral plot of Table G2. Table G2 is calculated based on the modified irregular series (Table E3). The data is first-differenced and transformed as specified in the TRANSFORM statement. By default, the type of spectral estimate used to calculate the

spectral plot is the spectrum. If the PERIODOGRAM option is specified in the PROC X13 statement, then the periodogram of the series is used to calculate the spectral plot.

RESIDUAL(< *residual-plot-options* **>)**
produces the regARIMA model residual series plots if the CHECK statement is specified. The ACF, PACF, HIST, SQACF, and SPECTRUM plots are produced by default. The following *residual-plot-options* are available:

ACF
produces the plot of residual autocorrelations.

ALL
produces all the residual diagnostics plots that are appropriate for the particular analysis.

HIST
produces the histogram of the residuals and also the residual outliers and residual statistics tables that describe the residual histogram.

NONE
suppresses all the residual diagnostics plots.

PACF
produces the plot of residual partial-autocorrelations if PRINT=PACF is specified in the CHECK statement.

SPECTRUM
produces the spectral plot of Table GRs. Table GRs is calculated based on the regARIMA model residual series. By default, the type of spectral estimate used to calculate the spectral plot is the spectrum. If the PERIODOGRAM option is specified in the PROC X13 statement, then the periodogram of the series is used to calculate the spectral plot.

SQACF
produces the plot of squared residual autocorrelations.

SERIES(< *series-plot-options* **>)**
produces plots that are associated with the identification stage of the modeling. The ACF, PACF, and SPECTRUM plots are produced by default. The following *series-plot-options* are available:

ACF
produces the plot of autocorrelations.

ALL
produces all the plots that are associated with the identification stage.

NONE
suppresses all plots that are associated with the identification stage.

PACF
produces the plot of partial-autocorrelations.

SPECTRUM

produces the spectral plot of Table G0. Table G0 is calculated based on either Table A1, A19, B1 or E1, as specified by the SPECTRUMSERIES= option. The original data is first-differenced and transformed as specified in the TRANSFORM statement. By default, the type of spectral estimate that is used to calculate the spectral plot is the spectrum. If the PERIODOGRAM option is specified in the PROC X13 statement, then the periodogram of the series is used to calculate the spectral plot.

SEASONS=*number*

specifies the number of observations in a seasonal cycle. If the SEASONS= option is not specified and INTERVAL=QTR, then SEASONS=4 is assumed. If the SEASONS= option is not specified and INTERVAL=MONTH, then SEASONS=12 is assumed. If the SEASONS= option is specified, its value should not conflict with the values of the INTERVAL= option or the values of the date variable. See the preceding descriptions for the START=, DATE=, and INTERVAL= options for more details.

SPAN=(*mmmyy* **,***mmmyy* **)**

SPAN=(*'yyQq'* **,***'yyQq'* **)**

specifies the dates of the first and last observations to define a subset for processing. A single date in parentheses is interpreted to be the starting date of the subset. To specify only the ending date, use SPAN=(,*mmmyy*). If the starting or ending date is omitted, then the first or last date, respectively, of the input data set or BY group is assumed. Because the dates are input as strings and the quarterly dates begin with a numeric character, the specification for a quarterly date must be enclosed in quotation marks. A four-digit year can be specified; if a two-digit year is specified, the value specified in the YEARCUTOFF= SAS system option applies.

SPECTRUMSERIES=*table-name*

specifies the table name of the series that is used in the spectrum of the original series (Table G0). The table names that can be specified are A1, A19, B1, or E1. The default is B1.

START=*mmmyy*

START=*'yyQq'*

STARTDATE=*mmmyy*

STARTDATE=*'yyQq'*

specifies the date of the first observation. Unless the SPAN= option is used, the starting and ending dates are the dates of the first and last observations, respectively. Either this option or the DATE= option is required. When using this option, use either the INTERVAL= option or the SEASONS= option to specify monthly or quarterly data. If neither the INTERVAL= option nor the SEASONS= option is present, monthly data are assumed. Because the dates are input as strings and the quarterly dates begin with a numeric character, the specification for a quarterly date must be enclosed in quotation marks. A four-digit year can be specified; if a two-digit year is specified, the value specified in the YEARCUTOFF= SAS system option applies. When using the START= option with BY processing, the start date is applied to the first observation in each BY group.

ADJUST Statement

ADJUST *option* **;**

The ADJUST statement adjusts the series for leap year and length-of-period factors prior to estimating a regARIMA model. The "Prior Adjustment Factors" table is associated with the ADJUST statement.

The following *option* can appear in the ADJUST statement:

PREDEFINED=LOM | LOQ | LPYEAR

specifies length-of-month adjustment, length-of-quarter adjustment, or leap year adjustment. PREDEFINED=LOM and PREDEFINED=LOQ are equivalent because the actual adjustment is determined by the interval of the time series. Also, because leap year adjustment is a limited form of length-of-period adjustment, only one type of predefined adjustment can be specified. The PREDEFINED= option should not be used in conjunction with PREDEFINED=TD or PREDEFINED=TD1COEF in the REGRESSION statement or MODE=ADD or MODE=PSEUDOADD in the X11 statement. PREDEFINED=LPYEAR cannot be specified unless the series is log transformed.

If the series is to be transformed by using a Box-Cox or logistic transformation, the series is first adjusted according to the ADJUST statement, and then it is transformed.

In the case of a length-of-month adjustment for the series with observations Y_t, each observation is first divided by the number of days in that month, m_t, and then multiplied by the average length of month (30.4375), resulting in $(30.4375 \times Y_t)/m_t$. Length-of-quarter adjustments are performed in a similar manner, resulting in $(91.3125 \times Y_t)/q_t$, where q_t is the length in days of quarter t.

Forecasts of the transformed and adjusted data are transformed and adjusted back to the original scale for output.

ARIMA Statement

ARIMA *option* **;**

The ARIMA statement specifies the ARIMA part of the regARIMA model. This statement defines a pure ARIMA model if no REGRESSION statements, INPUT statements, or EVENT statements are specified. The ARIMA part of the model can include multiplicative seasonal factors.

The following *option* can appear in the ARIMA statement:

MODEL=((*p d q*) (*P D Q*)*s*)

specifies the ARIMA model. The format follows standard Box-Jenkins notation (Box, Jenkins, and Reinsel 1994). The nonseasonal AR and MA orders are given by *p* and *q*, respectively, while the seasonal AR and MA orders are given by *P* and *Q*. The number of differences and seasonal differences are given by *d* and *D*, respectively. The notation (*p d q*) and (*P D Q*) can also be specified as (*p*, *d*, *q*) and (*P*, *D*, *Q*). The maximum lag of any AR or MA parameter is 36. The maximum value of a difference order, *d* or *D*, is 144. All values for *p*, *d*, *q*, *P*, *D*, and *Q* should be nonnegative integers. The seasonality parameter, *s*, should be a positive integer. If *s* is omitted, it is set equal to the value that is specified in the SEASONS= option in the PROC X13 statement.

For example, the following statements specify an ARIMA (2,1,1)(1,1,0)12 model:

```
proc x13 data=ICMETI seasons=12 start=jan1968;
   arima model=((2,1,1)(1,1,0));
```

AUTOMDL Statement

AUTOMDL *options* **;**

The AUTOMDL statement invokes the automatic model selection procedure of the X-13ARIMA-SEATS method. This method is based largely on the TRAMO (time series regression with ARIMA noise, missing values, and outliers) method by Gómez and Maravall (1997a, b). If the AUTOMDL statement is used without the OUTLIER statement, then only missing values regressors are included in the regARIMA model. If both the AUTOMDL and the OUTLIER statements are used, then both missing values regressors and regressors for automatically identified outliers are included in the regARIMA model. For more information about missing value regressors, see the section "Missing Values" on page 3204.

If both the AUTOMDL statement and the ARIMA statement are present, the ARIMA statement is ignored. The ARIMA statement specifies the model, but the AUTOMDL statement allows the X13 procedure to select the model. If the AUTOMDL statement is specified and a data set is specified in the MDLINFOIN= option in the PROC X13 statement, then the AUTOMDL statement is ignored if the specified data set contains a model specification for the series. If no model for the series is specified in the MDLINFOIN= data set, the AUTOMDL or ARIMA statement is used to determine the model. Thus, it is possible to give a specific model for some series and automatically identify the model for other series by using both the MDLINFOIN= option and the AUTOMDL statement.

When the AUTOMDL statement is specified, the X13 procedure compares a model selected using a TRAMO method to a default model. The TRAMO method is implemented first, and involves two parts: identifying the orders of differencing and identifying the ARIMA model. The table "ARIMA Estimates for Unit Root Identification" provides details about the identification of the orders of differencing, and the table "Results of Unit Root Test for Identifying Orders of Differencing" shows the orders of differencing selected by TRAMO. The table "Models Estimated by Automatic ARIMA Model Selection Procedure" provides details regarding the TRAMO automatic model selection, and the table "Best Five ARIMA Models Chosen by Automatic Modeling" ranks the best five models estimated using the TRAMO method. The "Comparison of Automatically Selected Model and Default Model" table compares the model selected by the TRAMO method to a default model. At this point in the processing, if the default model is selected over the TRAMO model, then PROC X13 displays a note. No note is displayed if the TRAMO model is selected. The Ljung-Box Q statistic is then checked for acceptance, and the results are displayed in the "Check of the Residual Ljung-Box Q Statistic" table. The initial model selected at this point is displayed in the "Initial Automatic Model Selection" table. PROC X13 then performs final checks for unit roots, overdifferencing, and insignificant ARMA coefficients. The results of the final checks are displayed in the "Final Checks for Identified Model" table, which also indicates changes to the model order if the orders are changed. The last table, "Final Automatic Model Selection," shows the results of the automatic model selection; if the orders have been altered during the final checks, the Orders Altered column displays a value of Yes. An example of the automatic modeling selection procedure is shown in Example 45.4.

The following *options* can appear in the AUTOMDL statement:

ACCEPTDEFAULT
specifies that the default model be chosen if its Ljung-Box Q is acceptable.

ARMACV=*value*
specifies the threshold value for the t statistics that are associated with the highest-order ARMA coefficients. As a check of model parsimony, the parameter estimates and t statistics of the highest-order ARMA coefficients are examined to determine whether the coefficient is insignificant. An

ARMA coefficient is considered to be insignificant if the *t* value that is displayed in the table "Exact ARMA Maximum Likelihood Estimation" is below the value specified in the ARMACV= option and the absolute value of the parameter estimate is reliably close to zero. The absolute value is considered to be *reliably close to zero* if it is below 0.15 for 150 or fewer observations or is below 0.1 for more than 150 observations. If the highest-order ARMA coefficient is found to be insignificant, then the order of the ARMA model is reduced. For example, if AUTOMDL identifies a (3 1 1)(0 0 1) model and the parameter estimate of the seasonal MA lag of order 1 is –0.09 and its *t* value is –0.55, then the ARIMA model is reduced to at least (3 1 1)(0 0 0). After the model is reestimated, the check for insignificant coefficients is performed again. If ARMACV=0.54 is specified in the preceding example, then the coefficient is not found to be insignificant and the model is not reduced.

If a constant is allowed in the model and if the *t* value associated with the constant parameter estimate is below the ARMACV= critical value, then the constant is considered to be insignificant and is removed from the model. Note that if a constant is added to or removed from the model and then the ARIMA model changes, then the *t* statistic for the constant parameter estimate also changes. Thus, changing the ARMACV= value does not necessarily add or remove a constant term from the model.

The value specified in the ARMACV= option should be greater than zero. The default value is 1.0.

BALANCED

specifies that the automatic modeling procedure prefer balanced models over unbalanced models. A balanced model is one in which the sum of the AR, seasonal AR, differencing, and seasonal differencing orders equals the sum of the MA and seasonal MA orders. Specifying BALANCED gives the same preference as the TRAMO program. If BALANCED is not specified, all models are given equal consideration.

DIFFID=CONDITIONAL | EXACT | EXACTFIRST (Experimental)

specifies the estimation to be used in automatic difference identification when Hannen-Rissanen fails. You can specify the following values:

CONDITIONAL uses conditional likelihood estimation.

EXACT uses exact likelihood estimation.

EXACTFIRST attempts to estimate the parameters by using exact likelihood for the first *diffiditer* iterations, where *diffiditer* is specified in the DIFFIDITER= option. If the estimation does not converge within *diffiditer* iterations, then conditional likelihood is used to estimate the parameters.

The effects of this option are displayed in the Estimation Method column in the "ARIMA Estimates for Unit Root Identification" table. By default, DIFFID=EXACTFIRST.

DIFFIDITER=*diffiditer* (Experimental)

specifies the maximum number of exact likelihood estimation iterations when DIFFID=EXACTFIRST is specified. If the number of iterations exceeds *diffiditer*, then conditional likelihood is used to estimate the remaining iterations. The default value is 500; this default differs from the default value of 200 in the US Census Bureau's implementation of X-13ARIMA-SEATS.

DIFFORDER=(*nonseasonal-order***, *seasonal-order*)**

specifies the fixed orders of differencing to be used in the automatic ARIMA model identification procedure. When the DIFFORDER= option is used, only the AR and MA orders are automatically identified. Acceptable values for the regular (nonseasonal) differencing orders are 0, 1, and 2;

acceptable values for the seasonal differencing orders are 0 and 1. If the MAXDIFF= option is also specified, then the DIFFORDER= option is ignored. There are no default values for DIFFORDER. If neither the DIFFORDER= option nor the MAXDIFF= option is specified, then the default is MAXDIFF=(2,1).

HRINITIAL
specifies that Hannan-Rissanen estimation be done before exact maximum likelihood estimation to provide initial values. If the HRINITIAL option is specified, then models for which the Hannan-Rissanen estimation has an unacceptable coefficient are rejected.

LJUNGBOXLIMIT=*value*
specifies acceptance criteria for the confidence coefficient of the Ljung-Box Q statistic. If the Ljung-Box Q for a final model is greater than this value, the model is rejected, the outlier critical value is reduced, and outlier identification is redone with the reduced value. See the REDUCECV option for more information. The value specified in the LJUNGBOXLIMIT= option must be greater than 0 and less than 1. The default value is 0.95.

MAXDIFF=(*nonseasonal-order***, *seasonal-order*)**
specifies the maximum orders of regular and seasonal differencing for the automatic identification of differencing orders. When MAXDIFF is specified, the differencing orders are identified first, and then the AR and MA orders are identified. Acceptable values for the regular differencing orders are 1 and 2. The only acceptable value for the seasonal differencing order is 1. If both the MAXDIFF= option and the DIFFORDER option= are specified, then the DIFFORDER= option is ignored. If neither the DIFFORDER= nor the MAXDIFF= option is specified, the default is MAXDIFF=(2,1).

MAXORDER=(*nonseasonal-order***, *seasonal-order*)**
specifies the maximum orders of nonseasonal and seasonal ARMA polynomials for the automatic ARIMA model identification procedure. The maximum order for the nonseasonal ARMA parameters is 4, and the maximum order for the seasonal ARMA is 2.

NOINT
suppresses the fitting of a constant or intercept parameter in the model.

PRINT=(*option-list***)**
specifies the tables to be displayed in the output. You can specify one or more of the following *options* (parentheses are optional; use a space between *options*):

NONE suppresses all automatic modeling output.

ALL includes all automatic modeling tables in the output if NONE is not specified in the *option-list*.

ONLY specifies that only the listed tables be output.

AUTOCHOICE displays the tables titled "Comparison of Automatically Selected Model and Default Model" and "Final Automatic Model Selection." The "Comparison of Automatically Selected Model and Default Model" table compares a default model to the model chosen by the TRAMO-based automatic modeling method. The "Final Automatic Model Selection" table indicates which model has been chosen automatically. These tables are output by default unless NONE or ONLY is specified in the *option-list*.

AUTOCHOICEMDL displays the table "Models Estimated by Automatic ARIMA Model Selection Procedure." This table summarizes the various models that were considered by the TRAMO automatic model selection method and their measures of fit.

AUTOLJUNGBOX displays the table "Check of the Residual Ljung-Box Q Statistic." This table is displayed only if the model is not accepted because the Ljung-Box Q statistic is greater than the acceptance limit. The details of the test and the changes made either to the model or to the model selection method are displayed.

BEST5MODEL displays the table "Best Five ARIMA Models Chosen by Automatic Modeling." This table ranks the five best models that were considered by the TRAMO automatic modeling method.

FINALCHECKS displays the table "Final Checks for Identified Model." This table displays the results of the final checks for model adequacy. The final checks can result in the orders of the initially identified model being altered. Any order changes or changes in the constant term are included in this table. This table is output by default unless NONE or ONLY is specified in the *option-list*.

INITCHOICEMDL displays the table "Initial Automatic Model Selection." The "Comparison of Automatically Selected Model and Default Model" table compares a default model to the model chosen by the TRAMO-based automatic modeling method. The chosen model can then be altered if the model fails the Ljung-Box Q statistic test. The "Initial Automatic Model Selection" table indicates which model has been chosen automatically after the Ljung-Box Q statistic test. This table is output by default unless NONE or ONLY is specified in the *option-list*.

UNITROOTTEST displays the table titled "Results of Unit Root Test for Identifying Orders of Differencing." This table displays the orders that were automatically selected by the AUTOMDL statement. Unless the nonseasonal and seasonal differences are specified using the DIFFORDER= option, the AUTOMDL statement automatically identifies the orders of differencing. This table is output by default unless NONE or ONLY is specified in the *option-list*.

UNITROOTTESTMDL displays the table titled "ARIMA Estimates for Unit Root Identification." This table summarizes the various models that were considered by the TRAMO automatic selection method while it identified the orders of differencing and the statistics associated with those models. The unit root identification method first attempts to obtain the coefficients by using the Hannan-Rissanen method. If Hannan-Rissanen estimation cannot be performed, the algorithm attempts to obtain the coefficients by using conditional likelihood estimation.

By default, PRINT=(UNITROOTTEST AUTOCHOICE INITCHOICEMDL FINALCHECKS).

REDUCECV=*value*

specifies the percentage by which the outlier critical value be reduced when a final model is found to have an unacceptable confidence coefficient for the Ljung-Box Q statistic. This value should be between 0 and 1. The default value is 0.14286.

BY Statement

BY *variables* **;**

A BY statement can be used with PROC X13 to obtain separate analyses on observations in groups defined by the BY variables. When a BY statement appears, the procedure expects the input DATA= data set to be sorted in order of the BY variables.

CHECK Statement

CHECK *options* **;**

The CHECK statement produces statistics for diagnostic checking of residuals from the estimated regARIMA model.

The following tables that are associated with diagnostic checking are displayed in the output: "Autocorrelation of regARIMA Model Residuals," "Partial Autocorrelation of regARIMA Model Residuals," "Autocorrelation of Squared regARIMA Model Residuals," "Outliers of the Unstandardized Residuals," "Summary Statistics for the Unstandardized Residuals," "Normality Statistics for regARIMA Model Residuals," and "Table G Rs: 10*LOG(SPECTRUM) of the regARIMA Model Residuals." If ODS graphics is enabled, the following plots that are associated with diagnostic checking output are produced: the autocorrelation function (ErrorACF) plot of the residuals, the partial autocorrelation function (ErrorPACF) plot of the residuals, the autocorrelation function (SqErrorACF) plot of the squared residuals, a histogram (ResidualHistogram) of the residuals, and a spectral plot (SpectralPlot) of the residuals. See the PLOTS=RESIDUAL option in the PROC X13 statement for further information about controlling the display of plots.

The residual histogram displayed by the X13 procedure shows the distribution of the unstandardized, uncentered regARIMA model residuals; the residual histogram displayed by the US Census Bureau's X-13ARIMA-SEATS seasonal adjustment program displays standardized and mean-centered residuals.

The following *options* can appear in the CHECK statement:

MAXLAG=*value*

specifies the number of lags for the residual sample autocorrelation function (ACF) and partial autocorrelation function (PACF). The default is 36 for monthly series and 12 for quarterly series. The minimum value for MAXLAG= is 1.

For the table "Autocorrelation of Squared regARIMA Model Residuals" and the corresponding SqErrorACF plot, the maximum number of lags calculated is 12 for monthly series and 4 for quarterly series. The MAXLAG= option can only reduce the number of lags for this table and plot.

PRINT=(*option-list***)**

specifies the diagnostic checking tables to be displayed. You can specify one or more of the following *options* (parentheses are optional; use a space between *options*):

NONE suppresses diagnostic checking output. If PRINT=NONE is specified and no other PRINT= option is specified, then none of the tables that are associated with diagnostic checking are displayed. However, PRINT=NONE has no effect if other PRINT= options are specified in the CHECK statement.

ALL specifies that all tables related to diagnostic checking be displayed.

ACF displays the table titled "Autocorrelation of regARIMA Model Residuals."

ACFSQUARED displays the table titled "Autocorrelation of Squared regARIMA Model Residuals."

NORM displays the table titled "Normality Statistics for regARIMA Model Residuals." Measures of normality included in this table are skewness, Geary's *a* statistic, and kurtosis.

PACF displays the table titled "Partial Autocorrelation of regARIMA Model Residuals."

RESIDUALOUTLIER displays the table titled "Outliers of the Unstandardized Residuals" if the residuals contain outliers. You can specify this option either as PRINT=RESIDUALOUTLIER or PRINT=RESOUTLIER.

RESIDUALSTATISTICS displays the table titled "Summary Statistics for the Unstandardized Residuals." You can specify this option either as PRINT=RESIDUALSTATISTICS or PRINT=RESSTAT.

SPECRESIDUAL displays the table titled "Table G Rs: 10*LOG(SPECTRUM) of the regARIMA Model Residuals."

By default, PRINT=(ACF ACFSQUARED NORM RESIDUALOUTLIER RESIDUALSTATISTICS SPECRESIDUAL).

ESTIMATE Statement

ESTIMATE *options* **;**

The ESTIMATE statement estimates the regARIMA model. The regARIMA model is specified by the REGRESSION, INPUT, EVENT, and ARIMA statements or by the MDLINFOIN= data set in the PROC X13 statement. Estimation output includes point estimates and standard errors for all estimated AR, MA, and regression parameters; the maximum likelihood estimate of the variance σ^2; t statistics for individual regression parameters; χ^2 statistics for assessing the joint significance of the parameters associated with certain regression effects (if included in the model); and likelihood-based model selection statistics (if the exact likelihood function is used). The regression effects for which χ^2 statistics are produced are fixed seasonal effects.

Tables displayed in the output associated with estimation are "Exact ARMA Likelihood Estimation Iteration Tolerances," "Average Absolute Percentage Error in within-Sample Forecasts," "ARMA Iteration History," "AR/MA Roots," "Exact ARMA Likelihood Estimation Iteration Summary," "Regression Model Parameter Estimates," "Chi-Squared Tests for Groups of Regressors," "Exact ARMA Maximum Likelihood Estimation," and "Estimation Summary."

The following *options* can appear in the ESTIMATE statement:

EXACT=ARMA | MA | NONE (Experimental)
specifies the likelihood function for estimation, likelihood evaluation, and forecasting. You can specify the following values:

ARMA uses the likelihood function that is exact for both AR and MA parameters.

MA uses the likelihood function that is exact for MA parameters, but conditional for AR parameters.

NONE uses the likelihood function that is conditional for both AR and MA parameters.

The ARMA estimation iterations are displayed in the "Iteration History" table, which is available when the ITPRINT option is specified. By default, EXACT=ARMA.

ITPRINT

displays the "Iteration History" table. This table includes detailed output for estimation iterations, including log-likelihood values, parameters, counts of function evaluations, and iterations. It is useful to examine the "Iteration History" table when errors occur within estimation iterations. By default, only successful iterations are displayed, unless the PRINTERR option is specified. An unsuccessful iteration is an iteration that is restarted due to a problem such as a root inside the unit circle. Successful iterations have a status of 0. If restarted iterations are displayed, a note at the end of the table gives definitions for status codes that indicate a restarted iteration. For restarted iterations, the number of function evaluations and the number of iterations is –1, which is displayed as missing. If regression parameters are included in the model, then both IGLS and ARMA iterations are included in the table. The number of function evaluations is a cumulative total.

MAXITER=*value*

specifies the maximum number of iterations used in estimating the AR and MA parameters. For models that include regression variables, this limit applies to the total number of ARMA iterations over all iterations of the iterative generalized least squares (IGLS) algorithm. For models without regression variables, *value* is the maximum number of iterations allowed for the set of ARMA iterations. By default, MAXITER=1500.

PRINTERR

causes restarted iterations to be included in the "Iteration History" table if ITPRINT is specified; creates the "Restarted Iterations" table if ITPRINT is not specified. Whether or not PRINTERR is specified, a WARNING message is printed to the log file if any iteration is restarted during estimation.

TOL=*value*

specifies the convergence tolerance for the nonlinear estimation. Absolute changes in the log-likelihood are compared to the TOL= value to check convergence of the estimation iterations. For models with regression variables, the TOL= value is used to check convergence of the IGLS iterations (where the regression parameters are reestimated for each new set of AR and MA parameters). For models without regression variables, there are no IGLS iterations, and the TOL= value is then used to check convergence of the nonlinear iterations that are used to estimate the AR and MA parameters. The default value is TOL=0.00001. The minimum tolerance value is a positive value based on the machine precision and the length of the series. If a tolerance less than the minimum supported value is specified, an error message is displayed and the series is not processed.

EVENT Statement

EVENT *variables* < / *options* > **;**

The EVENT statement specifies events to be included in the regression portion of the regARIMA model. Multiple EVENT statements can be specified. Dummy variable values for EVENT variables are generated by the X13 procedure, however, the EVENT variables are input as user-defined regression effects to the X-13ARIMA-SEATS method. Thus, the EVENT variables are treated in the same manner as it treats variables specified in the USERVAR= option in the REGRESSION statement. If a MDLINFOIN= data set is not specified in the PROC X13 statement, then all variables specified in the EVENT statements are applied to all BY groups and all time series that are processed. If a MDLINFOIN= data set is specified, then the EVENT statements apply only if no regression information for the BY group and series is available in the MDLINFOIN= data set. The events specified in the EVENT statements either must be SAS predefined events or must be defined in the data set specified in the INEVENT= option in the PROC X13 statement. For a summary of SAS predefined events, see the section "SAS Predefined Events" on page 3205.

The EVENT statement can also be used to include outlier, level-shift, and temporary change regressors that are available as predefined US Census Bureau variables in the X-13ARIMA-SEATS program. For example, the following statements specify an additive outlier in January 1970 and a level-shift that begins in July 1971:

```
proc x13 data=ICMETI seasons=12 start=jan1968;
   event AO01JAN1970D CBLS01JUL1971D;
```

The following statements specify an additive outlier in the second quarter 1970 and a temporary change that begins in the fourth quarter 1971:

```
proc x13 data=ICMETI seasons=4 start='1970q1';
   event AO01APR1970D TC01OCT1971D;
```

The following *options* can appear in the EVENT statement:

B=(*value* < **F** > . . . **)**

specifies initial or fixed values for the EVENT parameters in the order in which they appear in *variables*. Each B= list applies to the variable list that immediately precedes the slash.

For example, the following statements set an initial value of 1 for the event, x:

```
event y ;
event x / b=1 2 ;
```

In this example, the B= option applies only to the second EVENT statement. The value 2 is discarded because there is only one variable in the variable list.

To assign an initial value of 1 to the y regressor and 2 to the x regressor, use the following statements:

```
event y / b=1;
event x / b=2 ;
```

An **F** immediately following the numerical value indicates that this is not an initial value, but a fixed value. See Example 45.8 for an example that uses fixed parameters. In PROC X13, individual parameters can be fixed while other parameters in the same model are estimated.

USERTYPE=(*values*)

enables a user-defined variable to be processed in the same manner as a US Census predefined variable. You can specify the following *values*: AO, CONSTANT, EASTER, HOLIDAY, LABOR, LOM, LOMSTOCK, LOQ, LPYEAR, LS, RP, SCEASTER, SEASONAL, TC, TD, TDSTOCK, THANKS, or USER. For example, the US Census Bureau EASTER(*w*) regression effects are included the "RegARIMA Holiday Component" table (A7). Specify USERTYPE=EASTER to include an event variable that is processed exactly as the US Census predefined EASTER(*w*) variable, including inclusion in the A7 table. The NOAPPLY= option in the REGRESSION statement also changes the processing of variables based on the USERTYPE= value. Table 45.4 shows the regression types that are associated with each regression effects table.

Each USERTYPE= list applies to the variable list that immediately precedes the slash. The same rules for assigning B= values to regression variables apply for USERTYPE= options. For example, the following statements specify that the event in the variable MyEaster be processed exactly as the US Census predefined LOM variable:

```
event MyLOM;
event MyEaster / usertype=LOM EASTER;
```

In this example, the USERTYPE= option applies only to the MyEaster variable in the second EVENT statement. The USERTYPE value **EASTER** is discarded because there is only one variable in the variable list.

To assign the USERTYPE value **LOM** to the MyLOM variable and **EASTER** to the MyEaster variable, use the following statements:

```
event MyLOM / usertype=LOM;
event MyEaster / usertype=EASTER;
```

The following USERTYPE= options specify that the regression effect be removed from the seasonally adjusted series: EASTER, HOLIDAY, LABOR, LOM, LOMSTOCK, LOQ, LPYEAR, SCEASTER, SEASONAL, TD, TDSTOCK, THANKS, and USER. When a regression effect is removed from the seasonally adjusted series, the level (mean) of the seasonally adjusted series can be altered. It is often desirable to use a zero-mean (mean-adjusted) regressor for effects that are to be removed from the seasonally adjusted series. See Example 45.6 for an example showing the effects of specifying a zero-mean regressor.

FORECAST Statement

FORECAST *options* **;**

The FORECAST statement uses the estimated model to forecast the time series. The output contains point forecasts and forecast statistics for the transformed and original series. Whenever forecasts or backcasts (or both) are generated and seasonal adjustment is performed, the forecasts and backcasts are appended to the original series, and the seasonal adjustment procedures are applied to the forecast or backcast (or both) extended series. If the FORECAST statement is not specified, but a regARIMA model is specified using either the ARIMA or AUTOMDL statement, then the series is extended one year ahead by default.

Tables that contain forecasts, standard errors, and confidence limits are displayed in association with the FORECAST statement. If the data is transformed, then two tables are displayed: one table for the original data, and one table for the transformed data. Data from these tables can be output to a SAS data set using ODS. The auxiliary variable _SCALE_ is included in forecast data sets that are output using ODS. The value of _SCALE_ is "Original" or "Transformed" to indicate the scale of the data. The auxiliary variable can also be used in ODS SELECT and ODS OUTPUT statements. For example, you can specify the following statements to output the forecasts on the original scale to a data set forecasts and the forecasts on the transformed scale to a data set Tforecasts:

```
ods output Original.ForecastCL=forecasts;
ods output Transformed.ForecastCL=Tforecasts;
```

The following *options* can appear in the FORECAST statement:

ALPHA=α

specifies the size of the upper and lower confidence limits, which are calculated as $1-\alpha$, where α must be between 0 and 1. By default, ALPHA=0.05, which produces 95% confidence intervals.

LEAD=*value*

specifies the number of periods ahead to forecast for regARIMA extension of the series. The default is the number of periods in a year (4 or 12), and the maximum is 120. Setting LEAD=0 specifies that the series not be extended by forecasts for seasonal adjustment. The LEAD= value also controls the number of forecasts that are displayed in Table D10.A. However, if the series is not extended by forecasts (LEAD=0), then the default year of forecasts is displayed in Table D10.A. Forecast values in Table D10.A are calculated using the method shown on page 148 of Ladiray and Quenneville (2001) based on values that are displayed in Table D10. The regARIMA forecasts affect the D10.A forecasts only indirectly through the impact of the regARIMA forecasts on the seasonal factors that are shown in Table D10. If the SEATSDECOMP statement is specified, then *value* is increased to the minimum required for SEATS decomposition. See the section "SEATS Decomposition" on page 3213 for details.

NBACKCAST=*value*

BACKCAST=*value*

NBACK=*value*

specifies the number of periods to backcast for regARIMA extension of the series. The default is NBACKCAST=0, which specifies that the series not be extended with backcasts. The maximum number of backcasts is 120. When the OUTBACKCAST option is specified, the NBACKCAST= value also controls the number of backcasts that are output to the OUT= data set specified in the OUTPUT

statement. If the SEATSDECOMP statement is specified, then *value* is increased to the minimum required for SEATS decomposition. See the section "SEATS Decomposition" on page 3213 for details.

OUT1STEP
specifies that the one-step-ahead forecasts be computed and displayed in addition to the multistep forecasts. The default is to compute and display only the multistep forecasts beginning at the forecast horizon.

OUTBACKCAST
OUTBKCAST
determines whether backcasts are included in certain tables sent to the output data set. If OUTBACKCAST is specified, then backcast values are included in the output data set for tables A6, A7, A8, A9, A10, B1, D10, D10B, D10D, D16, D16B, and D18. The default is not to include backcasts.

OUTFCST
OUTFORECAST
determines whether forecasts are included in certain tables sent to the output data set. If OUTFORECAST is specified, then forecast values are included in the output data set for tables A6, A7, A8, A9, A10, B1, D10, D10B, D10D, D16, D16B, D18, and E18. The default is not to include forecasts. The OUTFORECAST option can be specified in either the X11 statement or the FORECAST statement with identical results.

ID Statement

ID *variables* **;**

If you are creating an output data set, use the ID statement to copy values of the ID variables, in addition to the table values, into the output data set. Or, if the VAR statement is omitted, all numeric variables that are not identified as BY variables, ID variables, the DATE= variable, or user-defined regressors are processed as time series. The ID statement has no effect when a VAR statement is specified and an output data set is not created. If the DATE= variable is specified in the PROC X13 statement, this variable is included automatically in the OUTPUT data set. If no DATE= variable is specified, the variable _DATE_ is added.

The date variable (or _DATE_) values outside the range of the actual data (from forecasting) are extrapolated, while all other ID variables are missing in the forecast horizon.

IDENTIFY Statement

IDENTIFY *options* **;**

The IDENTIFY statement produces plots of the sample autocorrelation function (ACF) and partial autocorrelation function (PACF) for identifying the ARIMA part of a regARIMA model. The sample ACF and PACF are produced for all combinations of the nonseasonal and seasonal differences of the data specified by the DIFF= and SDIFF= options.

The original series is first transformed as specified in the TRANSFORM statement.

If the model includes a regression component (specified using the REGRESSION, INPUT, and EVENT statements or the MDLINFOIN= data set in the PROC X13 statement), both the transformed series and

the regressors are differenced at the highest order that is specified in the DIFF= and SDIFF= option. The parameter estimates are calculated using the differenced data. Then the undifferenced regression effects (with the exception of a constant term) are removed from the undifferenced data to produce undifferenced regression residuals. The ACFs and PACFs are calculated for the specified differences of the undifferenced regression residuals.

If the model does not include a regression component, then the ACFs and PACFs are calculated for the specified differences of the transformed data.

Tables displayed in association with identification are "Autocorrelation of Model Residuals" and "Partial Autocorrelation of Model Residuals." If the model includes a regression component (specified using the REGRESSION, INPUT, and EVENT statements or the MDLINFOIN= data set in the PROC X13 statement), then the "Regression Model Parameter Estimates" table is also displayed if the PRINTREG option is specified.

The following *options* can appear in the IDENTIFY statement:

DIFF=(*order***,** *order***,** *order***)**
: specifies orders of nonseasonal differencing to use in model identification. The value 0 specifies no differencing, the value 1 specifies one nonseasonal difference $(1 - B)$, the value 2 specifies two nonseasonal differences $(1 - B)^2$, and so forth. The ACFs and PACFs are produced for all orders of nonseasonal differencing specified, in combination with all orders of seasonal differencing that are specified in the SDIFF= option. The default is DIFF=(0). You can specify up to three values for nonseasonal differences.

MAXLAG=*value*
: specifies the number of lags for the sample autocorrelation function (ACF) and partial autocorrelation function (PACF) of the regression residuals for model identification. The default is 36 for monthly series and 12 for quarterly series. MAXLAG applies to both tables and plots. The minimum value for MAXLAG= is 1.

PRINTREG
: causes the "Regression Model Parameter Estimates" table to be printed if the REGRESSION statement is present. By default, this table is not printed.

SDIFF=(*order***,** *order***,** *order***)**
: specifies orders of seasonal differencing to use in model identification. The value 0 specifies no seasonal differencing, the value 1 specifies one seasonal difference $(1 - B^s)$, the value 2 specifies two seasonal differences $(1 - B^s)^2$, and so forth. The value for s corresponds to the period specified in the SEASONS= option in the PROC X13 statement. The value of the SEASONS= option is supplied explicitly or is implicitly supplied through the INTERVAL= option or the values of the DATE= variable. The ACFs and PACFs are produced for all orders of seasonal differencing specified, in combination with all orders of nonseasonal differencing specified in the DIFF= option. The default is SDIFF=(0). You can specify up to three values for seasonal differences.

  For example, the following statement produces ACFs and PACFs for two levels of differencing: $(1 - B)$ and $(1 - B)(1 - B^s)$:

```
identify diff=(1) sdiff=(0, 1);
```

INPUT Statement

INPUT *variables* < / *options* > **;**

The INPUT statement specifies variables in the DATA= or AUXDATA= data set (which are specified in the PROC X13 statement) that are to be used as regressors in the regression portion of the regARIMA model. The variables in the data set should contain the values for each observation that define the regressor. Past values of regression variables should also be included in the DATA= or AUXDATA= data set if the time series listed in the VAR statement is to be extended with regARIMA backcasts. Similarly, future values of regression variables should also be included in the DATA= or AUXDATA= data set if the time series listed in the VAR statement is to be extended with regARIMA forecasts.

You can specify multiple INPUT statements. If you do not specify a MDLINFOIN= data set in the PROC X13 statement, then all variables listed in the INPUT statements are applied to all BY groups and all time series that are processed. If you specify a MDLINFOIN= data set, then the INPUT statements apply only if no regression information for the BY group and series is available in the MDLINFOIN= data set.

The INPUT statement provides the same functionality as the USERVAR= option in the REGRESSION statement. For more information about specifying user-defined regression variables, see the section "User-Defined Regression Variables" on page 3208, Example 45.6, and Example 45.11.

The following *options* can appear in the INPUT statement:

B=(*value* < **F** > . . . **)**
specifies initial or fixed values for the regression parameters in the order in which they appear in *variables*. Each B= list applies to the variable list that immediately precedes the slash.

For example, the following statements set an initial value of 1 for the user-defined regressor, x:

```
input y ;
input x / b=1 2 ;
```

In this example, the B= option applies only to the second INPUT statement. The value 2 is discarded because there is only one variable in the variable list.

To assign an initial value of 1 to the y regressor and 2 to the x regressor, use the following statements:

```
input y / b=1;
input x / b=2 ;
```

An **F** immediately following the numerical value indicates that this is not an initial value, but a fixed value. See Example 45.8 for an example that uses fixed parameters. In PROC X13, individual parameters can be fixed while other parameters in the same model are estimated.

USERTYPE=(*values***)**
enables a user-defined variable to be processed in the same manner as a US Census predefined variable. You can specify the following *values*: AO, CONSTANT, EASTER, HOLIDAY, LABOR, LOM, LOMSTOCK, LOQ, LPYEAR, LS, RP, SCEASTER, SEASONAL, TC, TD, TDSTOCK, THANKS, or USER. For example, the US Census Bureau EASTER(*w*) regression effects are included the "RegARIMA Holiday Component" table (A7). Specify USERTYPE=EASTER to include a user-defined variable that is processed exactly as the US Census predefined EASTER(*w*) variable, including inclusion in the A7 table. The NOAPPLY= option in the REGRESSION statement also changes the

processing of variables based on the USERTYPE= value. Table 45.4 shows the regression types that are associated with each regression effects table.

Each USERTYPE= list applies to the variable list that immediately precedes the slash. The same rules for assigning B= values to regression variables apply for USERTYPE= options. For example, the following statements specify that the user-defined regressor in the variable MyEaster be processed exactly as the US Census predefined LOM variable:

```
input MyLOM;
input MyEaster / usertype=LOM EASTER;
```

In this example, the USERTYPE= option applies only to the MyEaster variable in the second INPUT statement. The USERTYPE value EASTER is discarded because there is only one variable in the variable list.

To assign the USERTYPE value LOM to the MyLOM variable and EASTER to the MyEaster variable, use the following statements:

```
input MyLOM / usertype=LOM;
input MyEaster / usertype=EASTER;
```

The following USERTYPE= options specify that the regression effect be removed from the seasonally adjusted series: EASTER, HOLIDAY, LABOR, LOM, LOMSTOCK, LOQ, LPYEAR, SCEASTER, SEASONAL, TD, TDSTOCK, THANKS, and USER. When a regression effect is removed from the seasonally adjusted series, the level (mean) of the seasonally adjusted series can be altered. It is often desirable to use a zero-mean (mean-adjusted) regressor for effects that are to be removed from the seasonally adjusted series. See Example 45.6 for an example that specifies a zero-mean regressor.

OUTLIER Statement

OUTLIER *options* **;**

The OUTLIER statement specifies that the X13 procedure perform automatic detection of additive point outliers, temporary change outliers, level-shifts, or any combination of the three when using the specified model. After outliers are identified, the appropriate regression variables are incorporated into the model as "Automatically Identified Outliers," and the model is reestimated. This procedure is repeated until no additional outliers are found.

The OUTLIER statement also identifies potential outliers and lists them in the "Potential Outliers" table in the displayed output. Potential outliers are identified by decreasing the critical value by the value that is specified in the ALMOST= option.

In the output, the initial critical values used for outlier detection in a given analysis are displayed in the table "Critical Values to Use in Outlier Detection." Outliers that are detected and incorporated into the model are displayed in the output in the table "Regression Model Parameter Estimates," where the regression variable is listed as "Automatically Identified."

You can specify the following *options*:

ALMOST=*value*

specifies the difference between the critical value for an automatically identified outlier and a potential outlier that is "almost" identified. *value* is subtracted from the critical value that is used to identify outliers to form a critical value that more aggressively identifies potential outliers. Potential outliers are not included in the regARIMA model. However, potential outliers are displayed in the "Potential Outliers" table. *value* must be greater than 0. By default, ALMOST=0.5.

ALPHA=*value*

specifies the significance level to use for outlier identification, where critical values are calculated based on *value*. Any critical value that is specified in the CV=, AOCV=, LSCV=, or TCCV= option overrides the critical values that are calculated based on this option. *value* must be greater than 0 and less than or equal to 0.1. If you do not specify this option or the CV= option, the X-13ARIMA-SEATS method calculates the default initial critical value by assuming ALPHA=0.05.

AOCV=*value*

specifies a critical value to use for additive point outliers. If you specify this option, it overrides any default initial critical value for AO outliers. For more information, see the CV= option.

CV=*value*

specifies a default initial critical value to use for detecting all types of outliers. The absolute value of the *t* statistic that is associated with an outlier parameter estimate is compared with *value* to determine the significance of the outlier. If you do not specify this option, then the default initial critical value is computed based on the ALPHA= option, the CVMETHOD= option, and the number of observations for the model span that is used in the analysis. Table 45.2 shows initial critical values for various series lengths, which are based on the default values of the ALPHA= option and CVMETHOD= option. Increasing the critical value decreases the sensitivity of the outlier detection routine and can reduce the number of observations that are treated as outliers. The automatic model identification process might decrease the critical value by a certain percentage if the automatic model identification process fails to identify an acceptable model.

Table 45.2 Default Critical Values for Outlier Identification

Number of Observations	Outlier Critical Value
1	1.96
2	2.24
3	2.44
4	2.62
5	2.74
6	2.84
7	2.92
8	2.99
9	3.04
10	3.09
11	3.13
12	3.16
24	3.42
36	3.55
48	3.63
72	3.73

Table 45.2 *continued*

Number of Observations	Outlier Critical Value
96	3.80
120	3.85
144	3.89
168	3.92
192	3.95
216	3.97
240	3.99
264	4.01
288	4.03
312	4.04
336	4.05
360	4.07

CVMETHOD=CORRECTED | LJUNG

specifies the method to use to calculate the default initial critical value, based on the ALPHA= value and the number of observations for the model span that is used in the analysis. You can specify the following values:

CORRECTED uses a method that is a modification of the Ljung method in which critical values are interpolated based on the number of observations in the model span.

LJUNG uses a method that is based on the asymptotic formula described in Ljung (1993).

By default, CVMETHOD=CORRECTED.

LSCV=*value*

specifies a critical value to use for level-shift outliers. If you specify this option, it overrides any default initial critical value for LS outliers. For more information, see the CV= option.

LSRUN=*value*

specifies the maximum number of successive level-shift outliers to combine to form a temporary level-shift. Valid *values* for this option are 0 to 5, inclusive. If LSRUN=0 or LSRUN=1, no temporary level-shifts are evaluated. The evaluation of the temporary level-shifts is displayed in the "Tests for Cancellation of Level Shifts" table. By default, LSRUN=0.

METHOD=ADDALL | ADDONE

specifies whether to add outliers one at a time for each model estimation iteration or to add all outliers at once for each model estimation iteration. You can specify the following values:

ADDALL includes all significant outliers as regressors in the model, and then reestimates the model.

ADDONE adds the most significant outlier as a regressor in the model, and then reestimates the model.

For both methods, all candidate points for outliers are evaluated at each iteration and model estimation iterations continue until no remaining outliers are identified. By default, METHOD=ADDONE.

SPAN=(*mmmyy* **,***mmmyy* **)**

SPAN=(*'yyQq'* **,***'yyQq'* **)**

specifies the dates of the first and last observations to define a subset for searching for outliers. A single date in parentheses is interpreted to be the starting date of the subset. To specify only the ending date, use SPAN=(,*mmmyy*) or SPAN=(,*'yyQq'*). If the starting or ending date is omitted, then the first or last date, respectively, of the input data set or BY group is assumed. Because the dates are input as strings and the quarterly dates begin with a numeric character, the specification for a quarterly date must be enclosed in quotation marks. A four-digit year can be specified. If a two-digit year is specified, the value specified in the YEARCUTOFF= SAS system option applies.

TCCV=*value*

specifies a critical value to use for temporary change (TC) outliers. If you specify this option, it overrides any default initial critical value for TC outliers. For more information, see the CV= option.

TCRATE=*value*

specifies the rate of decay for temporary change outliers. *value* must be greater than 0 and less than 1. The default value is $(0.7)^{\frac{12}{period}}$, where *period* is the number of observations in one year.

TYPE=NONE

TYPE=(*outlier types***)**

lists the outlier types to be detected by the automatic outlier identification method. TYPE=NONE turns off outlier detection. The valid outlier types are AO, LS, and TC. The default is TYPE=(AO LS).

OUTPUT Statement

OUTPUT *OUT=SAS-data-set* < **YEARSEAS** > *tablename1 tablename2 . . .* **;**

The OUTPUT statement creates an output data set that contains specified tables. The data set is named by the OUT= option.

OUT=*SAS-data-set*

names the data set to contain the specified tables. If the OUT= option is omitted, the data set is named using the default DATA*n* convention.

YEARSEAS

YRSEAS

specifies that two additional variables be added to the OUT= data set. The two additional variables are the variables _YEAR_ and _SEASON_. The variable _YEAR_ contains the year of the date identifying the observation. The variable _SEASON_ contains the month for monthly data, or quarter for quarterly data, of the date that identifies the observation. For monthly data, the value of _SEASON_ is between 1 and 12. For quarterly data, the value of _SEASON_ is between 1 and 4. The _YEAR_ and _SEASON_ variables are useful when creating seasonal plots.

tablename1 tablename2 . . .

specify X13 *tablenames* that correspond to the title label used by the US Census Bureau X-13ARIMA-SEATS software. Specify one *tablename* for each table to be included in the output data set. Currently available *tablenames* are A1, A2, A6, A7, A8, A8AO, A8LS, A8TC, A9, A10, A19, B1, B7, B13, B17, B20, C1, C17, C20, D1, D7, D8, D8B, D8BX, D8BO, D8BL, D9, D10, D10B, D10D, D11,

D11A, D11F, D11R, D12, D13, D16, D16B, D18, E1, E2, E3, E5, E6, E6A, E6R, E7, E8, E18, and MV1. Specifying D8B is equivalent to specifying D8, D8BX, D8BO, and D8BL because Table D 8.B displays the D8 series along with labels for extremes (D8BX), outliers (D8BO), and level shifts (D8BL). If no table is specified in the OUTPUT statement, Table A1 is output to the OUT= data set by default.

The *tablenames* that can be used in the OUTPUT statement are listed in the section "Displayed Output, ODS Table Names, and OUTPUT Tablename Keywords" on page 3213. The following is an example of a VAR statement and an OUTPUT statement:

```
var sales costs;
output out=out_x13  b1 d11;
```

The default variable name used in the output data set is the input variable name followed by an underscore and the corresponding table name. The variable sales_B1 contains the Table B1 values for the variable sales, the variable costs_B1 contains the Table B1 values for the variable costs, the variable sales_D11 contains the Table D11 values for the variable sales, and the variable costs_D11 contains the Table D11 values for the variable costs. If necessary, the variable name is shortened so that the table name can be added. If the DATE= variable is specified in the PROC X13 statement, then that variable is included in the output data set; otherwise, a variable named _DATE_ is written to the OUT= data set as the date identifier.

PICKMDL Statement (Experimental)

PICKMDL *options* **;**

The PICKMDL statement uses models specified in the MDLINFOIN= data set to choose a regARIMA model. The MDLINFOIN= data set can include a variable that identifies different models. All observations with the same value for the model identification variable are considered to be relevant to the same model. A single model can be considered to consist of all the observations for a BY group that consists of the BY variables (if any), the _NAME_ variable if it exists, and the MDLVAR= variable. The default variable name that identifies the model is _MODEL_. Example 45.9 demonstrates the use of the PICKMDL statement.

For more information about using the US Census Bureau's PICKMDL method for model selection, see the section "PICKMDL Model Selection" on page 3212 .

You can specify the following *options* in the PICKMDL statement:

MDLVAR=*variable*
specifies the variable in the MDLINFOIN= data set that identifies the models.

METHOD= BEST | FIRST
specifies the method for choosing the regARIMA model. If you specify METHOD=BEST, the best model is chosen. If you specify METHOD=FIRST, the first acceptable model is chosen.

REGRESSION Statement

REGRESSION *regression-group-options* **;**

REGRESSION PREDEFINED= *variables* < / **B=(***value* < **F** > **...)** > **;**

REGRESSION USERVAR= *variables* < / **B=(***value* < **F** > **...) USERTYPE=(***values***)** > **;**

The REGRESSION statement includes regression variables in a regARIMA model or specifies regression variables whose effects are to be removed by the IDENTIFY statement to aid in ARIMA model identification. Include the PREDEFINED= option to select predefined regression variables. Include the USERVAR= option to specify user-defined regression variables.

Table 45.3 shows the X-13ARIMA-SEATS tables that contain regression factors. Tables A8AO, A8LS, and A8TC are available only when more than one outlier type is present in the model.

Table 45.3 X-13ARIMA-SEATS Regression Effects Tables

Table	Regression Effects
A6	Trading day effects
A7	Holiday effects including Easter, Labor Day, and Thanksgiving-Christmas
A8	Combined effects of outliers, level-shifts, ramps, and temporary changes
A8AO	Point outlier effects; available only when more than one outlier type is present in the model
A8LS	Level-shift and ramp effects; available only when more than one outlier type is present in the model
A8TC	Temporary change effects; available only when more than one outlier type is present in the model
A9	User-defined regression effects
A10	User-defined seasonal component effects

Missing values in the span of an input series automatically create missing value regressors. See the NOTRIMMISS option in the PROC X13 statement and the section "Missing Values" on page 3204 for further details about missing values.

Combining your model with additional predefined regression variables can result in a singularity problem. To successfully perform the regression if a singularity occurs, you might need to alter either the model or the choices of the regressors.

To seasonally adjust a series that uses a regARIMA model, the factors derived from regression are used as multiplicative or additive factors, depending on the mode of seasonal decomposition. Therefore, regressors that are appropriate to the mode of the seasonal decomposition should be defined, so that meaningful combined adjustment factors can be derived and adjustment diagnostics can be generated. For example, if a regARIMA model is applied to a log-transformed series, then the regression factors are expressed as ratios, which match the form of the seasonal factors that are generated by the multiplicative or log-additive adjustment modes. Conversely, if a regARIMA model is fit to the original series, then the regression factors are measured on the same scale as the original series, which matches the scale of the seasonal factors that are generated by the additive adjustment mode. Note that the default transformation (no transformation) and the default seasonal adjustment mode (multiplicative) are in conflict. Thus, when you specify the X11 statement and any of the REGRESSION, INPUT, or EVENT statements, you must also either use the TRANSFORM statement to specify a transformation or use the MODE= option in the X11 statement to specify a different mode to seasonally adjust the data that uses the regARIMA model.

According to Ladiray and Quenneville (2001), "X-12-ARIMA is based on the same principle [as the X-11 method] but proposes, in addition, a complete module, called Reg-ARIMA, that allows for the initial series to be corrected for all sorts of undesirable effects. These effects are estimated using regression models with ARIMA errors (Findley et al. [23])." The REGRESSION, INPUT, and EVENT statements specify these regression effects. Predefined effects that can be corrected in this manner are listed in the PREDEFINED= option. You can create your own definitions to remove other effects by using the USERVAR= option and the EVENT statement.

You can specify either the PREDEFINED= option or the USERVAR= option, but not both, in a single REGRESSION statement. You can use multiple REGRESSION statements.

You can specify the following *regression-group-options* in the REGRESSION statement. The *regression-group-options* apply to all regression variables in a regression group. For predefined regression variables, the regression group is predefined. For user-defined regression variables, you can specify the regression group in the USERTYPE= option.

AICTEST=(EASTER | TD | TD1COEF | TD1NOLPYEAR | TDNOLPYEAR | TDSTOCK | USER)

specifies that an AIC-based selection be used to determine whether a given set of regression variables are to be included with the specified regARIMA model. For example, if you specify a trading day model selection, then AIC values (with a correction for the length of the series, henceforth referred to as AICC) are derived for models with and without the specified trading day variable. By default, the model with a smaller AICC is used to generate forecasts, identify outliers, and so on. If you specify more than one type of regressor, the AIC tests are performed sequentially in this order: (a) trading day regressors, (b) Easter regressors, (c) user-defined regressors. If there are several variables of the same type (for example, several trading day regressors), then AIC-based selection is applied to them as a group. That is, either all variables of this type or none are included in the final model. If you do not specify this option, no automatic AIC-based selection is performed.

If you use the AUTOMDL statement to identify the model and you also specify this option, then this option affects the model selection process in the following manner:

- AIC-based selection tests are performed on the default model.
- A new series is created by removing the regression effects that are identified in the default model from the original series. The automatic model identification process attempts to identify a model that is based on the new series.
- After a model is automatically identified, AIC-based selection tests that use the automatically identified model are performed on the original series.
- The default model, including regressors that are identified by using AIC-based selection, is compared to the automatically identified model, which also might include regressors that are identified by using AIC-based selections. The regressors for the two models can differ.

For more information about the X-13ARIMA-SEATS automatic modeling method, see section 7.2 of the *X-13ARIMA-SEATS Reference Manual* (US Bureau of the Census 2009).

EASTERMEANS=(YR400 | YR500 | SPAN)

specifies how the monthly means, which are used to remove seasonality from the EASTER predefined regressor, are calculated. When PREDEFINED=EASTER(*w*) is specified in the REGRESSION statement, monthly means are computed internally over the 500-year range from years 1600 to 2099 by default. These monthly means are then used to remove seasonality from the Easter effect prior to

calculating the Easter regression coefficient. The EASTERMEANS= option is ignored if no predefined EASTER regressor is included in the regression model or if SCEASTER(*w*) is the only predefined Easter regressor specified. You can specify the following values:

SPAN computes short-term monthly means rather than long-term monthly means to remove seasonality in the Easter effect. In this case, the monthly means are computed over the same span of data that is used to calculate the coefficient of the EASTER(*w*) regressor.

YR400 computes monthly means over the 400-year range from years 1583–1982. This method was used in earlier versions of the X-13ARIMA-SEATS methodology.

YR500 computes monthly means over the 500-year range from years 1600–2099.

By default, EASTERMEANS=YR500.

NOAPPLY=(AO | HOLIDAY | LS | TC | TD | USER | USERSEASONAL)
specifies a list of the types of regression effects whose model-estimated values are not to be removed from the original series before performing the seasonal adjustment calculations that are specified by the X11 statement. The NOAPPLY= option applies to the regression component values displayed in the X11 seasonal adjustment method regARIMA component tables as shown in Table 45.4.

Table 45.4 NOAPPLY= Types and Regression Effects

NOAPPLY= Option	Regression Effects Table	Description
AO	A8AO	Point outliers
HOLIDAY	A7	Easter, Labor Day, and Thanksgiving-to-Christmas holiday effects
LS	A8LS	Level changes and ramps
TC	A8TC	Temporary changes
TD	A6	Trading day effects
USER	A9	User-defined regression effects
USERSEASONAL	A10	User-defined seasonal regression effects

You can specify the following regression variable specification options in the REGRESSION statement.

PREDEFINED=CONSTANT | EASTER(*value*) | LABOR(*value*) | LOM | LOMSTOCK | LOQ | LPYEAR
PREDEFINED=SCEASTER(*value*) | SEASONAL | SINCOS(*value . . .*) | TD | TD1COEF
PREDEFINED=TD1NOLPYEAR | TDNOLPYEAR | TDSTOCK(*value*) | THANK(*value*)
lists the predefined regression variables to be included in the model. Data values for these variables are calculated by the program, mostly as functions of the calendar. Table 45.5 gives definitions for the available predefined variables. The values LOM and LOQ are equivalent: the actual regression is controlled by the SEASONS= option in the PROC X13 statement. You can specify multiple predefined regression variables. The syntax for using both a length-of-month and a seasonal regression can be in one of the following forms:

```
regression predefined=lom seasonal;

regression predefined=(lom seasonal);

regression predefined=lom predefined=seasonal;
```

The following restrictions apply when you use more than one predefined regression variable:

- You can specify only one of TD, TDNOLPYEAR, TD1COEF, or TD1NOLPYEAR.
- You cannot specify LPYEAR with TD, TD1COEF, LOM, LOMSTOCK, or LOQ.
- You cannot specify LOM or LOQ with TD or TD1COEF.
- If you specify the SINCOS predefined regression variable, then you must also specify the INTERVAL= option or the SEASONS= option in the PROC X13 statement because there are restrictions on this regression variable that are based on the frequency of the data.

The predefined regression variables, EASTER, LABOR, SCEASTER, SINCOS, TDSTOCK, and THANK, require extra parameters. Only one TDSTOCK regressor can be implemented in the regression model. If you specify multiple TDSTOCK variables, PROC X13 uses the last TDSTOCK variable specified. For EASTER, LABOR, SCEASTER, SINCOS, and THANK, you can specify the variables with different parameters to implement multiple regressors in the model. For example, the following statement specifies two EASTER regressors with widths 7 and 14:

```
regression predefined=easter(7) easter(14);
```

For SINCOS, specifying a parameter includes both the sine and the cosine regressor except for the highest order allowed (2 for quarterly data and 6 for monthly data.) For quarterly data, the following statement is the most common use of the SINCOS variable; it includes three regressors in the model:

```
regression predefined=sincos(1,2);
```

For monthly data, the following statement is the most common use of the SINCOS variable; it includes 11 regressors in the model:

```
regression predefined=sincos(1,2,3,4,5,6);
```

Table 45.5 Predefined Regression Variables in X-13ARIMA-SEATS

Regression Effect	Variable Definitions
Trend constant CONSTANT	$(1-B)^{-d}(1-B^s)^{-D}I(t \geq 1)$ where $I(t \geq 1) = \begin{cases} 1 & \text{for } t \geq 1 \\ 0 & \text{for } t < 1 \end{cases}$
Easter holiday EASTER(w)	$E(w,t) = \frac{1}{w} \times n_t$ and n_t is the number of the w days before Easter that fall in month (or quarter) t. (Note: This variable is 0 except in February, March, and April (or first and second quarter). It is nonzero in February only for $w > 22$.) Restriction: $1 \leq w \leq 25$.
Labor Day LABOR(w)	$L(w,t) = \frac{1}{w} \times$ [no. of the w days before Labor Day that fall in month t] (Note: This variable is 0 except in August and September.) Restriction: $1 \leq w \leq 25$.

Table 45.5 *continued*

Regression Effect	Variable Definitions
Length-of-month (monthly flow) LOM	$m_t - \bar{m}$ where m_t = length of month t (in days) and $\bar{m} = 30.4375$ (average length of month)
Stock length-of-month LOMSTOCK	$SLOM_t = \begin{cases} m_t - \bar{m} - \mu(l) & \text{for } t = 1 \\ SLOM_{t-1} + m_t - \bar{m} & \text{otherwise} \end{cases}$ where $\bar{m}$ and m_t are defined in LOM and $\mu(l) = \begin{cases} 0.375 & \text{when first February in series is a leap year} \\ 0.125 & \text{when second February in series is a leap year} \\ -0.125 & \text{when third February in series is a leap year} \\ -0.375 & \text{when fourth February in series is a leap year} \end{cases}$
Length-of-quarter (quarterly flow) LOQ	$q_t - \bar{q}$ where q_t = length of quarter t (in days) and $\bar{q} = 91.3125$ (average length of quarter)
Leap year (monthly and quarterly flow) LPYEAR	$LY_t = \begin{cases} 0.75 & \text{in leap year February (first quarter)} \\ -0.25 & \text{in other Februaries (first quarter)} \\ 0 & \text{otherwise} \end{cases}$
Statistics Canada Easter (monthly or quarterly flow) SCEASTER(w)	If Easter falls before April w, let n_E be the number of the w days on or before Easter that fall in March. Then: $E(w,t) = \begin{cases} n_E/w & \text{in March} \\ -n_E/w & \text{in April} \\ 0 & \text{otherwise} \end{cases}$ If Easter falls on or after April w, then $E(w,t) = 0$. (Note: This variable is 0 except in March and April (or first and second quarter).) Restriction: $1 \leq w \leq 24$.
Fixed seasonal SEASONAL	$M_{1,t} = \begin{cases} 1 & \text{in January} \\ -1 & \text{in December} \\ 0 & \text{otherwise} \end{cases}$ $, \ldots, M_{11,t} = \begin{cases} 1 & \text{in November} \\ -1 & \text{in December} \\ 0 & \text{otherwise} \end{cases}$

Table 45.5 *continued*

Regression Effect	Variable Definitions
Fixed seasonal SINCOS(*j*) SINCOS($j_1, \ldots, j_n$)	$\sin(w_j t), \cos(w_j t)$, where $w_j = 2\pi j/s$, $1 \le j \le s/2$, and s is the seasonal period (drop $\sin(w_j t) \equiv 0$ for $j = s/2$) Restrictions: $1 \le j_i \le s/2$, $1 \le n \le s/2$.
Trading day TD, TDNOLPYEAR	$T_{1,t} =$ (number of Mondays) – (number of Sundays) $, \ldots, T_{6,t} =$ (number of Saturdays) – (number of Sundays)
One coefficient trading day TD1COEF, TD1NOLPYEAR	(number of weekdays) $- \frac{5}{2}$(number of Saturdays and Sundays)
Stock trading day TDSTOCK(*w*)	$D_{1,t} = \begin{cases} 1 & \tilde{w}\text{th day of month } t \text{ is a Monday} \\ -1 & \tilde{w}\text{th day of month } t \text{ is a Sunday} \\ 0 & \text{otherwise} \end{cases}$ $, \ldots, D_{6,t} = \begin{cases} 1 & \tilde{w}\text{th day of month } t \text{ is a Saturday} \\ -1 & \tilde{w}\text{th day of month } t \text{ is a Sunday} \\ 0 & \text{otherwise} \end{cases}$ where $\tilde{w}$ is the smaller of w and the length of month t. For end-of-month stock series, set w to 31; that is, specify TDSTOCK(31). Restriction: $1 \le w \le 31$.
Thanksgiving THANK(*w*)	ThC(w, t) = proportion of days from w days before Thanksgiving through December 24 that fall in month t (negative values of w indicate days after Thanksgiving). (Note: This variable is 0 except in November and December.) Restriction: $-8 \le w \le 17$.

USERVAR=(*variables*)

specifies variables in the DATA= or AUXDATA= data set (which are specified in the PROC X13 statement) that are to be used as regressors. The variables in the data set should contain the values for each observation that define the regressor. Regression variables should also include future values in the data set for the forecast horizon if the time series is to be extended with regARIMA forecasts. Regression variables should include past values if the time series is to be extended with regARIMA backcasts. Missing values are not permitted within the data span, including backcasts and forecasts, of the user-defined regressors. Example 45.6 shows how to create an input data set that contains both the series to be seasonally adjusted and a user-defined input variable. Example 45.11 shows how to create an auxiliary data set that contains a user-defined input variable. For more information about specifying user-defined regression variables see the section "User-Defined Regression Variables" on page 3208.

All regression variables in the USERVAR= option apply to all time series to be seasonally adjusted unless the MDLINFOIN= data set specifies different regression information. You cannot specify the PREDEFINED= option and the USERVAR= option in the same REGRESSION statement; however, you can specify multiple REGRESSION statements.

You can specify the following *options* for individual regression variables. Individual regression variable options are specified in the PREDEFINED= and USERVAR= options after the slash. The B= option can be specified in both the PREDEFINED= and USERVAR= options. Because the regression group is predefined for predefined variables, you can specify the USERTYPE= option only in the USERVAR= option.

B=(*value* < **F** > . . . **)**
specifies initial or fixed values for the regression parameters in the order in which they appear in a PREDEFINED= or USERVAR= option. Each B= list applies to the PREDEFINED= or USERVAR= variable list that immediately precedes the slash.

For example, the following statements set an initial value of 1 for the user-defined regressor, x:

```
regression predefined=LOM ;
regression uservar=x / b=1 2 ;
```

In this example, the B= option applies only to the USERVAR= option. The value 2 is discarded because there is only one variable in the USERVAR= list.

To assign an initial value of 1 to the LOM regressor and 2 to the x regressor, use the following statements:

```
regression predefined=LOM / b=1;
regression uservar=x / b=2 ;
```

An F immediately following the numerical value indicates that this is not an initial value, but a fixed value. See Example 45.8 for an example that uses fixed parameters. In PROC X13, individual parameters can be fixed while other parameters in the same model are estimated.

USERTYPE=(*values***)**
enables a variable that you define to be processed in the same manner as a US Census predefined variable. You can specify the following *values*: AO, CONSTANT, EASTER, HOLIDAY, LABOR, LOM, LOMSTOCK, LOQ, LPYEAR, LS, RP, SCEASTER, SEASONAL, TC, TD, TDSTOCK, THANKS, or USER. For example, the US Census Bureau EASTER(*w*) regression effects are included the "RegARIMA Holiday Component" table (A7). Specify USERTYPE=EASTER to define a variable that is processed exactly as the US Census predefined EASTER(*w*) variable, including inclusion in the A7 table. Each USERTYPE= list applies to the USERVAR= variable list that immediately precedes the slash. USERTYPE= does not apply to US Census predefined variables.

The same rules for assigning B= values to regression variables apply for USERTYPE= options. For example, the following statements specify that the user-defined regressor in the variable MyEaster be processed exactly as the US Census predefined LOM variable:

```
regression uservar=MyLOM;
regression uservar=MyEaster / usertype=LOM EASTER;
```

In this example, the USERTYPE= option applies only to the MyEaster variable in the second REGRESSION statement. The USERTYPE value EASTER is discarded because there is only one variable in the USERVAR= list.

To assign the USERTYPE value LOM to the MyLOM variable and EASTER to the MyEaster variable, use the following statements:

```
regression uservar=MyLOM / usertype=LOM;
regression uservar=MyEaster / usertype=EASTER;
```

The following USERTYPE= options specify that the regression effect be removed from the seasonally adjusted series: EASTER, HOLIDAY, LABOR, LOM, LOMSTOCK, LOQ, LPYEAR, SCEASTER, SEASONAL, TD, TDSTOCK, THANKS, and USER. When a regression effect is removed from the seasonally adjusted series, the level (mean) of the seasonally adjusted series can be altered. It is often desirable to use a zero-mean (mean-adjusted) regressor for effects that are to be removed from the seasonally adjusted series. See Example 45.6 for an example that specifies a zero-mean regressor.

SEATSDECOMP Statement (Experimental)

SEATSDECOMP OUT= *SAS-data-set* < *options* > **;**

The SEATSDECOMP statement creates an output data set (named by the OUT= option) that contains the SEATS decomposition series.

The following is an example of a VAR statement and a SEATSDECOMP statement:

```
var sales costs;
seatsdecomp out=SEATS_DECOMP;
```

The default variable name used in the output data set is the input variable name followed by an underscore and the corresponding table name. Because the B1 series is used as the original input series for the SEATS decomposition, the output data set SEATS_DECOMP from the example will contain the seasonal decomposition variables in the following order:

sales_OS	contains the Table B1 values for the variable sales.
sales_SC	contains the SEATS decomposition seasonal component for the variable sales.
sales_TC	contains the SEATS trend component values for the variable sales.
sales_SA	contains the SEATS seasonally adjusted series for the variable sales.
sales_IC	contains the SEATS irregular component for the variable sales.
costs_OS	contains the Table B1 values for the variable costs.
costs_SC	contains the SEATS decomposition seasonal component for the variable costs.
costs_TC	contains the SEATS trend component values for the variable costs.
costs_SA	contains the SEATS seasonally adjusted series for the variable costs.
costs_IC	contains the SEATS irregular component for the variable costs.

If necessary, the variable name is shortened so that the component name can be added. If you specify the DATE= variable in the PROC X13 statement, then that variable is included in the output data set; otherwise, a variable named _DATE_ is written to the OUT= data set as the date identifier. For further information about the output data set, see SEATSDECOMP OUT= Data Set.

You can specify the following *options* in the SEATSDECOMP statement:

LEAD=*value*

specifies the number of periods ahead to forecast for a regARIMA extension of the series. The default is twice the number of periods in a year (8 or 24), and the maximum is 120. In the SEATS computations, the number of backcasts and forecasts are the same, and the minimum number is also dependent on the ARIMA model orders. For more information, see the section "SEATS Decomposition" on page 3213. If you specify a LEAD= value that is less than the default, then the number of forecasts specified in the LEAD= option are displayed in the OUT= data set. If the value of the LEAD= option and NBACKCAST= options in the FORECAST statement are less than the required number for SEATS decomposition, then the values of the LEAD= and NBACKCAST= options in the FORECAST statement are increased.

NBACKCAST=*value*

BACKCAST=*value*

NBACK=*value*

specifies the number of periods to backcast for a regARIMA extension of the series. The default is twice the number of periods in a year (8 or 24), and the maximum is 120. In the SEATS computations, the number of backcasts and forecasts are the same, and the minimum number is also dependent on the ARIMA model orders. For more information, see the section "SEATS Decomposition" on page 3213. If you specify a NBACKCAST= value that is less than the default, then the number of backcasts specified in the NBACKCAST= option are displayed in the OUT= data set. If the value of the LEAD= option and NBACKCAST= option specified in the FORECAST statement are less than the required number for SEATS decomposition when SEATSDECOMP is specified, then the value of LEAD= and NBACKCAST= in the FORECAST statement will be increased.

OUT=*SAS-data-set*

names the data set to contain the SEATS decomposition series: original series, seasonal component, trend component, seasonally adjusted series, irregular component. If the OUT= option is omitted, the data set is named using the default DATA*n* convention.

YEARSEAS

YRSEAS

specifies that two additional variables be added to the OUT= data set: _YEAR_ and _SEASON_. The variable _YEAR_ contains the year of the date that identifies the observation. The variable _SEASON_ contains the month for monthly data, or quarter for quarterly data, of the date that identifies the observation. For monthly data, the value of _SEASON_ is between 1 and 12. For quarterly data, the value of _SEASON_ is between 1 and 4. The _YEAR_ and _SEASON_ variables are useful when you create seasonal plots.

TABLES Statement

TABLES *tablename1 tablename2 . . . options* **;**

The TABLES statement enables you to alter the display of the PROC X13 tables. You can specify the display of tables that are not displayed by default by PROC X13, and the NOSUM option enables you to suppress the printing of the period summary line in the time series tables.

tablename1 tablename2 . . .
specifies X13 *tablenames* that correspond to the title label used by the US Census Bureau X-13ARIMA-SEATS software. For each table to be included in the displayed output, you must specify the X13 *tablename* keyword. Currently available tables are A19, B7, B13, B17, B20, C1, C20, D1, D7, E1, E2, and E3. Although these tables are not displayed by default, their values are sometimes useful in understanding the X-13ARIMA-SEATS method. For further description of the available tables, see the section "Displayed Output, ODS Table Names, and OUTPUT Tablename Keywords" on page 3213.

NOSUM

NOSUMMARY

NOSUMMARYLINE
applies to the tables available for output in the OUTPUT Statement. By default, these tables include a summary line that gives the average, total, or standard deviation for the historical data by period. The NOSUM option suppresses the display of the summary line in the listing. Also, if the tables are output with ODS, the summary line is not an observation in the data set. Thus, the output to the data set is only the time series, both the historical data and the forecast data, if available.

TRANSFORM Statement

TRANSFORM *options* **;**

The TRANSFORM statement transforms or adjusts the series prior to estimating a regARIMA model. With this statement, the series can be Box-Cox (power) transformed. The "Prior Adjustment Factors" table is associated with the TRANSFORM statement.

Only one of the following *options* can appear in the TRANSFORM statement:

POWER=*value*
transforms the input series, Y_t, by using a Box-Cox power transformation,

$$Y_t \rightarrow y_t = \begin{cases} \log(Y_t) & \lambda = 0 \\ \lambda^2 + (Y_t^{\lambda} - 1)/\lambda & \lambda \neq 0 \end{cases}$$

The power λ must be specified (for example, POWER=0.33). The default is no transformation ($\lambda = 1$); that is, POWER=1. The log transformation (POWER=0), square root transformation (POWER=0.5), and the inverse transformation (POWER=–1) are equivalent to the corresponding FUNCTION= option.

Table 45.6 Power Values Related to the Census Bureau Function Argument

FUNCTION=	Transformation	Range for Y_t	Equivalent Power Argument
NONE	Y_t	All values	POWER=1
LOG	$\log(Y_t)$	$Y_t > 0$ for all t	POWER=0
SQRT	$2(\sqrt{Y_t} - 0.875)$	$Y_t \geq 0$ for all t	POWER=0.5
INVERSE	$2 - \frac{1}{Y_t}$	$Y_t \neq 0$ for all t	POWER=–1
LOGISTIC	$\log(\frac{Y_t}{1-Y_t})$	$0 < Y_t < 1$ for all t	No equivalent

FUNCTION=NONE | LOG | SQRT | INVERSE | LOGISTIC | AUTO

specifies the transformation to be applied to the series prior to estimating a regARIMA model. The transformation used by FUNCTION=NONE, LOG, SQRT, INVERSE, or LOGISTIC is related to the POWER= option as shown in Table 45.6. FUNCTION=AUTO uses selection based on Akaike's information criterion (AIC) to decide between a log transformation and no transformation. The default is FUNCTION=NONE.

However, the FUNCTION= and POWER= options are not completely equivalent. In some cases, using the FUNCTION= option causes the program to automatically select other options. For example, FUNCTION=NONE causes the default mode to be MODE=ADD in the X11 statement. Also, the choice of transformation invoked by the FUNCTION=AUTO option can impact the default mode of the X11 statement.

There are restrictions on the value used in the POWER= and FUNCTION= options when preadjustment factors for seasonal adjustment are generated from a regARIMA model. When seasonal adjustment is requested with the X11 statement, any value of the POWER option can be used for the purpose of forecasting the series with a regARIMA model. However, this is not the case when factors generated from the regression coefficients are used to adjust either the original series or the final seasonally adjusted series. In this case, the only accepted transformations are the log transformation, which can be specified as POWER=0 for multiplicative or log-additive seasonal adjustments, and no transformation, which can be specified as POWER=1 for additive seasonal adjustments. If no seasonal adjustment is performed, any POWER transformation can be used. The preceding restrictions also apply when FUNCTION=NONE and FUNCTION=LOG are specified.

USERDEFINED Statement

USERDEFINED *variables* **;**

The USERDEFINED statement is used to identify the variables in the input data set or auxiliary data set that are available for user-defined regression. Only numeric variables can be specified. Specifying variables in the USERDEFINED statement does not include the variables as regressors. If a variable is specified in the INPUT statement or USERVAR= option in the REGRESSION statement, it is not necessary to include that variable in the USERDEFINED statement. However, if a variable is specified in the MDLINFOIN= data set in the PROC X13 statement and is not specified in an INPUT statement or in the USERVAR= option in the REGRESSION statement, then the variable should be specified in the USERDEFINED statement in order to make the variable available for regression.

VAR Statement

VAR *variables* **;**

The VAR statement specifies the variables in the input data set that are to be analyzed by the procedure. Only numeric variables can be specified. If the VAR statement is omitted, all numeric variables are analyzed except those that appear in a BY statement, ID statement, INPUT statement, USERDEFINED statement, in the USERVAR= option in the REGRESSION statement, or the variable named in the DATE= option in the PROC X13 statement.

X11 Statement

X11 *options* **;**

The X11 statement is an optional statement for invoking seasonal adjustment by an enhanced version of the methodology of the US Census Bureau X-11 and X-11Q programs. You can control the type of seasonal adjustment decomposition calculated with the MODE= option. The output includes the final tables and diagnostics for the X-11 seasonal adjustment method listed in Table 45.7. Tables B7, B13, B17, B20, C1, E1, E2, E3, C20, D1, and D7 are not displayed by default; however, you can display these tables by requesting them in the TABLES statement.

Table 45.7 Tables Related to X11 Seasonal Adjustment

Table Name	Description
B1	Original series, adjusted for prior effects and forecast extended
B7	Preliminary trend-cycle, B iteration
B13	Irregular component, B iteration
B17	Preliminary weights for the irregular component
B20	Extreme values, B iteration
C1	Original series modified for outliers, trading day, and prior factors, C iteration
C17	Final weights for the irregular component
C20	Final extreme value adjustment factors
D1	Modified original data, D iteration
D7	Preliminary trend cycle, D iteration
D8	Final unmodified SI ratios (differences)
D8A	*F* tests for stable and moving seasonality, D8
D8B	Final unmodified SI ratios, with labels for outliers and extreme values
D9	Final replacement values for extreme SI ratios (differences), D iteration
D9A	Moving seasonality ratios for each period
SeasonalFilter	Seasonal filter statistics for Table D10
D10	Final seasonal factors
D10B	Seasonal factors, adjusted for user-defined seasonal
D10D	Final seasonal difference
D11	Final seasonally adjusted series
D11A	Final seasonally adjusted series with forced yearly totals
D11R	Rounded final seasonally adjusted series (with forced yearly totals)
TrendFilter	Trend filter statistics for Table D12
D12	Final trend cycle
D13	Final irregular component
D16	Combined seasonal and trading day factors
D16B	Final adjustment differences
D18	Combined calendar adjustment factors
E1	Original data modified for extremes
E2	Modified seasonally adjusted series
E3	Modified irregular series
E4	Ratio of yearly totals of original and seasonally adjusted series
E5	Percent changes (differences) in original series
E6	Percent changes (differences) in seasonally adjusted series
E6A	Percent changes (differences) in seasonally adjusted series with forced yearly totals (D11.A)
E6R	Percent changes (differences) in rounded seasonally adjusted series (D11.R)
E7	Percent changes (differences) in final trend component series
E8	Percent changes (differences) in original series adjusted for calendar factors (A18)
E18	Final adjustment ratios (original series to seasonally adjusted series)
F2A–F2I	X11 diagnostic summary
F3	Monitoring and quality assessment statistics
F4	Day of the week trading day component factors
G	Spectral plots

For more information about the X-11 seasonal adjustment diagnostics, see Shiskin, Young, and Musgrave (1967), Lothian and Morry (1978a), and Ladiray and Quenneville (2001).

You can specify the following *options* in the X11 statement:

FINAL=AO | LS | TC | USER |ALL

FINAL=(*options*)

lists the types of prior adjustment factors, obtained from the EVENT, REGRESSION, and OUTLIER statements, that are to be removed from the final seasonally adjusted series. Additive outliers are removed by specifying FINAL=AO. Level change and ramp outliers are removed by specifying FINAL=LS. Temporary change outliers are removed by specifying FINAL=TC. User-defined regressors or events (USERTYPE=USER) are removed by specifying FINAL=USER. All the preceding are removed by specifying FINAL=ALL or by specifying all the options in parentheses, FINAL=(AO LS TC USER). If this option is not specified, the final seasonally adjusted series contains these effects.

FORCE=TOTALS | ROUND | BOTH

specifies that the seasonally adjusted series be modified to: (a) force the yearly totals of the seasonally adjusted series and the original series to be the same (FORCE=TOTALS), (b) adjust the seasonally adjusted values for each calendar year so that the sum of the rounded seasonally adjusted series for any year equals the rounded annual total (FORCE=ROUND), or (c) first force the yearly totals, then round the adjusted series (FORCE=BOTH). When FORCE=TOTALS is specified, the differences between the annual totals is distributed over the seasonally adjusted values in a way that approximately preserves the month-to-month (or quarter-to-quarter) movements of the original series. For more details, see Huot (1975) and Cholette (1979). This forcing procedure is not recommended if the seasonal pattern is changing or if trading day adjustment is performed. Forcing the seasonally adjusted totals to be the same as the original series annual totals can degrade the quality of the seasonal adjustment, especially when the seasonal pattern is undergoing change. It is not natural if trading day adjustment is performed because the aggregate trading day effect over a year is variable and moderately different from zero.

MODE=ADD | MULT | LOGADD | PSEUDOADD

determines the mode of the seasonal adjustment decomposition to be performed. The four option choices correspond to additive, multiplicative, log-additive, and pseudo-additive decomposition, respectively. If this option is omitted, the procedure performs multiplicative adjustments. Table 45.8 shows the values of the MODE= option and the corresponding models for the original (O) and the seasonally adjusted (SA) series.

Table 45.8 Modes of Seasonal Adjustment and Their Models

Value of Mode Option	Name	Model for O	Model for SA
MULT	Multiplicative	$O = C \times S \times I$	$\mathrm{SA} = C \times I$
ADD	Additive	$O = C + S + I$	$\mathrm{SA} = C + I$
PSEUDOADD	Pseudo-additive	$O = C \times [S + I - 1]$	$\mathrm{SA} = C \times I$
LOGADD	Log-additive	$\log(O) = C + S + I$	$\mathrm{SA} = \exp(C + I)$

OUTFORECAST

OUTFCST

determines whether forecasts are included in certain tables sent to the output data set. If OUTFORECAST is specified, then forecast values are included in the output data set for tables A6, A7, A8, A9, A10, B1, D10, D10B, D10D, D16, D16B, D18, and E18. The default is not to include forecasts. The

OUTFORECAST option can be specified in either the X11 statement or the FORECAST statement with identical results.

SEASONALMA=S3X1 | S3X3 | S3X5 | S3X9 | S3X15 | STABLE | X11DEFAULT | MSR

SEASONALMA=*(filter-list-by-period)*

specifies which seasonal moving average (also called "seasonal filter") to use to estimate the seasonal factors. These seasonal moving averages are $n \times m$ *moving averages*, meaning that an n-term simple average is taken of a sequence of consecutive m-term simple averages. X11DEFAULT is the method used by the US Census Bureau's X-11-ARIMA program.

You can specify either a single filter option or a list. A single option indicates that all periods will use the same filter or the same method of identifying the filter. Alternately, you can specify the seasonal filters for each seasonal period by specifying SEASONALMA=*(filter-list-by-period)*, where *(filter-list-by-period)* lists the moving average filter for each period. For quarterly data, you must specify four filters; for monthly data, you must specify 12 filters. In the *filter-list-by-period*, you can specify S3X1, S3X3, S3X5, S3X9, or S3X15. For example, the following statement assigns a 3×1 moving average filter to the first quarter of a quarterly series and a 3×3 moving average to the second, third, and fourth quarters:

```
X11 SEASONALMA=( S3X1 S3X3 S3X3 S3X3 );
```

Table 45.9 describes the seasonal filter options available for the entire series:

Table 45.9 X-13ARIMA-SEATS Seasonal Filter Options and Descriptions

Filter Name	Description of Filter
S3X1	A 3×1 moving average
S3X3	A 3×3 moving average
S3X5	A 3×5 moving average
S3X9	A 3×9 moving average
S3X15	A 3×15 moving average
STABLE	Stable seasonal filter: a single seasonal factor for each calendar month or quarter is generated by calculating the simple average of all the values for each month or quarter (taken after detrending and outlier adjustment)
X11DEFAULT	Uses a 3×3 moving average to calculate the initial seasonal factors in each iteration and a 3×5 moving average to calculate the final seasonal factors
MSR	Filter chosen automatically by using the moving seasonality ratio of X-11-ARIMA/88 (Dagum 1988)

By default, SEASONALMA=MSR, which is the methodology of Statistic Canada's X-11-ARIMA/88 program.

SIGMALIM=(*lower limit***,** *upper limit* **)**

SIGMALIM=(*lower limit* **)**

SIGMALIM=(, *upper limit* **)**

specifies the lower and upper sigma limits in standard deviation units which are used to identify and down-weight extreme irregular values in the internal seasonal adjustment computations. One or both limits can be specified. The lower limit must be greater than 0 and not greater than the upper limit. If the lower sigma limit is not specified, then it defaults to a value of 1.5. The default upper sigma limit is 2.5. The comma must be used if the upper limit is specified.

Table 45.10 shows the effect of the SIGMALIM= option on the weights that are applied to the internal irregular values.

Table 45.10 Weights for Irregular Values

Weight	Sigma Limit
0	If $\frac{\lvert I_t-\mu\rvert}{\sigma_{1,I_t}} \geq$ *upper limit*
Partial weight	If *lower limit* $< \frac{\lvert I_t-\mu\rvert}{\sigma_{2,I_t}} <$ *upper limit*
1	If $\frac{\lvert I_t-\mu\rvert}{\sigma_{2,I_t}} \leq$ *lower limit*

In Table 45.10, μ is the theoretical mean of the irregular component, and σ_{1,I_t} and σ_{2,I_t} are the respective estimates of the standard deviation of the irregular component before and after extreme values are removed. The estimates of the standard deviation σ_{1,I_t} and σ_{2,I_t} vary with respect to t, and they are the same if no extreme values are removed. If they are different ($\sigma_{2,I_t} < \sigma_{1,I_t}$), then the first line in Table 45.10 is reevaluated with σ_{2,I_t}. In the special case where the lower limit equals the upper limit, the weight is 1 for $\frac{\lvert I_t-\mu\rvert}{\sigma_{2,I_t}} \leq$ *lower limit*, and 0 otherwise. For more information about how extreme irregular values are handled in the X11 computations, see Ladiray and Quenneville 2001, pp. 53–68, 122–125.

TRENDMA=*value*

specifies which Henderson moving average is used to estimate the final trend cycle. Any odd number greater than one and less than or equal to 101 can be specified (for example, TRENDMA=23). If the TRENDMA= option is not specified, the program selects a trend moving average based on statistical characteristics of the data. For monthly series, a 9-, 13-, or 23-term Henderson moving average is selected. For quarterly series, the program chooses either a 5- or a 7-term Henderson moving average.

TYPE=SA | SUMMARY | TREND

specifies the method used to calculate the final seasonally adjusted series (Table D11). The default method is TYPE=SA. This method assumes that the original series has not been seasonally adjusted. For method TYPE=SUMMARY, the trend cycle, irregular, trading day, and holiday factors are calculated, but not removed from the seasonally adjusted series. Thus, for TYPE=SUMMARY, Table D11 is the same as the original series. For TYPE=TREND, trading day, holiday, and prior adjustment factors are removed from the original series to calculate the seasonally adjusted series (Table D11) and also are used in the calculation of the final trend (Table D12).

Details: X13 Procedure

Data Requirements

The input data set must contain either quarterly or monthly time series, and the data must be sorted in chronological order within each BY group. For the standard X-13ARIMA-SEATS method, there must be at least three years of observations (12 for quarterly time series or 36 for monthly).

If an ARIMA model is specified in the ARIMA statement, AUTOMDL statement, PICKMDL statement, or the MDLINFOIN= data set, then more than three years of observations might be required in order to fit the ARIMA model and perform the computations associated with the seasonal decomposition and other diagnostics.

The minimum number of observations applies to each series listed in the VAR statement and within each BY group and is determined after any missing values are trimmed from the series.

Missing Values

PROC X13 can process a series with missing values.

Types of Missing Values

Missing values in a series are considered to be one of two types:

- A leading or trailing missing value occurs before the first nonmissing value or after the last nonmissing value, respectively, in the span of a series. The span of a series can be determined either explicitly by the SPAN= option or implicitly by the START= or DATE= option in the PROC X13 statement. By default, leading and trailing missing values are ignored. If you specify the NOTRIMMISS option in the PROC X13 statement, PROC X13 processes leading and trailing missing values according to the X-13ARIMA-SEATS missing value method.
- An embedded missing value occurs between the first nonmissing value and the last nonmissing value in the span of the series. PROC X13 processes embedded missing values according to the X-13ARIMA-SEATS missing value method.

X-13ARIMA-SEATS Missing Value Method

When the X-13ARIMA-SEATS method encounters a missing value, it inserts an additive outlier for the missing observation into the set of regression variables for the model of the series and then replaces the missing observation with a value large enough to be considered an outlier during model estimation. After the regARIMA model is estimated, the X-13ARIMA-SEATS method adjusts the original series by using factors that are generated from these missing value outlier regressors. The adjusted values are estimates of the missing values, and the adjusted series is displayed in Table MV1. The X-13ARIMA-SEATS missing value method requires the use of a regARIMA model to replace the missing values. Thus, either an ARIMA or AUTOMDL statement or the MDLINFOIN= option in the PROC X13 statement must be specified if there are embedded missing values in the time series.

SAS Predefined Events

SAS predefined events are summarized in this section. For complete details about SAS predefined events, see the section "EVENTKEY Statement" in *SAS Forecast Studio User's Guide*.

Table 45.11 shows a summary of the SAS predefined event keywords. Table 45.12 lists the holiday date keywords that can be used as SAS predefined events. Table 45.13 lists the seasonal date keywords that can be used as SAS predefined events.

Table 45.11 Definitions for EVENTKEY Predefined Event Keywords

Variable Name or Variable Name Format	Description	Qualifier Options
AO<*obs*>OBS AO<*date*>D AO<*datetime*>DT	Outlier	TYPE=POINT VALUE=1 BEFORE=(DURATION=0) AFTER=(DURATION=0)
LS<*obs*>OBS LS<*date*>D LS<*datetime*>DT	Level-shift	TYPE=LS VALUE=1 BEFORE=(DURATION=0) AFTER=(DURATION=ALL)
TLS<*obs*>OBS<n> TLS<*date*>D<n> TLS<*datetime*>DT<n>	Temporary level-shift	TYPE=LS VALUE=1 BEFORE=(DURATION=0) AFTER=(DURATION=<n>)
NLS<*obs*>OBS NLS<*date*>D NLS<*datetime*>DT	Negative level-shift	TYPE=LS VALUE=–1 BEFORE=(DURATION=0) AFTER=(DURATION=ALL)
CBLS<*obs*>OBS CBLS<*date*>D CBLS<*datetime*>DT	US Census Bureau level-shift	TYPE=LS VALUE=–1 SHIFT=–1 BEFORE=(DURATION=ALL) AFTER=(DURATION=0)
TC<*obs*>OBS TC<*date*>D TC<*datetime*>DT	Temporary change	TYPE=TC VALUE=1 BEFORE=(DURATION=0) AFTER=(DURATION=ALL)
<date keyword>	Date pulse	TYPE=POINT VALUE=1 BEFORE=(DURATION=0) AFTER=(DURATION=0) PULSE=DAY

Table 45.11 *continued*

Variable Name or Variable Name Format	Description	Qualifier Options
LINEAR QUAD CUBIC	Polynomial trends	TYPE=LIN TYPE=QUAD TYPE=CUBIC VALUE=1 BEFORE=(DURATION=ALL) AFTER=(DURATION=ALL) The default timing value is the 0 observation.
INVERSE LOG	Trends	TYPE=INV TYPE=LOG VALUE=1 BEFORE=(DURATION=0) AFTER=(DURATION=ALL) The default timing value is the 0 observation.
<seasonal keywords>	Seasonal	TYPE=POINT PULSE= depends on keyword VALUE=1 BEFORE=(DURATION=0) AFTER=(DURATION=0) Timing values are based on keyword.

Table 45.12 Holiday Date Keywords and Definitions

Date Keyword	Definition
BOXING	December 26th
CANADA	July 1st
CANADAOBSERVED	July 1st, or July 2nd if July 1st is a Sunday
CHRISTMAS	December 25th
COLUMBUS	Second Monday in October
EASTER	Easter Sunday
FATHERS	Third Sunday in June
HALLOWEEN	October 31st
LABOR	First Monday in September
MLK	Third Monday in January
MEMORIAL	Last Monday in May
MOTHERS	Second Sunday in May
NEWYEAR	January 1st
THANKSGIVING	Fourth Thursday in November
THANKSGIVINGCANADA	Second Monday in October
USINDEPENDENCE	July 4th
USPRESIDENTS	Third Monday in February (since 1971)
VALENTINES	February 14th
VETERANS	November 11th
VETERANSUSG	Veterans Day date that is observed by US government for Monday–Friday schedule
VETERANSUSPS	Veterans Day date that is observed by US government for Monday–Saturday schedule (US Post Office)
VICTORIA	Monday on or preceding May 24th

Table 45.13 Seasonal Date Keywords and Definitions

Date Keyword	Definition
SECOND_1, . . . , SECOND_60	Specified second
MINUTE_1, . . . , MINUTE_60	Beginning of the specified minute
HOUR_1, . . . , HOUR_24	Beginning of the specified hour
SUNDAY, . . . , SATURDAY	All Sundays, and so on, in the time series
WEEK_1, . . . , WEEK_53	First day of the *n*th week of the year (PULSE=WEEK.n shifts this date for $n \neq 1$)
TENDAY_1, TENDAY_4, . . . , TENDAY_34	The first day of the month
TENDAY_2, TENDAY_5, . . . , TENDAY_35	The 11th day of the month
TENDAY_3, TENDAY_6, . . . , TENDAY_36	The 21st day of the month
SEMIMONTH_1, SEMIMONTH_3, . . . , SEMIMONTH_23	The first day of the month
SEMIMONTH_2, SEMIMONTH_4, . . . , SEMIMONTH_24	The 16th day of the month
JANUARY, . . . , DECEMBER	The first day of the specified month
QTR_1, QTR_2, QTR_3, QTR_4	The first date of the quarter indicated after the underscore (PULSE=QTR.n shifts this date for $n \neq 1$)
SEMIYEAR_1, SEMIYEAR_2	The first date of the semiyear (PULSE=SEMIYEAR.n shifts this date for $n \neq 1$)

User-Defined Regression Variables

The X-13ARIMA-SEATS method enables you to define regression variables to be included in the regARIMA model. A user-defined regression variable is composed of a value at each time series observation that you provide; the entire variable is implemented as a regressor in the regARIMA model. The regARIMA model is used in the seasonal decomposition process to extend the series prior to X11 decomposition. Because the X-13ARIMA-SEATS method does not impute, forecast, nor backcast user-defined regression variables, you must provide a nonmissing value at each observation in the span of the time series to be modeled and also provide a nonmissing value at each observation to be forecast or backcast.

A user-defined regression variable can be included in either the PROC X13 DATA= or AUXDATA= data set. You can supply the values for the user-defined regression variable by one of the following methods:

- You can include them in an auxiliary data set. The auxiliary data set should have a date variable that corresponds to the date variable in the DATA= data set. The name of the auxiliary data set is specified in the AUXDATA= option in the PROC X13 statement. The name of the date variable that exists in both the DATA= and AUXDATA= data sets is specified in the DATE= option in the PROC X13 statement. The observations in the auxiliary data set must span the entire series plus any forecast and backcast period.

- You can include them in the DATA= data set. Because the number of observations and the date values are exactly the same for both user-defined regressors and time series values, you need to include forecast and backcast values for user-defined regression variables beyond the span of the time series in one of the following ways:
  - You must specify leading missing values in the series to be seasonally adjusted for backcast periods. You must specify trailing missing values in the series to be seasonally adjusted for forecast periods. You must not use the NOTRIMMISS option in this case. The span of the series to be seasonally adjusted that is implied by trimming the leading and trailing missing values will be less than the span of the date values in the DATA= data set. Using this method, forecast error cannot be computed for the forecast and backcast periods.
  - You can use the SPAN= option in the PROC X13 statement to alter the span of the series to be seasonally adjusted to allow for backcast and forecast periods within the span of the date values in the DATA= data set. Using this method, forecast error can be computed for the forecast and backcast periods.

These methods of including user-defined regression variables in the regARMIA model are illustrated in Example 45.6 and Example 45.11.

If missing values for the user-defined regression variable are present within the span of the time series, including backcast and forecast observations, then an error message is displayed and the time series is not processed. If the span of the user-defined regression variable, or the span after leading and trailing missing values are trimmed, is not sufficient to cover the span of the series to be seasonally adjusted, including any backcasts and forecasts, then an error message is also displayed, and the time series is not processed.

Combined Test for the Presence of Identifiable Seasonality

The seasonal component of a time series, S_t, is defined as the intrayear variation that is repeated constantly (stable) or in an evolving fashion from year to year (moving seasonality). If the increase in the seasonal factors from year to year is too large, then the seasonal factors introduce distortion into the model. It is important to determine whether seasonality is identifiable without distorting the series.

For seasonality to be identifiable, the series should be identified as seasonal by using the "Test for the Presence of Seasonality Assuming Stability" and "Nonparametric Test for the Presence of Seasonality Assuming Stability." Also, since the presence of moving seasonality can cause distortion, it is important to evaluate the moving seasonality in conjunction with the stable seasonality to determine whether the seasonality is identifiable. The results of these tests are displayed in "F tests for Seasonality" (Table D8.A) in the X13 procedure.

The test for identifiable seasonality is performed by combining the F tests for stable and moving seasonality, along with a Kruskal-Wallis test for stable seasonality. The following description is based on Lothian and Morry (1978b). Other details can be found in Dagum (1988, 1983).

Let F_s and F_m denote the F value for the stable and moving seasonality tests, respectively. The combined test is performed as follows (see also Figure 45.3):

1. If the null hypothesis of no stable seasonality is not rejected at the 0.10% significance level ($P_S \geq 0.001$), then the series is considered to be nonseasonal. PROC X13 returns the conclusion, "Identifiable Seasonality Not Present."

2. If the null hypothesis in step 1 is rejected, then PROC X13 computes the following quantities:

$$T_1 = \frac{7}{F_s}$$

$$T_2 = \frac{3F_m}{F_s}$$

   Let T denote the simple average of T_1 and T_2:

$$T = \frac{(T_1 + T_2)}{2}$$

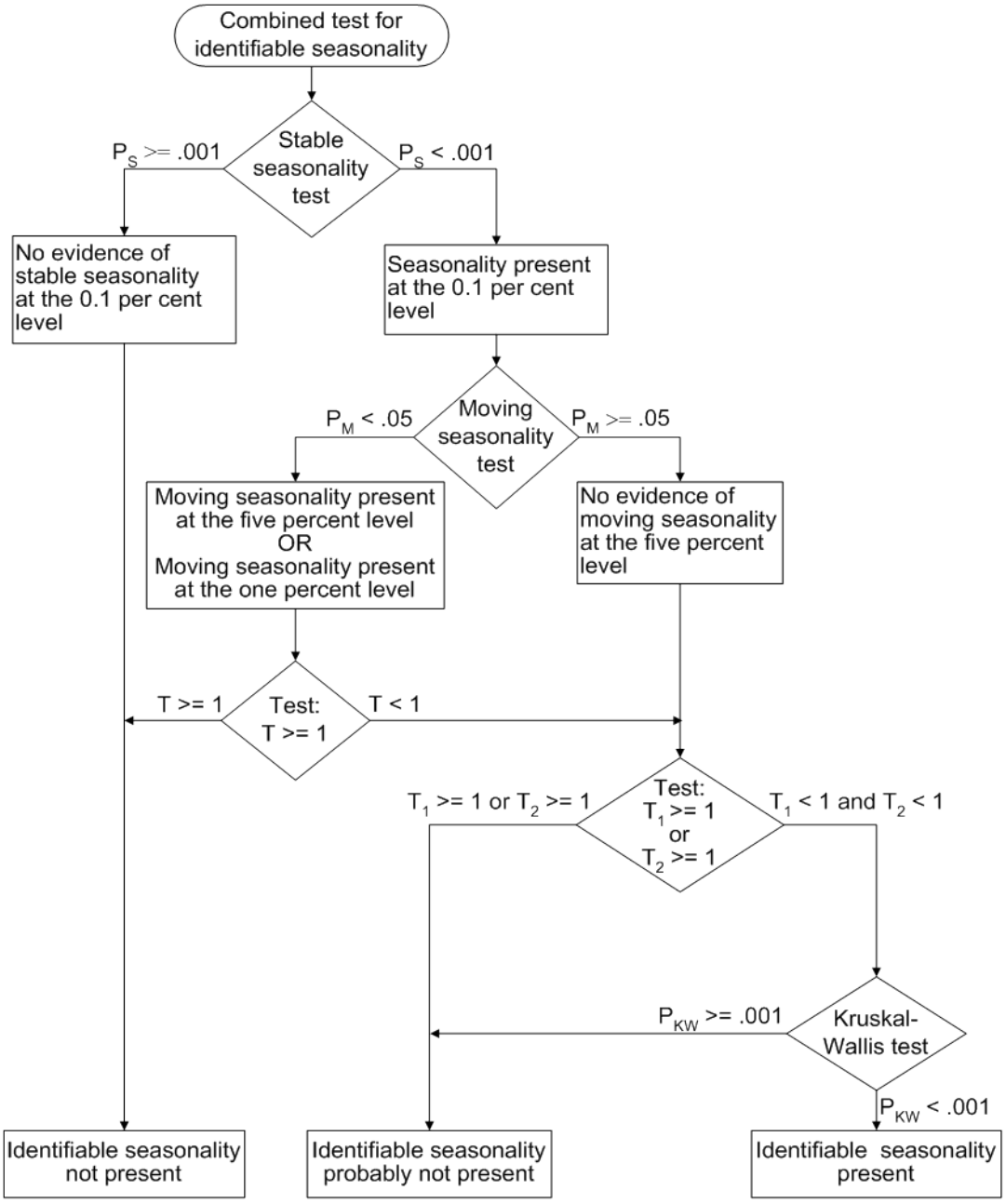
   If the null hypothesis of no moving seasonality is rejected at the 5.0% significance level ($P_M < 0.05$) and if $T \geq 1.0$, the null hypothesis of identifiable seasonality *not* present is not rejected and PROC X13 returns the conclusion, "Identifiable Seasonality Not Present."

3. If the null hypothesis of identifiable seasonality *not* present has not been accepted, but $T_1 \geq 1.0$, $T_2 \geq 1.0$, or the Kruskal-Wallis chi-squared test fails to reject at the 0.10% significance level ($P_{\mathrm{KW}} \geq 0.001$), then PROC X13 returns the conclusion "Identifiable Seasonality Probably Not Present."

4. If the null hypotheses of no stable seasonality associated with the F_S and Kruskal-Wallis chi-squared tests are rejected and if none of the combined measures described in steps 2 and 3 fail, then the null hypothesis of identifiable seasonality *not* present is rejected and PROC X13 returns the conclusion "Identifiable Seasonality Present."

Included in the displayed output of Table D8A is the table "Summary of Results and Combined Test for the Presence of Identifiable Seasonality." This table displays the T_1, T_2, and T values and the significance levels for the stable seasonality test, the moving seasonality test, and the Kruskal-Wallis test. The last item in the table is the result of the combined test for identifiable seasonality.

Figure 45.3 Combined Seasonality Test Flowchart



Computations

For more details about the computations used in PROC X13, see the *X-13ARIMA-SEATS Reference Manual* (US Bureau of the Census 2009).

For more details about the X-11 method of decomposition, see *Seasonal Adjustment with the X-11 Method* (Ladiray and Quenneville 2001).

PICKMDL Model Selection

You can request that the X-13ARIMA-SEATS method select a model in a manner similar to the method used in X-11-ARIMA (Dagum 1988, 1983) by specifying the PICKMDL statement. Information about this model selection (PICKMDL) is based on the description in the *X-13ARIMA-SEATS Reference Manual* (US Bureau of the Census 2009).

The default settings for the PICKMDL automatic model selection method classify a model as acceptable if all of the following conditions are true:

- The absolute average percentage error of the extrapolated values within the last three years of data is less than 15%.
- The p-value is greater than 5% for the fitted model's Ljung-Box Q statistic test of the lack of correlation in the model's residuals.
- There are no signs of overdifferencing. Overdifferencing is indicated if the sum of the nonseasonal MA parameter estimates (for models with at least one nonseasonal difference) is greater than 0.9.

No model is selected when none of the models in the MDLINFOIN= data set is acceptable.

The regARIMA model consists of a transformation, a regression component, and an ARIMA model component. For each series, the following conditions hold:

- If no regression is specified in the MDLINFOIN= data set model but regressors are specified using the INPUT, EVENT, or REGRESSION statements, then the ARIMA models from the MDLINFOIN= data set are tested in conjunction with the regression variables specified by the INPUT, EVENT, and REGRESSION statements.
- If no ARIMA model is specified in the MDLINFOIN= data set but an ARIMA model is specified using an ARIMA statement or TRANSFORM statement, then the regression information from each model specified in the MDLINFOIN= data set is used in conjunction with the ARIMA model specified by the TRANSFORM and ARIMA statements.
- If no model information is specified in the MDLINFOIN= data set, then any model information specified by the TRANSFORM, INPUT, REGRESSION, EVENT, and ARIMA statements is used, and the PICKMDL statement is not in effect for that series.

SEATS Decomposition

PROC X13 can decompose the B1 series by using the SEATS decomposition method described in Gómez and Maravall (1997a, b). The SEATS decomposition method is planned for inclusion in the US Census Bureau's X13 program, which is not yet available for release.

The SEATS method requires the series to be extended with the same number of backcast and forecast observations. The number of observations backcast and forecast must meet the following minimum criteria:

- The number of forecast and backcast observations must be at least twice the number of observations in a year, with a minimum of 8.
- The number of forecast and backcast observations must be at least $2 \times (q + Q * s)$, where the ARIMA model used to extend the series is $(pdq)(PDQ)s$ in standard Box-Jenkins notation.
- The number of forecast and backcast observations must be at least $p + d + q + (P + D + Q) * s$, where the ARIMA model used to extend the series is $(pdq)(PDQ)s$ in standard Box-Jenkins notation.

If you specify the SEATSDECOMP statement and the number of forecasts or backcasts (either the default number or the number you specify) is not sufficient for SEATS decomposition, then the number of forecasts or backcasts is increased to the minimum required.

Displayed Output, ODS Table Names, and OUTPUT Tablename Keywords

The options specified in PROC X13 control both the tables produced by the procedure and the tables available for output to the OUT= data set specified in the OUTPUT statement.

The displayed output is organized into tables identified by a part letter and a sequence number within the part. The seven major parts of the X13 procedure are as follows:

A	prior adjustments and regARIMA components (optional)
B	preliminary estimates of irregular component weights and trading day regression factors (X-11 method)
C	final estimates of irregular component weights and trading day regression factors
D	final estimates of seasonal, trend cycle, and irregular components
E	analytical tables
F	summary measures
G	charts

Table 45.14 describes the individual tables and charts. "P" indicates that the table is only displayed and is not available for output to the OUT= data set. Data from displayed tables can be extracted into data sets by using the Output Delivery System (ODS). For more information about the SAS Output Delivery System, see the *SAS Output Delivery System: User's Guide*. For more information about the features of the ODS Graphics system, including the many ways that you can control or customize the plots that are produced by SAS procedures, see Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*).

When tables available through the OUTPUT statement are output using ODS, the summary line is included in the ODS output by default. The summary line gives the average, standard deviation, or total by each period. The value –1 for YEAR indicates that the summary line is a total; the value –2 for YEAR indicates that the summary line is an average; and the value –3 for YEAR indicates that the line is a standard deviation. The value of YEAR for historical and forecast values is greater than or equal to zero. Thus, a negative value indicates a summary line. You can suppress the summary line altogether by specifying the NOSUM option in the TABLES statement. However, the NOSUM option also suppresses the display of the summary line in the displayed table.

"T" indicates that the table is available using the OUTPUT statement, but is not displayed by default; you must request that these tables be displayed by using the TABLES Statement. If there is no notation in the "Notes" column, then the table is available directly using the OUTPUT statement, without specifying the TABLES statement. If a table is not computed, then it is not displayed; if it is requested in the OUTPUT statement, then the variable in the OUT= data set contains missing values. The actual number of tables displayed depends on the options and statements specified.

Table 45.14 Table Names and Descriptions

Table	Description	Notes
Tables Associated with Model Order Identification		
ModelDescription	Regression model used in ARIMA model identification	P
ACF	Autocorrelation function	P
PACF	Partial autocorrelation function	P
Tables Associated with Automatic Modeling		
UnitRootTestModel	ARIMA estimates for unit root identification	P
UnitRootTest	Results of unit root test for identifying orders of differencing	P
AutoChoiceModel	Models estimated by automatic ARIMA model selection procedure	P
AutoLjungBox	Check of the residual Ljung-Box Q statistic	P
Best5Model	Best five ARIMA models chosen by automatic modeling	P
AutomaticModelChoice	Comparison of automatically selected model and default model	P
InitialModelChoice	Initial automatic model selection	P
FinalModelChecks	Final checks for identified model	P
FinalModelChoice	Final automatic model choice	P
Diagnostic Tables		
ErrorACF	Autocorrelation of regARIMA model residuals	P
ErrorPACF	Partial autocorrelation of regARIMA model residuals	P
SqErrorACF	Autocorrelation of squared regARIMA model residuals	P
ResidualOutliers	Outliers of the unstandardized residuals	P
ResidualStatistics	Summary statistics for the unstandardized residuals	P
NormalityStatistics	Normality statistics for regARIMA model residuals	P
G	Spectral analysis of regARIMA model residuals	P

Table 45.14 *continued*

Table	Description	Notes
Modeling Tables		
MissingExtreme	Extreme or missing values	P
ARMAIterationTolerances	Exact ARMA likelihood estimation iteration tolerances	P
IterHistory	ARMA iteration history	P
OutlierDetection	Critical values to use in outlier detection	P
PotentialOutliers	Potential outliers	P
TLSTest	Tests for cancellation of level-shifts	P
ARMAIterationSummary	Exact ARMA likelihood estimation iteration summary	P
ModelDescription	Model description for regARIMA model estimation	P
RegParameterEstimates	Regression model parameter estimates	P
RegressorGroupChiSq	Chi-squared tests for groups of regressors	P
ARMAParameterEstimates	Exact ARMA maximum likelihood estimation	P
AvgFcstErr	Average absolute percentage error in within-sample or without-sample forecasts or backcasts	P
Roots	Seasonal or nonseasonal AR or MA roots	P
MLESummary	Estimation summary	P
ForecastCL	Forecasts, standard errors, and confidence limits	P
MV1	Original series adjusted for missing value regressors	
Sequenced Tables		
A1	Original series	
A2	Prior-adjustment factors	
A6	RegARIMA trading day component	
A7	RegARIMA holiday component	
A8	RegARIMA combined outlier component	
A8AO	RegARIMA AO outlier component	
A8LS	RegARIMA level change outlier component	
A8TC	RegARIMA temporary change outlier component	
A9	RegARIMA user-defined regression component	
A10	RegARIMA user-defined seasonal component	
A19	RegARIMA outlier adjusted original data	T
B1	Prior-adjusted or original series	
B7	Preliminary trend-cycle, B iteration	T
B13	Irregular component, B iteration	T
B17	Preliminary weights for the irregular component	T
B20	Extreme values, B iteration	T
C1	Original series modified for outliers, trading day, and prior factors, C iteration	T
C17	Final weight for irregular components	
C20	Final extreme value adjustment factors	T
D1	Modified original data, D iteration	T
D7	Preliminary trend cycle, D iteration	T
D8	Final unmodified SI ratios	

Table 45.14 *continued*

Table	Description	Notes
D8A	Seasonality tests	P
D8B	Final unmodified SI ratios, with labels for outliers and extreme values	
D9	Final replacement values for extreme SI ratios	
D9A	Moving seasonality ratio	P
SeasonalFilter	Seasonal filter statistics for table D10	P
D10	Final seasonal factors	
D10B	Seasonal factors, adjusted for user-defined seasonal	
D10D	Final seasonal difference	
D11	Final seasonally adjusted series	
D11A	Final seasonally adjusted series with forced yearly totals	
D11F	Factors applied to get adjusted series with forced yearly totals	
D11R	Rounded final seasonally adjusted series (with forced yearly totals)	
TrendFilter	Trend filter statistics for table D12	P
D12	Final trend cycle	
D13	Final irregular series	
D16	Combined adjustment factors	
D16B	Final adjustment differences	
D18	Combined calendar adjustment factors	
E1	Original data modified for extremes	
E2	Modified seasonally adjusted series	
E3	Modified irregular series	
E4	Ratios of annual totals	P
E5	Percent changes in original series	
E6	Percent changes in final seasonally adjusted series	
E6A	Percent changes (differences) in seasonally adjusted series with forced yearly totals (D11.A)	
E6R	Percent changes (differences) in rounded seasonally adjusted series (D11.R)	
E7	Differences in final trend cycle	
E8	Percent changes (differences) in original series adjusted for calendar factors (A18)	
E18	Final adjustment ratios (original series to seasonally adjusted series)	
F2A-I	Summary measures	P
F3	Quality assessment statistics	P
F4	Day of the week trading day component factors	P
G	Spectral analysis	P

Table D 8.B

Table D8B displays the same series as Table D8. However, additional information is provided about the D8 series. The following values are displayed as labels for each observation of the series:

- The first label column indicates whether the D8 series value is extreme as determined by the X-11 extreme value method. An extreme observation is marked with an asterisk in the first label column. This data value is 0 or 1. If D8B or D8BX is specified in the OUTPUT statement, this value is output as the D8BX series to the data set that is specified in the OUT= option in the OUTPUT statement.
- The second label column contains the number of AO, TC, or RP outliers, if any, that affect the observation. This data value is 0 if no outliers affect the observation. Only the nonzero values are displayed in the table. If D8B or D8BO is specified in the OUTPUT statement, the number of outliers is output as the D8BO series to the data set that is specified in the OUT= option in the OUTPUT statement.
- The third label column indicates whether the observation is affected by level shift outliers as determined by an X-13ARIMA-SEATS method. This data value contains the number of level shifts that affect the observation. A nonzero value is displayed as "L". If D8B or D8BL is specified in the OUTPUT statement, the data values are output as the D8BL series to the data set that is specified in the OUT= option in the OUTPUT statement.

If any observations in Table D 8.B are affected by extremes, outliers, or level shifts, then notes that indicate the number of observations affected in each category are displayed at the end of the table.

Using Auxiliary Variables to Subset Output Data Sets

The X13 procedure can produce more than one table with the same name. For example, the following IDENTIFY statement produces ACF and PACF tables for two levels of differencing:

```
identify diff=(1) sdiff=(0, 1);
```

Auxiliary variables in the output data can be used to subset the data. In this example, the auxiliary variables Diff and SDiff specify the levels of regular and seasonal differencing that are used to compute the ACF. The following statements show how to retrieve the ACF results for the first differenced series:

```
ods select acf;
ods output acf=acf;
proc x13 data=sashelp.air date=date;
   identify diff=(1) sdiff=(0,1);
run;
title "Regular Difference=1 Seasonal Difference=0";
data acfd1D0;
   set acf(where=(Diff=1 and Sdiff=0));
run;
```

In addition to any BY variables, the auxiliary variables in the ACF and PACF data sets are _NAME_, _TYPE_, Transform, Adjust, Regressors, Diff and SDiff. Auxiliary variables can be related to the group as shown in the Results Viewer (for example, BY variables, _NAME_, and _TYPE_). However, they can also be variables in the template where printing is suppressed by using PRINT=OFF. Auxiliary variables such as Transform, Adjust, and Regressors are not displayed in the ACF and PACF tables because similar information is displayed in the ModelDescription table that immediately precedes the ACF and PACF tables. The variables Diff and SDiff are not displayed because the levels of differencing are included in the title of the ACF and PACF tables.

The BY variables and the _NAME_ variable are available for all ODS OUTPUT data sets that are produced by the X13 procedure. The _TYPE_ variable is available for all ODS OUTPUT data sets that are produced during the model identification and model estimation stages. The _TYPE_ variable enables you to determine whether data in a table, such as the ModelDescription table, originated from the model identification stage or the model estimation stage.

The forecast data sets contain the auxiliary variable _SCALE_. The value of _SCALE_ is "Original" or "Transformed" to indicate the scale of the data. The auxiliary variable _SCALE_ is the same as the group in the Results Viewer. It is not displayed in the forecast tables because the table titles indicate the scale of the data.

ODS Graphics

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*).

Before you create graphs, ODS Graphics must be enabled (for example, with the ODS GRAPHICS ON statement). For more information about enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" in that chapter.

The overall appearance of graphs is controlled by ODS styles. Styles and other aspects of using ODS Graphics are discussed in the section "A Primer on ODS Statistical Graphics" in that chapter.

This section describes the use of ODS for creating graphics with the X13 procedure.

The graphs available through ODS Graphics are ACF plots, PACF plots, a residual histogram, spectral graphs, and forecasting plots. ACF and PACF plots for regARIMA model identification are not available unless the IDENTIFY statement is used. ACF plots, PACF plots, the residual histogram, and the residual spectral graph for diagnosis of the regARIMA model residuals are not available unless the CHECK statement is used. Forecasting plots are not available unless the FORECAST statement is used. A spectral plot of the original series is always available; however, additional spectral plots are provided when the X11 statement and CHECK statement are used. When ODS Graphics is not enabled, the ACF, PACF, and spectral analysis are displayed as columns of a table. The residual histogram is available only when ODS Graphics is enabled. To obtain a table that contains values related to the residual histogram, use the ODS OUTPUT statement.

ODS Graph Names

PROC X13 assigns a name to each graph it creates by using ODS. You can use these names to selectively reference the graphs. The names are listed in Table 45.15.

Table 45.15 ODS Graphs Produced by PROC X13

ODS Graph Name	Plot Description	PROC X13 PLOTS= Option
ACFPlot	Autocorrelation of regression residuals	SERIES(ACF)
ErrorACFPlot	Autocorrelation of regARIMA model residuals	RESIDUAL(ACF)
ErrorPACFPlot	Partial autocorrelation of regARIMA model residuals	RESIDUAL(PACF)
ForecastsOnlyPlot	Forecasts only of the original series	FORECAST(FORECASTONLY)
ForecastsOnlyPlot	Forecasts only of the transformed series	FORECAST(TRANSFORECASTONLY)
ForecastsPlot	Forecasts of the original series	FORECAST(FORECAST)
ForecastsPlot	Forecasts of the transformed series	FORECAST(TRANSFORECAST)
ModelForecastsPlot	Model and forecasts of the original series	FORECAST(MODELFORECASTS)
ModelForecastsPlot	Model and forecasts of the transformed series	FORECAST(TRANSMODELFORECASTS)
ModelPlot	Model of the original series	FORECAST(MODELS)
ModelPlot	Model of the transformed series	FORECAST(TRANSMODELS)
PACFPlot	Partial autocorrelation of regression residuals	SERIES(PACF)
ResidualHistogram	Distribution of regARIMA residuals	RESIDUAL(HIST)
SpectralPlot	Spectral plot of the seasonally adjusted series	ADJUSTED(SPECTRUM)
SpectralPlot	Spectral plot of irregular series	IRREGULAR(SPECTRUM)
SpectralPlot	Spectral plot of the regARIMA model residuals	RESIDUAL(SPECTRUM)
SpectralPlot	Spectral plot of the original series	SERIES(SPECTRUM)
SqErrorACFPlot	Autocorrelation of squared regARIMA model residuals	RESIDUAL(SQACF)

OUT= Data Set

You can use the OUTPUT statement to output the component series computed in the X-13ARIMA-SEATS decomposition.

The OUT= data set specified in the OUTPUT statement contains the BY variables (if any), the ID variables (if any), and the DATE= variable if the DATE= option is specified or the variable _DATE_ if the DATE= option is not specified. If user-defined regressor variables or EVENT variables are specified, they are included. In addition, the various components specified by the table names in the OUTPUT statement are included in the OUT= data set.

The OUTPUT OUT= data set can contain the following variables:

BY variables	are the BY variables used to subset the series by BY groups. The BY variables included in this data set match the BY variables, if any, used to process the series in the DATA= data set.
ID variables	enable the series observations to be identified using further information. The ID variables included in this data set match the ID variables, if any, specified in the ID statement and input from the DATA= data set.
DATE variable	is the time ID variable used to process the time series. It is either the variable specified in the DATE= option in the PROC X13 statement or the variable _DATE_ generated by the START= option in the PROC X13 statement.
YEAR variable	contains a value for the year of the date variable for the observation. This variable is included in the OUT= data set if YEARSEAS is specified in the OUTPUT statement.
SEASON variable	contains a value for the month or quarter of the date variable for the observation. This variable is included in the OUT= data set if YEARSEAS is specified in the OUTPUT statement.
User-defined variables	are variables specified in the INPUT statement or the USERVAR= option in the REGRESSION statement. The values of these variables are copied from the DATA= data set or from the AUXDATA= data set.
EVENT variables	variables specified in the EVENT statement. The values of these variables are computed based on the event definition and the dates of the time series observations.
Table variables	contains the data from the X-13ARIMA-SEATS decomposition tables: A1, A2, A6, A7, A8, A8AO, A8LS, A8TC, A9, A10, A19, B1, B7, B13, B17, B20, C1, C17, C20, D1, D7, D8, D8BX, D8BO, D8BL, D9, D10, D10B, D10D, D11, D11A, D11F, D11R, D12, D13, D16, D16B, D18, E1, E2, E3, E5, E6, E6A, E6R, E7, E8, E18, and MV1. The variable name used in the output data set is the input variable name followed by an underscore and the corresponding table name.

SEATSDECOMP OUT= Data Set

You can use the SEATSDECOMP statement to output the component series that is computed using the SEATS method of seasonal decomposition.

The OUT= data set specified in the SEATSDECOMP statement contains the BY variables (if any), the ID variables (if any), and either the DATE= variable if the DATE= option is specified or the variable _DATE_ if the DATE= option is not specified. If user-defined regressor variables or EVENT variables are specified, they are included. In addition, the five components computed by the SEATS decomposition method are included in the OUT= data set for each series.

The SEATSDECOMP OUT= data set can contain the following variables:

BY variables	are the BY variables used to subset the series by BY groups. The BY variables included in this data set match the BY variables (if any) that are used to process the series in the DATA= data set.
ID variables	enable the series observations to be identified using further information. The ID variables included in this data set match the ID variables (if any) that are specified in the ID statement and input from the DATA= data set.
DATE variable	is the time ID variable used to process the time series. It is either the variable specified in the DATE= option in the PROC X13 statement or the variable _DATE_ that is generated by the START= option in the PROC X13 statement.
YEAR variable	contains a value for the year of the date variable for the observation. This variable is included in the OUT= data set if YEARSEAS is specified in the OUTPUT statement.
SEASON variable	contains a value for the month or quarter of the date variable for the observation. This variable is included in the OUT= data set if YEARSEAS is specified in the OUTPUT statement.
User-defined variables	are variables specified in the INPUT statement or the USERVAR= option in the REGRESSION statement. The values of these variables are copied from the DATA= data set or from the AUXDATA= data set.
EVENT variables	are variables that are specified in the EVENT statement. The values of these are computed based on the event definition and the dates of the time series observations.
Component variables	contains the data from the SEATS decomposition tables. The variable name used in the output data set is the input variable name followed by an underscore and the corresponding table name.
<variable>_OS	contains the original series for SEATS decomposition. This is the B1 series from the X-13ARIMA-SEATS method.
<variable>_SC	contains the seasonal component series that is calculated by SEATS decomposition.
<variable>_TC	contains the trend component series that is calculated by SEATS decomposition.
<variable>_SA	contains the seasonally adjusted series that is calculated by SEATS decomposition.
<variable>_IC	contains the irregular series that is calculated by SEATS decomposition.

Special Data Sets

The X13 procedure can input the MDLINFOIN= and output the MDLINFOOUT= data sets. The structure of both of these data sets is the same. The difference is that when the MDLINFOIN= data set is read, only information relative to specifying a model is processed, whereas the MDLINFOOUT= data set contains the results of estimating a model. The X13 procedure can also read data sets that contain event definition data. The structure of these data sets is the same as in the SAS® High Performance Forecasting system.

MDLINFOIN= and MDLINFOOUT= Data Sets

The MDLINFOIN= and MDLINFOOUT= data sets can contain the following variables:

BY variables enable the model information to be specified by BY groups. BY variables can be included in this data set that match the BY variables used to process the series. If no BY variables are included, then the models specified by _NAME_ in the MDLINFOIN= data set apply to all BY groups in the DATA= data set.

NAME should contain the variable name of the time series to which a particular model is to be applied. Omit the _NAME_ variable if you are specifying the same model for all series in a BY group.

MODELTYPE specifies whether the observation contains regression or ARIMA information. The value of _MODELTYPE_ should be either REG to supply regression information or ARIMA to supply model information. If valid regression information exists in the MDLINFOIN= data set for a BY group and series being processed, then the REGRESSION, INPUT, and EVENT statements are ignored for that BY group and series. Likewise, if valid ARIMA model information exists in the data set, then the AUTOMDL, ARIMA, and TRANSFORM statements are ignored. Valid values for the other variables in the data set depend on the value of the _MODELTYPE_ variable. Although other values of _MODELTYPE_ might be permitted in other SAS procedures, PROC X13 recognizes only REG and ARIMA.

MODELPART further qualifies the regression information in the observation. For _MODELTYPE_=REG, valid values of _MODELPART_ are INPUT, EVENT, and PREDEFINED. A value of INPUT indicates that this observation refers to the user-defined variable whose name is given in _DSVAR_. Likewise, a value of EVENT indicates that the observation refers to the SAS or user-defined event whose name is given in _DSVAR_. PREDEFINED indicates that the name given in _DSVAR_ is a predefined US Census Bureau variable. If only ARIMA model information is included in the data set (that is, all observations have _MODELTYPE_=ARIMA), then the _MODELPART_ variable can be omitted. For observations where _MODELTYPE_=ARIMA, valid values for _MODELPART_ are FORECAST, ".", or blank.

COMPONENT further qualifies the regression or ARIMA information in the observation. For _MODELTYPE_=REG, the only valid value of _COMPONENT_ is SCALE. For _MODELTYPE_=ARIMA, the valid values of _COMPONENT_ are TRANSFORM, CONSTANT, NONSEASONAL, and SEASONAL. TRANSFORM indicates that the observation contains the information that would be supplied in the TRANSFORM statement. CONSTANT is specified to control the constant term in the model. NONSEASONAL and SEASONAL refer to the AR, MA, and differencing terms in the ARIMA model.

PARMTYPE further qualifies the regression or ARIMA information in the observation. For _MODELTYPE_=REG, the value of _PARMTYPE_ is the same as the value of the USERTYPE=

option in the REGRESSION statement. Since the USERTYPE= option applies only to user-defined events and variables, the value of _PARMTYPE_ does not alter processing in observations where _MODELPART_=PREDEFINED. However, it is consistent to use a value for _PARMTYPE_ that matches the US Census Bureau predefined variable. For the constant term in the model information, _PARMTYPE_ should be SCALE. For transformation information, the value of _PARMTYPE_ should be NONE, LOG, LOGIT, SQRT, or BOXCOX. For _MODELTYPE_=ARIMA, valid values of _PARMTYPE_ are AR, MA, and DIF.

DSVAR specifies the variable name associated with the current observation. For _MODELTYPE_=REG, the value of _DSVAR_ is the name of the user-defined variable, the event, or the US Census Bureau predefined variable. For _MODELTYPE_=ARIMA, _DSVAR_ should match the name of the series being processed. If the ARIMA model information applies to more than one series, then _DSVAR_ can be blank or ".", equivalently.

VALUE contains a numerical value that is used as a parameter for certain types of information. For example, the PREDEFINED=EASTER(6) option in the REGESSION statement is implemented in the MDLINFOIN= data set by using _DSVAR_=EASTER and _VALUE_=6. For a BOXCOX transformation, _VALUE_ is set equal to the λ parameter value. For _COMPONENT_=SEASONAL, if _VALUE_ is nonmissing, then _VALUE_ is used as the seasonal period. If _VALUE_ is missing for _COMPONENT_=SEASONAL, then the seasonal period is determined by the interval of the series.

FACTOR applies only to the AR and MA portions of the ARIMA model. The value of _FACTOR_ identifies the factor of the given AR or MA term. Therefore, the value of _FACTOR_ is the same for all observations that are related to the same factor.

LAG identifies the degree for differencing and AR and MA lags. If _COMPONENT_=SEASONAL, then the value in _LAG_ is multiplied by the seasonal period indicated by the value of _VALUE_.

SHIFT contains the shift value for transfer functions. This value is not processed by PROC X13, but it might be processed by other procedures in which transfer functions can be specified.

NOEST indicates whether a parameter associated with the observation is to be estimated. For example, the NOINT option is indicated by _COMPONENT_=CONSTANT with _NOEST_=1 and _EST_=0. _NOEST_=1 indicates that the value in _EST_ is a fixed value. _NOEST_ pertains to the constant term, to AR and MA parameters, and to regression parameters.

EST contains an initial or fixed value for a parameter associated with the observation that is to be estimated. _NOEST_=1 indicates the value in _EST_ is a fixed value. _EST_ pertains to the constant term, to AR and MA parameters, and to regression parameters.

STDERR contains output information about estimated parameters. The variable _STDERR_ is not processed by the MDLINFOIN= data set for PROC X13. In the MDLINFOOUT= data set, _STDERR_ contains the standard error that pertains to the estimated parameter in the variable _EST_.

TVALUE contains output information about estimated parameters. The variable _TVALUE_ is not processed by the MDLINFOIN= data set for PROC X13. In the MDLINFOOUT= data set, _TVALUE_ contains the *t* value that pertains to the estimated parameter in the variable _EST_.

PVALUE contains output information about estimated parameters. The variable _PVALUE_ is not processed by the MDLINFOIN= data set for PROC X13. In the MDLINFOOUT= data set, _PVALUE_ contains the *p*-value that pertains to the estimated parameter in the variable _EST_.

INEVENT= Data Set

The INEVENT= data set can contain the following variables. When a variable is omitted from the data set, that variable is assumed to have the default value for all observations. The default values are specified in the list.

NAME specifies the event variable name. _NAME_ is displayed with the case preserved. Since _NAME_ is a SAS variable name, the event can be referenced by using any case. The _NAME_ variable is required; there is no default.

CLASS specifies the class of event: SIMPLE, COMBINATION, PREDEFINED. The default for _CLASS_ is SIMPLE.

KEYNAME contains either a date keyword (SIMPLE EVENT), a predefined event variable name (PREDEFINED EVENT), or an event name (COMBINATION EVENT). All _KEYNAME_ values are displayed in upper case. However, if the _KEYNAME_ value refers to an event name, then the actual name can be of mixed case. The default for _KEYNAME_ is no keyname, designated by ".".

STARTDATE contains either the date timing value or the first date timing value to use in a do-list. The default for _STARTDATE_ is no date, designated by a missing value.

ENDDATE contains the last date timing value to use in a do-list. The default for _ENDDATE_ is no date, designated by a missing value.

DATEINTRVL contains the interval for the date do-list. The default for _DATEINTRVL_ is no interval, designated by ".".

STARTDT contains either the datetime timing value or the first datetime timing value to use in a do-list. The default for _STARTDT_ is no datetime, designated by a missing value.

ENDDT contains the last datetime timing value to use in a do-list. The default for _ENDDT_ is no datetime, designated by a missing value.

DTINTRVL contains the interval for the datetime do-list. The default for _DTINTRVL_ is no interval, designated by ".".

STARTOBS contains either the observation number timing value or the first observation number timing value to use in a do-list. The default for _STARTOBS_ is no observation number, designated by a missing value.

ENDOBS contains the last observation number timing value to use in a do-list. The default for _ENDOBS_ is no observation number, designated by a missing value.

OBSINTRVL contains the interval length of the observation number do-list. The default for _OBSINTRVL_ is no interval, designated by ".".

TYPE specifies the type of event. The valid values of _TYPE_ are POINT, LS, RAMP, TR, TEMPRAMP, TC, LIN, LINEAR, QUAD, CUBIC, INV, INVERSE, LOG, and LOGARITHMIC. The default for _TYPE_ is POINT.

VALUE specifies the value for nonzero observation. The default for _VALUE_ is 1.0.

PULSE specifies the interval that defines the units for the duration values. The default for _PULSE_ is no interval, designated by ".".

_DUR_BEFORE_ specifies the number of durations before the timing value. The default for _DUR_BEFORE_ is 0.

_DUR_AFTER_ specifies the number of durations after the timing value. The default for _DUR_AFTER_ is 0.

_SLOPE_BEF_ determines whether the curve is GROWTH or DECAY before the timing value for _TYPE_=RAMP, _TYPE_=TEMPRAMP, and _TYPE_=TC. Valid values are GROWTH and DECAY. The default for _SLOPE_BEF_ is GROWTH.

_SLOPE_AFT_ determines whether the curve is GROWTH or DECAY after the timing value for _TYPE_=RAMP, _TYPE_=TEMPRAMP, and _TYPE_=TC. Valid values are GROWTH and DECAY. The default for _SLOPE_AFT_ is GROWTH unless _TYPE_=TC; then the default is DECAY.

SHIFT specifies the number of _PULSE_= intervals to shift the timing value. The shift can be positive (forward in time) or negative (backward in time). If _PULSE_= is not specified, then the shift is in observations. The default for _SHIFT_ is 0.

TCPARM specifies the parameter for EVENT of TYPE=TC. The default for _TCPARM_ is 0.5.

RULE specifies the rule to use when combining events or when timing values of an event overlap. The valid values of _RULE_ are ADD, MAX, MIN, MINNZ, MINMAG, and MULT. The default for _RULE_ is ADD.

PERIOD specifies the frequency interval at which the event should be repeated. If this value is missing, then the event is not periodic. The default for _PERIOD_ is no interval, designated by ".".

LABEL specifies the label or description for the event. If a label is not specified, then the default label value is displayed as ".". For events that produce dummy variables, either the user-supplied label or the default label is used. For COMPLEX events, the _LABEL_ value is merely a description of the group of events.

OUTSTAT= Data Set

The OUTSTAT= data set can contain the following variables:

BY variables sorts the statistics into BY groups. BY variables are included in this data set that match the BY variables used to process the series.

NAME specifies the variable name of the time series to which the statistics apply.

STAT describes the statistic that is stored in VALUE or CVALUE. STAT takes on the following values:

- Period — the period of the series, 4 or 12.
- Mode — the mode of the seasonal adjustment from the X11 statement. Possible values are ADD, MULT, LOGADD, and PSEUDOADD.

Start	the beginning of the model span expressed as *monyyyy* for monthly series or *yyyy*Q*q* for quarterly series.
End	the end of the model span expressed as *monyyyy* for monthly series or *yyyy*Q*q* for quarterly series.
NbFcst	the number of forecast observations.
SigmaLimLower	the lower sigma limit in standard deviation units.
SigmaLimUpper	the upper sigma limit in standard deviation units.
pLBQ_24	the Ljung-Box *Q* statistic of the residuals at lag 24, for monthly series. Note that lag 12 (pLBQ_12) and lag 16 (pLBQ_16) are included in the data set for quarterly series.
D8Fs	the stable seasonality *F* test value from Table D8.
D8Fm	the moving seasonality *F* test value from Table D8.
ISRatio	the final irregular-to-seasonal ratio from Table F 2.H.
SMA_ALL	the final seasonal moving average filter for all periods.
RSF	the residual seasonality *F* test value for Table D11 for the entire series.
RSF3	the residual seasonality *F* test value for Table D11 for the last three years.
RSFA	the residual seasonality *F* test value for Table D11.A for the entire series.
RSF3A	the residual seasonality *F* test value for Table D11.A for the last three years.
RSFR	the residual seasonality *F* test value for Table D11.R for the entire series.
RSF3R	the residual seasonality *F* test value for Table D11.R for the last three years.
TMA	the Henderson trend moving average filter selected.
ICRatio	the final irregular-to-trend cycle ratio from Table F 2.H.
E5sd	the standard deviation from Table E5.
E6sd	the standard deviation from Table E6.
E6Asd	the standard deviation from Table E6.A.
MCD	months of cyclical dominance.
Q	the overall level Q from Table F3.
Q2	Q overall level without M2 from Table F3.

FMT	indicates whether the format is numeric or character. FMT="NUM" if the value is numeric and stored in the VALUE variable. FMT="CHAR" if the value is a string and stored in the CVALUE variable.
VALUE	contains the numerical value of the statistic or missing if the statistic is of type character.
CVALUE	contains the character value of the text statistic or blank if the statistic is of type numeric.

Examples: X13 Procedure

Example 45.1: ARIMA Model Identification

This example shows typical PROC X13 statements that are used for ARIMA model identification. This example invokes the X13 procedure and uses the TRANSFORM and IDENTIFY statements. It specifies the time series data, takes the logarithm of the series (TRANSFORM statement), and generates ACFs and PACFs for the specified levels of differencing (IDENTIFY statement). The ACFs and PACFs for DIFF=1 and SDIFF=1 are shown in Output 45.1.1, Output 45.1.2, Output 45.1.3, and Output 45.1.4. The data set is the same as in the section "Basic Seasonal Adjustment" on page 3154.

The graphical displays are available when ODS Graphics is enabled. For more information about the graphics available in the X13 procedure, see the section "ODS Graphics" on page 3218.

```
proc x13 data=sales date=date;
   var sales;
   transform power=0;
   identify diff=(0,1) sdiff=(0,1);
run;
```

Output 45.1.1 ACFs (Nonseasonal Order=1 Seasonal Order=1)

The X13 Procedure

Autocorrelation of Regression Residuals for ARIMA Model Identification

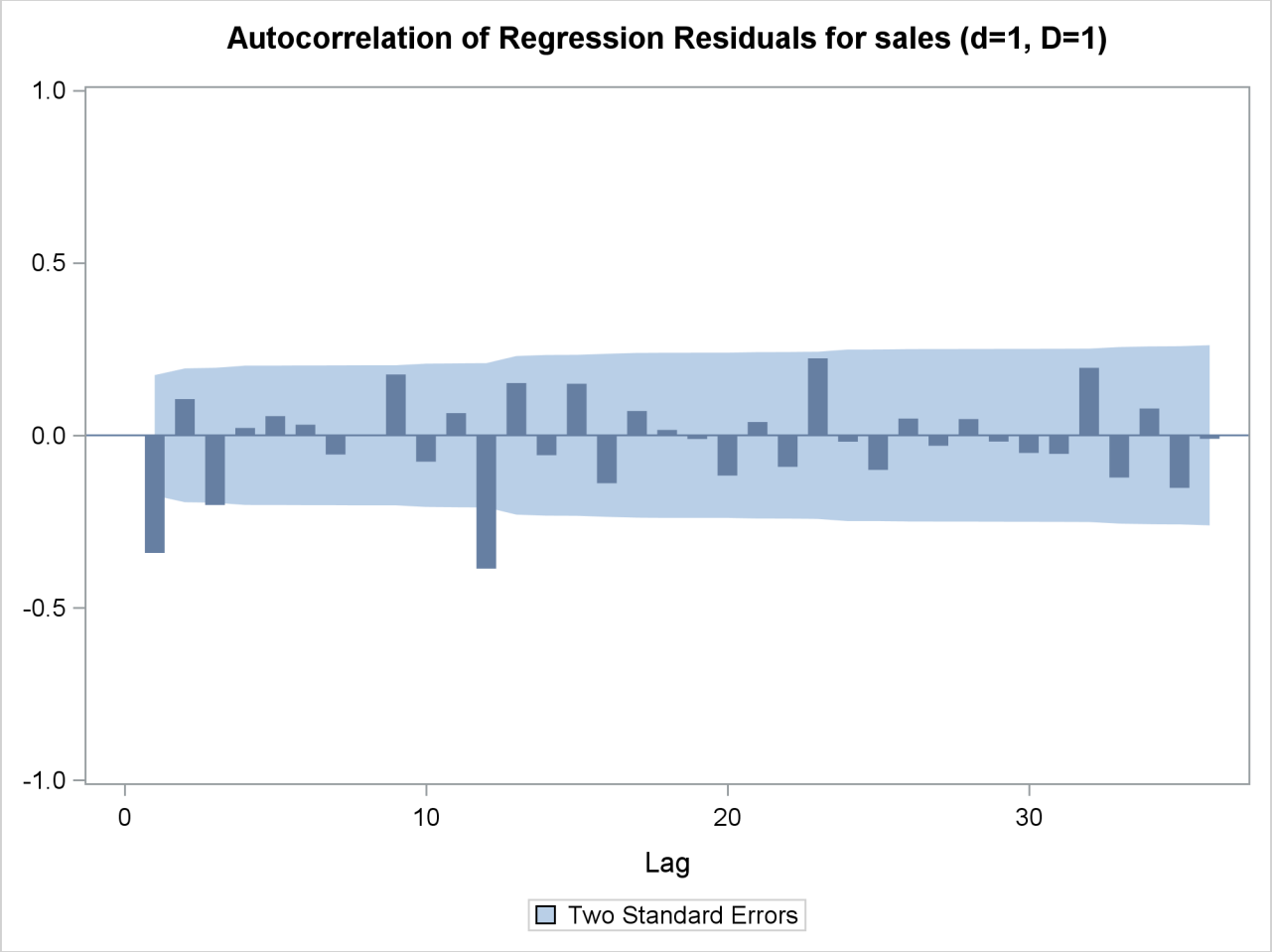
For Variable sales

Differencing: Nonseasonal Order=1 Seasonal Order=1

Lag	Correlation	Standard Error	Chi-Square	DF	Pr > ChiSq
1	-0.34112	0.08737	15.5957	1	<.0001
2	0.10505	0.09701	17.0860	2	0.0002
3	-0.20214	0.09787	22.6478	3	<.0001
4	0.02136	0.10101	22.7104	4	0.0001
5	0.05565	0.10104	23.1387	5	0.0003
6	0.03080	0.10128	23.2709	6	0.0007
7	-0.05558	0.10135	23.7050	7	0.0013
8	-0.00076	0.10158	23.7050	8	0.0026
9	0.17637	0.10158	28.1473	9	0.0009
10	-0.07636	0.10389	28.9869	10	0.0013
11	0.06438	0.10432	29.5887	11	0.0018
12	-0.38661	0.10462	51.4728	12	<.0001
13	0.15160	0.11501	54.8664	13	<.0001
14	-0.05761	0.11653	55.3605	14	<.0001
15	0.14957	0.11674	58.7204	15	<.0001
16	-0.13894	0.11820	61.6452	16	<.0001
17	0.07048	0.11944	62.4045	17	<.0001
18	0.01563	0.11975	62.4421	18	<.0001
19	-0.01061	0.11977	62.4596	19	<.0001
20	-0.11673	0.11978	64.5984	20	<.0001
21	0.03855	0.12064	64.8338	21	<.0001
22	-0.09136	0.12074	66.1681	22	<.0001
23	0.22327	0.12126	74.2099	23	<.0001
24	-0.01842	0.12436	74.2652	24	<.0001
25	-0.10029	0.12438	75.9183	25	<.0001
26	0.04857	0.12500	76.3097	26	<.0001
27	-0.03024	0.12514	76.4629	27	<.0001
28	0.04713	0.12520	76.8387	28	<.0001
29	-0.01803	0.12533	76.8943	29	<.0001
30	-0.05107	0.12535	77.3442	30	<.0001
31	-0.05377	0.12551	77.8478	31	<.0001
32	0.19573	0.12569	84.5900	32	<.0001
33	-0.12242	0.12799	87.2543	33	<.0001
34	0.07775	0.12888	88.3401	34	<.0001
35	-0.15245	0.12924	92.5584	35	<.0001
36	-0.01000	0.13061	92.5767	36	<.0001

Note: The P-values approximate the probability of observing a Chi-Square at least this large when the model fitted is correct. When DF is positive, small values of P, customarily those below 0.05, indicate model inadequacy.

Output 45.1.2 Plot for ACFs (Nonseasonal Order=1 Seasonal Order=1)



Output 45.1.3 PACFs (Nonseasonal Order=1 Seasonal Order=1)

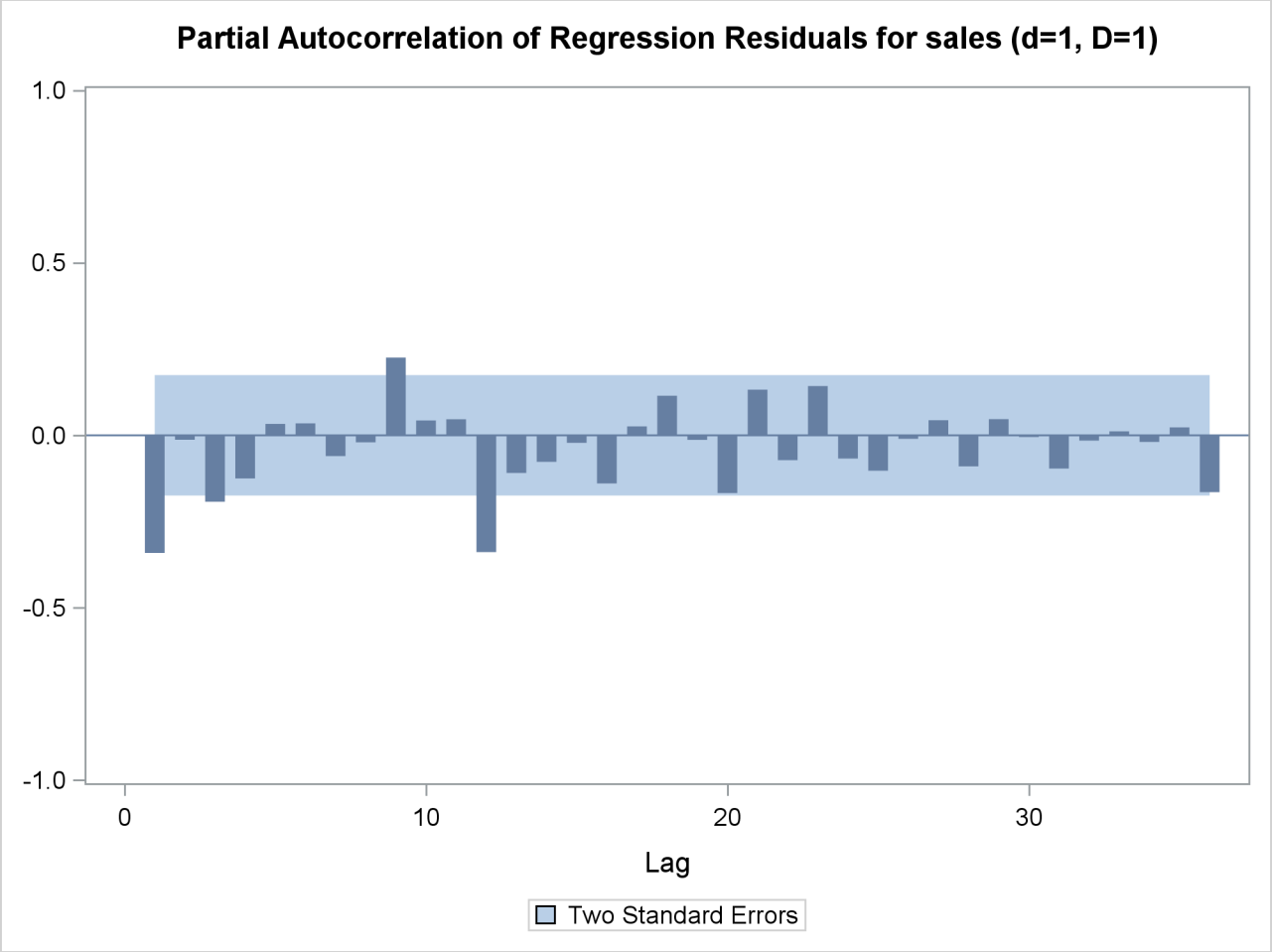
Partial Autocorrelations of Regression Residuals for ARIMA Model Identification

For Variable sales

Differencing: Nonseasonal Order=1 Seasonal Order=1

Lag	Correlation	Standard Error
1	-0.34112	0.08737
2	-0.01281	0.08737
3	-0.19266	0.08737
4	-0.12503	0.08737
5	0.03309	0.08737
6	0.03468	0.08737
7	-0.06019	0.08737
8	-0.02022	0.08737
9	0.22558	0.08737
10	0.04307	0.08737
11	0.04659	0.08737
12	-0.33869	0.08737
13	-0.10918	0.08737
14	-0.07684	0.08737
15	-0.02175	0.08737
16	-0.13955	0.08737
17	0.02589	0.08737
18	0.11482	0.08737
19	-0.01316	0.08737
20	-0.16743	0.08737
21	0.13240	0.08737
22	-0.07204	0.08737
23	0.14285	0.08737
24	-0.06733	0.08737
25	-0.10267	0.08737
26	-0.01007	0.08737
27	0.04378	0.08737
28	-0.08995	0.08737
29	0.04690	0.08737
30	-0.00490	0.08737
31	-0.09638	0.08737
32	-0.01528	0.08737
33	0.01150	0.08737
34	-0.01916	0.08737
35	0.02303	0.08737
36	-0.16488	0.08737

Output 45.1.4 Plot for PACFs (Nonseasonal Order=1 Seasonal Order=1)



Example 45.2: Model Estimation

After studying the output from Example 45.1 and identifying the ARIMA part of the model as, for example, (0 1 1)(0 1 1) 12, you can replace the IDENTIFY statement with the ARIMA and ESTIMATE statements as follows:

```
proc x13 data=sales date=date;
   var sales;
   transform power=0;
   arima model=( (0,1,1)(0,1,1) );
   estimate;
run ;
```

The parameter estimates and estimation summary statistics are shown in Output 45.2.1.

Output 45.2.1 Estimation Data

The X13 Procedure

Exact ARMA Likelihood Estimation Iteration Tolerances	
For Variable sales	
Maximum Total ARMA Iterations	1500
Convergence Tolerance	1.0E-05

Average absolute percentage error in within-sample forecasts:	
For Variable sales	
Last year:	2.81
Last-1 year:	6.38
Last-2 year:	7.69
Last three years:	5.63

Exact ARMA Likelihood Estimation Iteration Summary	
For Variable sales	
Number of ARMA iterations	6
Number of Function Evaluations	19

Exact ARMA Maximum Likelihood Estimation

For Variable sales

Parameter	Lag	Estimate	Standard Error	t Value	Pr > |t|
Nonseasonal MA	1	0.40181	0.07887	5.09	<.0001
Seasonal MA	12	0.55695	0.07626	7.30	<.0001

Estimation Summary	
For Variable sales	
Number of Observations	144
Number of Residuals	131
Number of Parameters Estimated	3
Variance Estimate	1.3E-03
Standard Error Estimate	3.7E-02
Standard Error of Variance	1.7E-04
Log likelihood	244.6965
Transformation Adjustment	-735.2943
Adjusted Log likelihood	-490.5978
AIC	987.1956
AICC (F-corrected-AIC)	987.3845
Hannan Quinn	990.7005
BIC	995.8211

Example 45.3: Seasonal Adjustment

Assuming that the model in Example 45.2 is satisfactory, a seasonal adjustment that uses forecast extension can be performed by adding the X11 statement to the procedure. By default, the data is forecast one year ahead at the end of the series.

```
ods output D8A#1=SalesD8A_1;
ods output D8A#2=SalesD8A_2;
ods output D8A#3=SalesD8A_3;
ods output D8A#4=SalesD8A_4;
proc x13 data=sales date=date;
   var sales;
   transform power=0;
   arima model=( (0,1,1)(0,1,1) );
   estimate;
   x11;
run;

title 'Stable Seasonality Test';
proc print data=SalesD8A_1 LABEL;
run;

title 'Nonparametric Stable Seasonality Test';
proc print data=SalesD8A_2 LABEL;
run;

title 'Moving Seasonality Test';
proc print data=SalesD8A_3 LABEL;
run;

title 'Combined Seasonality Test';
proc print data=SalesD8A_4 LABEL NOOBS;
   var _NAME_ Name1 Label1 cValue1;
run;
```

Table D8A, which contains the seasonality tests, is shown in Output 45.3.1.

Output 45.3.1 Table D8A as Displayed

The X13 Procedure

Table D 8.A: F-tests for Seasonality For Variable sales

Test for the Presence of Seasonality Assuming Stability				
	Sum of Squares	DF	Mean Square	F-Value
Between Months	23571.41	11	2142.855	190.9544 **
Residual	1481.28	132	11.22182	
Total	25052.69	143		

** Seasonality present at the 0.1 percent level.

Output 45.3.1 *continued*

Nonparametric Test for the Presence of Seasonality Assuming Stability		
Kruskal-Wallis Statistic	DF	Probability Level
131.9546	11	.00%

Seasonality present at the one percent level.

Moving Seasonality Test				
	Sum of Squares	DF	Mean Square	F-Value
Between Years	259.2517	10	25.92517	3.370317 **
Error	846.1424	110	7.692204	

****Moving seasonality present at the one percent level.**

Summary of Results and Combined Test for the Presence of Identifiable Seasonality	
Seasonality Tests:	Probability Level
Stable Seasonality F-test	0.000
Moving Seasonality F-test	0.001
Kruskal-Wallis Chi-square Test	0.000
Combined Measures:	Value
T1 = 7/F_Stable	0.04
T2 = 3*F_Moving/F_Stable	0.05
T = (T1 + T2)/2	0.04
Combined Test of Identifiable Seasonality:	Present

The four ODS statements in the preceding example direct output from the D8A tables into four data sets: SalesD8A_1, SalesD8A_2, SalesD8A_3, and SalesD8A_4. It is best to direct the output to four different data sets because the four tables associated with Table D8A have varying formats. The ODS data sets are shown in Output 45.3.2, Output 45.3.3, Output 45.3.4, and Output 45.3.5.

Output 45.3.2 Table D8A Output in Data Set SalesD8A_1

Stable Seasonality Test

Obs	_NAME_	FT_SRC	Sum of Squares	DF	Mean Square	F-Value	FT_AST
1	sales	Between Months	23571.41	11	2142.855	190.9544	**
2	sales	Residual	1481.28	132	11.22182	.	
3	sales	Total	25052.69	143	.	.	

Output 45.3.3 Table D8A Output in Data Set SalesD8A_2

Nonparametric Stable Seasonality Test

Obs	_NAME_	Kruskal-Wallis Statistic	DF	Probability Level
1	sales	131.9546	11	.00%

Output 45.3.4 Table D8A Output in Data Set SalesD8A_3

Moving Seasonality Test

Obs	_NAME_	FT_SRC	Sum of Squares	DF	Mean Square	F-Value	FT_AST
1	sales	Between Years	259.2517	10	25.92517	3.370317	**
2	sales	Error	846.1424	110	7.692204		.

Output 45.3.5 Table D8A Output in Data Set SalesD8A_4

Combined Seasonality Test

NAME	Name1	Label1	cValue1
sales		Seasonality Tests:	Probability Level
sales			
sales	P_STABLE	Stable Seasonality F-test	0.000
sales	P_MOV	Moving Seasonality F-test	0.001
sales	P_KW	Kruskal-Wallis Chi-square Test	0.000
sales			
sales		Combined Measures:	Value
sales			
sales	T1	T1 = 7/F_Stable	0.04
sales	T2	T2 = 3*F_Moving/F_Stable	0.05
sales	T	T = (T1 + T2)/2	0.04
sales			
sales	IDSeasTest	Combined Test of Identifiable Seasonality:	Present

Example 45.4: RegARIMA Automatic Model Selection

This example demonstrates regARIMA modeling and TRAMO-based automatic model selection, which is available with the AUTOMDL statement. ODS SELECT statements are used to limit the displayed output to the model selection and estimation stages. The same data set is used as in the previous examples.

```
title 'TRAMO Automatic Model Identification';
ods select UnitRootTestModel
           UnitRootTest
           AutoChoiceModel
           Best5Model
           AutomaticModelChoice
           InitialModelChoice
           FinalModelChecks
           FinalModelChoice
           AutomdlNote;
proc x13 data=sales date=date;
   var sales;
   transform function=log;
   regression predefined=td;
   automdl maxorder=(1,1)
           print=all;
   estimate;
   x11;
   output out=out a1 a2 a6 b1 c17 c20 d1 d7 d8 d9 d10
                  d11 d12 d13 d16 d18;
run;

proc print data=out(obs=21);
   title 'Output Variables Related to Trading Day Regression';
run;
```

The automatic model selection output is shown in Output 45.4.1, Output 45.4.2, and Output 45.4.3. The first table, "ARIMA Estimate for Unit Root Identification" in Output 45.4.1, gives details of the method that TRAMO uses to automatically select the orders of differencing. The second table, "Results of Unit Root Test for Identifying Orders of Differencing" in Output 45.4.1, shows that a regular difference order of 1 and a seasonal difference order of 1 has been determined by TRAMO. The third table, "Models Estimated by Automatic ARIMA Model Selection Procedure" in Output 45.4.2, shows all the models examined by the TRAMO-based method. The fourth table, "Best Five ARIMA Models Chosen by Automatic Modeling" in Output 45.4.3, shows the top five models in order of rank and their BIC2 statistic. The fifth table, "Comparison of Automatically Selected Model and Default Model" in Output 45.4.3, compares the model selected by the TRAMO model to the default regARIMA model of the X-13ARIMA-SEATS method. The sixth table, "Initial Automatic Model Selection" in Output 45.4.3, shows which model was selected between the two models that are compared in the table "Comparison of Automatically Selected Model and Default Model." (When available, the table "Check of the Residual Ljung-Box Q Statistic" in Output 45.4.3 contains additional information about the initial model choice.) The seventh table, "Final Checks for Identified Model" in Output 45.4.3, displays the results of the final model checks for model adequacy. The eighth table, "Final Automatic Model Selection" in Output 45.4.3, shows which model was actually selected.

Output 45.4.1 Output from the AUTOMDL Statement

TRAMO Automatic Model Identification

The X13 Procedure

ARIMA Estimates for Unit Root Identification

For Variable sales

Model Number	Estimation Method	Estimated Model	Parameter	ARMA Estimate
1	H-R	(2, 0, 0) (1, 0, 0)	NS_AR_1	0.67540
	H-R	(2, 0, 0) (1, 0, 0)	NS_AR_2	0.28425
	H-R	(2, 0, 0) (1, 0, 0)	S_AR_12	0.91963
2	H-R	(1, 1, 1) (1, 0, 1)	NS_AR_1	0.13418
	H-R	(1, 1, 1) (1, 0, 1)	S_AR_12	0.98500
	H-R	(1, 1, 1) (1, 0, 1)	NS_MA_1	0.47884
	H-R	(1, 1, 1) (1, 0, 1)	S_MA_12	0.51726
3	H-R	(1, 1, 1) (1, 1, 1)	NS_AR_1	-0.39269
	H-R	(1, 1, 1) (1, 1, 1)	S_AR_12	0.06223
	H-R	(1, 1, 1) (1, 1, 1)	NS_MA_1	-0.09570
	H-R	(1, 1, 1) (1, 1, 1)	S_MA_12	0.58536

Results of Unit Root Test for Identifying Orders of Differencing

For Variable sales

Regular difference order	Seasonal difference order	Mean Significant
1	1	no

Output 45.4.2 Output from the AUTOMDL Statement

Models estimated by Automatic ARIMA Model Selection procedure

For Variable sales

		ARMA		Statistics of Fit	
Model Number	**Estimated Model**	**Parameter**	**Estimate**	**BIC**	**BIC2**
1	**(3, 1, 0) (0, 1, 0)**	**NS_AR_1**	-0.33524		
	(3, 1, 0) (0, 1, 0)	**NS_AR_2**	-0.05558		
	(3, 1, 0) (0, 1, 0)	**NS_AR_3**	-0.15649		
	(3, 1, 0) (0, 1, 0)			1024.469	-3.40549
2	**(3, 1, 0) (0, 1, 1)**	**NS_AR_1**	-0.33186		
	(3, 1, 0) (0, 1, 1)	**NS_AR_2**	-0.05823		
	(3, 1, 0) (0, 1, 1)	**NS_AR_3**	-0.15200		
	(3, 1, 0) (0, 1, 1)	**S_MA_12**	0.55279		
	(3, 1, 0) (0, 1, 1)			993.7880	-3.63970
3	**(3, 1, 0) (1, 1, 0)**	**NS_AR_1**	-0.38673		
	(3, 1, 0) (1, 1, 0)	**NS_AR_2**	-0.08768		
	(3, 1, 0) (1, 1, 0)	**NS_AR_3**	-0.18143		
	(3, 1, 0) (1, 1, 0)	**S_AR_12**	-0.47336		
	(3, 1, 0) (1, 1, 0)			1000.224	-3.59057
4	**(3, 1, 0) (1, 1, 1)**	**NS_AR_1**	-0.34352		
	(3, 1, 0) (1, 1, 1)	**NS_AR_2**	-0.06504		
	(3, 1, 0) (1, 1, 1)	**NS_AR_3**	-0.15728		
	(3, 1, 0) (1, 1, 1)	**S_AR_12**	-0.12163		
	(3, 1, 0) (1, 1, 1)	**S_MA_12**	0.47073		
	(3, 1, 0) (1, 1, 1)			998.0548	-3.60713
5	**(0, 1, 0) (0, 1, 1)**	**S_MA_12**	0.60446		
	(0, 1, 0) (0, 1, 1)			996.8560	-3.61628
6	**(0, 1, 1) (0, 1, 1)**	**NS_MA_1**	0.36272		
	(0, 1, 1) (0, 1, 1)	**S_MA_12**	0.55599		
	(0, 1, 1) (0, 1, 1)			986.6405	-3.69426
7	**(1, 1, 0) (0, 1, 1)**	**NS_AR_1**	-0.32734		
	(1, 1, 0) (0, 1, 1)	**S_MA_12**	0.55834		
	(1, 1, 0) (0, 1, 1)			987.1500	-3.69037
8	**(1, 1, 1) (0, 1, 1)**	**NS_AR_1**	0.17833		
	(1, 1, 1) (0, 1, 1)	**NS_MA_1**	0.52867		
	(1, 1, 1) (0, 1, 1)	**S_MA_12**	0.56212		
	(1, 1, 1) (0, 1, 1)			991.2363	-3.65918
9	**(0, 1, 1) (0, 1, 0)**	**NS_MA_1**	0.36005		
	(0, 1, 1) (0, 1, 0)			1017.770	-3.45663

Output 45.4.3 Output from the AUTOMDL Statement

Best Five ARIMA Models Chosen by Automatic Modeling
For Variable sales

Rank	Estimated Model	BIC2
1	(0, 1, 1) (0, 1, 1)	-3.69426
2	(1, 1, 0) (0, 1, 1)	-3.69037
3	(1, 1, 1) (0, 1, 1)	-3.65918
4	(0, 1, 0) (0, 1, 1)	-3.61628
5	(0, 1, 1) (0, 1, 0)	-3.45663

Comparison of Automatically Selected Model and Default Model
For Variable sales

		Statistics of Fit		
Source of Candidate Models	Estimated Model	Confidence Coefficient of the Ljung-Box Q Statistic	Residual Standard Error	Number of Outliers
Automatic Model Choice	(0, 1, 1) (0, 1, 1)	0.62561	0.03546	0
Airline Model (Default)	(0, 1, 1) (0, 1, 1)	0.62561	0.03546	0

Initial Automatic Model Selection
For Variable sales

Source of Model	Estimated Model
Automatic Model Choice	(0, 1, 1) (0, 1, 1)

Final Checks for Identified Model
For Variable sales

Test	Result	Model Change
Check for Unit Roots	No unit root.	No Change
Check for Nonseasonal Overdifferencing	Nonseasonal MA not within 0.00 of 1.0 - model passes.	No Change
Check for insignificant ARMA coefficients	No insignificant ARMA coefficients found.	No Change

Final Automatic Model Selection
For Variable sales

Source of Model	Orders Altered	Estimated Model
Automatic Model Choice	No	(0, 1, 1) (0, 1, 1)

Table 45.16 and Output 45.4.4 illustrate the regARIMA modeling method. Table 45.16 shows the relationship between the output variables in PROC X13 that results from a regARIMA model. Note that some of these formulas apply only to this example. Output 45.4.4 shows the values of these variables for the first 21 observations in the example.

Table 45.16 regARIMA Output Variables and Descriptions

Table	Title	Type	Formula
A1	Time series data (for the span analyzed)	Data	Input
A2	Prior-adjustment factors leap year (from trading day regression) adjustments	Factor	Calculated from regression
A6	RegARIMA trading day component leap year prior adjustments included from Table A2	Factor	Calculated from regression
B1	Original series (prior adjusted) (adjusted for regARIMA factors)	Data	B1 = A1/A6 * * because only TD specified
C17	Final weights for irregular component	Factor	Calculated using moving standard deviation
C20	Final extreme value adjustment factors	Factor	Calculated using C16 and C17
D1	Modified original data, D iteration	Data	D1 = B1/C20 ** D1 = C19/C20 ** C19 = B1 in this example
D7	Preliminary trend cycle, D iteration	Data	Calculated using Henderson moving average
D8	Final unmodified SI ratios	Factor	D8 = B1/D7 *** D8 = C19/D7 *** TD specified in regression
D9	Final replacement values for SI ratios	Factor	If C17 shows extreme values, D9 = D1/D7; D9 = . otherwise
D10	Final seasonal factors	Factor	Calculated using moving averages
D11	Final seasonally adjusted data (also adjusted for trading day)	Data	D11 = B1/D10 **** D11 = C19/D10 **** B1 = C19 for this example
D12	Final trend cycle	Data	Calculated using Henderson moving average
D13	Final irregular component	Factor	D13 = D11/D12
D16	Combined adjustment factors (includes seasonal, trading day factors)	Factor	D16 = A1/D11
D18	Combined calendar adjustment factors (includes trading day factors)	Factor	D18 = D16/D10 D18 = A6 ***** ***** regression TD is the only calendar adjustment factor in this example

Output 45.4.4 Output Variables Related to Trading Day Regression

Output Variables Related to Trading Day Regression

Obs	DATE	sales_A1	sales_A2	sales_A6	sales_B1	sales_C17	sales_C20	sales_D1	sales_D7	sales_D8
1	SEP78	112	1.00000	1.01328	110.532	1.00000	1.00000	110.532	124.138	0.89040
2	OCT78	118	1.00000	0.99727	118.323	1.00000	1.00000	118.323	124.905	0.94731
3	NOV78	132	1.00000	0.98960	133.388	1.00000	1.00000	133.388	125.646	1.06161
4	DEC78	129	1.00000	1.00957	127.777	1.00000	1.00000	127.777	126.231	1.01225
5	JAN79	121	1.00000	0.99408	121.721	1.00000	1.00000	121.721	126.557	0.96179
6	FEB79	135	0.99115	0.99115	136.205	1.00000	1.00000	136.205	126.678	1.07521
7	MAR79	148	1.00000	1.00966	146.584	1.00000	1.00000	146.584	126.825	1.15580
8	APR79	148	1.00000	0.99279	149.075	1.00000	1.00000	149.075	127.038	1.17347
9	MAY79	136	1.00000	0.99406	136.813	1.00000	1.00000	136.813	127.433	1.07360
10	JUN79	119	1.00000	1.01328	117.440	1.00000	1.00000	117.440	127.900	0.91822
11	JUL79	104	1.00000	0.99727	104.285	1.00000	1.00000	104.285	128.499	0.81156
12	AUG79	118	1.00000	0.99678	118.381	1.00000	1.00000	118.381	129.253	0.91589
13	SEP79	115	1.00000	1.00229	114.737	0.98630	0.99964	114.778	130.160	0.88151
14	OCT79	126	1.00000	0.99408	126.751	0.88092	1.00320	126.346	131.238	0.96581
15	NOV79	141	1.00000	1.00366	140.486	1.00000	1.00000	140.486	132.699	1.05869
16	DEC79	135	1.00000	0.99872	135.173	1.00000	1.00000	135.173	134.595	1.00429
17	JAN80	125	1.00000	0.99406	125.747	0.00000	0.95084	132.248	136.820	0.91906
18	FEB80	149	1.02655	1.03400	144.100	1.00000	1.00000	144.100	139.215	1.03509
19	MAR80	170	1.00000	0.99872	170.217	1.00000	1.00000	170.217	141.559	1.20245
20	APR80	170	1.00000	0.99763	170.404	1.00000	1.00000	170.404	143.777	1.18520
21	MAY80	158	1.00000	1.00966	156.489	1.00000	1.00000	156.489	145.925	1.07239

Obs	sales_D9	sales_D10	sales_D11	sales_D12	sales_D13	sales_D16	sales_D18
1	.	0.90264	122.453	124.448	0.98398	0.91463	1.01328
2	.	0.94328	125.438	125.115	1.00258	0.94070	0.99727
3	.	1.06320	125.459	125.723	0.99790	1.05214	0.98960
4	.	0.99534	128.375	126.205	1.01720	1.00487	1.00957
5	.	0.97312	125.083	126.479	0.98896	0.96735	0.99408
6	.	1.05931	128.579	126.587	1.01574	1.04994	0.99115
7	.	1.17842	124.391	126.723	0.98160	1.18980	1.00966
8	.	1.18283	126.033	126.902	0.99315	1.17430	0.99279
9	.	1.06125	128.916	127.257	1.01303	1.05495	0.99406
10	.	0.91663	128.121	127.747	1.00293	0.92881	1.01328
11	.	0.81329	128.226	128.421	0.99848	0.81107	0.99727
12	.	0.91135	129.897	129.316	1.00449	0.90841	0.99678
13	0.88182	0.90514	126.761	130.347	0.97249	0.90722	1.00229
14	0.96273	0.93820	135.100	131.507	1.02732	0.93264	0.99408
15	.	1.06183	132.306	132.937	0.99525	1.06571	1.00366
16	.	0.99339	136.072	134.720	1.01004	0.99212	0.99872
17	0.96658	0.97481	128.996	136.763	0.94321	0.96902	0.99406
18	.	1.06153	135.748	138.996	0.97663	1.09762	1.03400
19	.	1.17965	144.295	141.221	1.02177	1.17814	0.99872
20	.	1.18499	143.802	143.397	1.00283	1.18218	0.99763
21	.	1.06005	147.624	145.591	1.01397	1.07028	1.00966

Example 45.5: Automatic Outlier Detection

This example demonstrates the use of the OUTLIER statement to automatically detect and remove outliers from a time series to be seasonally adjusted. The data set is the same as in the section "Basic Seasonal Adjustment" on page 3154 and the previous examples. Adding the OUTLIER statement to Example 45.3 requests that outliers be detected by using the default critical value as described in the section "OUTLIER Statement" on page 3183. The tables associated with outlier detection for this example are shown in Output 45.5.1. The first table shows the critical values; the second table shows that a single potential outlier was identified; the third table shows the estimates for the ARMA parameters. Since no outliers are included in the regression model, the "Regression Model Parameter Estimates" table is not displayed. Because only a potential outlier was identified, and not an actual outlier, in this case the A1 series and the B1 series are identical.

```
title 'Automatic Outlier Identification';
proc x13 data=sales date=date;
   var sales;
   transform function=log;
   arima model=( (0,1,1)(0,1,1) );
   outlier;
   estimate;
   x11;
   output out=nooutlier a1 b1 d10;
run ;
```

Output 45.5.1 PROC X13 Output When Potential Outliers Are Identified

Automatic Outlier Identification

The X13 Procedure

Critical Values to use in Outlier Detection For Variable sales	
Begin	SEP1978
End	AUG1990
Observations	144
Method	Add One
AO Critical Value	3.889838
LS Critical Value	3.889838

Note: The following time series values might later be identified as outliers when data are added or revised. They were not identified as outliers in this run either because their test t-statistics were slightly below the critical value or because they were eliminated during the backward deletion step of the identification procedure, when a non-robust t-statistic is used.

Potential Outliers

For Variable sales

Type of Outlier	Date	t Value for AO	t Value for LS
AO	**NOV1989**	-3.48	-1.51

Output 45.5.1 *continued*

Exact ARMA Maximum Likelihood Estimation

For Variable sales

Parameter	Lag	Estimate	Standard Error	t Value	Pr > |t|
Nonseasonal MA	**1**	0.40181	0.07887	5.09	<.0001
Seasonal MA	**12**	0.55695	0.07626	7.30	<.0001

In the next example, reducing the critical value to 3.3 causes the outlier identification routine to more aggressively identify outliers as shown in Output 45.5.2. The first table shows the critical values. The second table shows that three additive outliers and a level-shift have been included in the regression model. The third table shows how the inclusion of outliers in the model affects the ARMA parameters.

```
proc x13 data=sales date=date;
   var sales;
   transform function=log;
   arima model=((0,1,1) (0,1,1));
   outlier cv=3.3;
   estimate;
   x11;
   output out=outlier a1 a8  a8ao a8ls b1 d10;
run;

proc print data=outlier(obs=45);
run;
```

Output 45.5.2 PROC X13 Output When Outliers Are Identified

Automatic Outlier Identification

The X13 Procedure

Critical Values to use in Outlier Detection For Variable sales	
Begin	SEP1978
End	AUG1990
Observations	144
Method	Add One
AO Critical Value	3.3
LS Critical Value	3.3

Regression Model Parameter Estimates

For Variable sales

Type	Parameter	NoEst	Estimate	Standard Error	t Value	Pr > |t|
Automatically Identified	AO JAN1981	Est	0.09590	0.02168	4.42	<.0001
	LS FEB1983	Est	-0.09673	0.02488	-3.89	0.0002
	AO OCT1983	Est	-0.08032	0.02146	-3.74	0.0003
	AO NOV1989	Est	-0.10323	0.02480	-4.16	<.0001

Exact ARMA Maximum Likelihood Estimation

For Variable sales

Parameter	Lag	Estimate	Standard Error	t Value	Pr > |t|
Nonseasonal MA	**1**	0.33205	0.08239	4.03	<.0001
Seasonal MA	**12**	0.49647	0.07676	6.47	<.0001

The first 45 observations of the A1, A8, A8AO, A8LS, B1, and D10 series are displayed in Output 45.5.3. You can confirm the following relationships from the data:

$$A8 = A8AO \times A8LS$$

$$B1 = A1/A8$$

The seasonal factors are stored in the variable sales_D10.

Output 45.5.3 PROC X13 Output Series Related to Outlier Detection

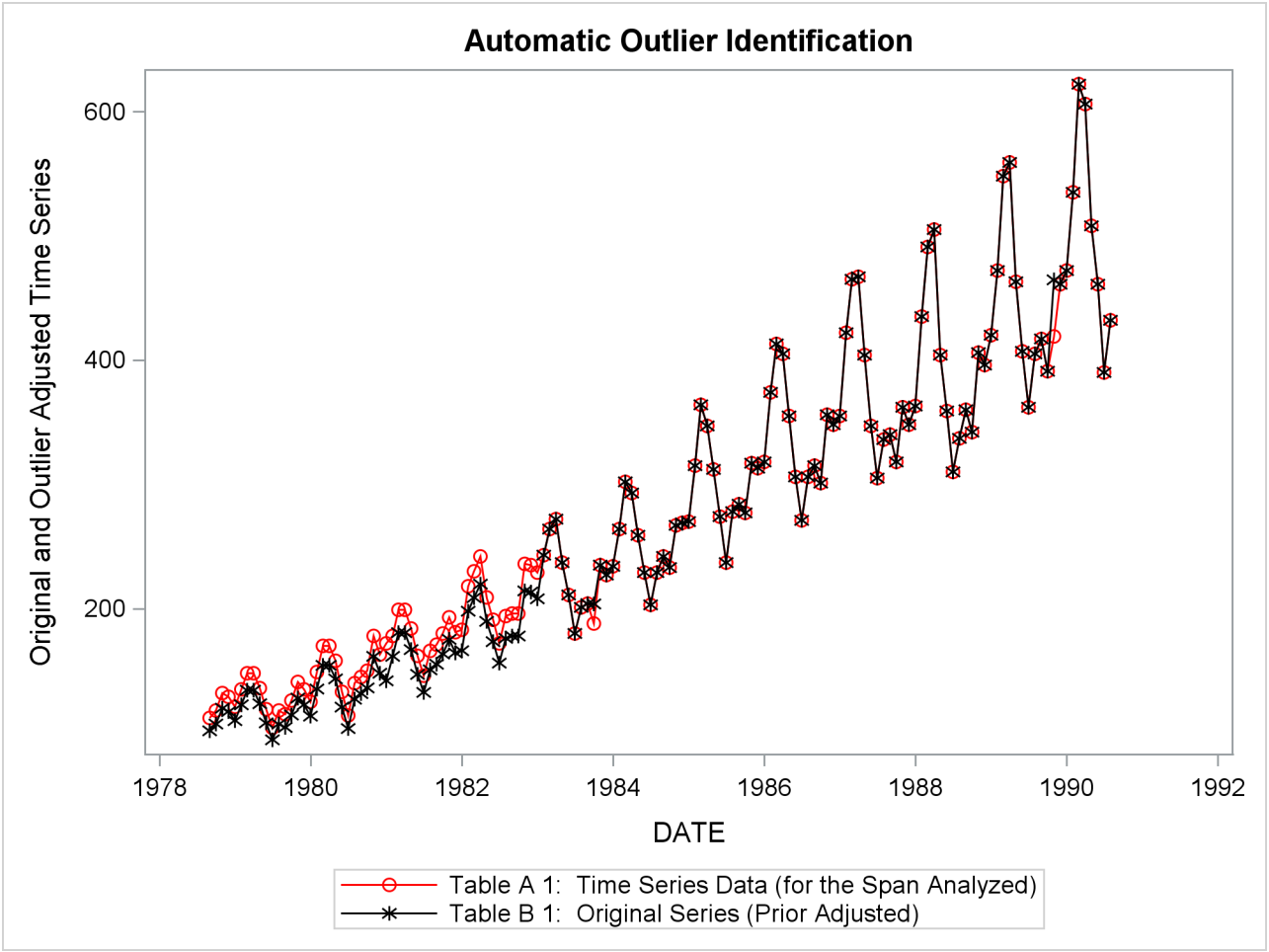
Automatic Outlier Identification

Obs	DATE	sales_A1	sales_A8	sales_A8AO	sales_A8LS	sales_B1	sales_D10
1	SEP78	112	1.10156	1.00000	1.10156	101.674	0.90496
2	OCT78	118	1.10156	1.00000	1.10156	107.121	0.94487
3	NOV78	132	1.10156	1.00000	1.10156	119.830	1.04711
4	DEC78	129	1.10156	1.00000	1.10156	117.107	1.00119
5	JAN79	121	1.10156	1.00000	1.10156	109.844	0.94833
6	FEB79	135	1.10156	1.00000	1.10156	122.553	1.06817
7	MAR79	148	1.10156	1.00000	1.10156	134.355	1.18679
8	APR79	148	1.10156	1.00000	1.10156	134.355	1.17607
9	MAY79	136	1.10156	1.00000	1.10156	123.461	1.07565
10	JUN79	119	1.10156	1.00000	1.10156	108.029	0.91844
11	JUL79	104	1.10156	1.00000	1.10156	94.412	0.81206
12	AUG79	118	1.10156	1.00000	1.10156	107.121	0.91602
13	SEP79	115	1.10156	1.00000	1.10156	104.397	0.90865
14	OCT79	126	1.10156	1.00000	1.10156	114.383	0.94131
15	NOV79	141	1.10156	1.00000	1.10156	128.000	1.04496
16	DEC79	135	1.10156	1.00000	1.10156	122.553	0.99766
17	JAN80	125	1.10156	1.00000	1.10156	113.475	0.94942
18	FEB80	149	1.10156	1.00000	1.10156	135.263	1.07172
19	MAR80	170	1.10156	1.00000	1.10156	154.327	1.18663
20	APR80	170	1.10156	1.00000	1.10156	154.327	1.18105
21	MAY80	158	1.10156	1.00000	1.10156	143.433	1.07383
22	JUN80	133	1.10156	1.00000	1.10156	120.738	0.91930
23	JUL80	114	1.10156	1.00000	1.10156	103.490	0.81385
24	AUG80	140	1.10156	1.00000	1.10156	127.093	0.91466
25	SEP80	145	1.10156	1.00000	1.10156	131.632	0.91302
26	OCT80	150	1.10156	1.00000	1.10156	136.171	0.93086
27	NOV80	178	1.10156	1.00000	1.10156	161.589	1.03965
28	DEC80	163	1.10156	1.00000	1.10156	147.972	0.99440
29	JAN81	172	1.21243	1.10065	1.10156	141.864	0.95136
30	FEB81	178	1.10156	1.00000	1.10156	161.589	1.07981
31	MAR81	199	1.10156	1.00000	1.10156	180.653	1.18661
32	APR81	199	1.10156	1.00000	1.10156	180.653	1.19097
33	MAY81	184	1.10156	1.00000	1.10156	167.036	1.06905
34	JUN81	162	1.10156	1.00000	1.10156	147.064	0.92446
35	JUL81	146	1.10156	1.00000	1.10156	132.539	0.81517
36	AUG81	166	1.10156	1.00000	1.10156	150.695	0.91148
37	SEP81	171	1.10156	1.00000	1.10156	155.234	0.91352
38	OCT81	180	1.10156	1.00000	1.10156	163.405	0.91632
39	NOV81	193	1.10156	1.00000	1.10156	175.206	1.03194
40	DEC81	181	1.10156	1.00000	1.10156	164.312	0.98879
41	JAN82	183	1.10156	1.00000	1.10156	166.128	0.95699
42	FEB82	218	1.10156	1.00000	1.10156	197.901	1.09125
43	MAR82	230	1.10156	1.00000	1.10156	208.795	1.19059
44	APR82	242	1.10156	1.00000	1.10156	219.688	1.20448
45	MAY82	209	1.10156	1.00000	1.10156	189.731	1.06355

From the two previous examples, you can examine how outlier detection affects the seasonally adjusted series. Output 45.5.4 shows a plot of A1 versus B1 in the series where outliers are detected. B1 has been adjusted for the additive outliers and the level-shift.

```
proc sgplot data=outlier;
   series x=date y=sales_A1 / name='A1' markers
                              markerattrs=(color=red symbol='circle')
                              lineattrs=(color=red);
   series x=date y=sales_B1 / name='B1' markers
                              markerattrs=(color=black symbol='asterisk')
                              lineattrs=(color=black);
   yaxis label='Original and Outlier Adjusted Time Series';
run;
```

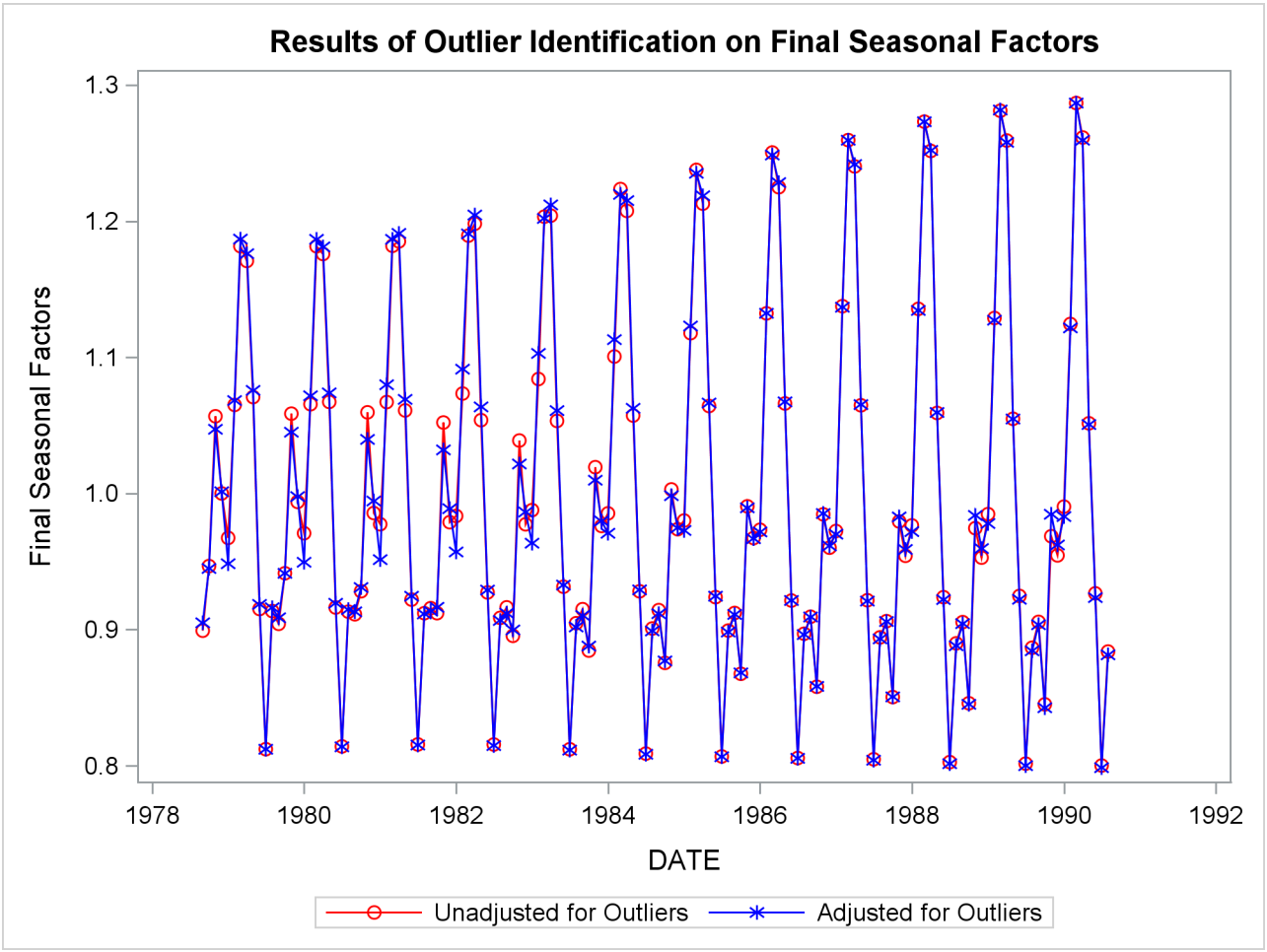
Output 45.5.4 Original Series and Outlier Adjusted Series



Output 45.5.5 compares the seasonal factors (table D10) of the series unadjusted for outliers to the series adjusted for outliers. The seasonal factors are based on the B1 series.

```
data both;
   merge nooutlier(rename=(sales_D10=unadj_D10)) outlier;
run;

title 'Results of Outlier Identification on Final Seasonal Factors';
proc sgplot data=both;
   series x=date y=unadj_D10 / name='unadjusted' markers
                               markerattrs=(color=red symbol='circle')
                               lineattrs=(color=red)
                               legendlabel='Unadjusted for Outliers';
   series x=date y=sales_D10 / name='adjusted' markers
                               markerattrs=(color=blue symbol='asterisk')
                               lineattrs=(color=blue)
                               legendlabel='Adjusted for Outliers';
   yaxis label='Final Seasonal Factors';
run;
```

Output 45.5.5 Seasonal Factors Based on Original and Outlier Adjusted Series



Example 45.6: User-Defined Regressors

This example demonstrates the use of the USERVAR= option in the REGRESSION statement to include user-defined regressors in the regARIMA model. The user-defined regressors must be defined as nonmissing values for the span of the series being modeled plus any backcast or forecast values. Suppose you have the data set SALESDATA with 132 monthly observations beginning in January of 1949.

```
title 'Data Set to be Seasonally Adjusted';
data salesdata;
   set sashelp.air(obs=132);
run;
```

Because the regARIMA model forecasts one year ahead, you must define the regressor for 144 observations that start in January of 1949. You can construct a simple length-of-month regressor by using the following DATA step:

```
title 'User-defined Regressor for Data to be Seasonally Adjusted';
data regressors(keep=date LengthOfMonth);
   set sashelp.air;
   LengthOfMonth = INTNX('MONTH',date,1) - date;
run;
```

In this example, the two data sets are merged to use them as input to PROC X13. You can also use the AUXDATA= data set to input user-defined regressors. See Example 45.11 for more information. The BY statement is used to align the regressors with the time series by the time ID variable DATE.

```
title 'Data Set Containing Series and Regressors';
data datain;
   merge regressors salesdata;
   by date;
run;

proc print data=datain(firstobs=121);
run;
```

The last 24 observations of the input data set are displayed in Output 45.6.1. The regressor variable is defined for one year (12 observations) beyond the span of the time series to be seasonally adjusted.

Output 45.6.1 PROC X13 Input Data Set with User-Defined Regressor

Data Set Containing Series and Regressors

Obs	DATE	LengthOfMonth	AIR
121	JAN59	31	360
122	FEB59	28	342
123	MAR59	31	406
124	APR59	30	396
125	MAY59	31	420
126	JUN59	30	472
127	JUL59	31	548
128	AUG59	31	559
129	SEP59	30	463
130	OCT59	31	407
131	NOV59	30	362
132	DEC59	31	405
133	JAN60	31	.
134	FEB60	29	.
135	MAR60	31	.
136	APR60	30	.
137	MAY60	31	.
138	JUN60	30	.
139	JUL60	31	.
140	AUG60	31	.
141	SEP60	30	.
142	OCT60	31	.
143	NOV60	30	.
144	DEC60	31	.

The DATAIN data set is now ready to be used as input to PROC X13. The DATE= variable and the user-defined regressors are automatically excluded from the variables to be seasonally adjusted.

```
title 'regARIMA Model with User-defined Regressor';
proc x13 data=datain date=DATE interval=MONTH plots=none;
   transform function=log;
   regression uservar=LengthOfMonth / usertype=lom;
   automdl;
   x11;
   output out=out a1 d11;
run;
```

The parameter estimates for the regARIMA model are shown in Output 45.6.2

Output 45.6.2 PROC X13 Output for User-Defined Regression Parameter

regARIMA Model with User-defined Regressor

The X13 Procedure

Regression Model Parameter Estimates

For Variable AIR

Type	Parameter	NoEst	Estimate	Standard Error	t Value	Pr > |t|
User Defined	LengthOfMonth	Est	0.04683	0.01834	2.55	0.0119

Exact ARMA Maximum Likelihood Estimation

For Variable AIR

Parameter	Lag	Estimate	Standard Error	t Value	Pr > |t|
Nonseasonal MA	**1**	0.33678	0.08506	3.96	0.0001
Seasonal MA	**12**	0.54078	0.07726	7.00	<.0001

Another way to include user-defined regressors in the regARIMA model is to specify the SPAN= option in the PROC X13 statement. The following user-defined regressor is similar to the one defined previously. However, this length-of-month regressor is mean adjusted. Using a zero-mean regressor prevents the regressor from altering the level of the series. In this instance, the series to be seasonally adjusted, AIR, and the regression variable, LengthOfMonth, have nonmissing observations at all time periods in the data set DATAIN.

```
title 'User-defined Regressor for Data to be Seasonally Adjusted, Mean Adjusted';
data datain(keep=date AIR LengthOfMonth);
   set sashelp.air;
   LengthOfMonth = INTNX('MONTH',date,1) - date - 30.4375;
run;
```

Because the default forecast period is one year ahead, the span of the series must be limited to one year before the end of the regression variable definition to forecast using the regression variable LengthOfMonth,

```
title 'regARIMA Model with Zero-Mean User-defined Regressor';
proc x13 data=datain date=DATE interval=MONTH span=(,DEC1959) plots=none;
   transform function=log;
   regression uservar=LengthOfMonth / usertype=lom;
   automdl;
   x11;
   output out=outzm a1 d11;
run;
```

The parameter estimates for the regARIMA model that are estimated using a zero-mean regressor are shown in Output 45.6.3

Output 45.6.3 PROC X13 Output for Zero-Mean User-Defined Regression Parameter

regARIMA Model with Zero-Mean User-defined Regressor

The X13 Procedure

Regression Model Parameter Estimates

For Variable AIR

Type	Parameter	NoEst	Estimate	Standard Error	t Value	Pr > |t|
User Defined	LengthOfMonth	Est	0.04683	0.01834	2.55	0.0119

Exact ARMA Maximum Likelihood Estimation

For Variable AIR

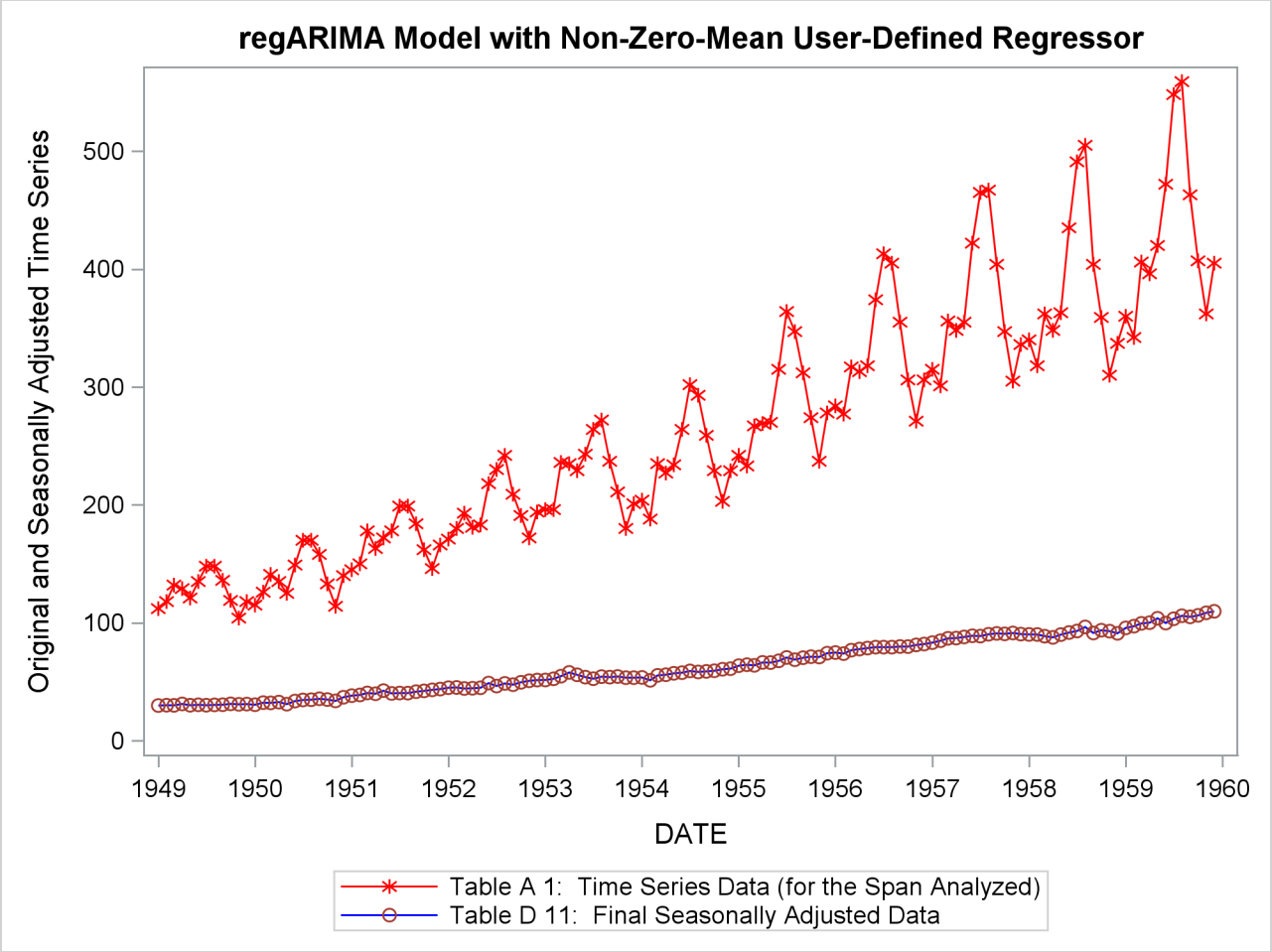
Parameter	Lag	Estimate	Standard Error	t Value	Pr > |t|
Nonseasonal MA	**1**	0.33678	0.08506	3.96	0.0001
Seasonal MA	**12**	0.54078	0.07726	7.00	<.0001

Specifying USERTYPE=LOM causes the regression effect to be removed from the seasonally adjusted series. The effect of the mean of the regression variable on the seasonally adjusted series can be seen by examining the plots of the original series and the seasonally adjusted series.

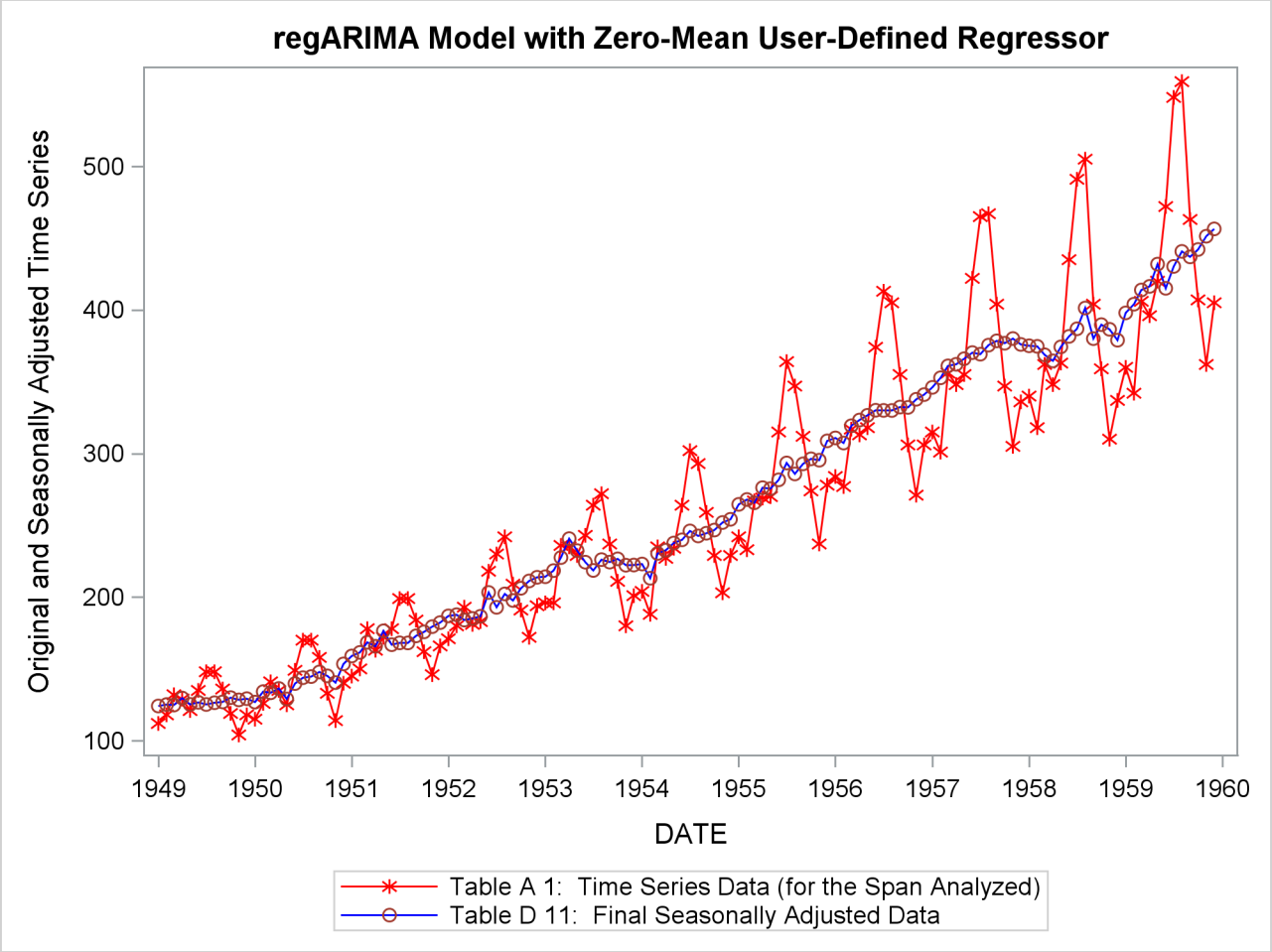
```
title 'regARIMA Model with Non-Zero-Mean User-Defined Regressor';
proc sgplot data=out;
   series x=date y=air_A1 / name = "A1" markers
                            markerattrs=(color=red symbol='asterisk')
                            lineattrs=(color=red);
   series x=date y=air_D11 / name= "D11" markers
                             markerattrs=(symbol='circle')
                             lineattrs=(color=blue);
   yaxis label='Original and Seasonally Adjusted Time Series';
run;

title 'regARIMA Model with Zero-Mean User-Defined Regressor';
proc sgplot data=outzm;
   series x=date y=air_A1 / name = "A1" markers
                            markerattrs=(color=red symbol='asterisk')
                            lineattrs=(color=red);
   series x=date y=air_D11 / name= "D11" markers
                             markerattrs=(symbol='circle')
                             lineattrs=(color=blue);
   yaxis label='Original and Seasonally Adjusted Time Series';
run;
```

The graph of the original and seasonally adjusted series in Output 45.6.4 shows that the level of the seasonally adjusted series has been altered due to the user-defined regressor. The graph of the original and seasonally adjusted series in Output 45.6.5 shows that the level of the seasonally adjusted series is the same as the original series since the user-defined regressor has zero-mean.

Output 45.6.4 Plot of Original and Seasonally Adjusted Data



Output 45.6.5 Plot of Original and Seasonally Adjusted Data (Zero-Mean Regressor)



When actual values are available for the forecast periods, information about forecast error is available in the output. Output 45.6.6 shows the table "Forecasts and Standard Errors of the Transformed Data on the Original Scale" for a series with missing values in the forecast period. Output 45.6.7 shows the table "Forecasts and Standard Errors of the Transformed Data on the Original Scale" for a series with actual values in the forecast period. Thus, it is more desirable to use SPAN= option to limit the span of a series if the actual values are available for the forecast period.

Output 45.6.6 PROC X13 Forecasts for Series Extended with Missing Values

Forecasts and Standard Errors of the Transformed Data

On the Original scale

For Variable AIR

Date	Forecast	Standard Error	95% Confidence Limits	
JAN1960	419.600	14.85053	391.509	449.705
FEB1960	416.480	19.05188	380.826	455.472
MAR1960	466.697	22.66762	424.402	513.208
APR1960	454.468	24.53242	408.951	505.051
MAY1960	473.876	27.91366	422.353	531.684
JUN1960	547.601	34.74893	483.769	619.855
JUL1960	623.318	42.20549	546.139	711.405
AUG1960	631.731	45.30824	549.231	726.623
SEP1960	527.221	39.81839	455.011	610.890
OCT1960	462.774	36.63020	396.605	539.984
NOV1960	407.155	33.64286	346.608	478.277
DEC1960	452.702	38.91914	382.913	535.212

Output 45.6.7 PROC X13 Forecasts for Series with Actual Values in Forecast Periods

Forecasts and Standard Errors of the Transformed Data

On the Original scale

For Variable AIR

Date	Data	Forecast	Forecast Error	Standard Error	t Value	95% Confidence Limits	
JAN1960	417.000	419.600	-2.600	14.85053	-0.18	391.509	449.705
FEB1960	391.000	416.480	-25.480	19.05188	-1.34	380.826	455.472
MAR1960	419.000	466.697	-47.697	22.66762	-2.10	424.402	513.208
APR1960	461.000	454.468	6.532	24.53242	0.27	408.951	505.051
MAY1960	472.000	473.876	-1.876	27.91366	-0.07	422.353	531.684
JUN1960	535.000	547.601	-12.601	34.74893	-0.36	483.769	619.855
JUL1960	622.000	623.318	-1.318	42.20549	-0.03	546.139	711.405
AUG1960	606.000	631.731	-25.731	45.30824	-0.57	549.231	726.623
SEP1960	508.000	527.221	-19.221	39.81839	-0.48	455.011	610.890
OCT1960	461.000	462.774	-1.774	36.63020	-0.05	396.605	539.984
NOV1960	390.000	407.155	-17.155	33.64286	-0.51	346.608	478.277
DEC1960	432.000	452.702	-20.702	38.91914	-0.53	382.913	535.212

Example 45.7: MDLINFOIN= and MDLINFOOUT= Data Sets

This example illustrates the use of MDLINFOIN= and MDLINFOOUT= data sets. Using the data set shown, PROC X13 step identifies the model with outliers as displayed in Output 45.7.1. Output 45.7.2 shows the data set that represents the chosen model.

```
data b1;
   input y @@;
   datalines;
   112 118 132 129
   121 135 148 148
   136 119 104 118
   115 126 141 135
   125 149 270 170
   158 133 114 140
;

title 'Model Identification Output to MDLINFOOUT= Data Set';
proc x13 data=b1 start='1980q1' interval=qtr MdlInfoOut=mdl;
   automdl;
   outlier;
run ;

proc print data=mdl;
run;
```

Output 45.7.1 Displayed Model Identification with Outliers

Model Identification Output to MDLINFOOUT= Data Set

The X13 Procedure

Critical Values to use in Outlier Detection
For Variable y

Begin	1980Q1
End	1985Q4
Observations	24
Method	Add One
AO Critical Value	3.419415
LS Critical Value	3.419415

Final Automatic Model Selection
For Variable y

Source of Model	Orders Altered	Estimated Model
Automatic Model Choice	**No**	**(2, 1, 0) (0, 0, 0)**

Regression Model Parameter Estimates
For Variable y

Type	Parameter	NoEst	Estimate	Standard Error	t Value	Pr > |t|
Automatically Identified	AO 1984Q3	Est	102.36589	5.96584	17.16	<.0001

Output 45.7.1 *continued*

Exact ARMA Maximum Likelihood Estimation

For Variable y

Parameter	Lag	Estimate	Standard Error	t Value	Pr > |t|
Nonseasonal AR	1	0.40892	0.20213	2.02	0.0554
	2	-0.53710	0.20975	-2.56	0.0178

Output 45.7.2 PROC X13 MDLINFOOUT= Data Set Model with Outlier Detection

Model Identification Output to MDLINFOOUT= Data Set

Obs	_NAME_	_MODELTYPE_	_MODELPART_	_COMPONENT_	_PARMTYPE_	_DSVAR_	_VALUE_	_FACTOR_
1	y	REG	EVENT	SCALE	AO	AO01JUL1984D	.	.
2	y	ARIMA	FORECAST	NONSEASONAL	DIF	y	.	.
3	y	ARIMA	FORECAST	NONSEASONAL	AR	y	.	1
4	y	ARIMA	FORECAST	NONSEASONAL	AR	y	.	1

Obs	_LAG_	_SHIFT_	_NOEST_	_EST_	_STDERR_	_TVALUE_	_PVALUE_	_STATUS_	_SCORE_	_LABEL_
1	.	.	0	102.366	5.96584	17.1587	0.000000	.		
2	1	.	.	.	.	.	.	.		
3	1	.	0	0.409	0.20213	2.0231	0.055385	.		
4	2	.	0	-0.537	0.20975	-2.5606	0.017830	.		

Suppose that after examining the output from the preceding example, you decide that an Easter regressor should be added to the model. The following statements create a data set with the model identified above and adds a US Census Bureau Predefined Easter(25) regressor. The new model data set to be used as input in the MDLINFOIN= option is displayed in the data set shown in Output 45.7.3.

```
data pluseaster;
   _NAME_ = 'y';
   _MODELTYPE_ = 'REG';
   _MODELPART_ = 'PREDEFINED';
   _COMPONENT_ = 'SCALE';
   _PARMTYPE_ = 'EASTER';
   _DSVAR_ = 'EASTER';
   _VALUE_ = 25;
run;

data mdlpluseaster;
   set mdl;
run;

proc append base=mdlpluseaster data=pluseaster force;
run;

proc print data=mdlpluseaster;
run;
```

Output 45.7.3 MDLINFOIN= Data Set Model with Easter(25) Regression Added

Model Identification Output to MDLINFOOUT= Data Set

Obs	_NAME_	_MODELTYPE_	_MODELPART_	_COMPONENT_	_PARMTYPE_	_DSVAR_	_VALUE_	_FACTOR_
1	y	REG	EVENT	SCALE	AO	AO01JUL1984D	.	.
2	y	ARIMA	FORECAST	NONSEASONAL	DIF	y	.	.
3	y	ARIMA	FORECAST	NONSEASONAL	AR	y	.	1
4	y	ARIMA	FORECAST	NONSEASONAL	AR	y	.	1
5	y	REG	PREDEFINED	SCALE	EASTER	EASTER	25	.

Obs	_LAG_	_SHIFT_	_NOEST_	_EST_	_STDERR_	_TVALUE_	_PVALUE_	_STATUS_	_SCORE_	_LABEL_
1	.	.	0	102.366	5.96584	17.1587	0.000000	.		
2	1	.	.	.	.	.	.	.		
3	1	.	0	0.409	0.20213	2.0231	0.055385	.		
4	2	.	0	-0.537	0.20975	-2.5606	0.017830	.		
5	.	.	.	.	.	.	.	.		

The following statements estimate the regression and ARIMA parameters by using the model described in the new data set mdlpluseaster. The results of estimating the new model are shown in Output 45.7.4.

```
proc x13 data=b1 start='1980q1' interval=qtr
   MdlInfoIn=mdlpluseaster MdlInfoOut=mdl2;
   estimate;
run;
```

Output 45.7.4 Estimate Model with Added Easter(25) Regression

Model Identification Output to MDLINFOOUT= Data Set

The X13 Procedure

Regression Model Parameter Estimates

For Variable y

Type	Parameter	NoEst	Estimate	Standard Error	t Value	Pr > |t|
Easter	Easter[25]	Est	6.73250	4.73335	1.42	0.1696
User Defined	AO01JUL1984D	Est	105.83795	6.12689	17.27	<.0001

Exact ARMA Maximum Likelihood Estimation

For Variable y

Parameter	Lag	Estimate	Standard Error	t Value	Pr > |t|
Nonseasonal AR	1	0.45233	0.20676	2.19	0.0401
	2	-0.54855	0.21583	-2.54	0.0190

The new model estimation results are displayed in the data set mdl2 shown in Output 45.7.5.

```
proc print data=mdl2;
run;
```

Output 45.7.5 MDLINFOOUT= Data Set, Estimation of Model with Easter(25) Regression Added

Model Identification Output to MDLINFOOUT= Data Set

Obs	_NAME_	_MODELTYPE_	_MODELPART_	_COMPONENT_	_PARMTYPE_	_DSVAR_	_VALUE_	_FACTOR_
1	y	REG	PREDEFINED	SCALE	EASTER	EASTER	25	.
2	y	REG	EVENT	SCALE	AO	AO01JUL1984D	.	.
3	y	ARIMA	FORECAST	NONSEASONAL	DIF	y	.	.
4	y	ARIMA	FORECAST	NONSEASONAL	AR	y	.	1
5	y	ARIMA	FORECAST	NONSEASONAL	AR	y	.	1

Obs	_LAG_	_SHIFT_	_NOEST_	_EST_	_STDERR_	_TVALUE_	_PVALUE_	_STATUS_	_SCORE_	_LABEL_
1	.	.	0	6.733	4.73335	1.4224	0.16961	.		
2	.	.	0	105.838	6.12689	17.2743	0.00000	.		
3	1	.	.	.	.	.	.	.		
4	1	.	0	0.452	0.20676	2.1877	0.04014	.		
5	2	.	0	-0.549	0.21583	-2.5415	0.01899	.		

Example 45.8: Setting Regression Parameters

This example illustrates the use of fixed regression parameters in PROC X13. Suppose that you have the same data set as in the section "Basic Seasonal Adjustment" on page 3154. You can specify the following statements to use TRAMO to automatically identify a model that includes a US Census Bureau Easter(25) regressor:

```
title 'Estimate Easter(25) Parameter';
proc x13 data=sales date=date MdlInfoOut=mdlout1;
   var sales;
   regression predefined=easter(25);
   automdl;
run ;
```

The displayed results are shown in Output 45.8.1.

Output 45.8.1 Automatic Model ID with Easter(25) Regression

Estimate Easter(25) Parameter

The X13 Procedure

Regression Model Parameter Estimates

For Variable sales

Type	Parameter	NoEst	Estimate	Standard Error	t Value	Pr > |t|
Easter	Easter[25]	Est	-5.09298	3.50786	-1.45	0.1489

Output 45.8.1 *continued*

Exact ARMA Maximum Likelihood Estimation

For Variable sales

Parameter	Lag	Estimate	Standard Error	t Value	Pr > |t|
Nonseasonal AR	**1**	0.62148	0.09279	6.70	<.0001
	2	0.23354	0.10385	2.25	0.0262
	3	-0.07191	0.09055	-0.79	0.4285
Nonseasonal MA	**1**	0.97377	0.03771	25.82	<.0001
Seasonal MA	**12**	0.10558	0.10205	1.03	0.3028

The MDLINFOOUT= data set, mdlout1, that contains the model and parameter estimates is shown in Output 45.8.2.

```
proc print data=mdlout1;
run;
```

Output 45.8.2 MDLINFOOUT= Data Set, Estimation of Automatic Model ID with Easter(25) Regression

Estimate Easter(25) Parameter

Obs	_NAME_	_MODELTYPE_	_MODELPART_	_COMPONENT_	_PARMTYPE_	_DSVAR_	_VALUE_	_FACTOR_
1	sales	REG	PREDEFINED	SCALE	EASTER	EASTER	25	.
2	sales	ARIMA	FORECAST	NONSEASONAL	DIF	sales	.	.
3	sales	ARIMA	FORECAST	SEASONAL	DIF	sales	.	.
4	sales	ARIMA	FORECAST	NONSEASONAL	AR	sales	.	1
5	sales	ARIMA	FORECAST	NONSEASONAL	AR	sales	.	1
6	sales	ARIMA	FORECAST	NONSEASONAL	AR	sales	.	1
7	sales	ARIMA	FORECAST	NONSEASONAL	MA	sales	.	1
8	sales	ARIMA	FORECAST	SEASONAL	MA	sales	.	2

Obs	_LAG_	_SHIFT_	_NOEST_	_EST_	_STDERR_	_TVALUE_	_PVALUE_	_STATUS_	_SCORE_	_LABEL_
1	.	.	0	-5.09298	3.50786	-1.4519	0.14894	.		
2	1	.	.	.	.	.	.	.		
3	1	.	.	.	.	.	.	.		
4	1	.	0	0.62148	0.09279	6.6980	0.00000	.		
5	2	.	0	0.23354	0.10385	2.2488	0.02621	.		
6	3	.	0	-0.07191	0.09055	-0.7942	0.42851	.		
7	1	.	0	0.97377	0.03771	25.8240	0.00000	.		
8	1	.	0	0.10558	0.10205	1.0346	0.30277	.		

To fix the Easter(25) parameter while adding a regressor that is weighted according to the number of Saturdays in a month, either use the REGRESSION and EVENT statements or create a MDLINFOIN= data set. The following statements show the method for using the REGRESSION statement to fix the EASTER parameter and the EVENT statement to add the SATURDAY regressor. The output is shown in Output 45.8.3.

```
title 'Use SAS Statements to Alter Model';
proc x13 data=sales date=date MdlInfoOut=mdlout2grm;
   var sales;
   regression predefined=easter(25) / b=-5.029298 F;
   event Saturday;
   automdl;
run ;
```

Output 45.8.3 Automatic Model ID with Fixed Easter(25) and Saturday Regression

Use SAS Statements to Alter Model

The X13 Procedure

Regression Model Parameter Estimates

For Variable sales

Type	Parameter	NoEst	Estimate	Standard Error	t Value	Pr > |t|
User Defined	Saturday	Est	3.23225	1.16701	2.77	0.0064
Easter	Easter[25]	Fixed	-5.02930	.	.	.

Exact ARMA Maximum Likelihood Estimation

For Variable sales

Parameter	Lag	Estimate	Standard Error	t Value	Pr > |t|
Nonseasonal AR	1	-0.32506	0.08256	-3.94	0.0001

To fix the EASTER regressor and add the new SATURDAY regressor by using a DATA step, you can create the data set mdlin2 as shown. The data set mdlin2 is displayed in Output 45.8.4.

```
title 'Use a SAS DATA Step to Create a MdlInfoIn= Data Set';
data plusSaturday;
   _NAME_ = 'sales';
   _MODELTYPE_ = 'REG';
   _MODELPART_ = 'EVENT';
   _COMPONENT_ = 'SCALE';
   _PARMTYPE_ = 'USER';
   _DSVAR_ = 'SATURDAY';
run;

data mdlin2;
   set mdlout1;
   if ( _DSVAR_ = 'EASTER' ) then do;
       _NOEST_ = 1;
       _EST_ = -5.029298;
       end;
run;

proc append base=mdlin2 data=plusSaturday force;
run;

proc print data=mdlin2;
run;
```

Output 45.8.4 MDLINFOIN= Data Set, Fixed Easter(25) and Added Saturday Regression, Previously Identified Model

Use a SAS DATA Step to Create a MdlInfoIn= Data Set

Obs	_NAME_	_MODELTYPE_	_MODELPART_	_COMPONENT_	_PARMTYPE_	_DSVAR_	_VALUE_	_FACTOR_
1	sales	REG	PREDEFINED	SCALE	EASTER	EASTER	25	.
2	sales	ARIMA	FORECAST	NONSEASONAL	DIF	sales	.	.
3	sales	ARIMA	FORECAST	SEASONAL	DIF	sales	.	.
4	sales	ARIMA	FORECAST	NONSEASONAL	AR	sales	.	1
5	sales	ARIMA	FORECAST	NONSEASONAL	AR	sales	.	1
6	sales	ARIMA	FORECAST	NONSEASONAL	AR	sales	.	1
7	sales	ARIMA	FORECAST	NONSEASONAL	MA	sales	.	1
8	sales	ARIMA	FORECAST	SEASONAL	MA	sales	.	2
9	sales	REG	EVENT	SCALE	USER	SATURDAY	.	.

Obs	_LAG_	_SHIFT_	_NOEST_	_EST_	_STDERR_	_TVALUE_	_PVALUE_	_STATUS_	_SCORE_	_LABEL_
1	.	.	1	-5.02930	3.50786	-1.4519	0.14894	.		
2	1	.	.	.	.	.	.	.		
3	1	.	.	.	.	.	.	.		
4	1	.	0	0.62148	0.09279	6.6980	0.00000	.		
5	2	.	0	0.23354	0.10385	2.2488	0.02621	.		
6	3	.	0	-0.07191	0.09055	-0.7942	0.42851	.		
7	1	.	0	0.97377	0.03771	25.8240	0.00000	.		
8	1	.	0	0.10558	0.10205	1.0346	0.30277	.		
9	.	.	.	.	.	.	.	.		

The data set mdlin2 can be used to replace the regression and model information contained in the REGRESSION, EVENT, and AUTOMDL statements. Note that the model specified in the mdlin2 data set is the same model as the automatically identified model. The following example uses the mdlin2 data set as input; the results are displayed in Output 45.8.5.

```
title 'Use Updated Data Set to Alter Model';
proc x13 data=sales date=date MdlInfoIn=mdlin2 MdlInfoOut=mdlout2DS;
   var sales;
   estimate;
run ;
```

Output 45.8.5 Estimate MDLINFOIN= File for Model with Fixed Easter(25) and Saturday Regression, Previously Identified Model

Use Updated Data Set to Alter Model

The X13 Procedure

Regression Model Parameter Estimates

For Variable sales

Type	Parameter	NoEst	Estimate	Standard Error	t Value	Pr > |t|
User Defined	SATURDAY	Est	3.41762	1.07641	3.18	0.0019
Easter	Easter[25]	Fixed	-5.02930	.	.	.

Output 45.8.5 *continued*

Exact ARMA Maximum Likelihood Estimation

For Variable sales

Parameter	Lag	Estimate	Standard Error	t Value	Pr > |t|
Nonseasonal AR	**1**	0.62225	0.09175	6.78	<.0001
	2	0.30429	0.10109	3.01	0.0031
	3	-0.14862	0.08859	-1.68	0.0958
Nonseasonal MA	**1**	0.97125	0.03798	25.57	<.0001
Seasonal MA	**12**	0.11691	0.10000	1.17	0.2445

The following statements specify almost the same information as contained in the data set mdlin2. The ARIMA statement specifies the lags of the model. However, the initial AR and MA parameter values are the default. When using the mdlin2 data set as input, the initial values can be specified. The results are displayed in Output 45.8.6.

```
title 'Use SAS Statements to Alter Model';
proc x13 data=sales date=date MdlInfoOut=mdlout3grm;
   var sales;
   regression predefined=easter(25) / b=-5.029298 F;
   event Saturday;
   arima model=((3 1 1)(0 1 1));
   estimate;
run ;

proc print data=mdlout3grm;
run;
```

Output 45.8.6 MDLINFOOUT= Statement, Fixed Easter(25) and Added Saturday Regression, Previously Identified Model

Use SAS Statements to Alter Model

Obs	_NAME_	_MODELTYPE_	_MODELPART_	_COMPONENT_	_PARMTYPE_	_DSVAR_	_VALUE_	_FACTOR_
1	sales	REG	EVENT	SCALE	USER	Saturday	.	.
2	sales	REG	PREDEFINED	SCALE	EASTER	EASTER	25	.
3	sales	ARIMA	FORECAST	NONSEASONAL	DIF	sales	.	.
4	sales	ARIMA	FORECAST	SEASONAL	DIF	sales	.	.
5	sales	ARIMA	FORECAST	NONSEASONAL	AR	sales	.	1
6	sales	ARIMA	FORECAST	NONSEASONAL	AR	sales	.	1
7	sales	ARIMA	FORECAST	NONSEASONAL	AR	sales	.	1
8	sales	ARIMA	FORECAST	NONSEASONAL	MA	sales	.	1
9	sales	ARIMA	FORECAST	SEASONAL	MA	sales	.	2

Obs	_LAG_	_SHIFT_	_NOEST_	_EST_	_STDERR_	_TVALUE_	_PVALUE_	_STATUS_	_SCORE_	_LABEL_
1	.	.	0	3.41760	1.07640	3.1750	0.00187	.		
2	.	.	1	-5.02930	.	.	.	.		
3	1	.	.	.	.	.	.	.		
4	1	.	.	.	.	.	.	.		
5	1	.	0	0.62228	0.09175	6.7825	0.00000	.		
6	2	.	0	0.30431	0.10109	3.0103	0.00314	.		
7	3	.	0	-0.14864	0.08859	-1.6779	0.09578	.		
8	1	.	0	0.97128	0.03796	25.5882	0.00000	.		
9	1	.	0	0.11684	0.10000	1.1684	0.24481	.		

The MDLINFOOUT= data set provides a method for comparing the results of the model identification. The data set mdlout3grm that results from using the MODEL= option in the ARIMA statement can be compared to the data set mdlout2DS that results from using the MDLINFOIN= data set with initial values for the AR and MA parameters. The mdlout2DS data set is shown in Output 45.8.7, and the results of the comparison are shown in Output 45.8.8. The slight difference in the estimated parameters can be attributed to the difference in the initial values for the AR and MA parameters.

```
proc print data=mdlout2DS;
run;
```

Output 45.8.7 MDLINFOOUT= Data Set, Fixed Easter(25) and Added Saturday Regression, Previously Identified Model

Use SAS Statements to Alter Model

Obs	_NAME_	_MODELTYPE_	_MODELPART_	_COMPONENT_	_PARMTYPE_	_DSVAR_	_VALUE_	_FACTOR_
1	sales	REG	EVENT	SCALE	USER	SATURDAY	.	.
2	sales	REG	PREDEFINED	SCALE	EASTER	EASTER	25	.
3	sales	ARIMA	FORECAST	NONSEASONAL	DIF	sales	.	.
4	sales	ARIMA	FORECAST	SEASONAL	DIF	sales	.	.
5	sales	ARIMA	FORECAST	NONSEASONAL	AR	sales	.	1
6	sales	ARIMA	FORECAST	NONSEASONAL	AR	sales	.	1
7	sales	ARIMA	FORECAST	NONSEASONAL	AR	sales	.	1
8	sales	ARIMA	FORECAST	NONSEASONAL	MA	sales	.	1
9	sales	ARIMA	FORECAST	SEASONAL	MA	sales	.	2

Obs	_LAG_	_SHIFT_	_NOEST_	_EST_	_STDERR_	_TVALUE_	_PVALUE_	_STATUS_	_SCORE_	_LABEL_
1	.	.	0	3.41762	1.07641	3.1750	0.00187	.		
2	.	.	1	-5.02930	.	.	.	.		
3	1	.	.	.	.	.	.	.		
4	1	.	.	.	.	.	.	.		
5	1	.	0	0.62225	0.09175	6.7817	0.00000	.		
6	2	.	0	0.30429	0.10109	3.0100	0.00314	.		
7	3	.	0	-0.14862	0.08859	-1.6776	0.09584	.		
8	1	.	0	0.97125	0.03798	25.5712	0.00000	.		
9	1	.	0	0.11691	0.10000	1.1691	0.24451	.		

```
title 'Compare Results of SAS Statement Input and MdlInfoIn= Input';
proc compare base= mdlout3grm compare=mdlout2DS;
var _EST_;
run ;
```

Output 45.8.8 Compare Parameter Estimates from Different MDLINFOOUT= Data Sets

Compare Results of SAS Statement Input and MdlInfoIn= Input

Value Comparison Results for Variables

Obs	Value of Parameter Estimate: Base _EST_	Compare _EST_	Diff.	% Diff
1	3.4176	3.4176	0.0000227	0.000665
5	0.6223	0.6222	-0.000033	-0.005259
6	0.3043	0.3043	-0.000021	-0.006983
7	-0.1486	-0.1486	0.0000236	-0.0159
8	0.9713	0.9713	-0.000024	-0.002459
9	0.1168	0.1169	0.0000763	0.0653

Example 45.9: Creating an MDLINFO= Data Set for Use with the PICKMDL Statement

This example illustrates how you can create a data set for use in the PICKMDL statement that contains five commonly used ARIMA models:

- ARIMA (0 1 1)(0 1 1)s
- ARIMA (0 1 2)(0 1 1)s
- ARIMA (2 1 0)(0 1 1)s
- ARIMA (0 2 2)(0 1 1)s
- ARIMA (2 1 2)(0 1 1)s

The following macro code creates a MDLINFOIN= data set for a general ARIMA model:

```
%macro makemodel(name,p,d,q,sp,sd,sq,model);
   data "&name" (keep= _MODELTYPE_ _MODELPART_ _COMPONENT_
                       _DSVAR_ _PARMTYPE_ _FACTOR_ _LAG_
                         _LABEL_ );
      length _MODELTYPE_ _MODELPART_ _COMPONENT_ _DSVAR_
             _PARMTYPE_ $32;
      length _FACTOR_ _LAG_ 8;
      length _LABEL_ $32;

      _MODELTYPE_="ARIMA";
      _MODELPART_="FORECAST";
      _DSVAR_=".";

      _LABEL_="("||"&p"||" "||"&d"||" "||"&q"||")("||
              "&sp"||" "||"&sd"||" "||"&sq"||")s";

      /* nonseasonal AR factors */
      _COMPONENT_="NONSEASONAL";
      _PARMTYPE_="AR";
      _FACTOR_=1;
      do _LAG_=1 to &p;
         output;
         end;

      /* seasonal AR factors */
      _COMPONENT_="SEASONAL";
      _PARMTYPE_="AR";
      _FACTOR_=2;
      do _LAG_=1 to &sp;
         output;
         end;
```

```
      /* nonseasonal MA factors */
      _COMPONENT_="NONSEASONAL";
      _PARMTYPE_="MA";
      _FACTOR_=1;
      do _LAG_=1 to &q;
         output;
         end;

      /* seasonal MA factors */
      _COMPONENT_="SEASONAL";
      _PARMTYPE_="MA";
      _FACTOR_=2;
      do _LAG_=1 to &sq;
         output;
         end;

      /* nonseasonal DIF */
      _COMPONENT_="NONSEASONAL";
      _PARMTYPE_="DIF";
      _FACTOR_=1;
      _LAG_=1;
      do i_=1 to &d;
         output;
         end;

      /* seasonal DIF */
      _COMPONENT_="SEASONAL";
      _PARMTYPE_="DIF";
      _FACTOR_=2;
      _LAG_=1;
      do i_=1 to &sd;
         output;
         end;

   run;
   data sasuser.&name;
     length _MODEL_ $32;
     set "&name";
     _MODEL_ = "&model";
   run;

%mend makemodel;
```

The following SAS statements use the macro to generate a data set with some commonly used models for use in the PICKMDL statement:

```
%makemodel(x13mdl1,0,1,1,0,1,1,Model1);
%makemodel(x13mdl2,0,1,2,0,1,1,Model2);
%makemodel(x13mdl3,2,1,0,0,1,1,Model3);
%makemodel(x13mdl4,0,2,2,0,1,1,Model4);
%makemodel(x13mdl5,2,1,2,0,1,1,Model5);

data Models;
   length _NAME_ $32;
   set sasuser.x13mdl1 sasuser.x13mdl2 sasuser.x13mdl3
       sasuser.x13mdl4 sasuser.x13mdl5;
   _NAME_ = 'sales';
run;
```

The Models data set is shown in Output 45.9.1.

```
title '5 Commonly Used Models';
proc print data=Models;
run ;
```

Output 45.9.1 A Data Set That Contains Models for Use with the PICKMDL Statement

5 Commonly Used Models

Obs	_NAME_	_MODEL_	_MODELTYPE_	_MODELPART_	_COMPONENT_	_DSVAR_
1	sales	Model1	ARIMA	FORECAST	NONSEASONAL	.
2	sales	Model1	ARIMA	FORECAST	SEASONAL	.
3	sales	Model1	ARIMA	FORECAST	NONSEASONAL	.
4	sales	Model1	ARIMA	FORECAST	SEASONAL	.
5	sales	Model2	ARIMA	FORECAST	NONSEASONAL	.
6	sales	Model2	ARIMA	FORECAST	NONSEASONAL	.
7	sales	Model2	ARIMA	FORECAST	SEASONAL	.
8	sales	Model2	ARIMA	FORECAST	NONSEASONAL	.
9	sales	Model2	ARIMA	FORECAST	SEASONAL	.
10	sales	Model3	ARIMA	FORECAST	NONSEASONAL	.
11	sales	Model3	ARIMA	FORECAST	NONSEASONAL	.
12	sales	Model3	ARIMA	FORECAST	SEASONAL	.
13	sales	Model3	ARIMA	FORECAST	NONSEASONAL	.
14	sales	Model3	ARIMA	FORECAST	SEASONAL	.
15	sales	Model4	ARIMA	FORECAST	NONSEASONAL	.
16	sales	Model4	ARIMA	FORECAST	NONSEASONAL	.
17	sales	Model4	ARIMA	FORECAST	SEASONAL	.
18	sales	Model4	ARIMA	FORECAST	NONSEASONAL	.
19	sales	Model4	ARIMA	FORECAST	NONSEASONAL	.
20	sales	Model4	ARIMA	FORECAST	SEASONAL	.
21	sales	Model5	ARIMA	FORECAST	NONSEASONAL	.
22	sales	Model5	ARIMA	FORECAST	NONSEASONAL	.

Obs	_PARMTYPE_	_FACTOR_	_LAG_	_LABEL_
1	MA	1	1	(0 1 1)(0 1 1)s
2	MA	2	1	(0 1 1)(0 1 1)s
3	DIF	1	1	(0 1 1)(0 1 1)s
4	DIF	2	1	(0 1 1)(0 1 1)s
5	MA	1	1	(0 1 2)(0 1 1)s
6	MA	1	2	(0 1 2)(0 1 1)s
7	MA	2	1	(0 1 2)(0 1 1)s
8	DIF	1	1	(0 1 2)(0 1 1)s
9	DIF	2	1	(0 1 2)(0 1 1)s
10	AR	1	1	(2 1 0)(0 1 1)s
11	AR	1	2	(2 1 0)(0 1 1)s
12	MA	2	1	(2 1 0)(0 1 1)s
13	DIF	1	1	(2 1 0)(0 1 1)s
14	DIF	2	1	(2 1 0)(0 1 1)s
15	MA	1	1	(0 2 2)(0 1 1)s
16	MA	1	2	(0 2 2)(0 1 1)s
17	MA	2	1	(0 2 2)(0 1 1)s
18	DIF	1	1	(0 2 2)(0 1 1)s
19	DIF	1	1	(0 2 2)(0 1 1)s
20	DIF	2	1	(0 2 2)(0 1 1)s
21	AR	1	1	(2 1 2)(0 1 1)s
22	AR	1	2	(2 1 2)(0 1 1)s

Output 45.9.1 *continued*

5 Commonly Used Models

Obs	_NAME_	_MODEL_	_MODELTYPE_	_MODELPART_	_COMPONENT_	_DSVAR_
23	sales	Model5	ARIMA	FORECAST	NONSEASONAL	.
24	sales	Model5	ARIMA	FORECAST	NONSEASONAL	.
25	sales	Model5	ARIMA	FORECAST	SEASONAL	.
26	sales	Model5	ARIMA	FORECAST	NONSEASONAL	.
27	sales	Model5	ARIMA	FORECAST	SEASONAL	.

Obs	_PARMTYPE_	_FACTOR_	_LAG_	_LABEL_
23	MA	1	1	(2 1 2)(0 1 1)s
24	MA	1	2	(2 1 2)(0 1 1)s
25	MA	2	1	(2 1 2)(0 1 1)s
26	DIF	1	1	(2 1 2)(0 1 1)s
27	DIF	2	1	(2 1 2)(0 1 1)s

The following statements specify that the PICKMDL method be used to choose a model from the list of models that are defined in the Models data set. The default METHOD=FIRST option is used to select the model. The chosen model is shown in the mdlchosen data set in Output 45.9.2.

```
proc x13 data=sales date=date mdlinfoin=Models mdlinfoout=mdlchosen;
   var sales;
   transform function=log;
   pickmdl method=first;
run;

title 'Chosen Model';
proc print data=mdlchosen;
run ;
```

Output 45.9.2 The Model Chosen from the Five Commonly Used Models

Chosen Model

Obs	_NAME_	_MODEL_	_MODELTYPE_	_MODELPART_	_COMPONENT_	_PARMTYPE_	_DSVAR_	_VALUE_	_FACTOR_
1	sales	MODEL1	ARIMA	FORECAST	TRANSFORM	LOG	sales	.	.
2	sales	MODEL1	ARIMA	FORECAST	NONSEASONAL	DIF	sales	.	.
3	sales	MODEL1	ARIMA	FORECAST	SEASONAL	DIF	sales	.	.
4	sales	MODEL1	ARIMA	FORECAST	NONSEASONAL	MA	sales	.	1
5	sales	MODEL1	ARIMA	FORECAST	SEASONAL	MA	sales	.	2

Obs	_LAG_	_SHIFT_	_NOEST_	_EST_	_STDERR_	_TVALUE_	_PVALUE_	_STATUS_	_SCORE_	_LABEL_
1	.	.	.	.	.	.	.	.	.	(0 1 1)(0 1 1)s
2	1	.	.	.	.	.	.	.	.	(0 1 1)(0 1 1)s
3	1	.	.	.	.	.	.	.	.	(0 1 1)(0 1 1)s
4	1	.	0	0.40181	0.078870	5.09458	.000001192	.	.	(0 1 1)(0 1 1)s
5	1	.	0	0.55695	0.076255	7.30369	2.4359E-11	.	.	(0 1 1)(0 1 1)s

The following statements reverse the order of the models in the input data set. The default METHOD=FIRST option is used to select the model. The chosen model is shown in the mdlchosen data set in Output 45.9.3. With METHOD=FIRST, a different model is chosen because the order is changed.

```
data Models;
   length _NAME_ $32;
   set sasuser.x13mdl5 sasuser.x13mdl4 sasuser.x13mdl3
       sasuser.x13mdl2 sasuser.x13mdl1 ;
   _NAME_ = 'sales';
run;

proc x13 data=sales date=date mdlinfoin=Models mdlinfoout=mdlchosen;
   var sales;
   transform function=log;
   pickmdl method=first;
run;

title 'Chosen Model';
proc print data=mdlchosen;
run ;
```

Output 45.9.3 The Model Chosen from the Five Commonly Used Models, Reversed Order

Chosen Model

Obs	_NAME_	_MODEL_	_MODELTYPE_	_MODELPART_	_COMPONENT_	_PARMTYPE_	_DSVAR_	_VALUE_	_FACTOR_
1	sales	MODEL3	ARIMA	FORECAST	TRANSFORM	LOG	sales	.	.
2	sales	MODEL3	ARIMA	FORECAST	NONSEASONAL	DIF	sales	.	.
3	sales	MODEL3	ARIMA	FORECAST	SEASONAL	DIF	sales	.	.
4	sales	MODEL3	ARIMA	FORECAST	NONSEASONAL	AR	sales	.	1
5	sales	MODEL3	ARIMA	FORECAST	NONSEASONAL	AR	sales	.	1
6	sales	MODEL3	ARIMA	FORECAST	SEASONAL	MA	sales	.	1

Obs	_LAG_	_SHIFT_	_NOEST_	_EST_	_STDERR_	_TVALUE_	_PVALUE_	_STATUS_	_SCORE_	_LABEL_
1	.	.	.	.	.	.	.	.		(2 1 0)(0 1 1)s
2	1	.	.	.	.	.	.	.		(2 1 0)(0 1 1)s
3	1	.	.	.	.	.	.	.		(2 1 0)(0 1 1)s
4	1	.	0	-0.36159	0.086055	-4.20188	0.00005	.		(2 1 0)(0 1 1)s
5	2	.	0	-0.06366	0.086141	-0.73905	0.46120	.		(2 1 0)(0 1 1)s
6	1	.	0	0.56109	0.072814	7.70588	0.00000	.		(2 1 0)(0 1 1)s

The following example shows the use of PICKMDL statement option METHOD=BEST to select the model. The chosen model is shown in the mdlchosen data set in Output 45.9.4. With METHOD=BEST, a different model is chosen than either of the previous models chosen. Because the order in which the models occur in the MDLINFOIN= data set affects model selection when METHOD=FIRST is specified, it is a common practice to list models from the simplest model to the most complex in the MDLINFOIN= data set that is used in conjunction with the PICKMDL statement.

```
proc x13 data=sales date=date mdlinfoin=Models mdlinfoout=mdlchosen;
   var sales;
   transform function=log;
   pickmdl method=best;
run;

title 'Chosen Model';
proc print data=mdlchosen;
run ;
```

Output 45.9.4 The Model Chosen from the Five Commonly Used Models, METHOD=BEST

Chosen Model

Obs	_NAME_	_MODEL_	_MODELTYPE_	_MODELPART_	_COMPONENT_	_PARMTYPE_	_DSVAR_	_VALUE_	_FACTOR_
1	sales	MODEL2	ARIMA	FORECAST	TRANSFORM	LOG	sales	.	.
2	sales	MODEL2	ARIMA	FORECAST	NONSEASONAL	DIF	sales	.	.
3	sales	MODEL2	ARIMA	FORECAST	SEASONAL	DIF	sales	.	.
4	sales	MODEL2	ARIMA	FORECAST	NONSEASONAL	MA	sales	.	1
5	sales	MODEL2	ARIMA	FORECAST	NONSEASONAL	MA	sales	.	1
6	sales	MODEL2	ARIMA	FORECAST	SEASONAL	MA	sales	.	2

Obs	_LAG_	_SHIFT_	_NOEST_	_EST_	_STDERR_	_TVALUE_	_PVALUE_	_STATUS_	_SCORE_	_LABEL_
1	.	.	.	.	.	.	.	.		(0 1 2)(0 1 1)s
2	1	.	.	.	.	.	.	.		(0 1 2)(0 1 1)s
3	1	.	.	.	.	.	.	.		(0 1 2)(0 1 1)s
4	1	.	0	0.39613	0.086126	4.59937	0.00001	.		(0 1 2)(0 1 1)s
5	2	.	0	0.03961	0.086163	0.45966	0.64652	.		(0 1 2)(0 1 1)s
6	1	.	0	0.55903	0.076446	7.31277	0.00000	.		(0 1 2)(0 1 1)s

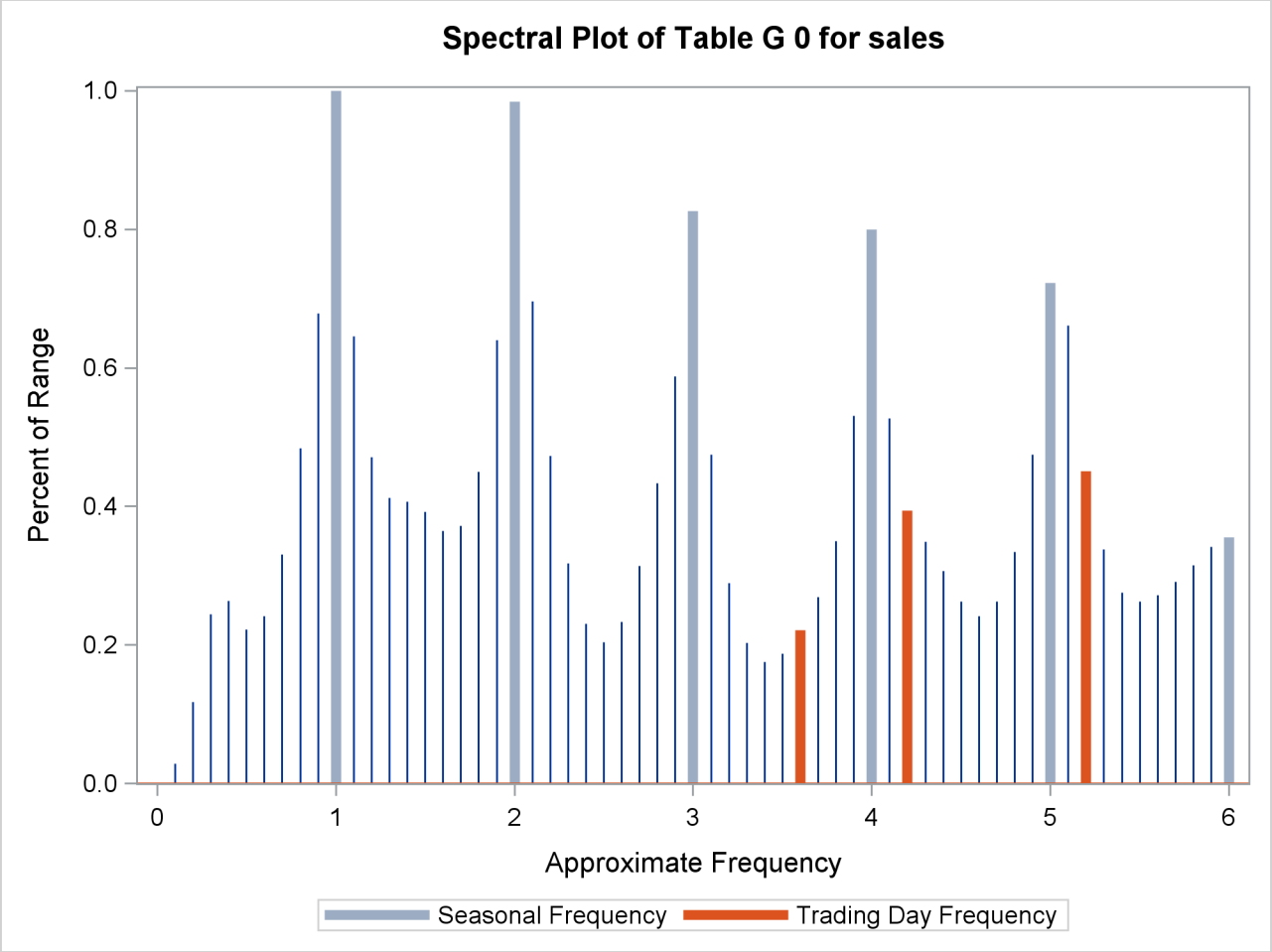
Example 45.10: Illustration of ODS Graphics

This example illustrates the use of ODS Graphics. Using the same data set as in the section "Basic Seasonal Adjustment" on page 3154 and the previous examples, a spectral plot of the original series is displayed in Output 45.10.1.

The graphical displays are available when ODS Graphics is enabled. For specific information about the graphics available in the X13 procedure, see the section "ODS Graphics" on page 3218.

```
proc x13 data=sales date=date;
   var sales;
run;
```

Output 45.10.1 Spectral Plot for Original Data



Example 45.11: AUXDATA= Data Set

This example demonstrates the use of the AUXDATA= data set to input user-defined regressors for use in the regARIMA model. User-defined regressors are often economic indicators, but in this example a user-defined regressor is generated in the following statements:

```
data auxreg(keep=date lengthofmonth);
   set sales;
   lengthofmonth = (INTNX('MONTH',date,1) - date) - (365/12);
   format date monyy.;
run;
```

When you use the AUXDATA= data set, it is not necessary to merge the user-defined regressor data set with the DATA= data set. The following statements input the regressor lengthofmonth in the data set auxreg. The regressor lengthofmonth is specified in the REGRESSION statement, and the data set auxreg is specified in the AUXDATA= option in the PROC X13 statement.

```
title 'Align lengthofmonth Regressor from Auxreg to First Three Years';
ods select regParameterEstimates;
proc x13 data=sales(obs=36) date=date auxdata=auxreg;
   var sales;
   regression uservar=lengthofmonth;
   arima model=((0 1 1) (0 1 1));
   estimate;
run;

title 'Align lengthofmonth Regressor from Auxreg to Second Three Years';
ods select regParameterEstimates;
proc x13 data=sales(firstobs=37 obs=72) date=date auxdata=auxreg;
   var sales;
   regression uservar=lengthofmonth;
   arima model=((0 1 1) (0 1 1));
   estimate;
run;
```

Output 45.11.1 and Output 45.11.2 display the parameter estimates for the two series.

Output 45.11.1 Using Regressors in the AUXDATA= Data for the First Three Years of Series

Align lengthofmonth Regressor from Auxreg to First Three Years

The X13 Procedure

Regression Model Parameter Estimates

For Variable sales

Type	Parameter	NoEst	Estimate	Standard Error	t Value	Pr > |t|
User Defined	lengthofmonth	Est	2.98046	5.36251	0.56	0.5840

Output 45.11.2 Using Regressors in the AUXDATA= Data for the Second Three Years of Series

Align lengthofmonth Regressor from Auxreg to Second Three Years

The X13 Procedure

Regression Model Parameter Estimates

For Variable sales

Type	Parameter	NoEst	Estimate	Standard Error	t Value	Pr > |t|
User Defined	lengthofmonth	Est	-0.51215	8.43145	-0.06	0.9521

The X13 procedure uses the date variable in the sales data set and the auxreg data set to align the user-defined regressors.

In the following example, the DATA= data set salesby contains BY groups. The X13 procedure aligns the regressor in the auxreg data set to each BY group in the salesby data set according to the variable date that is specified by the DATE= option in the PROC X13 statement. The variable date must be present in the auxreg data set to align the values.

```
data salesby;
     set sales(obs=72);
     if ( _n_ < 37 ) then by=1;
     else by=2;
run;
ods select regParameterEstimates;
title 'Align lengthofmonth Regressor from Auxreg to BY Groups';
proc x13 data=salesby date=date auxdata=auxreg;
   var sales;
   by by;
   regression uservar=lengthofmonth;
   arima model=((0 1 1) (0 1 1));
   estimate;
run;
```

The results in Output 45.11.3 match the previous results in Output 45.11.1 and Output 45.11.2.

Output 45.11.3 Using Regressors in the AUXDATA= Data with BY Groups

Align lengthofmonth Regressor from Auxreg to BY Groups

The X13 Procedure

by=1

Regression Model Parameter Estimates

For Variable sales

Type	Parameter	NoEst	Estimate	Standard Error	t Value	Pr > |t|
User Defined	lengthofmonth	Est	2.98046	5.36251	0.56	0.5840

Align lengthofmonth Regressor from Auxreg to BY Groups

The X13 Procedure

by=2

Regression Model Parameter Estimates

For Variable sales

Type	Parameter	NoEst	Estimate	Standard Error	t Value	Pr > |t|
User Defined	lengthofmonth	Est	-0.51215	8.43145	-0.06	0.9521

References

Box, G. E. P., Jenkins, G. M., and Reinsel, G. C. (1994). *Time Series Analysis: Forecasting and Control*. 3rd ed. Englewood Cliffs, NJ: Prentice-Hall.

Cholette, P. A. (1979). *A Comparison and Assessment of Various Adjustment Methods of Sub-annual Series to Yearly Benchmarks*. StatCan Staff Paper STC2119, Seasonal Adjustment and Time Series Staff, Statistics Canada, Ottawa.

Dagum, E. B. (1983). *The X-11-ARIMA Seasonal Adjustment Method*. Technical Report 12-564E, Statistics Canada, Ottawa.

Dagum, E. B. (1988). *The X-11-ARIMA/88 Seasonal Adjustment Method: Foundations and User's Manual*. Ottawa: Statistics Canada.

Findley, D. F., Monsell, B. C., Bell, W. R., Otto, M. C., and Chen, B. C. (1998). "New Capabilities and Methods of the X-12-ARIMA Seasonal Adjustment Program." *Journal of Business and Economic Statistics* 16:127–176.

Gómez, V., and Maravall, A. (1997a). *Guide for Using the Programs TRAMO and SEATS, Beta Version*. Madrid: Banco de España.

Gómez, V., and Maravall, A. (1997b). *Programs TRAMO and SEATS: Instructions for the User, Beta Version*. Madrid: Banco de España.

Hillmer, S. C., and Tiao, G. C. (1982). "An ARIMA-Model-Based Approach to Seasonal Adjustment." *Journal of the American Statistical Association* 77:63–70.

Huot, G. (1975). *Quadratic Minimization Adjustment of Monthly or Quarterly Series to Annual Totals*. StatCan Staff Paper STC2104, Statistics Canada, Seasonal Adjustment and Time Series Staff, Ottawa.

Ladiray, D., and Quenneville, B. (2001). *Seasonal Adjustment with the X-11 Method*. New York: Springer-Verlag.

Ljung, G. M. (1993). "On Outlier Detection in Time Series." *Journal of the Royal Statistical Society, Series B* 55:559–567.

Lothian, J., and Morry, M. (1978a). *A Set of Quality Control Statistics for the X-11-ARIMA Seasonal Adjustment Method*. StatCan Staff Paper STC1788E, Seasonal Adjustment and Time Series Analysis Staff, Statistics Canada, Ottawa.

Lothian, J., and Morry, M. (1978b). *A Test for the Presence of Identifiable Seasonality When Using the X-11-ARIMA Program*. StatCan Staff Paper STC2118, Seasonal Adjustment and Time Series Analysis Staff, Statistics Canada, Ottawa.

Shiskin, J., Young, A. H., and Musgrave, J. C. (1967). *The X-11 Variant of the Census Method II Seasonal Adjustment Program*. Technical Report 15, US Department of Commerce, Bureau of the Census.

US Bureau of the Census (2009). *X-12-ARIMA Reference Manual, Version 0.3*. US Bureau of the Census, Washington, DC. http://www.census.gov/ts/x12a/v03/x12adocV03.pdf.

US Bureau of the Census (2010). *X-12-ARIMA Seasonal Adjustment Program, Version 0.3*. US Bureau of the Census, Washington, DC. http://www.census.gov/srd/www/x12a/.

US Bureau of the Census (2013a). *X-13ARIMA-SEATS Quick Reference for DOS, Version 1.1*. US Bureau of the Census, Washington, DC. http://www.census.gov/ts/x13as/pc/qrefX13ASpc.pdf.

US Bureau of the Census (2013b). *X-13ARIMA-SEATS Quick Reference for UNIX/Linux, Version 1.1*. US Bureau of the Census, Washington, DC. http://www.census.gov/ts/x13as/unix/qrefX13ASunix.pdf.

US Bureau of the Census (2013c). *X-13ARIMA-SEATS Reference Manual, Version 1.1*. US Bureau of the Census, Washington, DC. http://www.census.gov/ts/x13as/docX13ASHTML.pdf.

Part III

Data Access Engines

Chapter 46
The SASECRSP Interface Engine

Contents

Overview: SASECRSP Interface Engine **3280**
Introduction 3280
Opening a Database 3280
Using Your Opened Database 3282
Getting Started: SASECRSP Interface Engine **3282**
Structure of a SAS Data Set That Contains Time Series Data 3282
Reading CRSP Data Files 3283
Using the SAS DATA Step 3283
Using SAS Procedures 3283
Using the SAS Windowing Environment 3283
Using CRSP Date Formats, Informats, and Functions 3283
Syntax: SASECRSP Interface Engine **3284**
The LIBNAME *libref* SASECRSP Statement 3284
Details: SASECRSP Interface Engine **3288**
Using the INSET= Option 3288
The SAS Output Data Set 3290
Understanding CRSP Date Formats, Informats, and Functions 3291
Data Elements Reference: SASECRSP Interface Engine 3295
Available CRSP Stock Data Sets 3296
Available CRSP Indices Data Sets 3301
Examples: SASECRSP Interface Engine **3313**
Example 46.1: Specifying PERMNOs and Range in the LIBNAME Statement 3313
Example 46.2: Using the LIBNAME Statement to Access All Keys 3315
Example 46.3: Accessing One PERMNO without the RANGE= Option 3317
Example 46.4: Specifying Keys by Using the INSET= Option 3319
Example 46.5: Specifying Ranges for Individual Keys with the INSET= Option 3321
Example 46.6: Converting Dates by Using the CRSP Date Functions 3323
References **3324**

Overview: SASECRSP Interface Engine

Introduction

The SASECRSP interface engine in SAS/ETS software enables SAS users to access and process time series, events, portfolios, and group data that reside in Center for Research in Security Prices databases (CRSPAccess data). It also provides a seamless interface between CRSP and SAS data processing. Currently, the SASECRSP engine supports access of CRSP US Stock Databases and CRSP Indices Databases.

Opening a Database

The SASECRSP interface engine uses the LIBNAME statement to enable you to specify which CRSPAccess database you want to access and how you want to select time series or events from that database.

To specify the database, you supply the combination of a physical path to indicate the location of the CRSPAccess data files and a set identifier (SETID) to identify the selected database from those available at the physical path. Specify one SETID from Table 46.1. Notice that the CRSP environment variable CRSPDB_SASCAL must be defined before the SASECRSP engine can access the CRSPAccess database calendars that provide the time ID variables and enable the libref to be assigned successfully. If your database SETID is 250, use the SASEXCCM interface to access your data. For more information about the SASEXCCM interface engine, see Chapter 51, "The SASEXCCM Interface Engine." Because CRSP no longer supports the CPZ data format, the SASECRSP engine no longer supports the SETID 200 (CRSP/Compustat Merged, CCM) data access.

Table 46.1 CRSPAccess Databases SETIDs

SETID	Data Set
10	CRSP Stock, daily data
20	CRSP Stock, monthly data
400	CRSP Indices data, monthly index groups
420	CRSP Indices data, monthly index series
440	CRSP Indices data, daily index groups
460	CRSP Indices data, daily index series

Usually you do not want to open the entire CRSPAccess database, so for efficiency and ease of use, the SASECRSP engine supports a variety of options for performing data selection on your CRSPAccess database by using the LIBNAME statement. These options enable you to open and retrieve data for only the portion of the database that you want. The availability of some of these options depends on the type of database that you open.

CRSP US Stock Databases

When accessing the CRSP US Stock Databases, you can select which securities to access by specifying their PERMNOs with the PERMNO= option. A PERMNO is CRSP's unique permanent issue identification number and the primary key for its stock databases. Alternatively, a number of secondary keys can be used to select stock data. For example, you can use the PERMCO= option to read selected securities based

on CRSP's unique permanent company identification number, PERMCO. A full list of possible keys for accessing CRSP Stock data is shown in Table 46.2.

Table 46.2 Keys for Accessing CRSP Stock Data

Key	Access By
PERMNO	CRSP's unique permanent issue identification number. This is the primary key for CRSP Stock Databases.
PERMCO	CRSP's unique permanent company identification number
CUSIP	CUSIP number
HCUSIP	Historical CUSIP
SICCD	Standard industrial classification (SIC) code
TICKER	Ticker symbol (for active companies only)

CRSP/Compustat Merged Databases—No Longer Supported by the SASECRSP Engine

Use the SASEXCCM interface engine instead of the SASECRSP interface engine to access your Xpressfeed CCM data. The SASEXCCM interface engine provides data item handling access methods by using CRSPAccess version 3.23. For a detailed description of this new SAS/ETS interface engine, see Chapter 51, "The SASEXCCM Interface Engine."

Because CRSPAccess version 3.23 does not support CPZ data (legacy Compustat data format for SETID 200), the SASECRSP engine issues the following error messages when you specify the SETID=200 option and/or the CRSPLINKPATH= option:

```
ERROR: Use the SASEXCCM engine instead of the SASECRSP engine for CCM access.
       The CPZ data format needed for SETID=200 and the CRSPLINKPATH= options
       was last shipped in July 2011 and is no longer supported by CRSP.
       Use of the SASECRSP engine for this purpose is not allowed:
       Depreciated calendar configurations can result in fatal errors,
       corrupted memory, tracebacks, exceptions, or incorrect results
       for all libref assignments that follow the deassignment
       of a CCM/CRSPLINKPATH libref.
ERROR: Engine is unable to open crspdb CPZ200606
       with SETID 200. Check that your CRSP database contains the
       crsp_ca_ref_2.bin file.
```

CRSP Indices Databases

When accessing the CRSP Indices Databases, you can select which indices to access by specifying their INDNOs. INDNO is the primary key for the CRSP Indices Databases. You can specify which INDNO to use by specifying the INDNO= option. No secondary key access is supported for CRSP Indices. A full list of possible keys for accessing CRSP Indices data is shown in Table 46.3.

Table 46.3 Keys for Accessing CRSP Indices Data

Key	Access By
INDNO	CRSP's unique permanent index identifier number. This is the primary key for CRSP Indices Databases. It enables you to specify which index series or groups you want to select.

Regardless of which database you access, you can always use the INSET= and RANGE= options for subsetting and selection. The RANGE= option subsets the data by date. The INSET= option enables you to specify which issues or companies to select from the CRSP Indices data by using an input SAS data set.

Using Your Opened Database

After the libref is assigned, the database is opened. You can retrieve data for any member that you want in the opened database. For a complete description of available data sets and their fields, see the section "Data Elements Reference: SASECRSP Interface Engine" on page 3295. You can also use the SAS DATA step to perform further subsetting and to store the resulting time series in a SAS data set. Because CRSP and SAS use three different date representations, you can use the CRSP date formats, informats, and functions that the SASECRSP engine provides for your data processing needs. For more information about dates in the SASECRSP engine, see the section "Understanding CRSP Date Formats, Informats, and Functions" on page 3291 and Example 46.6 later in this chapter.

The SASECRSP engine supports Linux X64 (64-bit), Solaris Sun Ultra Sparc, Solaris on Intel x86, and Windows. Windows no longer requires you to install the CRSPAccess API, because it is now distributed automatically by your SAS/ETS installation. Prior to running SASECRSP, your Windows setup requires that the environment variable, CRSPDB_SASCAL, be set to the path where your database calendar files reside.

Getting Started: SASECRSP Interface Engine

Structure of a SAS Data Set That Contains Time Series Data

SAS requires time series data to be in a specific form that is recognizable by the SAS System. This form is a two-dimensional array, called a SAS data set, whose columns correspond to series variables and whose rows correspond to measurements of these variables at certain points in time. The time at which observations are recorded can be included in the data set as a time ID variable. Because CRSP sets the date at the end of a time period instead of at the beginning, the SASECRSP interface engine follows this convention. For example, the time ID variable for any particular month in a monthly time series is set to the last trading day of that month.

The SASECRSP engine provides several different time ID variables, depending on the data member that is opened. For most members, a time ID variable named CALDT is provided. CALDT provides a day-based calendar date and is in a CRSP date format. The dates are stored as an offset in an array of trading days or a trading-day calendar. Five different CRSP trading-day calendars are available; which calendar is used depends on the frequency of the data member. For example, the CRSP date for a daily time series refers to a daily trading-day calendar.

The five trading-day calendar frequencies are annual, quarterly, monthly, weekly, and daily. For convenience, the format and informat for the frequency field are set so that the CRSP date is automatically converted to an integer date representation when viewed or printed. For data programming, the SASECRSP engine provides 23 different user functions for date conversions between CRSP, SAS, and integer dates.

Reading CRSP Data Files

The SASECRSP engine supports reading time series, events, portfolios, and group data from CRSPAccess databases. The SETID that you specify determines the database that is read. See Table 46.1 for a list of possible databases. The CRSP environment variable CRSPDB_SASCAL must be defined before the SASECRSP engine can access the CRSPAccess database calendars that provide the time ID variables and enable the libref to be successfully assigned.

Using the SAS DATA Step

You can store the selected series in a SAS data set by using the SAS DATA step. You can also perform other operations on your data inside the DATA step. After the data are stored in a SAS data set, you can use them as you would use data in any other SAS data set.

Using SAS Procedures

You can print the output SAS data set by using the PRINT procedure, and you can report information about the contents of your data set by using the CONTENTS procedure.

You can also create a view of the CRSPAccess database by using the SQL procedure in conjunction with a SASECRSP libref.

Using the SAS Windowing Environment

You can see the available data sets in the SAS LIBNAME window of the SAS windowing environment. To do so, select the SASECRSP engine libref in the LIBNAME window that you have previously defined in your LIBNAME statement. You can view your SAS output observations by double-clicking the desired output data set libref in the LIBNAME window of the SAS windowing environment. Type **Viewtable** on the SAS command line to view any of your SASECRSP engine tables, views, or librefs both for input and output data sets.

Before you use the **Viewtable** command, it is recommended that you store your output data sets in a physical folder or library that is separate from the folder or library used for your input databases. (The default location for output data sets is the SAS Work library.)

Using CRSP Date Formats, Informats, and Functions

Historically, CRSP has used two different methods to represent dates, and SAS has used a third. The SASECRSP engine provides 23 functions, 15 informats, and 10 formats to enable you to easily translate the dates from one internal representation to another. For more information, see the section "Understanding CRSP Date Formats, Informats, and Functions" on page 3291.

Syntax: SASECRSP Interface Engine

The SASECRSP engine uses standard engine syntax. The options that the SASECRSP engine uses are summarized in Table 46.4.

Table 46.4 Summary of SASECRSP Engine Options

Option	Description
SETID=	specifies which CRSP database subset to open. This option is required. See Table 46.1 for complete list of supported SETIDs.
PERMNO=	specifies a CRSP PERMNO to be selected for access
PERMCO=	specifies a CRSP PERMCO to be selected for access
CUSIP=	specifies a current CUSIP to be selected for access
HCUSIP=	specifies a historic CUSIP to be selected for access
TICKER=	specifies a ticker to be selected for access (for active companies only)
SICCD=	specifies a SIC code to be selected for access
INDNO=	specifies a CRSP INDNO to be selected for access
RANGE=	specifies the range of data to keep in format 'YYYYMMDD-YYYYMMDD'
INSET=	uses a SAS data set named Setname as input for issues

The LIBNAME *libref* SASECRSP Statement

LIBNAME *libref* SASECRSP '*physical name*' *options* ;

The physical name that the LIBNAME statement requires should point to the directory of CRSPAccess data files where the CRSP database that you want to open is located. Note that the physical name **must** end in a slash for UNIX environments and a backslash for Windows environments.

The CRSP environment variable CRSPDB_SASCAL must be defined before the SASECRSP engine can access the CRSPAccess database calendars. The CRSP environment variable CRSPDB_SASCAL is necessary for the SASECRSP libref to be assigned successfully. This environment variable should be defined automatically either by the CRSP software installation or, in later versions, by the CRSP data installation. Because occasionally the variable is not set properly, always check to ensure that the CRSPDB_SASCAL environment variable is set to the location where your most recent CRSP data reside. Remember to include the final slash or backslash as required.

After the libref is assigned, you can access any of the available data sets or members within the opened database. For a complete description of available data sets and their fields, see the section "Data Elements Reference: SASECRSP Interface Engine" on page 3295.

You can specify the following options.

SETID=*crsp_setidnumber*

specifies the CRSP database that you want to read from. SETID= is a required option. Choose one SETID from the six possible values in Table 46.1. The SETID limits the frequency selection of time series that are included in the SAS data set.

For example, to access monthly CRSP US Stock data, you would use the following statements:

```
LIBNAME myLib sasecrsp 'physical-name'
                       SETID=20;
```

PERMNO=*crsp_permnumber*

enables you to select data from your CRSP database by the PERMNO (or other keys) that you specify. A PERMNO is CRSP's unique permanent issue identification number. There is no limit to the number of *crsp_permnumber* options that you can use. By default, the SASECRSP engine reads all keys for the CRSPAccess database that you specified in your SASECRSP libref.

From a performance standpoint, the PERMNO= option enables efficient random access and reads *only* the data for the PERMNOs specified.

For example, the following LIBNAME statement reads data only for Microsoft Corporation (PERMNO=10107) and International Business Machines Corporation (PERMNO=12490) by using the primary PERMNO key and thus is very efficient:

```
LIBNAME myLib sasecrsp 'physical-name'
   SETID=20
   PERMNO=10107
   PERMNO=12490;
```

The PERMCO=, CUSIP=, HCUSIP=, SICCD=, TICKER=, and INDNO= options behave similarly, and you can use them in conjunction with or in place of the PERMNO= option. For example, you could use the following statement to access monthly data for Microsoft and IBM:

```
LIBNAME myLib sasecrsp 'physical-name'
                       SETID=20
                       TICKER='MSFT'
                       CUSIP=59491810;
```

Details about the use of the other key selection options (PERMCO, CUSIP, HCUSIP, TICKER, SICCD, and INDNO) follow.

PERMNOs that you specify by using this option can select the companies or issues to keep for CRSP US Stock data, but PERMNO is not a supported option for CRSP Indices data. Use the INDNO= option with the CRSP Indices data and use the PERMNO= option with the CRSP US Stock data. Details on the use of key selection options for each type of database follows.

STK Databases

PERMNO is the primary key for CRSP Stock data. Every valid PERMNO that you specify with the PERMNO= option keeps exactly one issue.

IND Databases

INDNO is the primary key for accessing CRSP Indices data. PERMNO is not available as a key for the IND (CRSP Indices) Databases; use INDNO for efficient access of the IND Databases.

PERMCO=*crsp_permcompany*
is similar to the PERMNO= option in that it enables you to use the CRSP's unique permanent company identification key (PERMCO) to select the companies or issues to keep. There is no limit to the number of *crsp_permcompany* options that you can use.

STK Databases

PERMCO is a secondary key for accessing CRSP US Stock data. One PERMCO can map to multiple PERMNOs. Access by a PERMCO key is equivalent to access by all mapped PERMNOs.

IND Databases

Use INDNO for accessing CRSP Indices data. PERMCO is not available as a key for accessing CRSP Indices data; use INDNO instead.

CUSIP=*crsp_cusip*
is similar to the PERMNO= option in that it enables you to use the CUSIP key to select the companies or issues to keep. There is no limit to the number of *crsp_cusip* options that you can use.

STK Databases

CUSIP is a secondary key for accessing CRSP US Stock data. One CUSIP maps to one PERMNO.

IND Databases

Use INDNO for accessing CRSP Indices data. CUSIP is not available as a key for accessing CRSP Indices Databases; use INDNO instead.

HCUSIP=*crsp_hcusip*
is similar to the PERMNO= option in that it enables you to use the historical CUSIP key, HCUSIP, to select the companies or issues to keep. There is no limit to the number of *crsp_hcusip* options that you can use.

STK Databases

HCUSIP is a secondary key for accessing CRSP US Stock Databases. One HCUSIP maps to one PERMNO.

IND Databases

Use INDNO for accessing CRSP Indices Databases. HCUSIP is not available as a key for accessing CRSP Indices Databases; use INDNO instead.

TICKER=*crsp_ticker*
is similar to the PERMNO= option in that it enables you to use the TICKER key to select the companies or issues to keep. There is no limit to the number of *crsp_ticker* options that you can use.

STK Databases

TICKER is a secondary key for accessing CRSP US Stock Databases. One TICKER maps to one PERMNO. **NOTE:** Some PERMNOs are inaccessible by the TICKER key.

IND Databases

Use INDNO for accessing CRSP Indices Databases. TICKER is not available as a key for accessing CRSP Indices Databases; use INDNO instead.

SICCD=*crsp_siccd*
is similar to the PERMNO= option in that it enables you to use the Standard Industrial Classification (SIC) code (SICCD) to select the companies or issues to keep. There is no limit to the number of *crsp_siccd* options that you can use.

STK Databases

SICCD is a secondary key for accessing CRSP US Stock Databases. One SICCD can map to multiple PERMNOs. All PERMNOs that have been classified once under the specified SICCD are mapped and the data for them are retrieved. Access by the SICCD key is equivalent to access by all PERMNOs that have ever been classified under the specified SICCD key.

IND Databases

Use INDNO for accessing CRSP Indices Databases. SICCD is not available as a key for accessing CRSP Indices Databases; use INDNO instead.

INDNO=*crsp_indno*
is similar to the PERMNO= option in that it enables you to use CRSP's permanent index number INDNO to select the companies or issues to keep. There is no limit to the number of *crsp_indno* options that you can use.

STK Databases

INDNO is not available as a key for accessing CRSP US Stock Databases, but it can be used in the combined CRSP US Stock and Indices Databases.

IND Databases

INDNO is the primary key for accessing CRSP Indices Databases. Every INDNO that you specify keeps exactly one index series or group.

For example, you can use the following statement to access the CRSP NYSE Value-Weighted and Equal-Weighted daily market indices:

```
LIBNAME myLib3 sasecrsp 'physical-name'
                        SETID=460
                        INDNO=1000000   /* Value-Weighted */
                        INDNO=1000001;  /* Equal-Weighted */
```

RANGE='crsp_begdt-crsp_enddt'
limits the time range of data that are read from your CRSPAccess database. Specify this option in your LIBNAME *libref* SASECRSP statement, where *crsp_begdt* is the beginning date of the range in YYYYMMDD format and *crsp_enddt* is the ending date of the range in YYYYMMDD format.

For example, to access monthly stock data for Microsoft Corporation and for International Business Machines Corporation for the first quarter of 1999, you can use the following statement:

```
LIBNAME myLib sasecrsp 'physical-name'
                       SETID=20
                       PERMNO=10107
                       PERMNO=12490
                       RANGE='19990101-19990331';
```

The specified beginning and ending dates are interpreted as calendar dates.

You can use the RANGE= option for all members of CRSP US Stock and Indices Databases. CRSP data members are associated with only one date, and all CRSP data members have a date resolution to the day. For example, monthly time series, although they are monthly, resolve to the last trading day of the month.

INSET=*'setname[,keyfieldname,keyfieldtype,date1field,date2field]'*
specifies a SAS data set named *setname* as input for issues. The SASECRSP engine assumes that a default PERMNO field that contains selected CRSP PERMNOs is present in the data set. If optional parameters are used, they must all be specified. The only acceptable shorthand for dropping the parameters is to drop those at the very end, assuming they are all being omitted. Dropped parameters use their defaults.

You can specify the following parameters:

keyfieldname	labels the field that contains the keys to be selected. If unspecified, the default is PERMNO.
keyfieldtype	specifies the CRSPAccess key type of the provided keys. Possible key types are: PERMNO, PERMCO, CUSIP, HCUSIP, TICKER, SICCD, or INDNO. If unspecified, the default is "PERMNO".
date1field	specifies the beginning date of the specific date range restriction being applied to this key. If either *date1field* or *date2field* is omitted, then by default there is no date range restriction.
date2field	specifies the ending date of the specific date range restriction being applied to this key. If either *date1field* or *date2field* is omitted, then by default there is no date range restriction.

Individual date range restrictions that you specify by using the INSET= option can be used in combination with the RANGE= option in the LIBNAME statement. In such a case, only data from the intersection of the individual date restriction and the global RANGE= option date restriction are read.

Details: SASECRSP Interface Engine

Using the INSET= Option

The following examples illustrate the use of the INSET= option.

Basic INSET= Option Use: Providing a List of PERMNOs

This example uses the INSET= option to extract monthly data for a portfolio of three companies. No date range restriction is used.

```
data testin1;
   permno = 10107; output;
   permno = 12490; output;
```

```
   permno = 14322; output;
run;

LIBNAME mstk sasecrsp 'physical-name'
                      SETID=20
                      INSET='testin1';

proc print data=mstk.stkhead (keep=permno permco begdt enddt hcomnam htick);
run;
```

General Use of the INSET= Option to Specify Lists of Keys

This example illustrates the use of the INSET= option to select a few index series from the CRSP Indices data, and securities from the CRSP US Stock data. The libref **ind2** is used for accessing the CRSP Indices data by using the two specified INDNO keys. The libref **sec3** is used to access the CRSP US Stock data by using the three specified TICKER keys. Note the use of shorthand in specifying the INSET= option. The *date1field* and the *date2field*, and datetype arguments are all omitted, so the default of no range restriction applies (though the range restriction set by the RANGE= option in the LIBNAME statement still applies). For more information, including sample output, see Example 46.4

```
data indices;
    indno=1000000; output;  /* NYSE Value-Weighted Market Index */
    indno=1000001; output;  /* NYSE Equal-Weighted Market Index */
run;

libname ind2 sasecrsp "%sysget(CRSP_MSTK)" setid=420
    inset='indices,INDNO,INDNO' range='19990101-19990401';

title2 'Total Returns for NYSE Value- and Equal-Weighted Market Indices';
proc print data=ind2.tret label;
run;


data securities;
    ticker='BAC'; output;  /* Bank of America */
    ticker='DUK'; output;  /* Duke Energy */
    ticker='GSK'; output;  /* GlaxoSmithKline */
run;

libname sec3 sasecrsp "%sysget(CRSP_MSTK)" setid=20
                       inset='securities,TICKER,TICKER'
                       range='19970820-19970920';

title2 'PERMNOs and General Header Info of Selected TICKERs';
proc print data=sec3.stkhead (keep=permno htick htsymbol) label;
run;

title3 'Average Price for Bank of America, Duke and GlaxoSmithKline';
proc print data=sec3.prc label; run;
```

Key-Specific Date Range Restriction with the INSET= Option

Suppose you not only want to select keys with your INSET= option, but you also want to specify a date range restriction for each key individually. The following statements show how to do this. Again, shorthand enables you to omit the *date1field* and *date2field* arguments. The dates that are provided default to a calendar interpretation. For more information, including the sample output, see Example 46.5.

```
title2 'INSET=testin2 uses date ranges along with PERMNOs:';
title3 '10107, 12490, 14322, 25788';
title4 'Begin dates and end dates for each permno are used in the INSET';

data testin2;
    permno = 10107; date1 = 19980731; date2 = 19981231; output;
    permno = 12490; date1 = 19970101; date2 = 19971231; output;
    permno = 14322; date1 = 19950731; date2 = 19960131; output;
    permno = 25778; date1 = 19950101; date2 = 19950331; output;
run;

libname mstk2 sasecrsp "%sysget(CRSP_MSTK)" setid=20
                       inset='testin2,PERMNO,PERMNO,DATE1,DATE2';

data b;
    set mstk2.prc;
run;

proc print data=b;
run;
```

The SAS Output Data Set

You can use the SAS DATA step to write the selected CRSP data to a SAS data set. This enables you to easily analyze the data by using SAS. When you specify the name of the output data set in the DATA statement, the engine supervisor creates a SAS data set by using the specified name in either the SAS Work library or, if specified, the User library.

The contents of the SAS data set include the date of each observation, the series name of each series read from the CRSPAccess database, event variables, and the label or description of each series/event or array.

You can use PROC PRINT and PROC CONTENTS to print your output data set and its contents. Alternatively, you can view your SAS output observations by opening the desired output data set in a SAS Explorer window. You can also use PROC SQL with the SASECRSP engine libref to create a custom view of your data.

In general, CRSP missing values are represented as '.' in the SAS data set. When accessing the CRSP US STOCK data, the SASECRSP engine uses the mapping shown in Table 46.5 for converting CRSP missing values into SAS missing codes.

Table 46.5 Mapping of CRSP Stock Missing Values to SAS Missing Codes

CRSP Stock	SAS	Condition
–99	.	No valid price
–88	.A	Out of range
–77	.B	Off-exchange
–66	.C	No valid previous price
–55	.D	No delisting information
–44	.E	No valid comparison for an excess return

Understanding CRSP Date Formats, Informats, and Functions

CRSP has historically used two different methods to represent dates, whereas SAS has used a third. The three formats are SAS dates, CRSP dates, and integer dates. The SASECRSP engine provides 23 functions, 15 informats, and 10 formats to enable you to easily translate the dates from one internal representation to another. A SASECRSP engine libref must be assigned prior to your use of the CRSP date formats, informats, and functions. See Example 46.6, "Converting Dates Using the CRSP Date Functions."

SAS dates are stored internally as the number of days since January 1, 1960. The SAS method is an industry standard that provides a great deal of flexibility, including a wide variety of informats, formats, and functions.

CRSP dates are designed to ease time series storage and access. Internally, the dates are stored as an offset in an array of trading days or a trading-day calendar. There are five different CRSP trading-day calendars: Annual, Quarterly, Monthly, Weekly, and Daily. In this sense, there are five different types of CRSP dates, one for each frequency of calendar that it references. The CRSP method provides fewer missing values and makes trading period calculations very easy. However, many valid calendar dates are not available in the CRSP trading calendars, and you must be careful when you use other dates.

Integer dates are a way to represent dates that are platform-independent and maintain the correct sort order. However, the distance between dates is not maintained.

The best way to illustrate the various date formats is to use some sample data. Table 46.6 shows date representations for CRSP daily and monthly data.

Table 46.6 Date Representations for Daily and Monthly Data

Date	SAS Date	CRSP Date (Daily)	CRSP Date (Monthly)	Integer Date
July 31, 1962	942	21	440	19620731
August 31, 1962	973	44	441	19620831
Dec. 30, 1998	14,243	9190	NA*	19981230
Dec. 31, 1998	14,244	9191	877	19981231

* Not available if an exact match is requested.

Having an understanding of the internal differences in representing SAS dates, CRSP dates, and CRSP integer dates helps you use the SASECRSP engine formats, informats, and functions effectively. Always keep in mind the frequency of the CRSP calendar that you are accessing when you specify a CRSP date.

The CRSP Date Formats

CRSP dates use two types of formats, and five frequencies are available for each type. The two types are exact dates (CRSPDT*) and range dates (CRSPDR*), where the '*' can be A for annual, Q for quarterly, M for monthly, W for weekly, or D for daily. The ten types are CRSPDTA, CRSPDTQ, CRSPDTM, CRSPDTW, CRSPDTD, CRSPDRA, CRSPDRQ, CRSPDRM, CRSPDRW, and CRSPDRD.

Table 46.7 shows some samples that use the monthly and daily calendar as examples. The Annual (CRSPDTA and CRSPDRA), Quarterly (CRSPDTQ and CRSPDRQ), and Weekly (CRSPDTW and CRSPDRW) formats work analogously.

Table 46.7 Sample CRSPDT Formats for Daily and Monthly Data

Date	CRSP Date Daily, Monthly	CRSPDTD Daily Date	CRSPDRD Daily Range	CRSPDTM Monthly Date	CRSPDRM Monthly Range
July 31,1962	21, 440	19620731	19620731 +	19620731	19620630, 19620731
August 31,1962	44, 441	19620831	19620831 +	19620831	19620801, 19620831
Dec. 30,1998	9190, NA *	19981230	19981230 +	NA*	NA*
Dec. 31,1998	9191, 877	19981231	19981231 +	19981231	19981201, 19981231

+ Daily ranges look similar to monthly ranges if they are Mondays or immediately follow a trading holiday.
* When you are working with exact matches, no CRSP monthly date exists for December 30, 1998.

The @CRSP Date Informats

CRSP dates use three types of informats, and five frequencies are available for each type. The three types are exact (@CRSPDT*), range (@CRSPDR*), and backward (@CRSPDB*) dates, where the '*' can be A for annual, Q for quarterly, M for monthly, W for weekly, or D for daily. The 15 formats are @CRSPDTA, @CRSPDTQ, @CRSPDTM, @CRSPDTW, @CRSPDTD, @CRSPDRA, @CRSPDRQ, @CRSPDRM, @CRSPDRW, @CRSPDRD, @CRSPDBA, @CRSPDBQ, @CRSPDBM, @CRSPDBW, and @CRSPDBD.

The five CRSPDT* informats find exact matches only. The five CRSPDR* informats look for an exact match, and if an exact match is not found, they go forward, matching the CRSPDR* formats. The five CRSPDB* informats look for an exact match, and if an exact match is not found, they go backward.

Table 46.8 shows a sample that uses only the CRSP monthly calendar as an example. The daily, weekly, quarterly, and annual frequencies work analogously.

Table 46.8 Sample @CRSP Date Informats Using Monthly Data

Input Date (Integer Date)	CRSP Date CRSPDTM	CRSP Date CRSPDRM	CRSP Date CRSPDBM	CRSPDTM Monthly Date	CRSPDRM Monthly Range
19620731	440	440	440	19620731	19620630 to 19620731
19620815	.(missing)	441	440	See below+	See below*
19620831	441	441	441	19620831	19620801 to 19620831

+ Missing values are preserved. If 441, then 19620831. If 440, then 19620731.
* Missing values are preserved. If 441, then 19620801 to 19620831. If 440, then 19620630 to 19620731.

The CRSP Date Functions

Table 46.9 shows the 22 date functions that the SASECRSP engine provides. The engine uses these functions internally, but they are also available to end users. There are seven groups of functions. The first four groups have five functions each, one for each CRSP calendar frequency. The next two functions are for converting between SAS and integer date formats.

Table 46.9 CRSP Date Functions

Function Group	Function Name	Argument One	Argument Two	Return Value
CRSP dates to integer dates for December 31, 1998				
Annual	crspdcia	74	None	19981231
Quarterly	crspdciq	293	None	19981231
Monthly	crspdcim	877	None	19981231
Weekly	crspdciw	1905	None	19981231
Daily	crspdcid	9191	None	19981231
CRSP dates to SAS dates for December 31, 1998				
Annual	crspdcsa	74	None	14,244
Quarterly	crspdcsq	293	None	14,244
Monthly	crspdcsm	877	None	14,244
Weekly	crspdcsw	1905	None	14,244
Daily	crspdcsd	9191	None	14,244
Integer dates to CRSP dates exact is illustrated, but can be forward or backward				
Annual	crspdica	19981231	0	74
Quarterly	crspdicq	19981231	0	293
Monthly	crspdicm	19981231	0	877
Weekly	crspdicw	19981231	0	1905
Daily	crspdicd	19981231	0	9191
SAS dates to CRSP dates exact is illustrated, but can be forward or backward				
Annual	crspdsca	14,244	0	74
Quarterly	crspdscq	14,244	0	293
Monthly	crspdscm	14,244	0	877
Weekly	crspdscw	14,244	0	1905
Daily	crspdscd	14,244	0	9191
Integer dates to SAS dates for December 31, 1998				
Integer to SAS	crspdi2s	19981231	None	14,244
SAS dates to integer dates for December 31, 1998				
SAS to Integer	crspds2i	14,244	None	19981231

Data Elements Reference: SASECRSP Interface Engine

Data sets are made available based on the type of CRSP database that you open. Table 46.10 and Table 46.11 show summary views of the two types of CRSP databases (Stock and Indices) and the data sets that they make available. Tables that contain details about the data sets, including their specific fields, immediately follow the summary tables. You can also see the available data sets for an opened database via the SAS Explorer by opening a SASECRSP engine libref that you previously assigned.

Table 46.10 Summary of All Available Data Sets by CRSP Database Type

CRSP Database	Data Set Name	Reference Table Title	Reference Table
CRSP US Stock Database (STOCK)	STKHEAD	Header Identification and Summary Data	Table 46.12
	NAMES	Name History Array	Table 46.13
	DISTS	Distribution Event Array	Table 46.14
	SHARES	Shares Outstanding Observation Array	Table 46.15
	DELIST	Delisting History Array	Table 46.16
	NASDIN	NASDAQ Information Array	Table 46.17
	PRC	Price or Bid/Ask Average Time Series	Table 46.18
	RET	Returns Time Series	Table 46.18
	BIDLO	Bid or Low Price Time Series	Table 46.18
	ASKHI	Ask or High Price Time Series	Table 46.18
	BID	Bid Time Series	Table 46.18
	ASK	Ask Time Series	Table 46.18
	RETX	Returns Without Dividends Time Series	Table 46.18
	SPREAD	Spread Between Bid and Ask	Table 46.18
	ALTPRC	Price Alternate Time Series	Table 46.18
	VOL	Volume Time Series	Table 46.18
	NUMTRD	Number of Trades Time Series	Table 46.18
	ALTPRCDT	Price Alternate Date Time Series	Table 46.18
	PORT1	Portfolio Data for Portfolio Type 1	Table 46.19
	PORT2	Portfolio Data for Portfolio Type 2	Table 46.19
	PORT3	Portfolio Data for Portfolio Type 3	Table 46.19
	PORT4	Portfolio Data for Portfolio Type 4	Table 46.19
	PORT5	Portfolio Data for Portfolio Type 5	Table 46.19
	PORT6	Portfolio Data for Portfolio Type 6	Table 46.19
	PORT7	Portfolio Data for Portfolio Type 7	Table 46.19
	PORT8	Portfolio Data for Portfolio Type 8	Table 46.19
	PORT9	Portfolio Data for Portfolio Type 9	Table 46.19
	GROUP16	Group Data for Group Type 16	Table 46.19

Table 46.11 Summary of All Available Data Sets by CRSP Database Type

CRSP Database	Data Set Name	Reference Table Title	Reference Table
CRSP Indices Database (IND)	INDHEAD	Index Header Data	Table 46.20
	REBAL	Index Rebalancing History Arrays	Table 46.21
	REBAL	Index Rebalancing History Group Arrays	Table 46.22
	LIST	Index Membership List Arrays	Table 46.23
	LIST	Index Membership List Groups Arrays	Table 46.24
	USDCNT	Portfolio Used Count Array	Table 46.25
	TOTCNT	Portfolio Total Count Array	Table 46.26
	USDCNT	Portfolio Used Count Time Series Groups	Table 46.27
	TOTCNT	Portfolio Total Count Time Series Groups	Table 46.28
	USDVAL	Portfolio Used Value Array	Table 46.29
	TOTVAL	Portfolio Total Value Array	Table 46.30
	USDVAL	Portfolio Used Value Time Series Groups	Table 46.31
	TOTVAL	Portfolio Total Value Time Series Groups	Table 46.32
	TRET	Total Returns Time Series	Table 46.33
	ARET	Appreciation Returns Time Series	Table 46.34
	IRET	Income Returns Time Series	Table 46.35
	TRET	Total Returns Time Series Groups	Table 46.36
	ARET	Income Returns Time Series Groups	Table 46.37
	IRET	Income Returns Time Series Groups	Table 46.38
	TIND	Total Return Index Levels Time Series	Table 46.39
	AIND	Appreciation Index Levels Time Series	Table 46.40
	IIND	Income Index Levels Time Series	Table 46.41
	TIND	Total Return Index Levels Groups	Table 46.42
	AIND	Appreciation Index Levels Groups	Table 46.43
	IIND	Income Index Levels Time Series Groups	Table 46.44

Available CRSP Stock Data Sets

STKHEAD Data Set—Header Identification and Summary Data

Table 46.12 STKHEAD Data Set—Header Identification and Summary Data

Field	Label	Type
PERMNO	PERMNO	Numeric
PERMCO	PERMCO	Numeric
COMPNO	NASDAQ Company Number	Numeric
ISSUNO	NASDAQ Issue Number	Numeric
HEXCD	Exchange Code Header	Numeric
HSHRCD	Share Code Header	Numeric
HSICCD	Standard Industrial Classification Code	Numeric
BEGDT	Begin of Stock data	Numeric
ENDDT	End of Stock data	Numeric

Table 46.12 *continued*

Field	Label	Type
DLSTCD	Delisting Code Header	Numeric
HCUSIP	CUSIP Header	Character
HTICK	Ticker Symbol Header	Character
HCOMNAM	Company Name Header	Character
HTSYMBOL	Trading Symbol Header	Character
HNAICS	North American Industry Classification Header	Character
HPRIMEXC	Primary Exchange Header	Character
HTRDSTAT	Trading Status Header	Character
HSECSTAT	Security Status Header	Character

NAMES Data Set—Name History Array

Table 46.13 NAMES Data Set—Name History Array

Field	Label	Type
PERMNO	PERMNO	Numeric
NAMEDT	Names Date	Numeric
NAMEENDT	Names Ending Date	Numeric
SHRCD	Share Code	Numeric
EXCHCD	Exchange Code	Numeric
SICCD	Standard Industrial Classification Code	Numeric
NCUSIP	CUSIP	Numeric
TICKER	Ticker Symbol	Character
COMNAM	Company Name	Character
SHRCLS	Share Class	Numeric
TSYMBOL	Trading Symbol	Character
NAICS	North American Industry Classification System	Character
PRIMEXCH	Primary Exchange	Character
TRDSTAT	Trading Status	Character
SECSTAT	Security Status	Character

DISTS Data Set—Distribution Event Array

Table 46.14 DISTS Data Set—Distribution Event Array

Field	Label	Type
PERMNO	PERMNO	Numeric
DISTCD	Distribution Code	Numeric
DIVAMT	Dividend Cash Amount	Numeric
FACPR	Factor to Adjust Price	Numeric
FACSHR	Factor to Adjust Share	Numeric
DCLRDT	Distribution Declaration Date	Numeric
EXDT	Ex-Distribution Date	Numeric
RCRDDT	Record Date	Numeric
PAYDT	Payment Date	Numeric

Table 46.14 *continued*

Field	Label	Type
ACPERM	Acquiring PERMNO	Numeric
ACCOMP	Acquiring PERMCO	Numeric

SHARES Data Set—Shares Outstanding Observation Array

Table 46.15 SHARES Data Set—Shares Outstanding Observation Array

Field	Label	Type
PERMNO	PERMNO	Numeric
SHROUT	Shares Outstanding	Numeric
SHRSDT	Shares Observation Date	Numeric
SHRENDDT	Shares Observation End Date	Numeric
SHRFLG	Shares Outstanding Observation Flag	Numeric

DELIST Data Set—Delisting History Array

Table 46.16 DELIST Data Set—Delisting History Array

Field	Label	Type
PERMNO	PERMNO	Numeric
DLSTDT	Delisting Date	Numeric
DLSTCD	Delisting Code	Numeric
NWPERM	New PERMNO	Numeric
NWCOMP	New PERMCO	Numeric
NEXTD	Delisting Next Price Date	Numeric
DLAMT	Delisting Amount	Numeric
DLRETX	Delisting Return Without Dividends	Numeric
DLPRC	Delisting Price	Numeric
DLPDT	Delisting Amount Date	Numeric
DLRET	Delisting Return	Numeric

NASDIN Data Set—NASDAQ Information Array

Table 46.17 NASDIN Data Set—NASDAQ Information Array

Field	Label	Type
PERMNO	PERMNO	Numeric
TRTSCD	NASDAQ Traits Code	Numeric
TRTSDT	NASDAQ Traits Date	Numeric
TRTSENDT	NASDAQ Traits End Date	Numeric
NMSIND	NASDAQ National Market Indicator	Numeric
MMCNT	Market Maker Count	Numeric
NSDINX	NASD Index Code	Numeric

STOCK Time Series Data Sets

Table 46.18 STOCK Time Series Data Sets

Data Set Name, Long Name	Field	Label	Type
PRC	PERMNO	PERMNO	Numeric
Price or Bid/Ask	CALDT	Calendar Trading Date	Numeric
Average Time Series	PRC	Price or Bid/Ask Aver	Numeric
RET	PERMNO	PERMNO	Numeric
Returns Time Series	CALDT	Calendar Trading Date	Numeric
	RET	Returns	Numeric
ASKHI	PERMNO	PERMNO	Numeric
Ask or High Price	CALDT	Calendar Trading Date	Numeric
Time Series	ASKHI	Ask or High Price	Numeric
BIDLO	PERMNO	PERMNO	Numeric
Bid or Low Price	CALDT	Calendar Trading Date	Numeric
Time Series	BIDLO	Bid or Low Price	Numeric
BID	PERMNO	PERMNO	Numeric
Bid Time Series	CALDT	Calendar Trading Date	Numeric
	BID	Bid	Numeric
ASK	PERMNO	PERMNO	Numeric
Ask Time Series	CALDT	Calendar Trading Date	Numeric
	ASK	Ask	Numeric
RETX	PERMNO	PERMNO	Numeric
Returns without	CALDT	Calendar Trading Date	Numeric
Dividends	RETX	Returns w/o Dividends	Numeric
SPREAD	PERMNO	PERMNO	Numeric
Spread Between Bid	CALDT	Calendar Trading Date	Numeric
and Ask Time Series	SPREAD	Spread Between Bid Ask	Numeric
ALTPRC	PERMNO	PERMNO	Numeric
Price Alternate	CALDT	Calendar Trading Date	Numeric
Time Series	ALTPRC	Price Alternate	Numeric
VOL	PERMNO	PERMNO	Numeric
Volume Time Series	CALDT	Calendar Trading Date	Numeric
	VOL	Volume	Numeric
NUMTRD	PERMNO	PERMNO	Numeric
Number of Trades	CALDT	Calendar Trading Date	Numeric
Time Series	NUMTRD	Number of Trades	Numeric
ALTPRCDT	PERMNO	PERMNO	Numeric
Alternate Price	CALDT	Calendar Trading Date	Numeric
Date Time Series	ALTPRCDT	Alternate Price Date	Numeric

Portfolio and Group Data Sets

Table 46.19 Portfolio and Group Data Sets

Data Set	Fields	Label	Type
PORT1	PERMNO	PERMNO	Numeric
Portfolio data	CALDT	Calendar Trading Date	Numeric
for Portfolio	PORT1	Portfolio Assignment for Portfolio Type 1	Numeric
Type 1	STAT1	Portfolio Statistic for Portfolio Type 1	Numeric
PORT2	PERMNO	PERMNO	Numeric
Portfolio data	CALDT	Calendar Trading Date	Numeric
for Portfolio	PORT2	Portfolio Assignment for Portfolio Type 2	Numeric
Type 2	STAT2	Portfolio Statistic for Portfolio Type 2	Numeric
PORT3	PERMNO	PERMNO	Numeric
Portfolio data	CALDT	Calendar Trading Date	Numeric
for Portfolio	PORT3	Portfolio Assignment for Portfolio Type 3	Numeric
Type 3	STAT3	Portfolio Statistic for Portfolio Type 3	Numeric
PORT4	PERMNO	PERMNO	Numeric
Portfolio data	CALDT	Calendar Trading Date	Numeric
for Portfolio	PORT4	Portfolio Assignment for Portfolio Type 4	Numeric
Type 4	STAT4	Portfolio Statistic for Portfolio Type 4	Numeric
PORT5	PERMNO	PERMNO	Numeric
Portfolio data	CALDT	Calendar Trading Date	Numeric
for Portfolio	PORT5	Portfolio Assignment for Portfolio Type 5	Numeric
Type 5	STAT5	Portfolio Statistic for Portfolio Type 5	Numeric
PORT6	PERMNO	PERMNO	Numeric
Portfolio data	CALDT	Calendar Trading Date	Numeric
for Portfolio	PORT6	Portfolio Assignment for Portfolio Type 6	Numeric
Type 6	STAT6	Portfolio Statistic for Portfolio Type 6	Numeric
PORT7	PERMNO	PERMNO	Numeric
Portfolio data	CALDT	Calendar Trading Date	Numeric
for Portfolio	PORT7	Portfolio Assignment for Portfolio Type 7	Numeric
Type 7	STAT7	Portfolio Statistic for Portfolio Type 7	Numeric
PORT8	PERMNO	PERMNO	Numeric
Portfolio data	CALDT	Calendar Trading Date	Numeric
for Portfolio	PORT8	Portfolio Assignment for Portfolio Type 8	Numeric
Type 8	STAT8	Portfolio Statistic for Portfolio Type 8	Numeric
PORT9	PERMNO	PERMNO	Numeric
Portfolio data	CALDT	Calendar Trading Date	Numeric
for Portfolio	PORT9	Portfolio Assignment for Portfolio Type 9	Numeric
Type 9	STAT9	Portfolio Statistic for Portfolio Type 9	Numeric
GROUP16	PERMNO	PERMNO	Numeric
Group data	GRPDT	Group Beginning Date	Numeric
for Group	GRPENDDT	Group Ending Date	Numeric
Type 16	GRPFLAG	Group Flag of Associated Index	Numeric
	GRPSU	Group Subflag	Numeric

Available CRSP Indices Data Sets

INDHEAD Data Set—CRSP Index Header Data

Table 46.20 INDHEAD Data Set—CRSP Index Header Data

Field	Label	Type
INDNO	Permanent index identification number	Numeric
INDCO	Permanent index group identification number	Numeric
PRIMFLAG	Index primary link	Numeric
PORTNUM	Portfolio number if subset series	Numeric
INDNAME	Index Name	Character
GROUPNAM	Index Group Name	Character

REBAL Data Set—Index Rebalancing History Arrays

Table 46.21 REBAL Data Set—Index Rebalancing History Arrays

Field	Label	Type
INDNO	INDNO	Numeric
RBEGDT	Rebalancing beginning date	Numeric
RENDDT	Rebalancing ending date	Numeric
USDCNT	Count used as of rebalancing	Numeric
MAXCNT	Maximum count during period	Numeric
TOTCNT	Available count as of rebalancing	Numeric
ENDCNT	Count at end of period	Numeric
MINID	Identifier at minimum value	Numeric
MAXID	Identifier at maximum value	Numeric
MINSTA	Smallest statistic in period	Numeric
MAXSTA	Largest statistic in period	Numeric
MEDSTA	Median statistic in period	Numeric
AVGSTA	Average statistic in period	Numeric

REBAL Group Data Set—Index Rebalancing History Group Array

Table 46.22 REBAL Group Data Set—Index Rebalancing History Group Array

Field	Label	Type
INDNO	INDNO	Numeric
RBEGDT1	Rebalancing beginning date for port 1	Numeric
RBEGDT2	Rebalancing beginning date for port 2	Numeric
RBEGDT3	Rebalancing beginning date for port 3	Numeric
RBEGDT4	Rebalancing beginning date for port 4	Numeric
RBEGDT5	Rebalancing beginning date for port 5	Numeric
RBEGDT6	Rebalancing beginning date for port 6	Numeric

Table 46.22 *continued*

Field	Label	Type
RBEGDT7	Rebalancing beginning date for port 7	Numeric
RBEGDT8	Rebalancing beginning date for port 8	Numeric
RBEGDT9	Rebalancing beginning date for port 9	Numeric
RBEGDT10	Rebalancing beginning date for port 10	Numeric
RENDDT1	Rebalancing ending date for port 1	Numeric
RENDDT2	Rebalancing ending date for port 2	Numeric
RENDDT3	Rebalancing ending date for port 3	Numeric
RENDDT4	Rebalancing ending date for port 4	Numeric
RENDDT5	Rebalancing ending date for port 5	Numeric
RENDDT6	Rebalancing ending date for port 6	Numeric
RENDDT7	Rebalancing ending date for port 7	Numeric
RENDDT8	Rebalancing ending date for port 8	Numeric
RENDDT9	Rebalancing ending date for port 9	Numeric
RENDDT10	Rebalancing ending date for port 10	Numeric
USDCNT1	Count used as of rebalancing for port 1	Numeric
USDCNT2	Count used as of rebalancing for port 2	Numeric
USDCNT3	Count used as of rebalancing for port 3	Numeric
USDCNT4	Count used as of rebalancing for port 4	Numeric
USDCNT5	Count used as of rebalancing for port 5	Numeric
USDCNT6	Count used as of rebalancing for port 6	Numeric
USDCNT7	Count used as of rebalancing for port 7	Numeric
USDCNT8	Count used as of rebalancing for port 8	Numeric
USDCNT9	Count used as of rebalancing for port 9	Numeric
USDCNT10	Count used as of rebalancing for port10	Numeric
MAXCNT1	Maximum count during period for port 1	Numeric
MAXCNT2	Maximum count during period for port 2	Numeric
MAXCNT3	Maximum count during period for port 3	Numeric
MAXCNT4	Maximum count during period for port 4	Numeric
MAXCNT5	Maximum count during period for port 5	Numeric
MAXCNT6	Maximum count during period for port 6	Numeric
MAXCNT7	Maximum count during period for port 7	Numeric
MAXCNT8	Maximum count during period for port 8	Numeric
MAXCNT9	Maximum count during period for port 9	Numeric
MAXCNT10	Maximum count during period for port 10	Numeric
TOTCNT1	Available count as of rebalancing for port 1	Numeric
TOTCNT2	Available count as of rebalancing for port 2	Numeric
TOTCNT3	Available count as of rebalancing for port 3	Numeric
TOTCNT4	Available count as of rebalancing for port 4	Numeric
TOTCNT5	Available count as of rebalancing for port 5	Numeric
TOTCNT6	Available count as of rebalancing for port 6	Numeric
TOTCNT7	Available count as of rebalancing for port 7	Numeric
TOTCNT8	Available count as of rebalancing for port 8	Numeric
TOTCNT9	Available count as of rebalancing for port 9	Numeric
TOTCNT10	Available count as of rebalancing for port10	Numeric
ENDCNT1	Count at end of period for port 1	Numeric

Table 46.22 *continued*

Field	Label	Type
ENDCNT2	Count at end of period for port 2	Numeric
ENDCNT3	Count at end of period for port 3	Numeric
ENDCNT4	Count at end of period for port 4	Numeric
ENDCNT5	Count at end of period for port 5	Numeric
ENDCNT6	Count at end of period for port 6	Numeric
ENDCNT7	Count at end of period for port 7	Numeric
ENDCNT8	Count at end of period for port 8	Numeric
ENDCNT9	Count at end of period for port 9	Numeric
ENDCNT10	Count at end of period for port 10	Numeric
MINID1	Identifier at minimum value for port 1	Numeric
MINID2	Identifier at minimum value for port 2	Numeric
MINID3	Identifier at minimum value for port 3	Numeric
MINID4	Identifier at minimum value for port 4	Numeric
MINID5	Identifier at minimum value for port 5	Numeric
MINID6	Identifier at minimum value for port 6	Numeric
MINID7	Identifier at minimum value for port 7	Numeric
MINID8	Identifier at minimum value for port 8	Numeric
MINID9	Identifier at minimum value for port 9	Numeric
MINID10	Identifier at minimum value for port 10	Numeric
MAXID1	Identifier at maximum value for port 1	Numeric
MAXID2	Identifier at maximum value for port 2	Numeric
MAXID3	Identifier at maximum value for port 3	Numeric
MAXID4	Identifier at maximum value for port 4	Numeric
MAXID5	Identifier at maximum value for port 5	Numeric
MAXID6	Identifier at maximum value for port 6	Numeric
MAXID7	Identifier at maximum value for port 7	Numeric
MAXID8	Identifier at maximum value for port 8	Numeric
MAXID9	Identifier at maximum value for port 9	Numeric
MAXID10	Identifier at maximum value for port 10	Numeric
MINSTA1	Smallest statistic in period for port 1	Numeric
MINSTA2	Smallest statistic in period for port 2	Numeric
MINSTA3	Smallest statistic in period for port 3	Numeric
MINSTA4	Smallest statistic in period for port 4	Numeric
MINSTA5	Smallest statistic in period for port 5	Numeric
MINSTA6	Smallest statistic in period for port 6	Numeric
MINSTA7	Smallest statistic in period for port 7	Numeric
MINSTA8	Smallest statistic in period for port 8	Numeric
MINSTA9	Smallest statistic in period for port 9	Numeric
MINSTA10	Smallest statistic in period for port 10	Numeric
MAXSTA1	Largest statistic in period for port 1	Numeric
MAXSTA2	Largest statistic in period for port 2	Numeric
MAXSTA3	Largest statistic in period for port 3	Numeric
MAXSTA4	Largest statistic in period for port 4	Numeric
MAXSTA5	Largest statistic in period for port 5	Numeric
MAXSTA6	Largest statistic in period for port 6	Numeric

Table 46.22 *continued*

Field	Label	Type
MAXSTA7	Largest statistic in period for port 7	Numeric
MAXSTA8	Largest statistic in period for port 8	Numeric
MAXSTA9	Largest statistic in period for port 9	Numeric
MAXSTA10	Largest statistic in period for port 10	Numeric
MEDSTA1	Median statistic in period for port 1	Numeric
MEDSTA2	Median statistic in period for port 2	Numeric
MEDSTA3	Median statistic in period for port 3	Numeric
MEDSTA4	Median statistic in period for port 4	Numeric
MEDSTA5	Median statistic in period for port 5	Numeric
MEDSTA6	Median statistic in period for port 6	Numeric
MEDSTA7	Median statistic in period for port 7	Numeric
MEDSTA8	Median statistic in period for port 8	Numeric
MEDSTA9	Median statistic in period for port 9	Numeric
MEDSTA10	Median statistic in period for port 10	Numeric
AVGSTA1	Average statistic in period for port 1	Numeric
AVGSTA2	Average statistic in period for port 2	Numeric
AVGSTA3	Average statistic in period for port 3	Numeric
AVGSTA4	Average statistic in period for port 4	Numeric
AVGSTA5	Average statistic in period for port 5	Numeric
AVGSTA6	Average statistic in period for port 6	Numeric
AVGSTA7	Average statistic in period for port 7	Numeric
AVGSTA8	Average statistic in period for port 8	Numeric
AVGSTA9	Average statistic in period for port 9	Numeric
AVGSTA10	Average statistic in period for port 10	Numeric

LIST Data Set—Index Membership List Arrays

Table 46.23 LIST Data Set—Index Membership List Arrays

Field	Label	Type
INDNO	INDNO	Numeric
PERMNO	Issue identifier	Numeric
BEGDT	First date included	Numeric
ENDDT	Last date included	Numeric
SUBIND	Code for subcategory of list	Numeric
WEIGHT	Weight during range	Numeric

LIST Group Data Set—Index Membership List Group Arrays

Table 46.24 LIST Group Data Set—Index Membership List Group Arrays

Field	Label	Type
INDNO	INDNO	Numeric
PERMNO1	Issue identifier	Numeric

Table 46.24 *continued*

Field	Label	Type
BEGDT1	First date included	Numeric
ENDDT1	Last date included	Numeric
SUBIND1	Code for subcategory of list	Numeric
WEIGHT1	Weight during range	Numeric

USDCNT Data Set—Portfolio Used Count Array

Table 46.25 USDCNT Data Set—Portfolio Used Count Array

Field	Label	Type
INDNO	INDNO	Numeric
CALDT	Calendar Trading Date	Numeric
USDCNT	Portfolio Used Count	Numeric

TOTCNT Data Set—Portfolio Total Count Array

Table 46.26 TOTCNT Data Set—Portfolio Total Count Array

Field	Label	Type
INDNO	INDNO	Numeric
CALDT	Calendar Trading Date	Numeric
TOTCNT	Portfolio Used Count	Numeric

USDCNT Group Data Set—Portfolio Used Time Series Group

Table 46.27 USDCNT Group Data Set—Portfolio Used Time Series Group

Field	Label	Type
INDNO	INDNO	Numeric
CALDT	Calendar Trading Date	Numeric
USDCNT1	Used Count for Port 1	Numeric
USDCNT2	Used Count for Port 2	Numeric
USDCNT3	Used Count for Port 3	Numeric
USDCNT4	Used Count for Port 4	Numeric
USDCNT5	Used Count for Port 5	Numeric
USDCNT6	Used Count for Port 6	Numeric
USDCNT7	Used Count for Port 7	Numeric
USDCNT8	Used Count for Port 8	Numeric
USDCNT9	Used Count for Port 9	Numeric
USDCNT10	Used Count for Port 10	Numeric
USDCNT11	Used Count for Port 11	Numeric
USDCNT12	Used Count for Port 12	Numeric
USDCNT13	Used Count for Port 13	Numeric

Table 46.27 *continued*

Field	Label	Type
USDCNT14	Used Count for Port 14	Numeric
USDCNT15	Used Count for Port 15	Numeric
USDCNT16	Used Count for Port 16	Numeric
USDCNT17	Used Count for Port 17	Numeric

TOTCNT Group Data Set—Portfolio Total Count Time Series Groups

Table 46.28 TOTCNT Group Data Set—Portfolio Total Count Time Series Groups

Field	Label	Type
INDNO	INDNO	Numeric
CALDT	Calendar Trading Date	Numeric
TOTCNT1	Total Count for Port 1	Numeric
TOTCNT2	Total Count for Port 2	Numeric
TOTCNT3	Total Count for Port 3	Numeric
TOTCNT4	Total Count for Port 4	Numeric
TOTCNT5	Total Count for Port 5	Numeric
TOTCNT6	Total Count for Port 6	Numeric
TOTCNT7	Total Count for Port 7	Numeric
TOTCNT8	Total Count for Port 8	Numeric
TOTCNT9	Total Count for Port 9	Numeric
TOTCNT10	Total Count for Port10	Numeric
TOTCNT11	Total Count for Port11	Numeric
TOTCNT12	Total Count for Port12	Numeric
TOTCNT13	Total Count for Port13	Numeric
TOTCNT14	Total Count for Port14	Numeric
TOTCNT15	Total Count for Port15	Numeric
TOTCNT16	Total Count for Port16	Numeric
TOTCNT17	Total Count for Port17	Numeric

USDVAL Data Set—Portfolio Used Value Array

Table 46.29 USDVAL Data Set—Portfolio Used Value Array

Field	Label	Type
INDNO	INDNO	Numeric
CALDT	Calendar Trading Date	Numeric
USDVAL	Portfolio Used Value	Numeric

TOTVAL Data Set—Portfolio Total Value Array

Table 46.30 TOTVAL Data Set—Portfolio Total Value Array

Field	Label	Type
INDNO	INDNO	Numeric
CALDT	Calendar Trading Date	Numeric
TOTVAL	Portfolio Total Value	Numeric

USDVAL Group Data Set—Portfolio Used Value Time Series Groups

Table 46.31 USDVAL Group Data Set—Portfolio Used Value Time Series Groups

Field	Label	Type
INDNO	INDNO	Numeric
CALDT	Calendar Trading Date	Numeric
USDVAL1	Used Value for Port 1	Numeric
USDVAL2	Used Value for Port 2	Numeric
USDVAL3	Used Value for Port 3	Numeric
USDVAL4	Used Value for Port 4	Numeric
USDVAL5	Used Value for Port 5	Numeric
USDVAL6	Used Value for Port 6	Numeric
USDVAL7	Used Value for Port 7	Numeric
USDVAL8	Used Value for Port 8	Numeric
USDVAL9	Used Value for Port 9	Numeric
USDVAL10	Used Value for Port 10	Numeric
USDVAL11	Used Value for Port 11	Numeric
USDVAL12	Used Value for Port 12	Numeric
USDVAL13	Used Value for Port 13	Numeric
USDVAL14	Used Value for Port 14	Numeric
USDVAL15	Used Value for Port 15	Numeric
USDVAL16	Used Value for Port 16	Numeric
USDVAL17	Used Value for Port 17	Numeric

TOTVAL Group Data Set—Portfolio Total Value Time Series Groups

Table 46.32 TOTVAL Group Data Set—Portfolio Total Value Time Series Groups

Field	Label	Type
INDNO	INDNO	Numeric
CALDT	Calendar Trading Date	Numeric
TOTVAL1	Total Value for Port 1	Numeric
TOTVAL2	Total Value for Port 2	Numeric
TOTVAL3	Total Value for Port 3	Numeric
TOTVAL4	Total Value for Port 4	Numeric
TOTVAL5	Total Value for Port 5	Numeric

Table 46.32 *continued*

Field	Label	Type
TOTVAL6	Total Value for Port 6	Numeric
TOTVAL7	Total Value for Port 7	Numeric
TOTVAL8	Total Value for Port 8	Numeric
TOTVAL9	Total Value for Port 9	Numeric
TOTVAL10	Total Value for Port10	Numeric
TOTVAL11	Total Value for Port11	Numeric
TOTVAL12	Total Value for Port12	Numeric
TOTVAL13	Total Value for Port13	Numeric
TOTVAL14	Total Value for Port14	Numeric
TOTVAL15	Total Value for Port15	Numeric
TOTVAL16	Total Value for Port16	Numeric
TOTVAL17	Total Value for Port17	Numeric

TRET Data Set—Total Returns Time Series

Table 46.33 TRET Data Set—Total Returns Time Series

Field	Label	Type
INDNO	INDNO	Numeric
CALDT	Calendar Trading Date	Numeric
TRET	Total Returns	Numeric

ARET Data Set—Appreciation Returns Time Series

Table 46.34 ARET Data Set—Appreciation Returns Time Series

Field	Label	Type
INDNO	INDNO	Numeric
CALDT	Calendar Trading Date	Numeric
ARET	Appreciation Returns Time Series	Numeric

IRET Data Set—Income Returns Time Series

Table 46.35 IRET Data Set—Income Returns Time Series

Field	Label	Type
INDNO	INDNO	Numeric
CALDT	Calendar Trading Date	Numeric
IRET	Income Returns	Numeric

TRET Group Data Set—Total Returns Time Series Groups

Table 46.36 TRET Group Data Set—Total Returns Time Series Groups

Field	Label	Type
INDNO	INDNO	Numeric
CALDT	Calendar Trading Date	Numeric
TRET1	Total Returns for Port 1	Numeric
TRET2	Total Returns for Port 2	Numeric
TRET3	Total Returns for Port 3	Numeric
TRET4	Total Returns for Port 4	Numeric
TRET5	Total Returns for Port 5	Numeric
TRET6	Total Returns for Port 6	Numeric
TRET7	Total Returns for Port 7	Numeric
TRET8	Total Returns for Port 8	Numeric
TRET9	Total Returns for Port 9	Numeric
TRET10	Total Returns for Port 10	Numeric
TRET11	Total Returns for Port 11	Numeric
TRET12	Total Returns for Port 12	Numeric
TRET13	Total Returns for Port 13	Numeric
TRET14	Total Returns for Port 14	Numeric
TRET15	Total Returns for Port 15	Numeric
TRET16	Total Returns for Port 16	Numeric
TRET17	Total Returns for Port 17	Numeric

ARET Group Data Set—Appreciation Returns Time Series Groups

Table 46.37 ARET Group Data Set—Appreciation Returns Time Series Groups

Field	Label	Type
INDNO	INDNO	Numeric
CALDT	Calendar Trading Date	Numeric
ARET1	Appreciation Returns for Port 1	Numeric
ARET2	Appreciation Returns for Port 2	Numeric
ARET3	Appreciation Returns for Port 3	Numeric
ARET4	Appreciation Returns for Port 4	Numeric
ARET5	Appreciation Returns for Port 5	Numeric
ARET6	Appreciation Returns for Port 6	Numeric
ARET7	Appreciation Returns for Port 7	Numeric
ARET8	Appreciation Returns for Port 8	Numeric
ARET9	Appreciation Returns for Port 9	Numeric
ARET10	Appreciation Returns for Port 10	Numeric
ARET11	Appreciation Returns for Port 11	Numeric
ARET12	Appreciation Returns for Port 12	Numeric
ARET13	Appreciation Returns for Port 13	Numeric
ARET14	Appreciation Returns for Port 14	Numeric

Table 46.37 *continued*

Field	Label	Type
ARET15	Appreciation Returns for Port 15	Numeric
ARET16	Appreciation Returns for Port 16	Numeric
ARET17	Appreciation Returns for Port 17	Numeric

IRET Group Data Set—Income Returns Time Series Groups

Table 46.38 IRET Group Data Set—Income Returns Time Series Groups

Field	Label	Type
INDNO	INDNO	Numeric
CALDT	Calendar Trading Date	Numeric
IRET1	Income Returns for Port 1	Numeric
IRET2	Income Returns for Port 2	Numeric
IRET3	Income Returns for Port 3	Numeric
IRET4	Income Returns for Port 4	Numeric
IRET5	Income Returns for Port 5	Numeric
IRET6	Income Returns for Port 6	Numeric
IRET7	Income Returns for Port 7	Numeric
IRET8	Income Returns for Port 8	Numeric
IRET9	Income Returns for Port 9	Numeric
IRET10	Income Returns for Port 10	Numeric
IRET11	Income Returns for Port 11	Numeric
IRET12	Income Returns for Port 12	Numeric
IRET13	Income Returns for Port 13	Numeric
IRET14	Income Returns for Port 14	Numeric
IRET15	Income Returns for Port 15	Numeric
IRET16	Income Returns for Port 16	Numeric
IRET17	Income Returns for Port 17	Numeric

TIND Data Set—Total Return Index Levels Time Series

Table 46.39 TIND Data Set—Total Return Index Levels Time Series

Field	Label	Type
INDNO	INDNO	Numeric
CALDT	Calendar Trading Date	Numeric
TIND	Total Return Index Levels	Numeric

AIND Data Set—Appreciation Index Levels Time Series

Table 46.40 AIND Data Set—Appreciation Index Levels Time Series

Field	Label	Type
INDNO	INDNO	Numeric
CALDT	Calendar Trading Date	Numeric
AIND	Appreciation Index Levels	Numeric

IIND Data Set—Income Index Levels Time Series

Table 46.41 IIND Data Set—Income Index Levels Time Series

Field	Label	Type
INDNO	INDNO	Numeric
CALDT	Calendar Trading Date	Numeric
IIND	Income Index Levels	Numeric

TIND Group Data Set—Total Return Index Levels Time Series Groups

Table 46.42 TIND Group Data Set—Total Return Index Levels Time Series Groups

Field	Label	Type
INDNO	INDNO	Numeric
CALDT	Calendar Trading Date	Numeric
TIND1	Total Return Index Levels for Port 1	Numeric
TIND2	Total Return Index Levels for Port 2	Numeric
TIND3	Total Return Index Levels for Port 3	Numeric
TIND4	Total Return Index Levels for Port 4	Numeric
TIND5	Total Return Index Levels for Port 5	Numeric
TIND6	Total Return Index Levels for Port 6	Numeric
TIND7	Total Return Index Levels for Port 7	Numeric
TIND8	Total Return Index Levels for Port 8	Numeric
TIND9	Total Return Index Levels for Port 9	Numeric
TIND10	Total Return Index Levels for Port 10	Numeric
TIND11	Total Return Index Levels for Port 11	Numeric
TIND12	Total Return Index Levels for Port 12	Numeric
TIND13	Total Return Index Levels for Port 13	Numeric
TIND14	Total Return Index Levels for Port 14	Numeric
TIND15	Total Return Index Levels for Port 15	Numeric
TIND16	Total Return Index Levels for Port 16	Numeric
TIND17	Total Return Index Levels for Port 17	Numeric

AIND Group Data Set—Appreciation Index Levels Groups

Table 46.43 AIND Group Data Set—Appreciation Index Levels Groups

Field	Label	Type
INDNO	INDNO	Numeric
CALDT	Calendar Trading Date	Numeric
AIND1	Appreciation Index Levels for Port 1	Numeric
AIND2	Appreciation Index Levels for Port 2	Numeric
AIND3	Appreciation Index Levels for Port 3	Numeric
AIND4	Appreciation Index Levels for Port 4	Numeric
AIND5	Appreciation Index Levels for Port 5	Numeric
AIND6	Appreciation Index Levels for Port 6	Numeric
AIND7	Appreciation Index Levels for Port 7	Numeric
AIND8	Appreciation Index Levels for Port 8	Numeric
AIND9	Appreciation Index Levels for Port 9	Numeric
AIND10	Appreciation Index Levels for Port 10	Numeric
AIND11	Appreciation Index Levels for Port 11	Numeric
AIND12	Appreciation Index Levels for Port 12	Numeric
AIND13	Appreciation Index Levels for Port 13	Numeric
AIND14	Appreciation Index Levels for Port 14	Numeric
AIND15	Appreciation Index Levels for Port 15	Numeric
AIND16	Appreciation Index Levels for Port 16	Numeric
AIND17	Appreciation Index Levels for Port 17	Numeric

IIND Group Data Set—Income Index Levels Time Series Groups

Table 46.44 IIND Group Data Set—Income Index Levels Time Series Groups

Field	Label	Type
INDNO	INDNO	Numeric
CALDT	Calendar Trading Date	Numeric
IIND1	Income Index Levels for Port 1	Numeric
IIND2	Income Index Levels for Port 2	Numeric
IIND3	Income Index Levels for Port 3	Numeric
IIND4	Income Index Levels for Port 4	Numeric
IIND5	Income Index Levels for Port 5	Numeric
IIND6	Income Index Levels for Port 6	Numeric
IIND7	Income Index Levels for Port 7	Numeric
IIND8	Income Index Levels for Port 8	Numeric
IIND9	Income Index Levels for Port 9	Numeric
IIND10	Income Index Levels for Port 10	Numeric
IIND11	Income Index Levels for Port 11	Numeric
IIND12	Income Index Levels for Port 12	Numeric
IIND13	Income Index Levels for Port 13	Numeric
IIND14	Income Index Levels for Port 14	Numeric

Table 46.44 *continued*

Field	Label	Type
IIND15	Income Index Levels for Port 15	Numeric
IIND16	Income Index Levels for Port 16	Numeric
IIND17	Income Index Levels for Port 17	Numeric

Examples: SASECRSP Interface Engine

Example 46.1: Specifying PERMNOs and Range in the LIBNAME Statement

The following statements show how to set up a LIBNAME statement to extract data for certain selected PERMNOs during a specific time period. The result is shown in Output 46.1.1.

```
title2 'Define a range inside the data range';
title3 'My range is ( 19950101-19960630 )';

libname _all_ clear;
libname testit1 sasecrsp "%sysget(CRSP_MSTK)"
        setid=20
        permno=81871    /* Desired PERMNOs are selected   */
        permno=82200    /* via the libname PERMNO= option */
        permno=82224
        permno=83435
        permno=83696
        permno=83776
        permno=84788
        range='19950101-19960630';

proc print data=testit1.ask;
run;
```

Output 46.1.1 ASK Monthly Time Series Data with RANGE= Option

Define a range inside the data range
My range is (19950101-19960630)

Obs	PERMNO	CALDT	ASK
1	81871	19950731	18.25000
2	81871	19950831	19.25000
3	81871	19950929	26.00000
4	81871	19951031	26.00000
5	81871	19951130	25.50000
6	81871	19951229	24.25000
7	81871	19960131	22.00000
8	81871	19960229	32.50000
9	81871	19960329	30.25000
10	81871	19960430	33.75000
11	81871	19960531	27.50000
12	81871	19960628	30.50000
13	82200	19950831	49.50000
14	82200	19950929	62.75000
15	82200	19951031	88.00000
16	82200	19951130	138.50000
17	82200	19951229	139.25000
18	82200	19960131	164.25000
19	82200	19960229	51.00000
20	82200	19960329	41.62500
21	82200	19960430	61.25000
22	82200	19960531	68.25000
23	82200	19960628	62.50000
24	82224	19950929	46.50000
25	82224	19951031	48.50000
26	82224	19951130	47.75000
27	82224	19951229	49.75000
28	82224	19960131	49.00000
29	82224	19960229	47.00000
30	82224	19960329	53.00000
31	82224	19960430	55.50000
32	82224	19960531	54.25000
33	82224	19960628	51.00000
34	83435	19960430	30.25000
35	83435	19960531	28.00000
36	83435	19960628	21.00000
37	83696	19960628	19.12500

Example 46.2: Using the LIBNAME Statement to Access All Keys

To set up the libref to access all keys, no key options such as PERMNO=, TICKER=, or GVKEY= are specified in the LIBNAME statement and no INSET= option is used. Any of these options cause the SASECRSP engine to limit access to specified keys or specified insets. When no such options are specified, the SASECRSP engine correctly defaults to selecting all keys in the database. Other LIBNAME statement options, such as the RANGE= option, can still be used normally to limit the time span of the data—in other words, to define the date range of observations.

This example does not use key-specifying options. This forces the engine to default to all PERMNOs in the monthly STK database. The range that is specified in the LIBNAME statement behaves normally, and data are limited to the first two months of 1995.

```
title2 'Define a range inside the data range ';
title3 'My range is ( 19950101-19950228 )';

libname _all_ clear;
libname testit2 sasecrsp "%sysget(CRSP_MSTK)"
        setid=20
        range='19950101-19950228';
data a;
   set testit2.ask(obs=30);
run;

proc print data=a;
run;
```

The result is shown in Output 46.2.1.

Output 46.2.1 All PERMNOs of ASK Monthly Time Series Data with RANGE= Option

Define a range inside the data range
My range is (19950101-19950228)

Obs	PERMNO	CALDT	ASK
1	10001	19950131	8.00000
2	10001	19950228	8.00000
3	10002	19950131	13.50000
4	10002	19950228	13.50000
5	10003	19950131	2.12500
6	10003	19950228	2.25000
7	10009	19950131	18.00000
8	10009	19950228	18.75000
9	10010	19950131	5.37500
10	10010	19950228	4.87500
11	10011	19950131	14.62500
12	10011	19950228	13.50000
13	10012	19950131	2.25000
14	10012	19950228	2.12500
15	10016	19950131	7.00000
16	10016	19950228	8.50000
17	10018	19950131	1.12500
18	10018	19950228	1.12500
19	10019	19950131	10.62500
20	10019	19950228	11.62500
21	10021	19950131	11.75000
22	10021	19950228	12.00000
23	10025	19950131	18.50000
24	10025	19950228	19.00000
25	10026	19950131	11.00000
26	10026	19950228	11.75000
27	10028	19950131	1.87500
28	10028	19950228	2.00000
29	10032	19950131	12.50000
30	10032	19950228	12.75000

Example 46.3: Accessing One PERMNO without the RANGE= Option

The SASECRSP engine defaults to providing access to the entire range of available data when you do not restrict the range (that is, when you do not use the RANGE= option).

This example shows access of the entire range of available data for one particular PERMNO extracted from the monthly data set.

```
title2 'Select only PERMNO = 81871';
title3 'Valid trading dates (19890131--19981231)';
title4 'No range option, leave wide open';

libname _all_ clear;
libname testit3 sasecrsp "%sysget(CRSP_MSTK)"
        setid=20
        permno=81871;

data c;
   set testit3.ask;
run;

proc print data=c;
run;
```

The result is shown in Output 46.3.1.

Output 46.3.1 PERMNO=81871 of ASK Monthly Time Series Data without RANGE= Option

Select only PERMNO = 81871
Valid trading dates (19890131--19981231)
No range option, leave wide open

Obs	PERMNO	CALDT	ASK
1	81871	19950731	18.25000
2	81871	19950831	19.25000
3	81871	19950929	26.00000
4	81871	19951031	26.00000
5	81871	19951130	25.50000
6	81871	19951229	24.25000
7	81871	19960131	22.00000
8	81871	19960229	32.50000
9	81871	19960329	30.25000
10	81871	19960430	33.75000
11	81871	19960531	27.50000
12	81871	19960628	30.50000
13	81871	19960731	26.12500
14	81871	19960830	19.12500
15	81871	19960930	19.50000
16	81871	19961031	14.00000
17	81871	19961129	18.75000
18	81871	19961231	24.25000
19	81871	19970131	29.75000
20	81871	19970228	24.37500
21	81871	19970331	15.00000
22	81871	19970430	18.25000
23	81871	19970530	25.12500
24	81871	19970630	31.12500
25	81871	19970731	35.00000
26	81871	19970829	33.00000
27	81871	19970930	26.81250
28	81871	19971031	18.37500
29	81871	19971128	16.50000
30	81871	19971231	16.25000
31	81871	19980130	22.75000
32	81871	19980227	21.00000
33	81871	19980331	22.50000
34	81871	19980430	16.12500
35	81871	19980529	11.12500
36	81871	19980630	13.43750
37	81871	19980731	22.87500
38	81871	19980831	17.75000
39	81871	19980930	24.25000
40	81871	19981030	26.00000

Example 46.4: Specifying Keys by Using the INSET= Option

The INSET= option enables you to select any companies or issues for which you want data. This example selects two CRSP Index Series from the CRSP Indices data, and four securities from the CRSP US Stock data for data extraction. Note that because each CRSP database might be in a different location and must be opened separately, a total of two different librefs are used, one for each database.

```
data indices;
   indno=1000000; output;  /* NYSE Value-Weighted Market Index */
   indno=1000001; output;  /* NYSE Equal-Weighted Market Index */
run;

libname _all_ clear;
libname ind2 sasecrsp "%sysget(CRSP_MSTK)"
        setid=420
        inset='indices,INDNO,INDNO'
        range='19990101-19990401';

title2 'Total Returns for NYSE Value- and Equal-Weighted Market Indices';
proc print data=ind2.tret label;
run;
```

Output 46.4.1 shows the result of selecting two CRSP Index Series from the CRSP Indices data.

Output 46.4.1 IND Data Extracted Using INSET= Option

Total Returns for NYSE Value- and Equal-Weighted Market Indices

Obs	INDNO	Calendar Trading Date	Total Returns
1	1000000	19990129	0.012419
2	1000000	19990226	-0.024179
3	1000000	19990331	0.028591
4	1000001	19990129	-0.007700
5	1000001	19990226	-0.041183
6	1000001	19990331	0.015101

The following statements select three securities from the CRSP US Stock data by using TICKER keys in the INSET= option for data extraction:

```
data securities;
   ticker='BAC'; output;  /* Bank of America */
   ticker='DUK'; output;  /* Duke Energy */
   ticker='GSK'; output;  /* GlaxoSmithKline */
run;

libname sec3 sasecrsp "%sysget(CRSP_MSTK)"
        setid=20
        inset='securities,TICKER,TICKER'
        range='19970820-19970920';
```

```
title2 'PERMNOs and General Header Info of Selected TICKERs';
proc print data=sec3.stkhead(keep=permno htick htsymbol) label;
run;
title3 'Average Price for Bank of America, Duke and GlaxoSmithKline';
proc print data=sec3.prc label;
run;
```

Output 46.4.2 shows the STK header data for the TICKER keys that are specified by using the INSET= option.

Output 46.4.2 STK Header Data Extracted Using INSET= Option

PERMNOs and General Header Info of Selected TICKERs

Obs	PERMNO	Ticker Symbol Header	Trading Symbol Header
1	59408	BAC	BAC
2	27959	DUK	DUK
3	75064	GSK	GSK

Output 46.4.3 shows the STK price data for the TICKER keys that are specified by using the INSET= option.

Output 46.4.3 STK Price Data Extracted Using INSET= Option

PERMNOs and General Header Info of Selected TICKERs
Average Price for Bank of America, Duke and GlaxoSmithKline

Obs	PERMNO	Calendar Trading Date	Price or Bid/Ask Average
1	59408	19970829	59.75000
2	27959	19970829	48.43750
3	75064	19970829	39.93750

Example 46.5: Specifying Ranges for Individual Keys with the INSET= Option

Insets enable you to define options that are specific to each individual key. This example uses an inset to select four PERMNOs and specifies a different date restriction for each PERMNO.

```
title2 'INSET=testin2 uses date ranges along with PERMNOs:';
title3 '10107, 12490, 14322, 25788';
title4 'Begin dates and end dates for each permno are used in the INSET';

data testin2;
   permno = 10107; date1 = 19980731; date2 = 19981231; output;
   permno = 12490; date1 = 19970101; date2 = 19971231; output;
   permno = 14322; date1 = 19950731; date2 = 19960131; output;
   permno = 25778; date1 = 19950101; date2 = 19950331; output;
run;

libname _all_ clear;
libname mstk2 sasecrsp "%sysget(CRSP_MSTK)"
        setid=20
        inset='testin2,PERMNO,PERMNO,DATE1,DATE2';

data b;
   set mstk2.prc;
run;

proc print data=b;
run;
```

Output 46.5.1 shows CRSP US Stock price time series data selected by PERMNO in the INSET= option, where each PERMNO has its own time span specified in the INSET= option.

Output 46.5.1 PRC Monthly Time Series Using the INSET= Option

INSET=testin2 uses date ranges along with PERMNOs:
10107, 12490, 14322, 25788
Begin dates and end dates for each permno are used in the INSET

Obs	PERMNO	CALDT	PRC
1	10107	19980731	109.93750
2	10107	19980831	95.93750
3	10107	19980930	110.06250
4	10107	19981030	105.87500
5	10107	19981130	122.00000
6	10107	19981231	138.68750
7	12490	19970131	156.87500
8	12490	19970228	143.75000
9	12490	19970331	137.25000
10	12490	19970430	160.50000
11	12490	19970530	86.50000
12	12490	19970630	90.25000
13	12490	19970731	105.75000
14	12490	19970829	101.37500
15	12490	19970930	106.00000
16	12490	19971031	98.50000
17	12490	19971128	109.50000
18	12490	19971231	104.62500
19	14322	19950731	32.62500
20	14322	19950831	32.37500
21	14322	19950929	36.87500
22	14322	19951031	34.00000
23	14322	19951130	39.37500
24	14322	19951229	39.00000
25	14322	19960131	41.50000
26	25778	19950131	49.87500
27	25778	19950228	57.25000
28	25778	19950331	59.37500

Example 46.6: Converting Dates by Using the CRSP Date Functions

This example shows how to use the CRSP date functions and formats. The CRSPDTD formats are used for all the crspdt variables, and the YYMMDD format is used for the sasdt variables.

```
title2 'OUT= Data Set';
title3 'CRSP Functions for sasecrsp';

libname _all_ clear;

/* Always assign the LIBNAME sasecrsp first */
libname mstk sasecrsp "%sysget(CRSP_MSTK)"
        setid=20;

data a (keep = crspdt crspdt2 crspdt3
               sasdt sasdt2 sasdt3
               intdt intdt2 intdt3);
   format crspdt crspdt2 crspdt3 crspdtd8.;
   format sasdt sasdt2 sasdt3 yymmdd6.;
   format intdt intdt2 intdt3 8.;
   format exact 2.;
   crspdt = 1;
   sasdt = '2jul1962'd;
   intdt = 19620702;
   exact = 0;

/* Call the CRSP date to Integer function*/
   intdt2 = crspdcid(crspdt);

/* Call the SAS date to Integer function*/
   intdt3 = crspds2i(sasdt);

/* Call the Integer to CRSP date function*/
   crspdt2 = crspdicd(intdt,exact);

/* Call the SAS date to CRSP date conversion function*/
   crspdt3 = crspdscd(sasdt,exact);

/* Call the CRSP date to SAS date conversion function*/
   sasdt2 = crspdcsd(crspdt);

/* Call the Integer to SAS date conversion function*/
   sasdt3 = crspdi2s(intdt);
run;

title3 'PROC PRINT showing data for sasecrsp';
proc print data=a;
run;

title3 'PROC CONTENTS showing formats for sasecrsp';
proc contents data=a;
run;
```

Output 46.6.1 shows the OUT= data set that is created by the DATA step.

Output 46.6.1 Date Conversions by Using the CRSP Date Functions

OUT= Data Set
PROC PRINT showing data for sasecrsp

Obs	crspdt	crspdt2	crspdt3	sasdt	sasdt2	sasdt3	intdt	intdt2	intdt3
1	19251231	19620702	19620702	620702	251231	620702	19620702	19251231	19620702

Output 46.6.2 shows the contents of the OUT= data set by alphabetically listing the variables and their attributes.

Output 46.6.2 Contents of Date Conversions by Using the CRSP Date Functions

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format
1	crspdt	Num	8	CRSPDTD8.
2	crspdt2	Num	8	CRSPDTD8.
3	crspdt3	Num	8	CRSPDTD8.
7	intdt	Num	8	8.
8	intdt2	Num	8	8.
9	intdt3	Num	8	8.
4	sasdt	Num	8	YYMMDD6.
5	sasdt2	Num	8	YYMMDD6.
6	sasdt3	Num	8	YYMMDD6.

References

Center for Research in Security Prices (2002a). *CRSP Programmer's Guide*. Chicago: CRSP, Chicago Booth. http://www.crsp.chicagobooth.edu/documentation/.

Center for Research in Security Prices (2002b). *CRSP SFA Guide*. Chicago: CRSP, Chicago Booth. http://www.crsp.chicagobooth.edu/documentation/.

Center for Research in Security Prices (2003a). *CRSP Data Description Guide*. Chicago: CRSP, Chicago Booth. http://www.crsp.chicagobooth.edu/documentation/.

Center for Research in Security Prices (2003b). *CRSP Utilities Guide*. Chicago: CRSP, Chicago Booth. http://www.crsp.chicagobooth.edu/documentation/.

Center for Research in Security Prices (2003c). *CRSPAccess Database Format Release Notes*. Chicago: CRSP, Chicago Booth. http://www.crsp.chicagobooth.edu/documentation/.

Center for Research in Security Prices (2003d). *CRSP/Compustat Merged Database Guide*. Chicago: CRSP, Chicago Booth.

Chapter 47
The SASEFAME Interface Engine

Contents

Overview: SASEFAME Interface Engine **3326**
Getting Started: SASEFAME Interface Engine **3326**
Structure of a SAS Data Set That Contains Time Series Data 3326
Reading and Converting Fame Database Time Series 3327
Using the SAS DATA Step 3327
Using SAS Procedures 3327
Using the SAS Windowing Environment 3327
Remote Fame Data Access 3328
Creating Views of Time Series by Using SASEFAME LIBNAME Options 3328
Syntax: SASEFAME Interface Engine **3329**
LIBNAME *libref* SASEFAME Statement 3329
Details: SASEFAME Interface Engine **3333**
SAS Output Data Set 3333
Mapping Fame Frequencies to SAS Time Intervals 3334
Performing the Crosslist Selection Function 3336
Examples: SASEFAME Interface Engine **3338**
Example 47.1: Converting an Entire Fame Database 3338
Example 47.2: Reading Time Series from the Fame Database 3341
Example 47.3: Writing Time Series to the SAS Data Set 3342
Example 47.4: Limiting the Time Range of Data 3345
Example 47.5: Creating a View Using the SQL Procedure and the SASEFAME Engine 3349
Example 47.6: Reading Other Fame Data Objects with the FAMEOUT= Option . . . 3355
Example 47.7: Remote Fame Access by Using Fame CHLI 3358
Example 47.8: Selecting Time Series by Using the CROSSLIST= Option and KEEP Statement 3359
Example 47.9: Selecting Time Series by Using the CROSSLIST= Option and Fame Namelist 3362
Example 47.10: Selecting Time Series by Using the CROSSLIST= Option and WHERE=TICK 3364
Example 47.11: Selecting Boolean Case Series with the FAMEOUT= Option 3367
Example 47.12: Selecting Numeric Case Series with the FAMEOUT= Option 3369
Example 47.13: Selecting Date Case Series with the FAMEOUT= Option 3370
Example 47.14: Selecting String Case Series with the FAMEOUT= Option 3372
Example 47.15: Extracting Source for Formulas 3373
Example 47.16: Reading Time Series by Defining Fame Expression Groups in the INSET= Option with the KEEP= Clause 3374

Example 47.17: Optimizing Cache Sizes with the TUNEFAME= and TUNECHLI= Options 3376
References **3378**

Overview: SASEFAME Interface Engine

The SASEFAME interface engine provides a seamless interface between Fame and SAS data to enable SAS users to access and process time series, case series, and formulas that reside in a Fame database.

Fame is an integrated, front-to-back market data and historical database solution for storing and managing real-time and high-volume time series data that are used by leading institutions in the financial, energy, and public sectors, as well as by third-party content aggregators, software vendors, and individual investors. Fame provides real-time market data feeds and end-of-day data, a web-based desktop solution, application hosting, data delivery components, and tools for performing analytic modeling.

The SASEFAME engine uses the LIBNAME statement to enable you to specify the time series that you want to read from the Fame database and how you want to convert the selected time series to the same time scale. You can then use the SAS DATA step to perform further subsetting and to store the resulting time series in a SAS data set. You can perform more analysis (if desired) either in the same SAS session or in a later session.

The SASEFAME interface engine supports Windows, Solaris, AIX, and Linux Opteron hosts.

Getting Started: SASEFAME Interface Engine

Structure of a SAS Data Set That Contains Time Series Data

The SAS System represents time series data in a two-dimensional array, called a SAS data set, whose columns correspond to series variables and whose rows correspond to measurements of these variables at certain time periods. The time periods at which observations are recorded can be included in the data set as a time ID variable. The SASEFAME engine provides a time ID variable named DATE. The DATE variable can be represented by any of the time intervals shown in the section “Mapping Fame Frequencies to SAS Time Intervals” on page 3334.

Reading and Converting Fame Database Time Series

The SASEFAME engine supports reading and converting time series that reside in Fame databases. The SASEFAME engine uses Fame's Work database to temporarily store the converted time series. All series that are specified by the Fame wildcard are written to the Fame Work database. For conversion of very large databases, you might want to define the FAME_TEMP environment variable to point to a location where there is ample space for the Fame Work database.

The SASEFAME engine provides seamless access to Fame databases via Fame's C host language interface (CHLI). Fame expressions that contain formulas and Fame functions can be input to the engine via the INSET= option.

The SASEFAME engine finishes the CHLI whenever a fatal error occurs. To restart the engine after a fatal error, terminate the current SAS session and open a new SAS session.

Using the SAS DATA Step

You can store the converted series in a SAS data set by using the SAS DATA step. You can also perform other operations on your data inside the DATA step. After your data are stored in a SAS data set, you can use this data set as you would any other SAS data set.

Using SAS Procedures

You can print the output SAS data set by using the PRINT procedure and report information about the contents of your data set by using the CONTENTS procedure, as in Example 47.1. You can create a view of the FAME database by using the SQL procedure's USING clause to reference the SASEFAME engine in your libref. See Example 47.5.

Using the SAS Windowing Environment

You can see the available data sets in the SAS LIBNAME window of the SAS windowing environment. To do so, select the SASEFAME engine libref in the LIBNAME window that you have previously defined in your LIBNAME statement. You can view your SAS output observations by double-clicking the desired output data set libref in the LIBNAME window of the SAS windowing environment. Type **Viewtable** on the SAS command line to view any of your SASEFAME engine tables, views, or librefs both for input and output data sets.

Before you use the **Viewtable** command, it is recommended that you store your output data sets in a physical folder or library that is separate from the folder or library used for your input databases. (The default location for output data sets is the SAS Work library.)

Remote Fame Data Access

The remote access feature of the SASEFAME engine uses the Fame CHLI to communicate with your remote Fame server. It is available to licensed CHLI customers who have Fame CHLI on both their remote and client machines.

As shown in Example 47.7, you simply provide the frdb_m port number and node name of your Fame master server in your *libref*. For more information, see the section "Starting the Master Server" in the *Guide to Fame Database Servers*.

Creating Views of Time Series by Using SASEFAME LIBNAME Options

You can perform selection based on names of your time series simply by using Fame wildcard specifications in the SASEFAME engine WILDCARD= option.

You can limit the time span of time series data by specifying a beginning and ending date range in the SASEFAME engine RANGE= option.

It is also easy to use the SAS input data set INSET= option to create a specific view of your Fame data. You can create multiple views by using multiple LIBNAME statements with customized options that are tailored to the unique views that you want to create.

You can list the INSET variables that you want to keep in your SAS data set by using the KEEP= clause. When you use INSET variables in conjunction with the input data set that you specify in the INSET= option, the SASEFAME engine can show any or all of your expression groups in the same view or in multiple views. The INSET= option defines the valid set of expression groups that you can reference in the KEEP= clause, as shown in Example 47.16.

The INSET variables define the BY variables that enable you to view cross sections (slices) of your data. When you use INSET variables in conjunction with the WHERE clause and the CROSSLIST= option, the SASEFAME engine can show any or all of your BY groups in the same view or in multiple views. When you use the INSET= option along with a WHERE clause that specifies the BY variables that you want to use in your view, you must also use the CROSSLIST= option, as shown in Example 47.10. You can use the CROSSLIST= option without using the INSET= option, as shown in Example 47.8 and Example 47.9.

Syntax: SASEFAME Interface Engine

The SASEFAME engine uses standard engine syntax. Table 47.1 summarizes the options used by the SASEFAME engine.

Table 47.1 Summary of LIBNAME *libref* SASEFAME Statement Options

Option	Description
CONVERT=	Specifies the Fame frequency and the Fame technique
WILDCARD=	Specifies a Fame wildcard to match data object series names within the Fame database
RANGE=	Specifies the range of data to keep in format 'ddmmmyyy'd – 'ddmmmyyyy'd
INSET=	Uses a SAS data set named *setname* and KEEP= *fame_expression_group* as selection input variables or WHERE= *fame_bygroup* as selection input for BY variables
CROSSLIST=	Specifies a Fame crosslist *fame_namelist* to perform selection based on the crossproduct of two Fame namelists
FAMEOUT=	Specifies the Fame data object class/type that you want output to the SAS data set
DBVERSION=	Echoes the present version number of the Fame Work database in the SAS log
TUNEFAME=	Tunes the Fame database engine's use of memory to reduce I/O in favor of a bigger virtual memory for caching database objects
TUNECHLI=	Tunes the CHLI database engine's use of memory to reduce I/O in favor of a bigger virtual memory for caching database objects

LIBNAME *libref* SASEFAME Statement

LIBNAME *libref* **SASEFAME** '*physical name*' *options* **;**

Because '*physical name*' specifies the location of the folder where your Fame database resides, it should end in a backslash if you are in a Windows environment or a forward slash if you are in a UNIX environment.

If you are accessing a remote Fame database by using the Fame CHLI, you can use the following syntax for '*physical name*':

'`#port_number @hostname physical_path_name`'

You can specify the following *options*.

CONVERT=(FREQ=*fame_frequency* **TECH=***fame_technique***)**

CONV=(FREQ=*fame_frequency* **TECH=***fame_technique***)**

specifies the Fame frequency and the Fame technique, just as you would in the Fame CONVERT

function. There are four possible values for *fame_technique*: *Constant* (default), *Cubic*, *Discrete*, and *Linear*. Table 47.4 shows the Fame frequencies that are supported by the SASEFAME engine.

For a more complete discussion of Fame frequencies and SAS time intervals, see the section "Mapping Fame Frequencies to SAS Time Intervals" on page 3334. For all possible *fame_frequency* values, see the section "Understanding Frequencies" in the *User's Guide to Fame*. For example:

```
LIBNAME libref sasefame 'physical-name'
        CONVERT=(TECH=CONSTANT FREQ=TWICEMONTHLY);
```

WILDCARD="*fame_wildcard***"**
WILD="*fame_wildcard***"**

limits the time series read from the Fame database. By default, the SASEFAME engine reads all time series in the Fame database that you name in your SASEFAME libref. The *fame_wildcard* is a quoted string that contains the Fame wildcard you want to use. The wildcard is used for matching against the data object names of series that you want to select from the Fame database that resides in the library you are assigning.

For more information about using wildcards, see the section "Specifying Wildcards" in the *User's Guide to Fame*.

For example, to read all time series in the TEST library that is being accessed by the SASEFAME engine, you would specify the following statement:

```
LIBNAME test sasefame 'physical name of test database'
        WILDCARD="?";
```

To read series that have names such as A_DATA, B_DATA, and C_DATA, you could specify the following statement:

```
LIBNAME test sasefame 'physical name of test database'
        WILDCARD="^_DATA";
```

When you use the WILDCARD= option, you limit the number of series that are read and converted to the desired frequency. This option can help you save resources when processing large databases or when processing a large number of observations, such as daily or hourly frequencies. Because the SASEFAME engine uses the Fame Work database to store the converted time series, using wildcards is recommended to prevent your workspace from getting too large. When the FAMEOUT= option is also specified, the wildcard is applied to the type of data object series that you specify in the FAMEOUT= option.

RANGE='*fame_begdt***'d-'***fame_enddt***'d**
DATERANGE='*fame_begdt***'d-'***fame_enddt***'d**
DATE='*fame_begdt***'d-'***fame_enddt***'d**
DATECASE='*fame_begdt***'d-'***fame_enddt***'d**

limits the time range of data that are read from your Fame database. The string *fame_begdt* is the beginning date in ddmmmyyyy format, and the string *fame_enddt* is the ending date of the range in ddmmmyyyy format; both strings must be enclosed in single quotation marks and followed by the letter d.

For example, to read a series with a date range that spans the first quarter of 1999, you could use the following statement:

```
LIBNAME test sasefame 'physical name of test database'
        RANGE='01jan1999'd - '31mar1999'd;
```

INSET=(*setname* **KEEP=***fame_expression_group* **)**
INSET=(*setname* **KEEPLIST=***fame_expression_group* **)**

specifies the name of a SAS data set (*setname*) and selects series that are generated by the expressions defined in *fame_expression_group*. You can define *fame_expression_group* by using Fame functions and Fame expressions. It is important to specify the length of the longest expression, or expressions might be truncated because the default length is the first defined variable in the DATA step. The INSET (input data set) must output each expression statement as a character string ending with a semicolon, enclosed in single quotation marks, and followed by another semicolon and an output statement. **NOTE:** The EXPRESS variable is assigned the character string expression, and is shown in Table 47.2. The following statements create an input data set, INSETA, and print it:

```
data inseta; /* Use this for training database */
  length express $52;
  express='{ibm.high,ibm.low,ibm.close};'; output;
  express='crosslist({gm,f,c},{volume});'; output;
  express='cvx.close;'; output;
  express='mave(ibm.close,30);'; output;
  express='cvx.close+ibm.close;'; output;
  express='ibm.close;'; output;
  express='close * shares/sum(close * shares);'; output;
  express='sum(pep.volume);'; output;
  express='mave(pep.close,20);'; output;
run;

proc print
   data=inseta;
run;
```

Next you can name the input data set that you want to use in the INSET= option, followed by the KEEP= variable that specifies the expression group you want to keep. Only series variables that are defined in the selected expression group are output to the output data set. You can define up to eight different expression groups in an INSET= option.

```
  libname lib5 sasefame "C:\PROGRA~1\FAME10\util"
     wildcard="?"
     convert=(frequency=business technique=constant)
     range='23jul1997'd - '25jul1997'd
     inset=( inseta KEEP=express)
     ;

data trout;
   set lib5.trainten;
run;

title1 'TRAINING DB, Pricing Timeseries for Expressions in INSET=';
title2 'OUT=TROUT from the PRINT Procedure';
proc print data=trout;
run;
```

Table 47.2 shows the eight expressions that are defined in INSETA.

Table 47.2 SAS Input Data Set, INSETA, Defined for Use in the INSET= Option

Observation	EXPRESS
1	cvx.close;
2	ibm.high,ibm.low,ibm.close;
3	mave(ibm.close,30);
4	crosslist(gm,f,c,volume);
5	cvx.close+ibm.close;
6	ibm.close;
7	sum(pep.volume);
8	mave(pep.close,20);

Table 47.3 shows the output data set, TROUT. The output data set names each derived variable 'SASTEMPn' by appending the number, n, to match the observation number of the input data set's expression for that variable. For example, SASTEMP1 names the series derived by 'cvx.close' in observation 1, and SASTEMP3 names the series derived by the expression 'mave(ibm.close,30);' in observation 3. Because SASTEMP2 is a simple name list of three series, the original series names are used.

Table 47.3 TRAINING DB, Pricing Timeseries for Expressions in INSETA for OUT=TROUT from the PRINT Procedure

DATE	C.VOLUME	VOLUME	GM.VOLUME	IBM.CLOSE	IBM.HIGH
23JUL1997	33791.88	45864.05	37392	52.5625	53.5000
24JUL1997	41828.85	29651.34	27771	53.9063	54.2188
25JUL1997	46979.83	36716.77	24969	53.5000	54.2188

IBM.LOW	SASTEMP1	SASTEMP3	SASTEMP5	SASTEMP6	SASTEMP8
51.5938	38.4063	.	90.9688	52.5625	.
52.2500	38.4375	.	92.3438	53.9063	.
52.8125	39.0000	.	92.5000	53.5000	.

Note that SASTEMP3 and SASTEMP8 have no observations in the date range July 23, 1997, to July 25, 1997, so the missing value symbol '.' appears for those observations.

INSET=(*setname* **WHERE=***fame_bygroup* **)**

specifies a SAS data set (*setname*) as input for a BY group such as a ticker, and uses the *fame_bygroup* to select time series that are named using the following convention. Selected variable names are glued together by the BY-group name (such as a ticker symbol) concatenated with the glue character (such as DOT) to the series name that is specified in the CROSSLIST= option or in the *fame_bygroup*.

For more information, see the section "Performing the Crosslist Selection Function" on page 3336.

CROSSLIST=(< *fame_namelist1*, **>** *fame_namelist2* **)**

CROSS=(< *fame_namelist1*, **>** *fame_namelist2* **)**

performs a crossproduct of the members of the first namelist with the members of the second namelist, using a glue symbol "." to join the two. If one of the time series that are listed in *fame_namelist2* does not exist, the SASEFAME engine stops processing the remainder of the namelist. For more information, see the section "Performing the Crosslist Selection Function" on page 3336.

FAMEOUT=*fame_data_object_class_type*

specifies the class and type of the Fame data series objects to include in your SAS output data set. The possible values for *fame_data_object_class_type* are FORMULA, TIME, BOOLEAN, CASE, DATE, and STRING. Case series can be numeric, boolean, string, and date, or they can be generated using formulas that resolve to series. The SASEFAME engine resolves all formulas that belong to the type of series data object that you specify in the FAMEOUT= option. If the FAMEOUT= option is not specified, numeric time series are output to the SAS data set. FAMEOUT=CASE defaults to case series of numeric type. If you want another type of case series in your output, then you must specify it. Scalar data objects are not supported.

DBVERSION=ON | OFF

specifies whether to display the version number of the Fame Work database. DBVERSION=ON specifies that the SAS log show the version number (3 or 4) of the Fame Work database. By default, DBVERSION=OFF.

TUNEFAME=NODES *fameengine_size_virtual_memory_MB*

specifies the number of megabytes to use for the cache size for the Fame API (CHLI). The *fameengine_size_virtual_memory_MB* can range from a minimum of 0.1 MB (100 KB) to a maximum of 17,592,186,000,000 MB. For more information, see Example 47.17.

TUNECHLI=NODES *famechliengine_size_virtual_memory_MB*

specifies the number of megabytes to use for the cache size for the Fame API (CHLI). The *famechliengine_size_virtual_memory_MB* can range from a minimum of 0.1 MB (100 KB) to a maximum of 17,592,186,000,000 MB. For more information, see Example 47.17.

Details: SASEFAME Interface Engine

SAS Output Data Set

You can use the SAS DATA step to write the selected time series from your Fame database to a SAS data set. This enables you to easily analyze the data by using the SAS System. You can specify the name of the output data set in the DATA statement. This causes the engine supervisor to create a SAS data set by using the specified name in either the SAS Work library or, if specified, the Sasuser library. For more information about naming your SAS data set, see the section "SAS Data Sets: Data Set Names" in the *SAS Language Reference: Concepts*.

The contents of the SAS data set that contains time series include the date of each observation, the name of each series read from the Fame database as specified by the WILDCARD= option, and the label or Fame description of each series. Missing values are represented as '.' in the SAS data set. You can see the available

data sets in the SAS LIBNAME window of the SAS windowing environment by selecting the SASEFAME libref in the LIBNAME window that you have previously used in your LIBNAME statement. You can use PROC PRINT and PROC CONTENTS to print your output data set and its contents. You can use PROC SQL and the SASEFAME engine to create a view of your SAS data set. You can view your SAS output observations by double-clicking the desired output data set libref in the LIBNAME window of the SAS windowing environment.

The DATE variable in the SAS data set contains the date of the observation. For Fame weekly intervals that end on a Friday, Fame reports the date on the Friday that ends the week, whereas the SAS System reports the date on the Saturday that begins the week.

A more detailed discussion of how to map Fame frequencies to SAS time intervals follows. For other types of data, such as Boolean case series, numeric case series, date case series, string case series, and extracting source for formulas, see Example 47.11, Example 47.12, Example 47.13, Example 47.14, and Example 47.15, respectively.

Mapping Fame Frequencies to SAS Time Intervals

Table 47.4 summarizes the mapping of Fame frequencies to SAS time intervals. Fame frequencies often have a sample unit in parentheses after the keyword frequency. This sample unit is an end-of-interval unit. SAS dates are represented by beginning-of-interval notation.

For more information about SAS time intervals, see Chapter 5, "Date Intervals, Formats, and Functions."

For more information about Fame frequencies, see the section "Understanding Frequencies" in the *User's Guide to Fame*.

Table 47.4 Mapping Fame Frequencies

Fame Frequency	SAS Time Interval
WEEKLY (SUNDAY)	WEEK.2
WEEKLY (MONDAY)	WEEK.3
WEEKLY (TUESDAY)	WEEK.4
WEEKLY (WEDNESDAY)	WEEK.5
WEEKLY (THURSDAY)	WEEK.6
WEEKLY (FRIDAY)	WEEK.7
WEEKLY (SATURDAY)	WEEK.1
BIWEEKLY (ASUNDAY)	WEEK2.2
BIWEEKLY (AMONDAY)	WEEK2.3
BIWEEKLY (ATUESDAY)	WEEK2.4
BIWEEKLY (AWEDNESDAY)	WEEK2.5
BIWEEKLY (ATHURSDAY)	WEEK2.6
BIWEEKLY (AFRIDAY)	WEEK2.7
BIWEEKLY (ASATURDAY)	WEEK2.1
BIWEEKLY (BSUNDAY)	WEEK2.9
BIWEEKLY (BMONDAY)	WEEK2.10
BIWEEKLY (BTUESDAY)	WEEK2.11
BIWEEKLY (BWEDNESDAY)	WEEK2.12

Table 47.4 *continued*

Fame Frequency	SAS Time Interval
BIWEEKLY (BTHURSDAY)	WEEK2.13
BIWEEKLY (BFRIDAY)	WEEK2.14
BIWEEKLY (BSATURDAY)	WEEK2.8
BIMONTHLY (NOVEMBER)	MONTH2.2
BIMONTHLY	MONTH2.1
QUARTERLY (OCTOBER)	QTR.2
QUARTERLY (NOVEMBER)	QTR.3
QUARTERLY	QTR.1
ANNUAL (JANUARY)	YEAR.2
ANNUAL (FEBRUARY)	YEAR.3
ANNUAL (MARCH)	YEAR.4
ANNUAL (APRIL)	YEAR.5
ANNUAL (MAY)	YEAR.6
ANNUAL (JUNE)	YEAR.7
ANNUAL (JULY)	YEAR.8
ANNUAL (AUGUST)	YEAR.9
ANNUAL (SEPTEMBER)	YEAR.10
ANNUAL (OCTOBER)	YEAR.11
ANNUAL (NOVEMBER)	YEAR.12
ANNUAL	YEAR.1
SEMIANNUAL (JULY)	SEMIYEAR.2
SEMIANNUAL (AUGUST)	SEMIYEAR.3
SEMIANNUAL (SEPTEMBER)	SEMIYEAR.4
SEMIANNUAL (OCTOBER)	SEMIYEAR.5
SEMIANNUAL (NOVEMBER)	SEMIYEAR.6
SEMIANNUAL	SEMIYEAR.1
YPP	Not supported
PPY	Not supported
SECONDLY	SECOND
MINUTELY	MINUTE
HOURLY	HOUR
DAILY	DAY
BUSINESS	WEEKDAY
TENDAY	TENDAY
TWICEMONTHLY	SEMIMONTH
MONTHLY	MONTH

Performing the Crosslist Selection Function

There are two methods of performing the crosslist selection function. The first method uses two Fame namelists, and the second method uses one namelist and one BY group specified in the WHERE= clause of the INSET= option.

For example, suppose that your Fame database has a string case series named TICKER, so that when the Fame NL function is used on TICKER, it returns the following namelist:

```
Ticker = {AOL, C, CVX, F, GM, HPQ, IBM, INDUA, INTC, SPX, SUNW, XOM}
```

Also suppose your time series are named in *fame_namelist2* as

```
{adjust, close, high, low, open, volume, uclose, uhigh, ulow, uopen, uvolume}
```

When you specify the following statements, the 132 variables shown in Table 47.5 are selected by the CROSSLIST= option:

```
LIBNAME test sasefame 'physical name of test database'
        RANGE='01jan1999'd - '31mar1999'd
        CROSSLIST=(nl(ticker),
                   {adjust, close, high, low, open, volume,
                    uclose, uhigh, ulow, uopen, uvolume})
        ;
```

Table 47.5 SAS Variables Selected by CROSSLIST= Option

AOL.ADJUST	C.ADJUST	CVX.ADJUST	F.ADJUST
AOL.CLOSE	C.CLOSE	CVX.CLOSE	F.CLOSE
AOL.HIGH	C.HIGH	CVX.HIGH	F.HIGH
AOL.LOW	C.LOW	CVX.LOW	F.LOW
AOL.OPEN	C.OPEN	CVX.OPEN	F.OPEN
AOL.UCLOSE	C.UCLOSE	CVX.UCLOSE	F.UCLOSE
AOL.UHIGH	C.UHIGH	CVX.UHIGH	F.UHIGH
AOL.ULOW	C.ULOW	CVX.ULOW	F.ULOW
AOL.UOPEN	C.UOPEN	CVX.UOPEN	F.UOPEN
AOL.UVOLUME	C.UVOLUME	CVX.UVOLUME	F.UVOLUME
AOL.VOLUME	C.VOLUME	CVX.VOLUME	F.VOLUME
GM.ADJUST	HPQ.ADJUST	IBM.ADJUST	INDUA.ADJUST
GM.CLOSE	HPQ.CLOSE	IBM.CLOSE	INDUA.CLOSE
GM.HIGH	HPQ.HIGH	IBM.HIGH	INDUA.HIGH
GM.LOW	HPQ.LOW	IBM.LOW	INDUA.LOW
GM.OPEN	HPQ.OPEN	IBM.OPEN	INDUA.OPEN
GM.UCLOSE	HPQ.UCLOSE	IBM.UCLOSE	INDUA.UCLOSE
GM.UHIGH	HPQ.UHIGH	IBM.UHIGH	INDUA.UHIGH
GM.ULOW	HPQ.ULOW	IBM.ULOW	INDUA.ULOW

Table 47.5 *continued*

GM.UOPEN	HPQ.UOPEN	IBM.UOPEN	INDUA.UOPEN
GM.UVOLUME	HPQ.UVOLUME	IBM.UVOLUME	INDUA.UVOLUME
GM.VOLUME	HPQ.VOLUME	IBM.VOLUME	INDUA.VOLUME
INTC.ADJUST	SPX.ADJUST	SUNW.ADJUST	XOM.ADJUST
INTC.CLOSE	SPX.CLOSE	SUNW.CLOSE	XOM.CLOSE
INTC.HIGH	SPX.HIGH	SUNW.HIGH	XOM.HIGH
INTC.LOW	SPX.LOW	SUNW.LOW	XOM.LOW
INTC.OPEN	SPX.OPEN	SUNW.OPEN	XOM.OPEN
INTC.UCLOSE	SPX.UCLOSE	SUNW.UCLOSE	XOM.UCLOSE
INTC.UHIGH	SPX.UHIGH	SUNW.UHIGH	XOM.UHIGH
INTC.ULOW	SPX.ULOW	SUNW.ULOW	XOM.ULOW
INTC.UOPEN	SPX.UOPEN	SUNW.UOPEN	XOM.UOPEN
INTC.UVOLUME	SPX.UVOLUME	SUNW.UVOLUME	XOM.UVOLUME
INTC.VOLUME	SPX.VOLUME	SUNW.VOLUME	XOM.VOLUME

Instead of using two namelists, you can use the WHERE= clause in an INSET= option to perform the crossproduct of the BY variables specified in your input data set via the WHERE= clause and the members named in your namelist. The following statements define a SAS input data set named INSETA to use as input for the CROSSLIST= option instead of using the Fame namelist:

```
DATA INSETA;
   LENGTH tick $5;
/* AOL, C, CVX, F, GM, HPQ, IBM, INDUA, INTC, SPX, SUNW, XOM */
   tick='AOL'; output;
   tick='C'; output;
   tick='CVX'; output;
   tick='F'; output;
   tick='GM'; output;
   tick='HPQ'; output;
   tick='IBM'; output;
   tick='INDUA'; output;
   tick='INTC'; output;
   tick='SPX'; output;
   tick='SUNW'; output;
   tick='XOM'; output;
RUN;

LIBNAME test sasefame 'physical name of test database'
        RANGE='01jan1999'd - '31mar1999'd
        INSET=(inseta, where=tick)
        CROSSLIST=(
              {adjust, close, high, low, open, volume,
               uclose, uhigh, ulow, uopen, uvolume})
        ;
```

Using a SAS INSET statement with a WHERE clause and using a Fame namelist in the CROSSLIST= statement are equivalent ways of performing the same selection function. In the preceding example, the Fame ticker namelist corresponds to the SAS input data set's BY variable named TICK.

Note that the *fame_bygroup* that you specify in the WHERE= clause must match the BY-variable name used in your input data set in order for the CROSSLIST= option to perform the desired selection. If one of the time series listed in *fame_namelist2* does not exist, the SASEFAME engine stops processing the remainder of the namelist. For complete results, make sure that your *fame_namelist2* is accurate and does not name unknown variables. The same holds true for *fame_namelist1* and the BY-variable values named in the input data set and used in the WHERE= clause.

Examples: SASEFAME Interface Engine

In this section, the examples were run on Windows, so the physical names used in the LIBNAME *libref* SASEFAME statement reflect the syntax necessary for that platform. In general, Windows environments use backslashes in their pathname, and the UNIX environments use forward slashes.

Example 47.1: Converting an Entire Fame Database

To enable conversion of all time series, no wildcard is specified, so the default "?" wildcard is used. Always consider both the number of time series and the number of observations generated by the conversion process. The converted series reside in the Fame Work database during the SAS DATA step. You can further limit your resulting SAS data set by using KEEP, DROP, or WHERE statements inside your DATA step.

The following statements convert a Fame database and print out its contents:

```
options pagesize=60 linesize=80 validvarname=any ;
%let FAME=%sysget(FAME);
%put(&FAME);
%let FAMETEMP=%sysget(FAME_TEMP);
%put(&FAMETEMP);

libname famedir sasefame "%sysget(FAME_DATA)"
        convert=(freq=annual technique=constant);

libname mydir "%sysget(FAME_TEMP)";

data mydir.a;  /* add data set to mydir */
   set famedir.oecd1;
   /* Read in oecd1.db data from the Organization */
   /*  For Economic Cooperation and Development    */
   where date between '01jan88'd and '31dec93'd;
run;

proc print data=mydir.a;
run;
```

In the preceding example, the Fame database is called OECD1.DB, and it resides in the famedir directory. The DATA statement names the SAS output data set 'a' that will reside in mydir. All time series in the Fame OECD1.DB database will be converted to an annual frequency and reside in the mydir.a SAS data set. Because the time series variable names contain the special glue symbol '.', the SAS option statement specifies VALIDVARNAME=ANY. For more information about this option, see the *SAS System Options: Reference*. The Fame environment variable is the location of the Fame installation. In the Windows environment, the log would look like this:

```
1             options validvarname=any;

2             %let FAME=%sysget(FAME);
3             %put(&FAME);
(C:\PROGRA~1\FAME)
4             %let FAMETEMP=%sysget(FAME_TEMP);
5             %put(&FAMETEMP);
(\\ge\U11\saskff\fametemp\)
6
7             libname famedir sasefame "&FAME\util"
8                      convert=(freq=annual technique=constant);
NOTE: Libref FAMEDIR was successfully assigned as follows:
      Engine:        FAMECHLI
      Physical Name: C:\PROGRA~1\FAME\util
9
10            libname mydir '\\dntsrc\usrtmp\saskff';
NOTE: Libref MYDIR was successfully assigned as follows:
      Engine:        V9
      Physical Name: \\dntsrc\usrtmp\saskff
11
12            data mydir.a;  /* add data set to mydir */
13               set famedir.oecd1;
AUS.DIRDES -- SERIES (NUMERIC by ANNUAL)
AUS.DIRDES copied to work data base as AUS.DIRDES.
```

For more about the glue DOT character, see the section "Gluing Names Together" in the *User's Guide to Fame*. In the preceding log, the variable name AUS.DIRDES uses the glue DOT between AUS and DIRDES.

The PROC PRINT statement produces the results shown in Output 47.1.1, which displays all observations in the mydir.a SAS data set.

Output 47.1.1 Listing of OUT=MYDIR.A of the OECD1 Fame Data

Obs	DATE	AUS.DIRDES	AUS.HERD	AUT.DIRDES	AUT.HERD	BEL.DIRDES	BEL.HERD	CAN.DIRDES	CAN.HERD
1	1988	750	1072.90	.	.	374	16572.70	1589.60	2006
2	1989	.	.	.	.	.	18310.70	1737.00	2214
3	1990	.	.	.	.	.	18874.20	1859.20	2347
4	1991	.	.	.	.	.	.	1959.60	2488

Obs	CHE.DIRDES	CHE.HERD	DEU.DIRDES	DEU.HERD	DNK.DIRDES	DNK.HERD	ESP.DIRDES	ESP.HERD
1	632.100	1532	3538.60	8780.00	258.100	2662	508.200	55365.5
2	.	1648	3777.20	9226.60	284.800	2951	623.600	69270.5
3	.	.	2953.30	9700.00	.	.	723.600	78848.0
4	.	.	.	.	.	.	.	89908.0

Obs	FIN.DIRDES	FIN.HERD	FRA.DIRDES	FRA.HERD	GBR.DIRDES	GBR.HERD	GRC.DIRDES	GRC.HERD
1	247.700	1602.0	2573.50	19272.00	2627.00	1592.00	60.600	6674.50
2	259.700	1725.5	2856.50	21347.80	2844.10	1774.20	119.800	14485.20
3	271.000	1839.0	3005.20	22240.00	.	.	.	.
4	.	.	.	.	.	.	.	.

Obs	IRL.DIRDES	IRL.HERD	ISL.DIRDES	ISL.HERD	ITA.DIRDES	ITA.HERD	JPN.DIRDES	JPN.HERD	NLD.DIRDES
1	49.6000	37.0730	.	.	1861.50	2699927	9657.20	2014073	883
2	50.2000	39.0130	10.3000	786.762	1968.00	2923504	10405.90	2129372	945
3	51.7000	.	11.0000	902.498	2075.00	3183071	.	2296992	.
4	.	.	11.8000	990.865	2137.80	3374000	.	.	.

Obs	NLD.HERD	NOR.DIRDES	NOR.HERD	NZL.DIRDES	NZL.HERD	PRT.DIRDES	PRT.HERD	SWE.DIRDES
1	2105	.	.	.	.	111.5	10158.20	.
2	2202	308.900	2771.40	78.7000	143.800	.	.	1076
3	.	.	.	.	.	.	.	.
4	.	352.000	3100.00	.	.	.	.	.

Obs	SWE.HERD	TUR.DIRDES	TUR.HERD	USA.DIRDES	USA.HERD	YUG.DIRDES	YUG.HERD
1	.	174.400	74474	20246.20	20246.20	233.000	29.81
2	11104	212.300	143951	22159.50	22159.50	205.100	375.22
3	.	.	.	23556.10	23556.10	.	2588.50
4	.	.	.	24953.80	24953.80	.	.

Example 47.2: Reading Time Series from the Fame Database

This example uses the Fame wildcard option to limit the number of series converted. The following statements show how to read only series whose names begin with WSPCA:

```
options validvarname=any;

%let FAME=%sysget(FAME);
%put(&FAME);
%let FAMETEMP=%sysget(FAME_TEMP);
%put(&FAMETEMP);

libname lib1 sasefame "%sysget(FAME_DATA)"
        wildcard="wspca?"
        convert=(technique=constant freq=twicemonthly );

libname lib2 "%sysget(FAME_TEMP)";

data lib2.twild(label='Annual Series from the FAMEECON.db');
   set lib1.subecon;
   where date between '01jan93'd and '31dec93'd;
   /* keep only  */
   keep date wspca;
run;

proc contents data=lib2.twild;
run;

proc print data=lib2.twild;
run;
```

Output 47.2.1 and Output 47.2.2 show the results of using WILDCARD="WSPCA?".

Output 47.2.1 Contents of OUT=LIB2.TWILD of the SUBECON Fame Data

The CONTENTS Procedure

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Informat	Label
1	DATE	Num	8	DATE9.	9.	Date of Observation
2	WSPCA	Num	8			STANDARD & POOR'S WEEKLY BOND YIELD: COMPOSITE, A

The WILDCARD="WSPCA?" option limits reading to only those series whose names begin with WSPCA. The KEEP statement further restricts the SAS data set to include only the series named WSPCA and the DATE variable. The time interval that is used for the conversion is TWICEMONTHLY.

Output 47.2.2 Listing of OUT=LIB2.TWILD of the SUBECON Fame Data

Obs	DATE	WSPCA
1	01JAN1993	8.59400
2	16JAN1993	8.50562
3	01FEB1993	8.47000
4	16FEB1993	8.31000
5	01MAR1993	8.27000
6	16MAR1993	8.29250
7	01APR1993	8.32400
8	16APR1993	8.56333
9	01MAY1993	8.37867
10	16MAY1993	8.26313
11	01JUN1993	8.21333
12	16JUN1993	8.14400
13	01JUL1993	8.09067
14	16JUL1993	8.09937
15	01AUG1993	7.98533
16	16AUG1993	7.91600

Example 47.3: Writing Time Series to the SAS Data Set

The following statements use the DROP statement to exclude certain time series from the SAS data set. (You can also use the KEEP statement to include certain series in the SAS data set.)

```
options validvarname=any;

%let FAME=%sysget(FAME);
%put(&FAME);
%let FAMETEMP=%sysget(FAME_TEMP);
%put(&FAMETEMP);

libname famedir sasefame "%sysget(FAME_DATA)"
        convert=(freq=annual technique=constant);

libname mydir "%sysget(FAME_TEMP)";

data mydir.a;  /* add data set to mydir */
   set famedir.oecd1;
   drop 'ita.dirdes'n--'jpn.herd'n 'tur.dirdes'n--'usa.herd'n;
   where date between '01jan88'd and '31dec93'd;
run;

title1 "OECD1: TECH=Constant, FREQ=Annual";
title2 "Drop Using N-literals";

proc print data=mydir.a;
run;
```

Output 47.3.1 shows the results.

Output 47.3.1 Listing of OUT=MYDIR.A of the OECD1 Fame Data

OECD1: TECH=Constant, FREQ=Annual
Drop Using N-literals

Obs	DATE	AUS.DIRDES	AUS.HERD	AUT.DIRDES	AUT.HERD	BEL.DIRDES	BEL.HERD	CAN.DIRDES	CAN.HERD
1	1988	750	1072.90	.	.	374	16572.70	1589.60	2006
2	1989	.	.	.	.	.	18310.70	1737.00	2214
3	1990	.	.	.	.	.	18874.20	1859.20	2347
4	1991	.	.	.	.	.	.	1959.60	2488

Obs	CHE.DIRDES	CHE.HERD	DEU.DIRDES	DEU.HERD	DNK.DIRDES	DNK.HERD	ESP.DIRDES	ESP.HERD
1	632.100	1532	3538.60	8780.00	258.100	2662	508.200	55365.5
2	.	1648	3777.20	9226.60	284.800	2951	623.600	69270.5
3	.	.	2953.30	9700.00	.	.	723.600	78848.0
4	.	.	.	.	.	.	.	89908.0

Obs	FIN.DIRDES	FIN.HERD	FRA.DIRDES	FRA.HERD	GBR.DIRDES	GBR.HERD	GRC.DIRDES	GRC.HERD
1	247.700	1602.0	2573.50	19272.00	2627.00	1592.00	60.600	6674.50
2	259.700	1725.5	2856.50	21347.80	2844.10	1774.20	119.800	14485.20
3	271.000	1839.0	3005.20	22240.00	.	.	.	.
4	.	.	.	.	.	.	.	.

Obs	IRL.DIRDES	IRL.HERD	ISL.DIRDES	ISL.HERD	NLD.DIRDES	NLD.HERD	NOR.DIRDES	NOR.HERD
1	49.6000	37.0730	.	.	883	2105	.	.
2	50.2000	39.0130	10.3000	786.762	945	2202	308.900	2771.40
3	51.7000	.	11.0000	902.498	.	.	.	.
4	.	.	11.8000	990.865	.	.	352.000	3100.00

Obs	NZL.DIRDES	NZL.HERD	PRT.DIRDES	PRT.HERD	SWE.DIRDES	SWE.HERD	YUG.DIRDES	YUG.HERD
1	.	.	111.5	10158.20	.	.	233.000	29.81
2	78.7000	143.800	.	.	1076	11104	205.100	375.22
3	.	.	.	.	.	.	.	2588.50
4	.	.	.	.	.	.	.	.

Note that the SAS option VALIDVARNAME=ANY was used at the beginning of this example because special characters are present in the time series names. SAS variables that contain certain special characters are called *n-literals* and are referenced in SAS code, as shown in this example.

You can rename your SAS variables by using the RENAME statement. The following statements show how to use *n-literals* when selecting variables that you want to keep and how to rename some of your kept variables:

```
options validvarname=any;

%let FAME=%sysget(FAME);
%put(&FAME);
%let FAMETEMP=%sysget(FAME_TEMP);
%put(&FAMETEMP);

libname famedir sasefame "%sysget(FAME_DATA)"
        convert=(freq=annual technique=constant);
```

```
libname mydir "%sysget(FAME_TEMP)";

data mydir.a;  /* add data set to mydir */
   set famedir.oecd1;
   /* keep and rename  */
   keep date 'ita.dirdes'n--'jpn.herd'n 'tur.dirdes'n--'usa.herd'n;
   rename 'ita.dirdes'n='italy.dirdes'n
          'jpn.dirdes'n='japan.dirdes'n
          'tur.dirdes'n='turkey.dirdes'n
          'usa.dirdes'n='united.states.of.america.dirdes'n ;
run;

title1 "OECD1: TECH=Constant, FREQ=Annual";
title2 "keep statement using n-literals";
title3 "rename statement using n-literals";

proc print data=mydir.a;
run;
```

Output 47.3.2 shows the results.

Output 47.3.2 Listing of OUT=MYDIR.A of the OECD1 Fame Data

OECD1: TECH=Constant, FREQ=Annual
keep statement using n-literals
rename statement using n-literals

Obs	DATE	italy.dirdes	ITA.HERD	japan.dirdes	JPN.HERD	turkey.dirdes	TUR.HERD
1	1985	1344.90	1751008	8065.70	1789780	144.800	22196
2	1986	1460.60	2004453	8290.10	1832575	136.400	26957
3	1987	1674.40	2362102	9120.80	1957921	121.900	32309
4	1988	1861.50	2699927	9657.20	2014073	174.400	74474
5	1989	1968.00	2923504	10405.90	2129372	212.300	143951
6	1990	2075.00	3183071	.	2296992	.	.
7	1991	2137.80	3374000	.	.	.	.

Obs	united.states.of.america.dirdes	USA.HERD
1	14786.00	14786.00
2	16566.90	16566.90
3	18326.10	18326.10
4	20246.20	20246.20
5	22159.50	22159.50
6	23556.10	23556.10
7	24953.80	24953.80

Example 47.4: Limiting the Time Range of Data

You can also limit the time range of the data in the SAS data set by using the RANGE= option in the LIBNAME statement or the WHERE statement in the DATA step to process the time ID variable DATE only when it falls in the range you are interested in.

All data for 1988, 1989, and 1990 are included in the SAS data set that is generated by using the RANGE='01JAN1988'D - '31DEC1990'D option or the WHERE DATE BETWEEN '01JAN88'D AND '31DEC90'D statement. The difference is that the RANGE= option uses less space in the Fame Work database. If you have a very large database and you want to use less space in your Fame Work database while you are processing the OECD1 database, you should use the RANGE= option as shown in the following statements:

```
options validvarname=any;

%let FAME=%sysget(FAME);
%put(&FAME);
%let FAMETEMP=%sysget(FAME_TEMP);
%put(&FAMETEMP);

libname famedir SASEFAME "%sysget(FAME_DATA)"
        convert=(freq=annual technique=constant)
        range='01jan1988'd - '31dec1990'd;

libname mydir "%sysget(FAME_TEMP)";

data mydir.a;  /* add data set to mydir */
   set famedir.oecd1;
   /* range on the libref restricts the dates *
    * read from famedir's oecd1 database      */
run;

title1 "OECD1: TECH=Constant, FREQ=Annual";
proc print data=mydir.a;
run;
```

Output 47.4.1 shows the results.

Output 47.4.1 Listing of OUT=MYDIR.A of the OECD1 Fame Data Using RANGE= Option

OECD1: TECH=Constant, FREQ=Annual

Obs	DATE	AUS.DIRDES	AUS.HERD	AUT.DIRDES	AUT.HERD	BEL.DIRDES	BEL.HERD	CAN.DIRDES	CAN.HERD
1	1988	750	1072.90	.	.	374	16572.70	1589.60	2006
2	1989	.	.	.	.	.	18310.70	1737.00	2214
3	1990	.	.	.	.	.	18874.20	1859.20	2347

Obs	CHE.DIRDES	CHE.HERD	DEU.DIRDES	DEU.HERD	DNK.DIRDES	DNK.HERD	ESP.DIRDES	ESP.HERD
1	632.100	1532	3538.60	8780.00	258.100	2662	508.200	55365.5
2	.	1648	3777.20	9226.60	284.800	2951	623.600	69270.5
3	.	.	2953.30	9700.00	.	.	723.600	78848.0

Obs	FIN.DIRDES	FIN.HERD	FRA.DIRDES	FRA.HERD	GBR.DIRDES	GBR.HERD	GRC.DIRDES	GRC.HERD
1	247.700	1602.0	2573.50	19272.00	2627.00	1592.00	60.600	6674.50
2	259.700	1725.5	2856.50	21347.80	2844.10	1774.20	119.800	14485.20
3	271.000	1839.0	3005.20	22240.00	.	.	.	.

Obs	IRL.DIRDES	IRL.HERD	ISL.DIRDES	ISL.HERD	ITA.DIRDES	ITA.HERD	JPN.DIRDES	JPN.HERD	NLD.DIRDES
1	49.6000	37.0730	.	.	1861.5	2699927	9657.20	2014073	883
2	50.2000	39.0130	10.3000	786.762	1968.0	2923504	10405.90	2129372	945
3	51.7000	.	11.0000	902.498	2075.0	3183071	.	2296992	.

Obs	NLD.HERD	NOR.DIRDES	NOR.HERD	NZL.DIRDES	NZL.HERD	PRT.DIRDES	PRT.HERD	SWE.DIRDES
1	2105	.	.	.	.	111.5	10158.20	.
2	2202	308.900	2771.40	78.7000	143.800	.	.	1076
3	.	.	.	.	.	.	.	.

Obs	SWE.HERD	TUR.DIRDES	TUR.HERD	USA.DIRDES	USA.HERD	YUG.DIRDES	YUG.HERD
1	.	174.400	74474	20246.20	20246.20	233.000	29.81
2	11104	212.300	143951	22159.50	22159.50	205.100	375.22
3	.	.	.	23556.10	23556.10	.	2588.50

The following statements show how you can use the WHERE statement in the DATA step to process the time ID variable DATE only when it falls in the range you are interested in:

```
options validvarname=any;

%let FAME=%sysget(FAME);
%put(&FAME);
%let FAMETEMP=%sysget(FAME_TEMP);
%put(&FAMETEMP);

libname famedir SASEFAME "%sysget(FAME_DATA)"
        convert=(freq=annual technique=constant);

libname mydir "%sysget(FAME_TEMP)";

data mydir.a;  /* add data set to mydir */
   set famedir.oecd1;
   /* where only  */
   where date between '01jan88'd and '31dec90'd;
run;

title1 "OECD1: TECH=Constant, FREQ=Annual";
proc print data=mydir.a;
run;
```

In Output 47.4.2, you can see that the result from the WHERE statement is the same as the result in Output 47.4.1 from using the RANGE= option.

Output 47.4.2 Listing of OUT=MYDIR.A of the OECD1 Fame Data Using WHERE Statement

OECD1: TECH=Constant, FREQ=Annual

Obs	DATE	AUS.DIRDES	AUS.HERD	AUT.DIRDES	AUT.HERD	BEL.DIRDES	BEL.HERD	CAN.DIRDES	CAN.HERD
1	1988	750	1072.90	.	.	374	16572.70	1589.60	2006
2	1989	.	.	.	.	.	18310.70	1737.00	2214
3	1990	.	.	.	.	.	18874.20	1859.20	2347

Obs	CHE.DIRDES	CHE.HERD	DEU.DIRDES	DEU.HERD	DNK.DIRDES	DNK.HERD	ESP.DIRDES	ESP.HERD
1	632.100	1532	3538.60	8780.00	258.100	2662	508.200	55365.5
2	.	1648	3777.20	9226.60	284.800	2951	623.600	69270.5
3	.	.	2953.30	9700.00	.	.	723.600	78848.0

Obs	FIN.DIRDES	FIN.HERD	FRA.DIRDES	FRA.HERD	GBR.DIRDES	GBR.HERD	GRC.DIRDES	GRC.HERD
1	247.700	1602.0	2573.50	19272.00	2627.00	1592.00	60.600	6674.50
2	259.700	1725.5	2856.50	21347.80	2844.10	1774.20	119.800	14485.20
3	271.000	1839.0	3005.20	22240.00	.	.	.	.

Obs	IRL.DIRDES	IRL.HERD	ISL.DIRDES	ISL.HERD	ITA.DIRDES	ITA.HERD	JPN.DIRDES	JPN.HERD	NLD.DIRDES
1	49.6000	37.0730	.	.	1861.5	2699927	9657.20	2014073	883
2	50.2000	39.0130	10.3000	786.762	1968.0	2923504	10405.90	2129372	945
3	51.7000	.	11.0000	902.498	2075.0	3183071	.	2296992	.

Obs	NLD.HERD	NOR.DIRDES	NOR.HERD	NZL.DIRDES	NZL.HERD	PRT.DIRDES	PRT.HERD	SWE.DIRDES
1	2105	.	.	.	.	111.5	10158.20	.
2	2202	308.900	2771.40	78.7000	143.800	.	.	1076
3	.	.	.	.	.	.	.	.

Obs	SWE.HERD	TUR.DIRDES	TUR.HERD	USA.DIRDES	USA.HERD	YUG.DIRDES	YUG.HERD
1	.	174.400	74474	20246.20	20246.20	233.000	29.81
2	11104	212.300	143951	22159.50	22159.50	205.100	375.22
3	.	.	.	23556.10	23556.10	.	2588.50

For more information about the KEEP, DROP, RENAME, and WHERE statements, see *SAS Language Reference: Concepts*.

Example 47.5: Creating a View Using the SQL Procedure and the SASEFAME Engine

The following statements create a view of OECD data by using the SQL procedure's FROM and USING clauses. For more information about SQL views, see the *BASE SAS Procedures Guide*.

```
title1 'famesql5: PROC SQL Dual Embedded Libraries w/ FAME option';
options validvarname=any;

%let FAME=%sysget(FAME);
%put(&FAME);
%let FAMETEMP=%sysget(FAME_TEMP);
%put(&FAMETEMP);

title2 'OECD1: Dual Embedded Library Allocations with FAME Option';
proc sql;
   create view fameview as
      select date, 'fin.herd'n
         from lib1.oecd1
      using libname lib1 sasefame "%sysget(FAME_DATA)"
               convert=(tech=constant freq=annual),
            libname temp "%sysget(FAME_TEMP)";
quit;

title2 'OECD1: Print of View from Embedded Library with FAME Option';
proc print data=fameview;
run;
```

Output 47.5.1 shows the results.

Output 47.5.1 Printout of the Fame View of OECD Data

famesql5: PROC SQL Dual Embedded Libraries w/ FAME option
OECD1: Print of View from Embedded Library with FAME Option

Obs	DATE	FIN.HERD
1	1985	1097.00
2	1986	1234.00
3	1987	1401.30
4	1988	1602.00
5	1989	1725.50
6	1990	1839.00
7	1991	.

The following statements create a view of the DRI Basic Economic data by using the SQL procedure's FROM and USING clauses:

```
title2 'SUBECON: Dual Embedded Library Allocations with FAME Option';
options validvarname=any;

%let FAME=%sysget(FAME);
%put(&FAME);
%let FAMETEMP=%sysget(FAME_TEMP);
%put(&FAMETEMP);

proc sql;
   create view fameview as
   select date, gaa
      from lib1.subecon
      using libname lib1 sasefame "%sysget(FAME_DATA)"
                         convert=(tech=constant freq=annual),
            libname temp "%sysget(FAME_TEMP)";
quit;

title2 'SUBECON: Print of View from Embedded Library with FAME Option';
proc print data=fameview;
run;
```

Output 47.5.2 shows the results.

Output 47.5.2 Printout of the Fame View of DRI Basic Economic Data

famesql5: PROC SQL Dual Embedded Libraries w/ FAME option
SUBECON: Print of View from Embedded Library with FAME Option

Obs	DATE	GAA
1	1946	.
2	1947	.
3	1948	23174
4	1949	19003
5	1950	24960
6	1951	21906
7	1952	20246
8	1953	20912
9	1954	21056
10	1955	27168
11	1956	27638
12	1957	26723
13	1958	22929
14	1959	29729
15	1960	28444
16	1961	28226
17	1962	32396
18	1963	34932
19	1964	40024
20	1965	47941
21	1966	51429
22	1967	49164
23	1968	51208
24	1969	49371
25	1970	44034
26	1971	52352
27	1972	62644
28	1973	81645
29	1974	91028
30	1975	89494
31	1976	109492
32	1977	130260
33	1978	154357
34	1979	173428
35	1980	156096
36	1981	147765
37	1982	113216
38	1983	133495
39	1984	146448
40	1985	128522
41	1986	111338
42	1987	160785
43	1988	210532
44	1989	201637
45	1990	218702
46	1991	210666

Output 47.5.2 *continued*

famesql5: PROC SQL Dual Embedded Libraries w/ FAME option
SUBECON: Print of View from Embedded Library with FAME Option

Obs	DATE	GAA
47	1992	.
48	1993	.

The following statements create a view of the DB77 database by using the SQL procedure's FROM and USING clauses:

```
title2 'DB77: Dual Embedded Library Allocations with FAME Option';
options validvarname=any;

%let FAME=%sysget(FAME);
%put(&FAME);
%let FAMETEMP=%sysget(FAME_TEMP);
%put(&FAMETEMP);

proc sql;
   create view fameview as
      select date, ann, 'qandom.x'n
      from lib1.db77
      using libname lib1 sasefame "%sysget(FAME_DATA)"
                    convert=(tech=constant freq=annual),
            libname temp "%sysget(FAME_TEMP)";
 quit;

title2 'DB77: Print of View from Embedded Library with FAME Option';
proc print data=fameview;
run;
```

Output 47.5.3 shows the results.

Output 47.5.3 Printout of the Fame View of DB77 Data

famesql5: PROC SQL Dual Embedded Libraries w/ FAME option
DB77: Print of View from Embedded Library with FAME Option

Obs	DATE	ANN	QANDOM.X
1	1959	.	0.56147
2	1960	.	0.51031
3	1961	.	.
4	1962	.	.
5	1963	.	.
6	1964	.	.
7	1965	.	.
8	1966	.	.
9	1967	.	.
10	1968	.	.
11	1969	.	.
12	1970	.	.
13	1971	.	.
14	1972	.	.
15	1973	.	.
16	1974	.	.
17	1975	.	.
18	1976	.	.
19	1977	.	.
20	1978	.	.
21	1979	.	.
22	1980	100	.
23	1981	101	.
24	1982	102	.
25	1983	103	.
26	1984	104	.
27	1985	105	.
28	1986	106	.
29	1987	107	.
30	1988	109	.
31	1989	111	.

The following statements create a view of the Data Resources Incorporated (DRI) Basic Economic data by using the SQL procedure's FROM and USING clauses:

```
title2 'DRIECON: Dual Embedded Library Allocations with FAME Option';
options validvarname=any;

%let FAME=%sysget(FAME);
%put(&FAME);
%let FAMETEMP=%sysget(FAME_TEMP);
%put(&FAMETEMP);

proc sql;
   create view fameview as
```

```
      select date, husts
      from lib1.driecon
      using libname lib1 sasefame "%sysget(FAME_DATA)"
                         convert=(tech=constant freq=annual)
                         range='01jan1980'd - '01jan2006'd ,
            libname temp "%sysget(FAME_TEMP)";
quit;

title2 'DRIECON: Print of View from Embedded Library with FAME Option';
proc print data=fameview;
run;
```

The SAS option VALIDVARNAME=ANY is used at the beginning of this example because special characters are present in the time series names. The output from this example shows how each Fame view is the output of the SASEFAME engine's processing. Different engine options could have been used in the USING LIBNAME clause if desired. Output 47.5.4 shows the results.

Output 47.5.4 Printout of the Fame View of DRI Basic Economic Data

famesql5: PROC SQL Dual Embedded Libraries w/ FAME option
DRIECON: Print of View from Embedded Library with FAME Option

Obs	DATE	HUSTS
1	1980	1292.2
2	1981	1084.2
3	1982	1062.2
4	1983	1703.0
5	1984	1749.5
6	1985	1741.8
7	1986	1805.4
8	1987	1620.5
9	1988	1488.1
10	1989	1376.1
11	1990	1192.7
12	1991	1013.9
13	1992	1199.7
14	1993	1287.6
15	1994	1457.0
16	1995	1354.1
17	1996	1476.8
18	1997	1474.0
19	1998	1616.9
20	1999	1666.5
21	2000	1568.7
22	2001	1602.7
23	2002	1704.9
24	2003	.

Example 47.6: Reading Other Fame Data Objects with the FAMEOUT= Option

This example shows how you can designate the data objects that are output to your SAS data set by using the FAMEOUT= option. In this example, the FAMEOUT=FORMULA option selects the formulas and their source definitions to be output. The RANGE= option is ignored because no time series are selected when FAMEOUT=FORMULA is specified.

```
options validvarname=any ls=90;

%let FAME=%sysget(FAME);
%put(&FAME);
%let FAMETEMP=%sysget(FAME_TEMP);
%put(&FAMETEMP);

libname lib6 sasefame "%sysget(FAME_DATA)"
   fameout=formula
   convert=(frequency=business technique=constant)
   range='02jan1995'd - '25jul1997'd
   wildcard="?YIELD?" ;

data crout;
   set lib6.training;
   keep 'S.GM.YIELD.A'n -- 'S.XON.YIELD.A'n ;
run;

title1 'Formulas from the TRAINING DB, FAMEOUT=FORMULA Option';
title2 'Using WILDCARD="?YIELD?"';
proc contents
   data=crout;
run;
```

Output 47.6.1shows the results.

Output 47.6.1 Contents of OUT=CROUT from the FAMEOUT=FORMULA Option of the Fame TRAINING Data

Formulas from the TRAINING DB, FAMEOUT=FORMULA Option Using WILDCARD="?YIELD?"

The CONTENTS Procedure

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
1	S.GM.YIELD.A	Char	82
2	S.GM__PP.YIELD.A	Char	82
3	S.HWP.YIELD.A	Char	82
4	S.IBM.YIELD.A	Char	82
5	S.INDUT.YIELD.A	Char	82
6	S.SPAL.YIELD.A	Char	82
7	S.SPALN.YIELD.A	Char	82
8	S.SUNW.YIELD.A	Char	82
9	S.XOM.YIELD.A	Char	82
10	S.XON.YIELD.A	Char	82

The FAMEOUT=FORMULA option restricts the SAS data set to include only formulas. The WILDCARD="?YIELD?" option further limits the selection of formulas to those whose names contain "YIELD".

```
options validvarname=any linesize=79;

title1 'Formulas from the TRAINING DB, FAMEOUT=FORMULA Option';
title2 'Using WILDCARD="?YIELD?"';
proc print
   data=crout noobs;
run;
```

Output 47.6.2 shows the results.

Output 47.6.2 Listing of OUT=CROUT from the FAMEOUT=FORMULA Option of the Fame TRAINING Data

Formulas from the TRAINING DB, FAMEOUT=FORMULA Option Using WILDCARD="?YIELD?"

S.GM.YIELD.A	S.GM__PP.YIELD.A
(%SPLC2TF(C37044210X01, IAD_DATE.H, IAD.H)/C37044210X01.CLOSE)*C37044210X01.ADJUST	(%SPLC2TF(C37044210X01, IAD_DATE.H, IAD.H)/C37044210X01.CLOSE)*C37044210X01.ADJUST

S.HWP.YIELD.A	S.IBM.YIELD.A
(%SPLC2TF(C42823610X01, IAD_DATE.H, IAD.H)/C42823610X01.CLOSE)*C42823610X01.ADJUST	(%SPLC2TF(C45920010X01, IAD_DATE.H, IAD.H)/C45920010X01.CLOSE)*C45920010X01.ADJUST

S.INDUT.YIELD.A	S.SPAL.YIELD.A
(%SPLC2TF(C00000110X00, IAD_DATE.H, IAD.H)/C00000110X00.CLOSE)*C00000110X00.ADJUST	(%SPLC2TF(C00000117X00, IAD_DATE.H, IAD.H)/C00000117X00.CLOSE)*C00000117X00.ADJUST

S.SPALN.YIELD.A	S.SUNW.YIELD.A
(%SPLC2TF(C00000117X00, IAD_DATE.H, IAD.H)/C00000117X00.CLOSE)*C00000117X00.ADJUST	(%SPLC2TF(C86681010X60, IAD_DATE.H, IAD.H)/C86681010X60.CLOSE)*C86681010X60.ADJUST

S.XOM.YIELD.A	S.XON.YIELD.A
(%SPLC2TF(C30231G10X01, IAD_DATE.H, IAD.H)/C30231G10X01.CLOSE)*C30231G10X01.ADJUST	(%SPLC2TF(C30231G10X01, IAD_DATE.H, IAD.H)/C30231G10X01.CLOSE)*C30231G10X01.ADJUST

Additional examples of the FAMEOUT= option are shown in Example 47.11, Example 47.12, Example 47.13, Example 47.14, and Example 47.15.

Example 47.7: Remote Fame Access by Using Fame CHLI

When you run Fame in a client/server environment and also have Fame CHLI capability to enable access to the server, you can access Fame remote data. Access the remote data by specifying the port number of the TCP/IP service that is defined for the frdb_m and the node name of the Fame master server in the physical path. In this example, the Fame server node name is STONES, and the port number is 5555, as was designated in the Fame master command. For more information about starting your Fame master server, see the section "Starting the Master Server" in the *Guide to Fame Database Servers*.

```
options ls=78;
title1 "DRIECON Database, Using FAME with Remote Access Via CHLI";
options validvarname=any;
libname test1 sasefame '#5555@stones $FAME/util';

data a;
   set test1.driecon;
   keep YP ZA ZB;
   where date between '01jan98'd and '31dec03'd;
run;

proc means data=a n;
run;
```

Output 47.7.1 shows the results.

Output 47.7.1 Summary Statistics for the Remote FAME Data

DRIECON Database, Using FAME with Remote Access Via CHLI

The MEANS Procedure

Variable	Label	N
YP	PERSONAL INCOME	5
ZA	CORPORATE PROFITS AFTER TAX EXCLUDING IVA	4
ZB	CORPORATE PROFITS BEFORE TAX EXCLUDING IVA	4

Example 47.8: Selecting Time Series by Using the CROSSLIST= Option and KEEP Statement

This example shows how to use two Fame namelists to perform selection. Note that *fame_namelist1* could be easily generated using the Fame WILDLIST function. For more about the WILDLIST function, see the section "The WILDLIST Function" in the *Fame Command Reference, Volume 2, Functions*. In the following statements, four tickers are selected in *fame_namelist1*, but when you use the KEEP statement, the resulting data set contains only the desired IBM ticker:

```
libname lib8 sasefame "%sysget(FAME_DATA)"
        convert=(frequency=business technique=constant)
        crosslist=(
           { IBM,SPALN,SUNW,XOM },
           { adjust, close, high, low, open, volume,
             uclose, uhigh, ulow,uopen,uvolume }
                 );

data trout;
   /* eleven companies, keep only the IBM ticker this time */
   set lib8.training;
   where date between '01mar02'd and '20mar02'd;
   keep IBM: ;
run;

title1 'TRAINING DB, Pricing Timeseries for IBM Ticker in CROSSLIST=';
proc contents
   data=trout;
run;

proc print
   data=trout;
run;
```

Output 47.8.1 and Output 47.8.2 show the results.

Output 47.8.1 Contents of the IBM Time Series in the Fame TRAINING Data

TRAINING DB, Pricing Timeseries for IBM Ticker in CROSSLIST=

The CONTENTS Procedure

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
1	IBM.ADJUST	Num	8
2	IBM.CLOSE	Num	8
3	IBM.HIGH	Num	8
4	IBM.LOW	Num	8
5	IBM.OPEN	Num	8
6	IBM.UCLOSE	Num	8
7	IBM.UHIGH	Num	8
8	IBM.ULOW	Num	8
9	IBM.UOPEN	Num	8
10	IBM.UVOLUME	Num	8
11	IBM.VOLUME	Num	8

Output 47.8.2 Listing of Ticker IBM Time Series in the Fame TRAINING Data

TRAINING DB, Pricing Timeseries for IBM Ticker in CROSSLIST=

Obs	IBM.ADJUST	IBM.CLOSE	IBM.HIGH	IBM.LOW	IBM.OPEN	IBM.UCLOSE	IBM.UHIGH
1	1	103.020	103.100	98.500	98.600	103.020	103.100
2	1	105.900	106.540	103.130	103.350	105.900	106.540
3	1	105.670	106.500	104.160	104.250	105.670	106.500
4	1	106.300	107.090	104.750	105.150	106.300	107.090
5	1	103.710	107.500	103.240	107.300	103.710	107.500
6	1	105.090	107.340	104.820	104.820	105.090	107.340
7	1	105.240	105.970	103.600	104.350	105.240	105.970
8	1	108.500	108.850	105.510	105.520	108.500	108.850
9	1	107.180	108.650	106.700	108.300	107.180	108.650
10	1	106.600	107.950	106.590	107.020	106.600	107.950
11	1	106.790	107.450	105.590	106.550	106.790	107.450
12	1	106.350	108.640	106.230	107.100	106.350	108.640
13	1	107.490	108.050	106.490	106.850	107.490	108.050
14	1	105.500	106.900	105.490	106.900	105.500	106.900

Obs	IBM.ULOW	IBM.UOPEN	IBM.UVOLUME	IBM.VOLUME
1	98.500	98.600	104890	104890
2	103.130	103.350	107650	107650
3	104.160	104.250	75617	75617
4	104.750	105.150	76874	76874
5	103.240	107.300	109720	109720
6	104.820	104.820	107260	107260
7	103.600	104.350	86391	86391
8	105.510	105.520	110640	110640
9	106.700	108.300	64086	64086
10	106.590	107.020	53335	53335
11	105.590	106.550	108640	108640
12	106.230	107.100	53048	53048
13	106.490	106.850	46148	46148
14	105.490	106.900	48367	48367

Example 47.9: Selecting Time Series by Using the CROSSLIST= Option and Fame Namelist

This example demonstrates selection by using the CROSSLIST= option. Only the ticker "IBM" is specified in the KEEP statement from the 11 companies in the Fame ticker namelist.

```
libname lib9 sasefame "%sysget(FAME_DATA)"
        convert=(frequency=business technique=constant)
        range='07jul1997'd - '25jul1997'd
        crosslist=( nl(ticker),
                    { adjust, close, high, low, open, volume,
                      uclose, uhigh, ulow, uopen, uvolume }
                  );

data crout;
   /* eleven companies in the FAME ticker namelist */
   set lib9.training;
   keep IBM: ;
run;

title1 'TRAINING DB, Pricing Timeseries for Eleven Tickers in CROSSLIST=';
title2 'Using TICKER namelist.';
proc print data=crout;
run;

proc contents data=crout;
run;
```

Output 47.9.1 and Output 47.9.2 show the results.

Output 47.9.1 Listing of OUT=CROUT Using CROSSLIST= Option in the Fame TRAINING Data

TRAINING DB, Pricing Timeseries for Eleven Tickers in CROSSLIST= Using TICKER namelist.

Obs	IBM.ADJUST	IBM.CLOSE	IBM.HIGH	IBM.LOW	IBM.OPEN	IBM.UCLOSE	IBM.UHIGH
1	0.5	47.2500	47.7500	47.0000	47.5000	94.500	95.500
2	0.5	47.8750	47.8750	47.2500	47.2500	95.750	95.750
3	0.5	48.0938	48.3438	47.6563	48.0000	96.188	96.688
4	0.5	47.8750	48.0938	47.0313	47.3438	95.750	96.188
5	0.5	47.8750	48.6875	47.8125	47.9063	95.750	97.375
6	0.5	47.6250	48.2188	47.0000	47.8125	95.250	96.438
7	0.5	48.0000	48.1250	46.6875	47.4375	96.000	96.250
8	0.5	48.8125	49.0000	47.6875	47.8750	97.625	98.000
9	0.5	49.8125	50.8750	48.5625	48.9063	99.625	101.750
10	0.5	52.2500	52.6250	50.0000	50.0000	104.500	105.250
11	0.5	51.8750	53.1563	51.0938	52.6250	103.750	106.313
12	0.5	51.5000	51.7500	49.6875	50.0313	103.000	103.500
13	0.5	52.5625	53.5000	51.5938	52.1875	105.125	107.000
14	0.5	53.9063	54.2188	52.2500	52.8125	107.813	108.438
15	0.5	53.5000	54.2188	52.8125	53.9688	107.000	108.438

Obs	IBM.ULOW	IBM.UOPEN	IBM.UVOLUME	IBM.VOLUME
1	94.000	95.000	129012	64506
2	94.500	94.500	102796	51398
3	95.313	96.000	177276	88638
4	94.063	94.688	127900	63950
5	95.625	95.813	137724	68862
6	94.000	95.625	128976	64488
7	93.375	94.875	149612	74806
8	95.375	95.750	215440	107720
9	97.125	97.813	315504	157752
10	100.000	100.000	463480	231740
11	102.188	105.250	328184	164092
12	99.375	100.063	368276	184138
13	103.188	104.375	219880	109940
14	104.500	105.625	204088	102044
15	105.625	107.938	146600	73300

Output 47.9.2 Contents of OUT=CROUT Using CROSSLIST= Option in the Fame TRAINING Data

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
1	IBM.ADJUST	Num	8
2	IBM.CLOSE	Num	8
3	IBM.HIGH	Num	8
4	IBM.LOW	Num	8
5	IBM.OPEN	Num	8
6	IBM.UCLOSE	Num	8
7	IBM.UHIGH	Num	8
8	IBM.ULOW	Num	8
9	IBM.UOPEN	Num	8
10	IBM.UVOLUME	Num	8
11	IBM.VOLUME	Num	8

Example 47.10: Selecting Time Series by Using the CROSSLIST= Option and WHERE=TICK

Instead of having a Fame namelist with the ticker symbols for companies whose data you are interested in, you can designate an input SAS data set (INSETA) that specifies the tickers to select. Specify your selection by using the WHERE clause in the INSET= option as follows:

```
data inseta;
   length tick $5;
   /* need $5 so SPALN is not truncated */

   tick='AOL';   output;
   tick='C';     output;
   tick='CPQ';   output;
   tick='CVX';   output;
   tick='F';     output;
   tick='GM';    output;
   tick='HWP';   output;
   tick='IBM';   output;
   tick='SPALN'; output;
   tick='SUNW';  output;
   tick='XOM';   output;
run;

libname lib10 sasefame "%sysget(FAME_DATA)"
        convert=(frequency=business technique=constant)
        range='07jul1997'd - '25jul1997'd
        inset=( inseta where=tick )
        crosslist=
            ( {adjust, close, high, low, open, volume,
               uclose, uhigh, ulow,uopen,uvolume} );

data trout;
   /* eleven companies with unique TICKs specified in INSETA */
   set lib10.training;
   keep IBM: ;
run;

title1 'TRAINING DB, Pricing Timeseries for Eleven Tickers in CROSSLIST=';
title2 'Using INSET with WHERE=TICK.';
proc print data=trout;
run;

proc contents data=trout;
run;
```

Output 47.10.1 and Output 47.10.2 show the results.

Output 47.10.1 Listing of OUT=TROUT Using CROSSLIST= and INSET= Options in the Fame TRAINING Data

TRAINING DB, Pricing Timeseries for Eleven Tickers in CROSSLIST= Using INSET with WHERE=TICK.

Obs	IBM.ADJUST	IBM.CLOSE	IBM.HIGH	IBM.LOW	IBM.OPEN	IBM.UCLOSE	IBM.UHIGH
1	0.5	47.2500	47.7500	47.0000	47.5000	94.500	95.500
2	0.5	47.8750	47.8750	47.2500	47.2500	95.750	95.750
3	0.5	48.0938	48.3438	47.6563	48.0000	96.188	96.688
4	0.5	47.8750	48.0938	47.0313	47.3438	95.750	96.188
5	0.5	47.8750	48.6875	47.8125	47.9063	95.750	97.375
6	0.5	47.6250	48.2188	47.0000	47.8125	95.250	96.438
7	0.5	48.0000	48.1250	46.6875	47.4375	96.000	96.250
8	0.5	48.8125	49.0000	47.6875	47.8750	97.625	98.000
9	0.5	49.8125	50.8750	48.5625	48.9063	99.625	101.750
10	0.5	52.2500	52.6250	50.0000	50.0000	104.500	105.250
11	0.5	51.8750	53.1563	51.0938	52.6250	103.750	106.313
12	0.5	51.5000	51.7500	49.6875	50.0313	103.000	103.500
13	0.5	52.5625	53.5000	51.5938	52.1875	105.125	107.000
14	0.5	53.9063	54.2188	52.2500	52.8125	107.813	108.438
15	0.5	53.5000	54.2188	52.8125	53.9688	107.000	108.438

Obs	IBM.ULOW	IBM.UOPEN	IBM.UVOLUME	IBM.VOLUME
1	94.000	95.000	129012	64506
2	94.500	94.500	102796	51398
3	95.313	96.000	177276	88638
4	94.063	94.688	127900	63950
5	95.625	95.813	137724	68862
6	94.000	95.625	128976	64488
7	93.375	94.875	149612	74806
8	95.375	95.750	215440	107720
9	97.125	97.813	315504	157752
10	100.000	100.000	463480	231740
11	102.188	105.250	328184	164092
12	99.375	100.063	368276	184138
13	103.188	104.375	219880	109940
14	104.500	105.625	204088	102044
15	105.625	107.938	146600	73300

Output 47.10.2 Contents of OUT=TROUT Using CROSSLIST= and INSET= Options in the Fame TRAINING Data

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
1	IBM.ADJUST	Num	8
2	IBM.CLOSE	Num	8
3	IBM.HIGH	Num	8
4	IBM.LOW	Num	8
5	IBM.OPEN	Num	8
6	IBM.UCLOSE	Num	8
7	IBM.UHIGH	Num	8
8	IBM.ULOW	Num	8
9	IBM.UOPEN	Num	8
10	IBM.UVOLUME	Num	8
11	IBM.VOLUME	Num	8

Example 47.11: Selecting Boolean Case Series with the FAMEOUT= Option

This example shows how to extract all Boolean case series from the ALLTYPES Fame database. The following statements write all Boolean case series to the SAS data set BOOOUT:

```
title1 '***famallt: FAMEOUT Option, Different Type Values***';
options validvarname=any;

%let FAME=%sysget(FAME);
%put(&FAME);
%let FAMETEMP=%sysget(FAME_TEMP);
%put(&FAMETEMP);

libname lib4 sasefame "%sysget(FAME_DATA)"
   fameout=boolcase wildcard="?" ;

data booout;
   set lib4.alltypes;
run;

title1 'ALLTYPES FAMEOUT=BOOLCASE for Boolean Case Series';
title2 'Using FAMEOUT=CASE BOOLEAN Option without Range';
proc contents
   data=booout;
run;

proc print
   data=booout;
run;
```

Output 47.11.1 and Output 47.11.2 show the results for the Boolean case.

Output 47.11.1 Contents of OUT=BOOOUT Using FAMEOUT=BOOLCASE for Boolean Case Series

ALLTYPES FAMEOUT=BOOLCASE for Boolean Case Series
Using FAMEOUT=CASE BOOLEAN Option without Range

The CONTENTS Procedure

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
1	BOO0	Num	8
2	BOO1	Num	8
3	BOO2	Num	8
4	BOOM	Num	8
5	BOO_RES	Num	8

Output 47.11.2 Listing of OUT=BOOOUT Using FAMEOUT=BOOLCASE for Boolean Case Series

ALLTYPES FAMEOUT=BOOLCASE for Boolean Case Series
Using FAMEOUT=CASE BOOLEAN Option without Range

Obs	BOO0	BOO1	BOO2	BOOM	BOO_RES
1	0	1	0	1	.
2	0	0	1	0	.
3	0	0	0	251	.
4	0	1	1	1	.
5	0	1	0	1	.
6	0	0	.	0	.
7	0	0	.	0	.
8	0	1	.	1	.
9	0	.	0	.	.
10	0	.	.	.	.
11	1	.	.	.	.
12	1	.	.	.	.
13	1	.	1	.	.
14	1	.	.	.	.
15	1	.	.	.	.
16	1	.	.	.	.
17	1	.	0	.	.
18	1	.	.	.	.
19	1	.	.	.	.
20	1	.	.	.	.

Example 47.12: Selecting Numeric Case Series with the FAMEOUT= Option

This example extracts numeric case series. In addition to the already existing numeric case series in the Fame database, you can also have formulas that expand to numeric case series. The SASEFAME engine resolves all formulas that belong to the class and type of series data object that you specify in the FAMEOUT= option. The following statements write all numeric case series to the SAS data set CSOUT:

```
libname lib5 sasefame "%sysget(FAME_DATA)"
   fameout=case wildcard="?" ;

data csout;
   set lib5.alltypes;
run;

title1 'Using FAMEOUT=CASE Option without Range';
title2 'ALLTYPES, FAMEOUT=CASE and Open Wildcard for Numeric Case Series';
proc contents
   data=csout;
run;

proc print
   data=csout;
run;
```

Output 47.12.1 and Output 47.12.2 show the results.

Output 47.12.1 Contents of OUT=CSOUT Using FAMEOUT=CASE and Open Wildcard for Numeric Case Series

Using FAMEOUT=CASE Option without Range
ALLTYPES, FAMEOUT=CASE and Open Wildcard for Numeric Case Series

The CONTENTS Procedure

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
1	FRM1	Num	8
2	NUM0	Num	8
3	NUM1	Num	8
4	NUM2	Num	8
5	NUMM	Num	8
6	NUM_RES	Num	8
7	PRC0	Num	8
8	PRC1	Num	8
9	PRC2	Num	8
10	PRCM	Num	8
11	PRC_RES	Num	8

Output 47.12.2 Listing of OUT=CSOUT Using FAMEOUT=CASE and Open Wildcard for Numeric Case Series

Using FAMEOUT=CASE Option without Range
ALLTYPES, FAMEOUT=CASE and Open Wildcard for Numeric Case Series

Obs	FRM1	NUM0	NUM1	NUM2	NUMM	NUM_RES	PRC0	PRC1	PRC2	PRCM	PRC_RES
1	0.00000	-9	0	1.33333	0	.	-18	0	1.33333	0	.
2	1.00000	-8	1	1.00000	1	.	-16	1	1.00000	1	.
3	0.66667	-7	2	0.66667	1.7014E38	.	-14	2	0.66667	1.7014E38	.
4	3.00000	-6	3	0.33333	3	.	-12	3	0.33333	3	.
5	4.00000	-5	4	0.00000	4	.	-10	4	0.00000	4	.
6	.	-4	5	.	5	.	-8	5	.	5	.
7	.	-3	6	.	6	.	-6	6	.	6	.
8	7.00000	-2	7	.	7	.	-4	7	.	7	.
9	.	-1	.	-1.33333	.	.	-2	.	-1.33333	.	.
10	.	0	.	.	.	.	0	.	.	.	.
11	.	1	.	.	.	.	2	.	.	.	.
12	.	2	.	.	.	.	4	.	.	.	.
13	.	3	.	-2.66667	.	.	6	.	-2.66667	.	.
14	.	4	.	.	.	.	8	.	.	.	.
15	.	5	.	.	.	.	10	.	.	.	.
16	.	6	.	.	.	.	12	.	.	.	.
17	.	7	.	-4.00000	.	.	14	.	-4.00000	.	.
18	.	8	.	.	.	.	16	.	.	.	.
19	.	9	.	.	.	.	18	.	.	.	.
20	.	10	.	.	.	.	20	.	.	.	.

Example 47.13: Selecting Date Case Series with the FAMEOUT= Option

This example shows how to extract date case series. In addition to the existing date case series in the Fame database, you can have formulas that resolve to date case series. The SASEFAME engine resolves all formulas that belong to the class and type of series data object that you specify in the FAMEOUT= option. The following statements write all date case series to the SAS data set CDOUT:

```
libname lib6 sasefame "%sysget(FAME_DATA)"
   fameout=datecase wildcard="?" ;

data cdout;
   set lib6.alltypes;
run;

title1 'Using FAMEOUT=DATECASE Option without Range';
title2 'ALLTYPES: FAMEOUT=DATECASE and Open Wildcard for Date Case Series';
proc contents
   data=cdout;
run;

proc print
   data=cdout;
run;
```

Output 47.13.1 and Output 47.13.2 show the results.

Output 47.13.1 Contents of OUT=CDOUT Using FAMEOUT=DATECASE

Using FAMEOUT=DATECASE Option without Range
ALLTYPES: FAMEOUT=DATECASE and Open Wildcard for Date Case Series

The CONTENTS Procedure

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Informat
1	DAT0	Num	8	YEAR4.	4.
2	DAT1	Num	8	YEAR4.	4.
3	DAT2	Num	8	YEAR4.	4.
4	DATM	Num	8	YEAR4.	4.
5	FRM2	Num	8	YEAR4.	4.

Output 47.13.2 Listing of OUT=CDOUT Using FAMEOUT=DATECASE

Using FAMEOUT=DATECASE Option without Range
ALLTYPES: FAMEOUT=DATECASE and Open Wildcard for Date Case Series

Obs	DAT0	DAT1	DAT2	DATM	FRM2
1	1991	1981	1987	1981	1987
2	1992	1982	1986	1982	1986
3	1993	1983	1985	1983	1985
4	1994	1984	1984	1984	1984
5	1995	1985	1983	1985	1983
6	1996	1986	.	1986	.
7	1997	1987	.	1987	.
8	1998	1988	.	1988	.
9	1999	.	1979	.	1979
10	2000	.	.	.	.
11	2001	.	.	.	.
12	2002	.	.	.	.
13	2003	.	1975	.	.
14	2004	.	.	.	.
15	2005	.	.	.	.
16	2006	.	.	.	.
17	2007	.	1971	.	.
18	2008	.	.	.	.
19	2009	.	.	.	.
20	2010	.	.	.	.

Example 47.14: Selecting String Case Series with the FAMEOUT= Option

This example shows how to extract string case series. In addition to the existing string case series in your Fame database, you can have formulas that resolve to string case series. The SASEFAME engine resolves all formulas that belong to the class and type of series data object that you specify in the FAMEOUT= option. The following statements write all string case series to the SAS data set CSTROUT:

```
libname lib7 sasefame "%sysget(FAME_DATA)"
   fameout=stringcase wildcard="?" ;

data cstrout;
   set lib7.alltypes;
run;

title1 'Using FAMEOUT=STRINGCASE Option without Range';
title2 'ALLTYPES, FAMEOUT=STRINGCASE and Open Wildcard for String Case Series';
proc contents
   data=cstrout;
run;

proc print
   data=cstrout;
run;
```

Output 47.14.1 and Output 47.14.2 show the results.

Output 47.14.1 Contents of OUT=CSTROUT Using FAMEOUT=STRINGCASE and Open Wildcard for String Case Series

Using FAMEOUT=STRINGCASE Option without Range
ALLTYPES, FAMEOUT=STRINGCASE and Open Wildcard for String Case Series

The CONTENTS Procedure

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
1	STR0	Char	16
2	STR1	Char	16
3	STR2	Char	16
4	STRM	Char	16

Output 47.14.2 Listing of OUT=CSTROUT Using FAMEOUT=STRINGCASE and Open Wildcard for String Case Series

Using FAMEOUT=STRINGCASE Option without Range
ALLTYPES, FAMEOUT=STRINGCASE and Open Wildcard for String Case Series

Obs	STR0	STR1	STR2	STRM
1	-9	0	1.333333	0
2	-8	1	1	1
3	-7	2	0.6666667	2
4	-6	3	0.3333333	3
5	-5	4	0	4
6	-4	5		5
7	-3	6		
8	-2	7		7
9	-1		-1.333333	
10	0			
11	1			
12	2			
13	3		-2.666667	
14	4			
15	5			
16	6			
17	7		-4	
18	8			
19	9			
20	10			

Example 47.15: Extracting Source for Formulas

This example shows how to extract the source for all the formulas in the Fame database by using the FAMEOUT=FORMULA and WILDCARD="?" options. The following statements show the source of all formulas written to the SAS data set CFOROUT. Another example of the FAMEOUT=FORMULA option is shown in Example 47.6.

```
libname lib8 sasefame "%sysget(FAME_DATA)"
   fameout=formula wildcard="?" ;

data cforout;
   set lib8.alltypes;
run;

title1 'Using FAMEOUT=FORMULA Option without Range';
proc contents
   data=cforout;
run;
```

Output 47.15.1 and Output 47.15.2 show the results.

Output 47.15.1 Contents of OUT=CFOROUT Using FAMEOUT=FORMULA and Open Wildcard

Using FAMEOUT=FORMULA Option without Range

The CONTENTS Procedure

Alphabetic List of Variables and Attributes			
#	Variable	Type	Len
1	S.DFRM	Char	27
2	S.FRM1	Char	27
3	S.FRM2	Char	27

```
title3 'ALLTYPES, FAMEOUT=FORMULA, and Open Wildcard for FORMULA Series';
proc print
   data=cforout noobs;
run;
```

Output 47.15.2 Listing of OUT=CFOROUT Using FAMEOUT=FORMULA and Open Wildcard

Using FAMEOUT=FORMULA Option without Range

ALLTYPES, FAMEOUT=FORMULA, and Open Wildcard for FORMULA Series

S.DFRM	S.FRM1	S.FRM2
IF DBOO THEN DPRC ELSE DNUM	IF BOO1 THEN NUM1 ELSE NUM2	IF BOO0 THEN DAT1 ELSE DAT2

If you want all series of every type, you can merge the resulting data sets. For more information about merging SAS data sets, see *SAS Language Reference: Concepts*.

Example 47.16: Reading Time Series by Defining Fame Expression Groups in the INSET= Option with the KEEP= Clause

To keep all the numeric time series that are listed in the expressions given in the input data set, INSETA, use the INSET=(*setname* KEEPLIST=*fame_expression_group*) and WILDCARD="?" options. The following statements show how to select time series that are specified in a KEEP expression group and are written to the SAS output data set:

```
data inseta; /* Use this for d8690 training database */
   length express $52;
   express='cvx.close;'; output;
   express='{ibm.high,ibm.low,ibm.close};'; output;
   express='mave(ibm.close,30);'; output;
   express='crosslist({gm,f,c},{volume});'; output;
   express='cvx.close+ibm.close;'; output;
   express='ibm.close;'; output;
   express='sum(pep.volume);'; output;
   express='mave(pep.close,20);'; output;
run;
```

```
title1 'TRAINING DB, Pricing Timeseries for Expressions in INSET=';
proc print
   data=inseta;
run;
```

Output 47.16.1 shows the expressions that are stored as observations in the input data set, INSETA.

Output 47.16.1 Listing of INSETA Defining Fame Expression Group

TRAINING DB, Pricing Timeseries for Expressions in INSET=

Obs	express
1	cvx.close;
2	{ibm.high,ibm.low,ibm.close};
3	mave(ibm.close,30);
4	crosslist({gm,f,c},{volume});
5	cvx.close+ibm.close;
6	ibm.close;
7	sum(pep.volume);
8	mave(pep.close,20);

The following statements show how to use the INSET= option to keep all time series that are represented in the input data set, INSETA, as the group variable EXPRESS:

```
libname libX sasefame "%sysget(FAME_DATA)"
   wildcard="?"
   convert=(frequency=business technique=constant)
   range='23jul1997'd - '25jul1997'd
   inset=( inseta KEEP=express)
   ;


data trout;
   set libX.trainten;
run;

title1 'TRAINING DB, Pricing Timeseries for Expressions in INSET=';
proc print data=trout;
run;

proc contents data=trout;
run;
```

Output 47.16.2 and Output 47.16.3 show the results.

Output 47.16.2 Listing of TROUT Using INSETA with KEEP=EXPRESS

TRAINING DB, Pricing Timeseries for Expressions in INSET=

Obs	DATE	C.VOLUME	VOLUME	GM.VOLUME	IBM.CLOSE	IBM.HIGH	IBM.LOW	SASTEMP1
1	23JUL1997	33791.88	45864.05	37392	52.5625	53.5000	51.5938	76.8125
2	24JUL1997	41828.85	29651.34	27771	53.9063	54.2188	52.2500	76.8750
3	25JUL1997	46979.83	36716.77	24969	53.5000	54.2188	52.8125	78.0000

Obs	SASTEMP3	SASTEMP5	SASTEMP6	SASTEMP8
1	47.0894	129.375	52.5625	37.6118
2	47.4289	130.781	53.9063	37.6250
3	47.7392	131.500	53.5000	37.6546

Output 47.16.3 Listing of Contents of TROUT

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Informat	Label
2	C.VOLUME	Num	8			
1	DATE	Num	8	DATE9.	9.	Date of Observation
4	GM.VOLUME	Num	8			
5	IBM.CLOSE	Num	8			
6	IBM.HIGH	Num	8			
7	IBM.LOW	Num	8			
8	SASTEMP1	Num	8			
9	SASTEMP3	Num	8			
10	SASTEMP5	Num	8			
11	SASTEMP6	Num	8			
12	SASTEMP8	Num	8			
3	VOLUME	Num	8			

Example 47.17: Optimizing Cache Sizes with the TUNEFAME= and TUNECHLI= Options

This example shows how to use the TUNEFAME= option, the TUNECHLI= option, and a RANGE= option to select pricing time series in the TRAINTEN database. The selected time series are written to the SAS output data set. The Fame database engine's virtual memory is given in megabytes (MB), so this example sets the cache size to 100 MB. The Fame CHLI engine's virtual memory is also given in megabytes (MB), so this example sets the CHLI cache size to 100 MB. These two settings correspond to the default settings. Both the Fame 4GL engine and the Fame CHLI engine can use a cache size that ranges from 0.1 MB to 17,592,186,000,000 MB.

```
libname lib5 sasefame "%sysget(FAME_DATA)"
  wildcard="?UHIGH"
  tunefame=nodes 100
  tunechli=nodes 100
  convert=(frequency=business technique=constant)
  range='23jul1997'd - '25jul1997'd
  ;

data trout(drop=C:);
   set lib5.trainten;
run;
title1 'TRAINTEN DB, Pricing Timeseries, TUNEFAME=NODES and TUNECHLI=NODES Options';
proc print data=trout;
run;

proc contents data=trout;
run;
```

Output 47.17.1 and Output 47.17.2 show the results.

Output 47.17.1 Listing of TRAINING DB, Pricing Timeseries, TUNEFAME=NODES, and TUNECHLI=NODES Options

TRAINTEN DB, Pricing Timeseries, TUNEFAME=NODES and TUNECHLI=NODES Options

Obs	DATE	DJ30IN.UHIGH	DJ__30.UHIGH	F.UHIGH	F___I.UHIGH	GM.UHIGH	GM__PP.UHIGH
1	23JUL1997	8199.15	8199.15	41.0625	41.0625	59.1250	59.1250
2	24JUL1997	8174.53	8174.53	42.0000	42.0000	59.2500	59.2500
3	25JUL1997	8200.31	8200.31	41.5000	41.5000	57.8125	57.8125

Obs	HPQ.UHIGH	HWP.UHIGH	IBM.UHIGH	INDUT.UHIGH	INTC.UHIGH	JAVA.UHIGH	JAVAD.UHIGH
1	67.3125	67.3125	107.000	8199.15	90.750	46.9375	46.9375
2	65.8750	65.8750	108.438	8174.53	90.625	46.8750	46.8750
3	66.1250	66.1250	108.438	8200.31	91.125	47.3750	47.3750

Obs	KO.UHIGH	PEP.UHIGH	SPAL.UHIGH	SPALN.UHIGH	SPALNS.UHIGH	SPX.UHIGH	SP_CI.UHIGH
1	70.7500	38.4375	941.800	941.800	941.800	941.800	941.800
2	70.4375	38.0625	941.510	941.510	941.510	941.510	941.510
3	70.9375	38.7500	945.650	945.650	945.650	945.650	945.650

Obs	SP__50.UHIGH	SP___C.UHIGH	SUNW.UHIGH	XOM.UHIGH	XON.UHIGH
1	941.800	941.800	46.9375	63.125	63.125
2	941.510	941.510	46.8750	62.000	62.000
3	945.650	945.650	47.3750	63.000	63.000

Output 47.17.2 Listing of Contents of TROUT for TUNEFAME=NODES and TUNECHLI=NODES Options

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Informat	Label
1	DATE	Num	8	DATE9.	9.	Date of Observation
2	DJ30IN.UHIGH	Num	8			
3	DJ__30.UHIGH	Num	8			
4	F.UHIGH	Num	8			
5	F___I.UHIGH	Num	8			
6	GM.UHIGH	Num	8			
7	GM__PP.UHIGH	Num	8			
8	HPQ.UHIGH	Num	8			
9	HWP.UHIGH	Num	8			
10	IBM.UHIGH	Num	8			
11	INDUT.UHIGH	Num	8			
12	INTC.UHIGH	Num	8			
13	JAVA.UHIGH	Num	8			
14	JAVAD.UHIGH	Num	8			
15	KO.UHIGH	Num	8			
16	PEP.UHIGH	Num	8			
17	SPAL.UHIGH	Num	8			
18	SPALN.UHIGH	Num	8			
19	SPALNS.UHIGH	Num	8			
20	SPX.UHIGH	Num	8			
21	SP_CI.UHIGH	Num	8			
22	SP__50.UHIGH	Num	8			
23	SP___C.UHIGH	Num	8			
24	SUNW.UHIGH	Num	8			
25	XOM.UHIGH	Num	8			
26	XON.UHIGH	Num	8			

For more information about tuning the use of virtual memory, read about TUNE CACHE nodes in the section "TUNE CACHE Option" in the online document *Fame 10 Online Help*.

References

DRI/McGraw-Hill (1997). *DataLink*. Lexington, MA: DRI/McGraw-Hill.

DRI/McGraw-Hill Data Search and Retrieval for Windows (1996). *DRIPRO User's Guide*. Lexington, MA: DRI/McGraw-Hill.

IHS Global Insight (2009). "Global Economic Data." Available at http://www.ihs.com/products/global-insight/industry-analysis/financial/global-economic-data.aspx.

Organisation for Economic Co-operation and Development (1992a). *Annual National Accounts, Vol. 1: Main Aggregates Content Documentation for Magnetic Tape Subscription*. Paris: OECD.

Organisation for Economic Co-operation and Development (1992b). *Annual National Accounts, Vol. 2: Detailed Tables Technical Documentation for Magnetic Tape Subscription*. Paris: OECD.

Organisation for Economic Co-operation and Development (1992c). *Main Economic Indicators Database Note*. Paris: OECD.

Organisation for Economic Co-operation and Development (1992d). *Main Economic Indicators Inventory*. Paris: OECD.

Organisation for Economic Co-operation and Development (1992e). *Main Economic Indicators OECD Statistics on Magnetic Tape Document*. Paris: OECD.

Organisation for Economic Co-operation and Development (1992f). *OECD Statistical Information Research and Inquiry System Magnetic Tape Format Documentation*. Paris: OECD.

Organisation for Economic Co-operation and Development (1992g). *Quarterly National Accounts Inventory of Series Codes*. Paris: OECD.

Organisation for Economic Co-operation and Development (1992h). *Quarterly National Accounts Technical Documentation*. Paris: OECD.

SunGard Solutions for Data Management (2009a). *FAME 10 Online Help*. Ann Arbor, MI: SunGard Solutions for Data Management. http://www.fame.sungard.com/support.html.

SunGard Solutions for Data Management (2009b). *FAME Command Reference for Release 9 and Earlier*. Ann Arbor, MI: SunGard Solutions for Data Management. http://www.fame.sungard.com/support.html.

SunGard Solutions for Data Management (2009c). *FAME Functions for FAME Release 9 and Earlier*. Ann Arbor, MI: SunGard Solutions for Data Management. http://www.fame.sungard.com/support.html.

SunGard Solutions for Data Management (2009d). *Guide to FAME Database Servers*. New York: SunGard Solutions for Data Management. http://www.fame.sungard.com/support.html.

SunGard Solutions for Data Management (2009e). *Reference Guide to Seamless C HL*. Ann Arbor, MI: SunGard Solutions for Data Management. http://www.fame.sungard.com/support.html.

SunGard Solutions for Data Management (2009f). *User's Guide to FAME*. Ann Arbor, MI: SunGard Solutions for Data Management. http://www.fame.sungard.com/support.html.

Chapter 48
The SASEFRED Interface Engine

Contents

Overview: SASEFRED Interface Engine **3382**
Getting Started: SASEFRED Interface Engine **3382**
Syntax: SASEFRED Interface Engine **3384**
The LIBNAME *libref* SASEFRED Statement 3386
Details: SASEFRED Interface Engine **3391**
Available Sources That Provide FRED Time Series Data 3391
FRED API Key 3392
Available Releases for Each Source That Provides FRED Time Series Data 3392
Available Time Series for Each Release ID 3393
Available Native Frequency for Each Series ID 3393
Vintage Dates for Each Series ID 3393
SAS Output Data Set 3394
SAS OUTXML File 3394
SAS XML Map File 3394
XFREDTPU SAS Data Set 3394
Reading Price Data by Using Indices 3395
Examples: SASEFRED Interface Engine **3397**
Example 48.1: Retrieving Data for Multiple Time Series 3397
Example 48.2: Retrieving Data by Using the Vintage Date 3399
Example 48.3: Selecting Time Series When Native Frequency Is Less Than Requested Frequency 3400
Example 48.4: Selecting Time Series When Native Frequency Is Greater Than Requested Frequency 3402
Example 48.5: Specifying One Series ID with Multiple Vintage Dates for the OUTPUT=2 Option 3403
Example 48.6: Specifying Two Series IDs with Multiple Vintage Dates and Descending Sort Order 3405
Example 48.7: Vintage Dates for a Specific Series with the URL= Option 3407
Example 48.8: Series for a Specific Release with the URL= Option 3408
Example 48.9: Series for Specific Tags with the URL= Option 3412
Example 48.10: Categories for a Specific Series with the URL= Option 3414
Example 48.11: Categories for a Specific Source with the URL= Option 3415
Example 48.12: Series for a Specific Category with the URL= Option 3416
Example 48.13: Sources for Today's Date with the URL= Option 3427
Example 48.14: Releases Available for Today's Date with the URL= Option 3429
References **3430**

Overview: SASEFRED Interface Engine

The SASEFRED interface engine enables SAS users to retrieve economic data from the FRED website, which is hosted by the Economic Research Division of the Federal Reserve Bank of St. Louis. FRED stands for Federal Reserve Economic Data. The FRED databases contain more than 61,000 economic data time series from 48 national and international sources, both public and private. These time series are updated at annual, quarterly, monthly, weekly, and daily intervals. The economic time series on the FRED website contain observation or measurement periods that are associated with data values.

The SASEFRED interface engine uses the LIBNAME statement to enable you to specify how to subset your FRED data and how to aggregate the selected time series at the same update frequency. You can then use the SAS DATA step to perform further subsetting and to store the resulting time series in a SAS data set. You can perform more analysis (if desired) either in the same SAS session or in a later session.

The SASEFRED interface engine supports 64-bit Windows and Linux X64 (LAX) platforms.

Note that the SASEFRED engine uses the FRED API, but it is not endorsed or certified by the Federal Reserve Bank of St. Louis, and that by using the SASEFRED interface, you are agreeing to comply with the FRED terms of use, which are described on the web page at the following URL: **http://api.stlouisfed.org/terms_of_use.html**.

Getting Started: SASEFRED Interface Engine

You can query the Federal Reserve Economic Data (FRED) databases to retrieve the observations or data values for a list of economic time series by specifying the series ID of each time series that you want to read into SAS and by specifying your unique FRED API key. To obtain your own unique API key, visit the FRED website at the following URL:

http://api.stlouisfed.org/api_key.html

The FRED API key is a 32-character alphanumeric lowercase string, such as 'abcdefghijklmnopqrstuvwxyz123456', and is represented by 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX' in the APIKEY= option in the following example. In addition, the example URLs in this section and in the section "Details: SASEFRED Interface Engine" on page 3391 use the same FRED API key as the argument *your_fred_apikey*.

After you have your assigned FRED API key and you have agreed to the terms of use, before downloading any copyright-protected data series, be aware that you are solely responsible for obtaining copyright permissions for any copyright-protected time series that you download (other than for personal use). To obtain a list of the copyright-protected data series, visit the web page at the following URL:

http://api.stlouisfed.org/fred/series/search?search_text=copyright&api_key=your_fred_apikey

Now that your are informed about the terms of use of the FRED data, you can use your FRED API key to access the FRED data, as shown in the following example.

The following statements enable you to access the exports of goods and services time series data from January 1, 1960, to January 1, 2012, on an annual basis. The observations are sorted by the time ID variable DATE.

```
options validvarname=any;
title 'Retrieve Data for the Exports of Goods and Services';
libname _all_ clear;

libname fred sasefred "%sysget(FRED)"
   OUTXML=exportgs
   XMLMAP="%sysget(FRED)exportgs.map"
   APIKEY='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
   IDLIST='bopxgsa';

data export_gsa;
   set fred.exportgs ;
run;

proc contents data=export_gsa; run;
proc print data=export_gsa(obs=15); run;
```

Figure 48.1 Getting Started with Exports of Goods and Services: export_gsa(obs=15)

Retrieve Data for the Exports of Goods and Services

Obs	date	realtime_start	realtime_end	BOPXGSA
1	1960-01-01	2015-03-19	2015-03-19	25.940
2	1961-01-01	2015-03-19	2015-03-19	26.403
3	1962-01-01	2015-03-19	2015-03-19	27.722
4	1963-01-01	2015-03-19	2015-03-19	29.620
5	1964-01-01	2015-03-19	2015-03-19	33.341
6	1965-01-01	2015-03-19	2015-03-19	35.285
7	1966-01-01	2015-03-19	2015-03-19	38.926
8	1967-01-01	2015-03-19	2015-03-19	41.333
9	1968-01-01	2015-03-19	2015-03-19	45.543
10	1969-01-01	2015-03-19	2015-03-19	49.220
11	1970-01-01	2015-03-19	2015-03-19	56.640
12	1971-01-01	2015-03-19	2015-03-19	59.677
13	1972-01-01	2015-03-19	2015-03-19	67.222
14	1973-01-01	2015-03-19	2015-03-19	91.242
15	1974-01-01	2015-03-19	2015-03-19	120.897

The XML data that the FRED website returns are placed in a file named by the OUTXML= option, in this case, EXPORTGS.xml. Note that the XML file extension is excluded from the filename given in the OUTXML= option. This XML data file resides in the location given inside the string enclosed in double quotation marks in the SASEFRED LIBNAME statement. So, in the preceding example, if the FRED environment variable is set to:

```
C:\freddata\
```

then the downloaded XML file is located at

```
C:\freddata\EXPORTGS.xml
```

An equivalent LIBNAME statement that does not use any environment variables could be as follows:

```
Libname fred sasefred "C:\freddata\"
   OUTXML=exportgs
   XMLMAP="C:\freddata\exportgs.map"
   APIKEY='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
   IDLIST='bopxgsa';
```

You could also use either a SAS macro variable or a system environment variable to store the value of your FRED API key so that the key does not appear explicitly in your SAS code. The XML map that is created is assigned the full pathname specified by the XMLMAP= option. The IDLIST= option specifies the list of time series that you want to retrieve. This option accepts a string, enclosed in single quotation marks, that denotes a list of one or more time series that you select (keep) in the resulting SAS data set. The result, Export_gsa, is named in the DATA step and is shown in Figure 48.1. It is more efficient to use the DATA step to store your FRED data in a SAS data set and then refer to the SAS data set directly in your PROC PRINT or PROC GPLOT statement, but you can also refer to the SASEFRED libref directly, as in the following statement:

```
proc print data=fred.exportgs; run;
```

This statement uses the member name, exportgs, in the PROC PRINT statement; this usage corresponds to specifying the OUTXML=EXPORTGS option. Although using this statement might seem easier, it is not as efficient, because every time you use the SASEFRED libref, the FRED interface reads the entire XML file again into SAS. It is best to refer to the SAS data set repeatedly rather than invoking the interface engine repeatedly. For another example that uses more SASEFRED LIBNAME statement options, see the section "Reading Price Data by Using Indices" on page 3395.

Syntax: SASEFRED Interface Engine

The SASEFRED interface engine uses standard engine syntax to read the observations or data values for one or more economic time series. Table 48.1 summarizes the options that the SASEFRED engine uses. In addition, there is one required option: API_KEY='*fred_api_key*'.

Table 48.1 Summary of LIBNAME *libref* SASEFRED Options

Option	Description
APIKEY=	Specifies the required FRED access key that enables you to access the data that the FRED website provides.
AUTOMAP=	Specifies whether or not to overwrite the existing XML map file.
IDLIST=	Specifies a list of time series IDs for accessing FRED data. To select more than one time series, list the unique time series IDs, separated by commas.
PROXY=	Specifies the proxy server to use (if you cannot connect otherwise)
START=	Specifies the start date for the observation period (YYYY-MM-DD formatted string, optional; the default is 9999-12-31 (latest available))
END=	Specifies the end date for the observation period (YYYY-MM-DD formatted string, optional; the default is 1776-07-04 (earliest available))
RTSTART=	Specifies the real-time start date for the observation period (YYYY-MM-DD formatted string, optional; the default is today.)
RTEND=	Specifies the real-time end date for the observation period (YYYY-MM-DD formatted string, optional; the default is today.)
FREQ=	Specifies the reporting frequency of the selected data (lower frequency to aggregate values to): ‘m’ for monthly, ‘d’ for daily. The FRED frequency aggregation feature converts higher-frequency data series to lower-frequency time series (such as converting a monthly time series to an annual time series). See Table 48.2 for the complete list of frequencies.
AGG=	Specifies the aggregation method used for frequency aggregation. The valid aggregation arguments are ’avg’, ’sum’, and ‘eop’; the default is ‘avg’.
LIMIT=	Specifies the maximum number of observations (rows) to return (integer between 1 and 100,000, optional; the default is 100,000)
OFFSET=	Specifies the number of rows (observations) to skip in the returned data set.
SORT=	Specifies the order of the results in ascending or descending observation_date order. The valid sort arguments are ‘asc’ and ‘desc’; the default is ‘asc’.
UNITS=	Specifies a data value transformation. The valid units arguments are ‘lin’, ‘chg’, ‘ch1’, ‘pch’, ‘pc1’, ‘pca’, ‘cch’, ‘cca’, and ‘log’; the default is ‘lin’. See Table 48.3 for more information about units.
VINTAGE=	Specifies one or more dates in history. Vintage dates are used to download data as they existed on that specific date in history (YYYY-MM-DD formatted string, optional; by default no vintage dates are set). You can request one or many vintage dates at a time; dates are in YYYY-MM-DD format and are separated by commas (no blanks allowed). For multiple vintage dates, specify OUTPUT=2 for all observations or OUTPUT=3 for only new or revised observations.
OUTPUT=	Specifies an output type. The valid output arguments are ‘1’ for Observations by Real-Time Period; ‘2’ for Observations by Vintage Date, All Observations; ‘3’ for Observations by Vintage Date, New and Revised Observations Only; and ‘4’ for Observations, Initial Release Only (integer, optional; the default is ‘1’).
FORMAT=	Specifies a file extension that indicates the type of file to retrieve. Only XML is supported for the SASEFRED interface engine in SAS/ETS 13.1 (SAS 9.4).
URL=	Specifies a URL from which to request useful information about available releases, vintage dates, tags, categories, sources, and series. The information is downloaded from the specified URL and stored in the XFREDTPU data set (a temporary utility data set), which can then be saved or renamed to a permanent SAS data set.

The LIBNAME *libref* SASEFRED Statement

LIBNAME *libref* **SASEFRED** '*physical-name*' *options* **;**

The LIBNAME statement assigns a SAS library reference (libref) to the physical path of the directory of FRED data files in which the downloaded FRED XML data are stored.

You must specify the following arguments:

"*physical name*"
specifies the location of the folder where your FRED XML data reside. Enclose the *physical name* in double quotation marks, and end it with a backslash if the folder is in a Windows environment or a forward slash if it is in a UNIX environment.

APIKEY='*fred_apikey*'
specifies the FRED access key that enables you to access the data provided by the FRED website. The FRED access key is a 32-character alphanumeric lowercase string. You can request your *fred_apikey* by visiting the website at the following URL:

http://api.stlouisfed.org/api_key.html

IDLIST=*'fred_idlist'*
specifies the list of time series to be included in the output SAS data set. This list is comma delimited and must be enclosed in single quotation marks.

You can also specify the following *options*.

AGG='AVG' | 'EOP' | 'SUM'
specifies the aggregation method used for frequency aggregation. You can specify the following values:

'AVG' aggregates by averaging the frequencies.

'EOP' aggregates by summing the frequencies.

'SUM' aggregates by using the end of the period.

By default, AGG='AVG'. This option has no effect if the frequency option (FREQ=) is not specified.

AUTOMAP=REPLACE | REUSE
specifies which XMLmap file to use. You can specify the following values:

REPLACE overwrites the existing XMLmap file and uses the most current XMLmap that is generated by the SASEFRED engine and named in the XMLMAP= option.

REUSE uses a preexisting XMLmap file that is named in the XMLMAP= option.

END=*'fred_enddate'*
specifies the end date for the time series in the format YYYY-MM-DD. The default is 9999-12-31 (latest available).

FORMAT=*fred_xmlformat*

specifies the format of the file to be received from the FRED website. Although FRED can report data in many formats, the SASEFRED engine for 9.4 supports the XML format (default).

FREQ=*'fred_frequency'*

specifies a lower frequency to aggregate values to. The FRED frequency aggregation feature converts higher-frequency time series to lower-frequency time series (such as converting a daily time series to a monthly time series). In FRED, the highest frequency is daily, and the lowest frequency is annual. There is no default value for no frequency aggregation. The valid frequency arguments are presented in Table 48.2.

NOTE: An error is returned if you specify a frequency that is higher than the native frequency of the series. For example, if a series has the native frequency 'Annually', it is not possible to aggregate the series to the higher 'Monthly' frequency by using the frequency parameter value 'm'. To find the native frequency of an economic time series, enter the following URL in your web browser. The output includes the 'Frequency' field, which shows native frequency of that time series.

```
http://api.stlouisfed.org/fred/series?series_id=DJCA&api_key=your_fred_apikey
```

NOTE: When a single time series is specified in the IDLIST= option and the FREQ= option is not specified or is an empty string, then the native frequency of that time series is used as the reporting frequency. When multiple time series are specified in the IDLIST= option, then the 'Annual' frequency is used as the reporting frequency unless the reporting frequency is specified in the FREQ= option. If any time series in the IDLIST= option list have a lower native frequency than the requested frequency, then those time series are dropped from the list and excluded from the output.

Table 48.2 FRED Frequency Codes

Frequency Code	Description
d	Displays data on a daily basis
w	Displays data on a weekly basis
bw	Displays data on a biweekly basis
m	Displays data on a monthly basis
q	Displays data on a quarterly basis
sa	Displays data on a semiannual basis
a	Displays data on an annual basis
wef	Displays data on a weekly (ending Friday) basis
weth	Displays data on a weekly (ending Thursday) basis
wew	Displays data on a weekly (ending Wednesday) basis
wetu	Displays data on a weekly (ending Tuesday) basis
wem	Displays data on a weekly (ending Monday) basis
wesu	Displays data on a weekly (ending Sunday) basis
wesa	Displays data on a weekly (ending Saturday) basis
bwew	Displays data on a biweekly (ending Wednesday) basis
bwem	Displays data on a biweekly (ending Monday) basis

LIMIT=*fred_limit*

specifies the maximum number of rows (time series observations) to return, where *fred_limit* is an integer between 1 and 100,000. LIMIT= is optional, and the default is LIMIT=100000.

MAPREF=*fred_xmlmapref*

specifies the fileref used for the map assignment. For an example of the SASEFRED engine that uses the MAPREF= and the XMLMAP= options in the FILENAME statement to assign a filename, as in the following, see the section "Reading Price Data by Using Indices" on page 3395:

```
FILENAME MyMap "U:\fred\test\gstart.map";
```

You can use the MAPREF= and XMLMAP= options to control where the map resides, what you name the map, and how you refer to it with a fileref. You can use the OUTXML= option to name your XML data file. It is placed in the folder designated by 'physical-name', and you can reference it by using the myLib libref in your SASEFRED LIBNAME statement. This is shown in the section "Getting Started: SASEFRED Interface Engine" on page 3382, inside the DATA step in the SET statement. The SET statement reads observations from the input data set myLib.gstart and stores them in a SAS data set named Company_pvol.

OFFSET=*fred_offset*

specifies the number of rows (time series observations) to skip before reading the time series observations from the FRED database, where *fred_offset* is an optional nonnegative integer. If you specify both the OFFSET= and LIMIT= options, the number of rows specified in the OFFSET= option is skipped before the count begins of the number of rows (specified in the LIMIT= option) that are returned. By default, OFFSET=0.

OUTPUT= 1 | 2 | 3 | 4

specifies the type of the file to be received from the FRED website. You can specify the following values:

1 specifies the type Observations by Real-Time Period.

2 specifies the type Observations by Vintage Date, All Observations.

3 specifies the type Observations by Vintage Date, New and Revised Observations Only.

4 specifies the type Observations, Initial Release Only.

If OUTPUT=1 and UNITS='lin', then you must specify a START= date that is later than the series observation start date, Obs_Start. If OUTPUT=3 or OUTPUT=4, then you must specify UNITS='lin'.

OUTXML=*fred_xmlfile*

specifies the name of the file where the XML data that are returned from the FRED website are stored. By default, OUTXML=FRED, which creates a file named *FRED1.xml*.

PROXY="*fred_proxyserver***"**

specifies the proxy server to use when a connection-refused error or a connection-timed-out error occurs. For *fred_proxyserver*, specify the server's HTTP address followed by a colon and the port number, and enclose that string with double quotation marks. For example, PROXY="http://inetgw.unx.sas.com:8118".

RTEND=*'fred_rtenddate'*
specifies the real-time end date for the time series in the format YYYY-MM-DD. When you use the OUTPUT=4 option, it is important to specify RTSTART='1776-07-04' and RTEND='9999-12-31'to get the available observations for the initial release of the data. Failure to do so can result in no observations being returned for the requested series. The default is today.

RTSTART=*'fred_rtstartdate'*
specifies the real-time start date for the time series in the format YYYY-MM-DD. When you use the OUTPUT=4 option, it is important to specify RTSTART='1776-07-04' and RTEND='9999-12-31'to get the available observations for the initial release of the data. Failure to do so can result in no observations being returned for the requested series. The default is today.

SORT='ASC' | 'DSC'
specifies the order of the time series observations. You can specify the following values:

'ASC' specifies that the time series observations are in ascending order.

'DSC' specifies that the time series observations are in descending order.

By default, SORT='ASC'.

START=*'fred_startdate'*
specifies the start date for the time series in the format YYYY-MM-DD. The default is 1776-07-04 (earliest available). When you use the OUTPUT=1 option (observation by real-time period) and the UNITS='chg' option, it is important to specify a date in the START= option that is later than the series observation start date, Obs_Start. Failure to do so forces the SASEFRED interface engine to change UNITS='chg' to UNITS='lin'.

UNITS=*'fred_units'*
specifies the data value transformation. The valid units arguments are 'lin', 'chg', 'ch1', 'pch', 'pc1', 'pca', 'cch', 'cca', and 'log'. the default is UNITS='lin' (for no transformation). The details of the arguments and the corresponding formulas are presented in Table 48.3. When you specify UNITS='chg' and OUTPUT=1 (observation by real-time period), it is important to specify a date in the START= option that is later than the series observation start date, Obs_Start. Failure to do so forces the SASEFRED interface engine to change UNITS='chg' to UNITS='lin'.

Table 48.3 FRED Transformation UNITS Codes

Units Code	Description	Formula
chg	Change	$x_t - x_{t-1}$
ch1	Change from one year ago	$x_t - x_{t-N}$
pch	Percentage change	$(\frac{x_t}{x_{t-1}} - 1) \times 100$
pc1	Percentage change from one year ago	$(\frac{x_t}{x_{t-N}} - 1) \times 100$
pca	Compounded annual rate of change	$(\frac{x_t}{x_{t-1}})^N - 1) \times 100$
cch	Continuously compounded rate of change	$(ln(x_t) - ln(x_{t-1})) \times 100$
cca	Continuously compounded annual rate of change	$((ln(x_t) - ln(x_{t-1}))100) \times N$
log	Natural log	$ln(x_t)$

x_t is the value of series x at time period t. N is the number of observations per year that differs by frequency: daily (N=260), annual (N=1), monthly (N=12), quarterly (N=4), biweekly (N=26), and weekly (N=52).

URL=*"fred_url_link/*<*query_type*>**?**<*query_option=value*>><**LIMIT=***limit*>*"*
queries for useful information (such as categories, tags, groups, and releases) and stores the information in a temporary utility data set named XFREDTPU. Specify the following fields within double quotation marks:

fred_url_link/
specifies the FRED URL that you want to use for the *fred_url_link*. All the examples in this chapter use the following URL:

```
http://api.stlouisfed.org/fred/
```

*query_type***?***query_option*
specifies the type of information that you want to query. You can specify the following *query_types* and *query_options*:

series/vintagedates?series_id=*series_id*
requests the vintage dates for the specified *series_id*, which you must also specify in the IDLIST= option. For an example of this type of query, see Example 48.7.

release/series?release_id=*release*
requests a list of the available series for the specified *release*. For an example of this type of query, see Example 48.8.

source/releases?

source/releases?source_id=*source_id*
requests a list of the releases available today or available for a specified *source_id*. For an example of this type of query, see Example 48.14.

You can also narrow this type of query by specifying the *source_id*; then only the releases that are available for the specified source are stored in the XFREDTPU data set. For an example of this type of query, see Example 48.11.

tags/series?tags_names=*value-list*
requests a list of the series that are available and whose tag names match the specified *value-list*. For an example of this type of query, see Example 48.9.

category/series?category_id=*category_id*
requests a list of the series that are available and whose category ID matches the specified *category_id*. For an example of this type of query, see Example 48.12.

sources?
requests a list of the sources available for today's date. For an example of this type of query, see Example 48.13.

series/categories?series_id=*series_ID*
requests a list of the categories available for a specified *series_ID*. For an example of this type of query, see Example 48.10.

LIMIT=*limit*
limits the number of query results that are returned, where *limit* must be an integer between 1 and 100,000, inclusive. By default, LIMIT=1000 for releases and release date requests and LIMIT=100,000 for time series requests.

VINTAGE=*'fred_vintage_date1,fred_vintage_date2,...,fred_vintage_dateN'*
specifies one or more vintage dates in history. The *fred_vintage_dates* are represented in YYYY-MM-DD format and are used to download the data for a time series as it existed on that specific date in history. The dates in the list are separated by commas (no blanks are allowed). When requesting multiple vintage dates, specify OUTPUT=2 to retrieve all observations or OUTPUT=3 to retrieve only new or revised observations. The default setting is no vintage dates.

Archival Federal Reserve economic data (ALFRED) enable you to retrieve vintage versions of economic data that were available on specific dates in history. To retrieve vintage versions of various time series, enter the following URL in your web browser:

http://alfred.stlouisfed.org/

To see a list of available vintage dates for each series, refer to the FRED documentation at the web page with the following URL:

http://api.stlouisfed.org/docs/fred/series_vintagedates.html

XMLMAP=*fred_xmlmapfile*
specifies the fully qualified name of the location where the XMLmap file is automatically stored. By default, XMLMAP=Fred.map.

Details: SASEFRED Interface Engine

The SASEFRED interface engine enables SAS users to access both ArchivaL Federal Reserve Economic Data (ALFRED) and FRED data that are provided by the FRED website. Normal use is called the *FRED mode*, for which the real-time period is the current day (today). In FRED mode, you are using the current facts: the information about the past that is available today. Economic data sources, releases, series, and observations can change their names or their observation data values over time. The real-time period marks when facts were true or when information was known until it changed. Economic data sources, releases, series, and observations are all assigned a real-time period. For most URL requests, the default real-time period is today. This can be thought of as FRED mode—what information about the past is available today. ALFRED users can change the real-time period to retrieve information that was known as of a point in history. ALFRED uses vintage dates, which are release dates for a series, excluding the release dates when the data values did not change.

Available Sources That Provide FRED Time Series Data

To obtain a list of the available sources of economic data, enter the following URL in your web browser. Table 48.4 shows some of the sources available.

http://api.stlouisfed.org/fred/sources?api_key=your_fred_apikey

Table 48.4 Some Available Sources of Economic Data

ID	Name
1	Board of Governors of the Federal Reserve System
3	Federal Reserve Bank of Philadelphia
4	Federal Reserve Bank of St. Louis
6	Federal Financial Institutions Examination Council
11	Dow Jones & Company
13	Institute for Supply Management
15	The White House: Council of Economic Advisers
16	The White House: Office of Management and Budget
17	U.S. Congress: Congressional Budget Office
18	U.S. Department of Commerce: Bureau of Economic Analysis
19	U.S. Department of Commerce: Census Bureau
21	U.S. Department of Housing and Urban Development

You can use the URL= option to store today's available sources (and associated information about the sources) in a SAS data set. For more information, see the sources query option. For an example see Example 48.13.

You can also use the URL= option to store today's available releases (and associated information about the releases) in a SAS data set. For more information, see the releases query option. For an example see Example 48.14.

FRED API Key

The API key that is used in these examples,'abcdefghijklmnopqrstuvwxyz123456', is for demonstration purposes only. To successfully download data from the FRED website, use your own FRED API key, which is a 32-character alphanumeric lowercase string. You can request your own API key by visiting the website at the following URL:

http://api.stlouisfed.org/api_key.html

Available Releases for Each Source That Provides FRED Time Series Data

Each of the FRED sources might have several releases. To get a list of the releases for a specific source, enter the following URL in your web browser and specify the ID that corresponds to that source. For example, the response to this request retrieves a list of all releases for Dow Jones & Company (source_ID=11).

http://api.stlouisfed.org/fred/source/releases?source_id=11&api_key=your_fred_apikey

Table 48.5 shows the list of releases for Dow Jones & Company.

Table 48.5 Releases for Dow Jones & Company

Release ID	Name	URL
72	Daily Treasury Inflation-Indexed Securities	--
102	Wall Street Journal	http://online.wsj.com/public/us
197	Dow Jones Averages	http://www.djaverages.com

Available Time Series for Each Release ID

Each release of economic sources contains several time series. To get the list of time series for a specific release, enter the following URL in your web browser and specify the ID that corresponds to that release. For example, the following URL retrieves a list of all time series for the Dow Jones Averages release (release_ID=197):

```
http://api.stlouisfed.org/fred/release/series?release_id=197&api_key=your_fred_apikey
```

Table 48.6 shows all the time series that are included in the Dow Jones Averages release.

Table 48.6 Time Series for the Release of Dow Jones Averages

Series ID	Title	Start	End	Frequency
DJCA	Dow Jones Composite Average	1934-01-02	2012-11-23	Daily
DJIA	Dow Jones Industrial Average	1896-05-26	2012-11-23	Daily
DJTA	Dow Jones Transportation Average	1896-10-26	2012-11-23	Daily
DJUA	Dow Jones Utility Average	1929-01-02	2012-11-23	Daily

You can use the URL= option to store the list of available time series for a particular release in a SAS data set. For more information, see the release/series query option.

Available Native Frequency for Each Series ID

To find the native frequency of an economic time series, enter the following URL in your web browser. The output includes the "Frequency" field, which shows the native frequency of that time series.

```
http://api.stlouisfed.org/fred/series?series_id=DJCA&api_key=your_fred_apikey
```

The response to the preceding request follows. As the response shows, the native frequency of the Dow Jones Composite Average (DJCA) time series is Daily (frequency=Daily).

<series id="DJCA" realtime_start="2012–11–26" realtime_end="2012–11–26" title="Dow Jones Composite Average" observation_start="1934–01–02" observation_end="2012–11–23" ***frequency="Daily, Close"*** *frequency_short="D" units="Index" units_short="Index" seasonal_adjustment="Not Seasonally Adjusted" seasonal_adjustment_short="NSA" last_updated="2012–11–26 09:05:12–06" popularity="48">*

Vintage Dates for Each Series ID

Vintage dates are the release dates for a time series, excluding those releases in which the data did not change. To obtain a list of vintage dates for a particular series, you can enter the following URL in your web browser and specify the series ID of the series that you are interested in. For example, the following URL retrieves a list of all vintage dates for the MICH series, showing the median expected price change (the next 12 months from the Survey of Consumers):

```
http://api.stlouisfed.org/fred/series/vintagedates?series_id=MICH&api_key=your_fred_apikey
```

The resulting list of observations is too long to show here—172 vintage dates, ranging from 1999-02-26 to 2013-05-31. You can get only the vintage dates that you want by specifying the VINTAGE= option.

You can use the URL= option to store the list of available vintage dates for a particular time series in a SAS data set. For more information, see the series/vintagedates query option.

SAS Output Data Set

You can use the SAS DATA step to write the selected FRED data to a SAS data set. This enables you to use SAS software to easily analyze the data. If you specify the name of the output data set in the DATA statement, the engine supervisor creates a SAS data set that has the specified name in either the SAS Work library or, if specified, the User library.

The contents of the SAS data set include the date of each observation and the series name of each series that is read from the FRED data source.

The SASEFRED interface engine maintains the sort order, so the time series are sorted in the resulting SAS data set by the order specified in the SORT= option, by date (time ID), and by variable (time series item name).

You can use the PRINT and CONTENTS procedures to print your output data set and its contents. Alternatively, you can view your SAS output observations by opening the desired output data set in a SAS Explorer window. You can also use the SQL procedure with your SASEFRED libref to create a custom view of your data.

SAS OUTXML File

The SAS XML (XML format) data that are returned from the FRED website are placed in a file named by the OUTXML= option. The SAS XML data file is placed in the location that is specified by the *physical-name* in your LIBNAME *libref* SASEFRED statement, which is described in the section "The LIBNAME *libref* SASEFRED Statement" on page 3386.

SAS XML Map File

The XML map that (by default) is automatically created is assigned the full pathname given by the XMLMAP= option in your LIBNAME *libref* SASEFRED statement. The map file is either reused (not overwritten) if you specify AUTOMAP=REUSE or overwritten by a new map if you specify AUTOMAP=REPLACE (the default). The SASEFRED engine invokes the XMLV2 engine to create the map and to read the data into SAS.

XFREDTPU SAS Data Set

You can use the URL= option to query for useful information such as categories, tags, groups, and releases and store the information in a temporary utility data set named XFREDTPU. After you have this information, you can use it for selecting the data you want to include in a subsequent SASEFRED *libref* statement. For more information about the seven possible types of XFREDTPU contents, see the URL= option.

Reading Price Data by Using Indices

The following statements enable you to access the S&P 500 Stock Price Index (IDLIST=SP500) and the Wilshire 5000 Price Index (IDLIST=WILL5000PR) on a monthly basis:

```
options validvarname=any;
title 'FRED Data: SP500 Stock Index and Wilshire 5000 Price Index';
LIBNAME myLib sasefred "%sysget(FRED)"
   OUTXML=gstart
   AUTOMAP=replace
   MAPREF=MyMap
   XMLMAP="%sysget(FRED)gstart.map"
   APIKEY='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
   IDLIST='sp500,will5000pr'
   START='2011-01-01'
   END='2012-01-01'
   FREQ='m'
   FORMAT=xml
   ;

data stock_price;
   set myLib.gstart ;
run;

proc contents data=stock_price; run;
proc print data=stock_price; run;
```

Figure 48.2 FRED Data: stock_price

FRED Data: SP500 Stock Index and Wilshire 5000 Price Index

Obs	date	realtime_start	realtime_end	SP500	WILL5000PR
1	2011-01-01	2015-03-19	2015-03-19	1282.62	13368.14
2	2011-02-01	2015-03-19	2015-03-19	1321.12	13772.27
3	2011-03-01	2015-03-19	2015-03-19	1304.49	13610.85
4	2011-04-01	2015-03-19	2015-03-19	1331.51	13920.50
5	2011-05-01	2015-03-19	2015-03-19	1338.31	13967.83
6	2011-06-01	2015-03-19	2015-03-19	1287.29	13441.17
7	2011-07-01	2015-03-19	2015-03-19	1325.18	13848.15
8	2011-08-01	2015-03-19	2015-03-19	1185.31	12296.04
9	2011-09-01	2015-03-19	2015-03-19	1173.88	12144.13
10	2011-10-01	2015-03-19	2015-03-19	1207.22	12459.48
11	2011-11-01	2015-03-19	2015-03-19	1226.41	12684.75
12	2011-12-01	2015-03-19	2015-03-19	1243.32	12850.31
13	2012-01-01	2015-03-19	2015-03-19	1300.58	13465.23

The SASEFRED interface engine supports the XML format. The XML data that the FRED website returns are placed in a file named by the OUTXML= option. The XML map that is automatically created is assigned the full pathname specified by the XMLMAP= option, and the fileref that is used for the map assignment is specified by the MAPREF= option. In the preceding example, the SASEFRED engine uses the MAPREF= and XMLMAP= options in the FILENAME statement to assign a filename:

```
FILENAME MyMap "U:\fred\test\gstart.map";
```

You can use the MAPREF= and XMLMAP= options to control where the map resides, what you name the map, and how you refer to it with a fileref. You can use the OUTXML= option to name your XML data file; it is described in the section "SAS OUTXML File" on page 3394. The XML data file is placed in the folder designated by 'physical-name', which is described in the section "The LIBNAME *libref* SASEFRED Statement" on page 3386. You can refer to your data by using the myLib libref in your SASEFRED LIBNAME statement. The myLib libref is shown inside the DATA step in the SET statement. The SET statement reads observations from the input data set myLib.gstart and stores them in a SAS data set named stock_price, as shown in Figure 48.2. You can also use the SAS DATA step to perform further processing and to store the resulting time series in a SAS data set; this process is described in the section "SAS Output Data Set" on page 3394.

To specify the list of time series that you want to retrieve, use the IDLIST= option. This option accepts a string enclosed in single quotation marks that denotes a list of time series that you select for the resulting SAS data set. The series IDs are separated by commas, so valid time series IDs cannot contain embedded commas or quotes. The stock_price data set contains two time series variables, sp500 and will5000pr, as specified in the IDLIST= option, and the observation range is controlled by the START= and END= options. The stock_price data set contains observations that range from January 1, 2011, to January 1, 2012, as specified by the START= and END= options. The frequency of the data is monthly, as indicated by the 'm' in the FREQ= option.

NOTE: The '%20' is a special character for URL encoding of blanks. If the time series ID that you name in the IDLIST= option contains a blank, you must use the '%20' wherever the blank appears in the time series name. If the time series ID contains an underscore, then you must use an underscore in the time series name. The underscore and the blank are not equivalent in the FRED databases, so make sure that you use the '%20' (URL encoded space) to designate blank characters.

Examples: SASEFRED Interface Engine

Example 48.1: Retrieving Data for Multiple Time Series

This example shows how to use multiple time series IDs to retrieve the average balance of payment basis data for the exports (BOPXGS) and imports (BOPMGS) of goods and services for the last 15 years, starting 1997-01-01 and ending 2011-01-01, with an annual frequency.

```
title 'Retrieve Balance of Payment Data for the Exports and Imports';
libname _all_ clear;

libname fred sasefred "%sysget(FRED)"
   OUTXML=fredex01
   AUTOMAP=replace
   MAPREF=MyMap
   XMLMAP="%sysget(FRED)fredex01.map"
   APIKEY='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
   IDLIST='bopxgs,bopmgs'
   START='1997-01-01'
   END='2011-01-01'
   FREQ='a'
   OUTPUT=1
   AGG='avg'
   FORMAT=xml;

data export_import;
   set fred.fredex01 ;
run;

proc contents data=export_import; run;
proc print data=export_import; run;
```

Output 48.1.1 Retrieve Balance of Payment Data for the Exports and Imports

Retrieve Balance of Payment Data for the Exports and Imports

Obs	date	realtime_start	realtime_end	BOPXGS	BOPMGS
1	1997-01-01	2015-03-19	2015-03-19	233.614	-260.682
2	1998-01-01	2015-03-19	2015-03-19	233.293	-274.829
3	1999-01-01	2015-03-19	2015-03-19	241.824	-306.479
4	2000-01-01	2015-03-19	2015-03-19	268.064	-361.193
5	2001-01-01	2015-03-19	2015-03-19	250.634	-341.011
6	2002-01-01	2015-03-19	2015-03-19	243.652	-348.391
7	2003-01-01	2015-03-19	2015-03-19	254.367	-377.839
8	2004-01-01	2015-03-19	2015-03-19	289.490	-441.961
9	2005-01-01	2015-03-19	2015-03-19	320.775	-499.336
10	2006-01-01	2015-03-19	2015-03-19	363.212	-553.641
11	2007-01-01	2015-03-19	2015-03-19	412.059	-588.403
12	2008-01-01	2015-03-19	2015-03-19	458.632	-635.814
13	2009-01-01	2015-03-19	2015-03-19	393.685	-489.628
14	2010-01-01	2015-03-19	2015-03-19	462.232	-585.896
15	2011-01-01	2015-03-19	2015-03-19	530.359	-667.515

Example 48.2: Retrieving Data by Using the Vintage Date

This example shows how to use the vintage date to retrieve data for exports of goods and services as they existed on that specific date in history. OUTPUT=3 retrieves the new and revised observations only, by the vintage date (VINTAGE=2012-06-14). If OUTPUT=3, then you must specify UNITS='lin'. In this example, the UNITS= option is not specified, so it assumes its default value, which is 'lin'. Specifying a different argument for the UNITS= option (such as 'chg') is invalid for OUTPUT= 3, so 'chg' is replaced by the default value ('lin').

```
title 'Retrieve Data for the Exports of Goods and Service by Using Vintage Date';
libname _all_ clear;

libname fred sasefred "%sysget(FRED)"
   OUTXML=fredex02
   AUTOMAP=replace
   MAPREF=MyMap
   XMLMAP="%sysget(FRED)fredex02.map"
   APIKEY='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
   IDLIST='bopxgsa'
   VINTAGE='2012-06-14'
   OUTPUT=3
   FORMAT=xml;

data export_vin;
   set fred.fredex02 ;
run;

proc contents data=export_vin; run;
proc print data=export_vin; run;
```

Output 48.2.1 Retrieve Data for the Exports of Goods and Services by Using the Vintage Date

Retrieve Data for the Exports of Goods and Service by Using Vintage Date

Obs	date	BOPXGSA_20120614
1	2009-01-01	1578.95
2	2010-01-01	1842.49
3	2011-01-01	2103.37

Example 48.3: Selecting Time Series When Native Frequency Is Less Than Requested Frequency

This example shows how to retrieve data for multiple time series that have different default frequencies. The time series are Domestic Financial Commercial Paper Outstanding (DFINCP), Domestic Nonfinancial Commercial Paper Outstanding (DNFINCP), Foreign Financial Commercial Paper Outstanding (FFINCP), Foreign Nonfinancial Commercial Paper Outstanding (FNFINCP), and Total Credit Market Assets Held by Domestic Financial Sectors (ABSITCMAHDFS). The native frequency of the first four time series is 'Weekly', and the native frequency of the last time series (ABSITCMAHDFS) is 'Quarterly'. Note that the requested frequency as it is specified by the FREQ= option is 'Weekly' (FREQ=w). The native frequency of the last time series (ABSITCMAHDFS) is lower than the requested frequency. Therefore, this time series is excluded from the list, and only the observations that correspond to the first four time series are presented. If you want to retrieve the observations for all five time series, then the value of the FREQ= option needs to be less than or equal to all the native frequencies (here, weekly and quarterly). In this case, the valid frequency parameters would be 'q', 'sa', and 'a'. See Example 48.4.

```
title 'Selecting Time Series When Native Frequency Is Less Than Requested Frequency';
libname _all_ clear;

libname fred sasefred "%sysget(FRED)"
   OUTXML=fredex03
   AUTOMAP=replace
   MAPREF=MyMap
   XMLMAP="%sysget(FRED)fredex03.map"
   APIKEY='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
   IDLIST='dfincp,dnfincp,ffincp,fnfincp,absitcmahdfs'
   START='2010-01-01'
   END='2010-05-20'
   FREQ='w'
   OUTPUT=1
   FORMAT=xml;

data diffNative_freqw;
   set fred.fredex03 ;
run;

proc contents data=diffNative_freqw; run;
proc print data=diffNative_freqw; run;
```

Output 48.3.1 Selecting Time Series When Native Frequency Is Less Than Requested Frequency

Selecting Time Series When Native Frequency Is Less Than Requested Frequency

Obs	date	realtime_start	realtime_end	DFINCP	DNFINCP	FFINCP	FNFINCP
1	2010-01-06	2015-03-19	2015-03-19	295.1	86.9	248.6	23.9
2	2010-01-13	2015-03-19	2015-03-19	315.0	90.1	241.6	25.8
3	2010-01-20	2015-03-19	2015-03-19	314.9	90.1	230.3	27.5
4	2010-01-27	2015-03-19	2015-03-19	370.0	83.0	232.7	31.3
5	2010-02-03	2015-03-19	2015-03-19	350.1	84.0	225.0	34.0
6	2010-02-10	2015-03-19	2015-03-19	363.9	81.3	223.7	36.2
7	2010-02-17	2015-03-19	2015-03-19	366.7	83.3	226.2	38.5
8	2010-02-24	2015-03-19	2015-03-19	385.8	77.1	231.8	39.0
9	2010-03-03	2015-03-19	2015-03-19	366.8	78.7	227.8	40.1
10	2010-03-10	2015-03-19	2015-03-19	380.1	79.2	229.3	40.1
11	2010-03-17	2015-03-19	2015-03-19	360.5	84.5	224.2	39.4
12	2010-03-24	2015-03-19	2015-03-19	355.1	82.7	218.5	39.8
13	2010-03-31	2015-03-19	2015-03-19	352.7	90.6	217.7	40.0
14	2010-04-07	2015-03-19	2015-03-19	335.2	95.8	217.6	40.0
15	2010-04-14	2015-03-19	2015-03-19	329.4	93.4	209.2	40.3
16	2010-04-21	2015-03-19	2015-03-19	326.8	93.1	211.8	41.6
17	2010-04-28	2015-03-19	2015-03-19	358.9	95.3	203.4	41.9
18	2010-05-05	2015-03-19	2015-03-19	353.8	91.3	200.8	43.4
19	2010-05-12	2015-03-19	2015-03-19	358.5	90.7	190.3	43.2
20	2010-05-19	2015-03-19	2015-03-19	330.0	92.4	180.5	40.9

Example 48.4: Selecting Time Series When Native Frequency Is Greater Than Requested Frequency

This example shows how to retrieve data for multiple time series that have different default frequencies. The time series are Domestic Financial Commercial Paper Outstanding (DFINCP), Domestic Nonfinancial Commercial Paper Outstanding (DNFINCP), Foreign Financial Commercial Paper Outstanding (FFINCP), Foreign Nonfinancial Commercial Paper Outstanding (FNFINCP), and Total Credit Market Assets Held by Domestic Financial Sectors (ABSITCMAHDFS). The native frequency of the first four time series is 'Weekly', and the native frequency of the last time series (ABSITCMAHDFS) is 'Quarterly'. The requested frequency as it is specified by the FREQ= option is 'Quarterly' (FREQ=q). The native frequency of all five time series is either greater than or equal to the requested frequency. Hence, the output includes the data for all time series.

```
title 'Selecting Time Series When Native Frequency Is Greater Than Requested Frequency';
libname _all_ clear;

libname fred sasefred "%sysget(FRED)"
   OUTXML=fredex04
   AUTOMAP=replace
   MAPREF=MyMap
   XMLMAP="%sysget(FRED)fredex04.map"
   APIKEY='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
   IDLIST='dfincp,dnfincp,ffincp,fnfincp,absitcmahdfs'
   START='2010-01-01'
   END='2010-05-20'
   FREQ='q'
   OUTPUT=1
   FORMAT=xml;

data diffNative_freqq;
   set fred.fredex04;
run;

proc contents data=diffNative_freqq; run;
proc print data=diffNative_freqq; run;
```

Output 48.4.1 Selecting Time Series When Native Frequency Is Greater Than Requested Frequency

Selecting Time Series When Native Frequency Is Greater Than Requested Frequency

Obs	date	realtime_start	realtime_end	DFINCP	DNFINCP	FFINCP	FNFINCP	ABSITCMAHDFS
1	2010-01-01	2015-03-19	2015-03-19	352.1	84.0	229.0	35	2552.98
2	2010-04-01	2015-03-19	2015-03-19	341.6	98.5	186.9	38	2424.03

Example 48.5: Specifying One Series ID with Multiple Vintage Dates for the OUTPUT=2 Option

This example demonstrates how to request the CBI time series, which show the change in private industries for three different vintage dates: 1947-08-17, 1966-08-11, and 1994-08-26. Using the early range of START='1942-01-01'and END='1947-04-01', you can get an idea of how the changes show up for each vintage date. If you specify OUTPUT=2, each time series is named by concatenating the series ID to the vintage date with an underscore.

```
title 'Specifying One Series ID with Multiple Vintage Dates for OUTPUT=2 Option';
libname _all_ clear;

libname fred sasefred "%sysget(FRED)"
   OUTXML=fredex05
   AUTOMAP=replace
   MAPREF=MyMap
   XMLMAP="%sysget(FRED)fredex05.map"
   APIKEY='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
   IDLIST='CBI'
   VINTAGE='1947-08-17,1966-08-11,1994-08-26'
   START='1942-01-01'
   END='1947-04-01'
   FREQ='q'
   OUTPUT=2
   UNITS='lin'
   FORMAT=xml;

data threeVinsCBI;
   set fred.fredex05;
run;

proc contents data=threeVinsCBI; run;
proc print data=threeVinsCBI; run;
```

Output 48.5.1 Specifying One Series ID with Multiple Vintage Dates for OUTPUT=2 Option

Specifying One Series ID with Multiple Vintage Dates for OUTPUT=2 Option

Obs	date	CBI_19470817	CBI_19660811	CBI_19940826
1	1942-01-01	3.9	.	.
2	1942-04-01	3.6	.	.
3	1942-07-01	-0.9	.	.
4	1942-10-01	-0.9	.	.
5	1943-01-01	-2.4	.	.
6	1943-04-01	-2.1	.	.
7	1943-07-01	1.1	.	.
8	1943-10-01	-1.5	.	.
9	1944-01-01	-2.4	.	.
10	1944-04-01	-3.2	.	.
11	1944-07-01	-1.0	.	.
12	1944-10-01	-1.3	.	.
13	1945-01-01	-2.8	.	.
14	1945-04-01	-1.5	.	.
15	1945-07-01	0.1	.	.
16	1945-10-01	-0.8	.	.
17	1946-01-01	2.3	5.9	5.7
18	1946-04-01	2.0	8.8	8.6
19	1946-07-01	4.9	6.1	5.9
20	1946-10-01	5.4	4.7	4.5
21	1947-01-01	2.7	0.4	0.4
22	1947-04-01	1.5	-1.0	-1.2

Example 48.6: Specifying Two Series IDs with Multiple Vintage Dates and Descending Sort Order

This example demonstrates how to request the ADJRES and ADJRESN time series, which show the St. Louis adjusted reserves, the first of which is seasonally adjusted and the second of which is not seasonally adjusted. The request is made for three different vintage dates, but only 2006-08-31 and 2013-06-13 yield data when you use the range of START='2004-01-01'and END='2012-12-01'. If you specify OUTPUT=2, each time series is named by concatenating the series ID to the vintage date with an underscore. For brevity, Output 48.6.1 shows only the first 10 and last 10 observations. The sort order is descending; that is why the dates start with the most recent observation and continue in biweekly (ending Wednesday) periods to the least recent.

```
title 'Specifying Two Series IDs with Multiple Vintage Dates and Descending Sort Order';
libname _all_ clear;

libname fred sasefred "%sysget(FRED)"
   OUTXML=fredex06
   AUTOMAP=replace
   MAPREF=MyMap
   XMLMAP="%sysget(FRED)fredex06.map"
   APIKEY='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
   IDLIST='ADJRES,ADJRESN'
   VINTAGE='2003-07-31,2006-08-31,2013-06-13'
   START='2004-01-01'
   END='2012-12-01'
   FREQ='bw'
   OUTPUT=2
   AGG='avg'
   SORT='desc'
   FORMAT=xml;

data fredPDD;
   set fred.fredex06;
run;

proc contents data=fredPDD; run;
title3 "First 10/Last 10 Obs, IDLIST=ADJRES,ADJRESN, and SORT=Descending";
%pri20nom(fredPDD);
title3;
```

Output 48.6.1 Specifying Two Series IDs with Multiple Vintage Dates and Descending Sort Order—First 10 and Last 10 Observations

Specifying Two Series IDs with Multiple Vintage Dates and Descending Sort Order

Obs	date	ADJRES_20130613	ADJRES_20060831	ADJRESN_20130613	ADJRESN_20060831
1	2012-11-28	1591.92	.	1583.96	.
2	2012-11-14	1583.90	.	1583.90	.
3	2012-10-31	1573.04	.	1568.32	.
4	2012-10-17	1563.23	.	1560.10	.
5	2012-10-03	1511.02	.	1518.58	.
6	2012-09-19	1587.55	.	1563.74	.
7	2012-09-05	1583.80	.	1594.89	.
8	2012-08-22	1618.63	.	1615.40	.
9	2012-08-08	1652.49	.	1639.27	.
10	2012-07-25	1620.07	.	1629.79	.
11	2004-05-12	95.89	95.871	94.74	94.720
12	2004-04-28	96.25	96.154	97.79	97.693
13	2004-04-14	93.38	93.293	93.38	93.293
14	2004-03-31	94.81	94.718	93.67	93.582
15	2004-03-17	94.28	94.146	93.91	93.769
16	2004-03-03	94.13	94.096	95.73	95.696
17	2004-02-18	92.05	92.001	93.24	93.197
18	2004-02-04	96.25	96.192	95.10	95.038
19	2004-01-21	96.54	96.511	97.60	97.573
20	2004-01-07	96.06	96.044	100.00	99.982

Example 48.7: Vintage Dates for a Specific Series with the URL= Option

The following statements demonstrate how to use the URL= option to obtain the VINTAGE_DATE and VINTAGE_DATES data sets for a specified series and how to create a permanent data set named *VINDAT1* in the MyLib SAS library. You must specify the series in both the URL= option and the IDLIST= option.

```
title 'Specifying the URL= Option to Create the VINTAGE_DATES Data Set';
libname _all_ clear;

libname mylib "U:\fred940\doc\";

libname fred1 sasefred "%sysget(FRED)"
   URL="http://api.stlouisfed.org/fred/series/vintagedates?series_id=N5005C1A027NBEA"
   APIKEY='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
   IDLIST= 'N5005C1A027NBEA'
   ;

data mylib.vindat1;
   set fred1.XFREDtpu;
run;

proc print
   data=mylib.vindat1;
run;

proc contents
   data=mylib.vindat1;
run;
```

The list of available vintage dates for the N5005C1A027NBEA series is read from the *XFREDtpu.xml* file that is downloaded by the SASEFRED engine. The contents are shown in Output 48.7.1. The engine automatically maps the data in the XML file and reads the data into the XFREDTPU data set when the SET statement is executed. When the DATA step runs, the data in the temporary utility data set are read and stored in the permanent data set named *vindat1.sas7bdat* in the MyLib library. A side effect of the DATA step is the automatic creation of two SAS data sets named *vintage_date.sas7bdat* and *vintage_dates.sas7bdat* in the FRED1 library's location.

Output 48.7.1 Specifying the URL= Option to Create the VINTAGE_DATES Data Set

Specifying the URL= Option to Create the VINTAGE_DATES Data Set

Obs	vintage_dates_ORDINAL	vintage_date_ORDINAL	vintage_date
1	1	1	2013-02-28
2	1	2	2013-03-28
3	1	3	2013-05-30
4	1	4	2013-07-31
5	1	5	2014-03-27
6	1	6	2014-05-29
7	1	7	2014-07-30

Example 48.8: Series for a Specific Release with the URL= Option

The following statements demonstrate how to use the URL= option to obtain the SERIES and SERIESS data sets for a specified release and how to create a permanent data set named SERIES2 in the MyLib SAS library:

```
title 'Specifying the URL= Option to Create the SERIES Data Set';
libname _all_ clear;

libname fred2 sasefred "%sysget(FRED)"
   URL="http://api.stlouisfed.org/fred/release/series?release_id=51"
   APIKEY='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
   ;

data series2;
   set fred2.XFREDtpu;
run;

proc contents
   data=series2;
run;
title3 "First 5/Last 5 Obs, SERIES2 Data Set";
%pri10nom(series2);
title3;
```

The returned data are stored in the XFREDTPU data set and are copied to the permanent data set named *series2.sas7bdat* in the MyLib library. A side effect of the DATA step is the automatic creation of two SAS data sets named *series.sas7bdat* and *seriess.sas7bdat* in the FRED2 library's location. Fifty-seven series are returned for release_id=51; Output 48.8.1 shows the first and last five observations of the SERIES data set.

Output 48.8.1 Specifying the URL= Option to Create the SERIES Data Set—First 5 and Last 5 Observations

Specifying the URL= Option to Create the SERIES Data Set

Obs	seriess_ORDINAL	series_ORDINAL	series_id	series_realtime_start	series_realtime_end
1	1	1	BOMTVLM133S	2015-03-19	2015-03-19
2	1	2	BOMVGMM133S	2015-03-19	2015-03-19
3	1	3	BOMVJMM133S	2015-03-19	2015-03-19

Obs	series_title	series_observation_start	series_observation_end	series_frequency	series_frequency_short
1	U.S. Imports of Services - Travel	1992-01-01	2015-01-01	Monthly	M
2	U.S. Imports of Services - U.S. Government Misc. Services (DISCONTINUED)	1992-01-01	2013-12-01	Monthly	M
3	U.S. Imports of Services - Direct Defense Expenditures (DISCONTINUED)	1992-01-01	2013-12-01	Monthly	M

Obs	series_units	series_units_short	series_seasonal_adjustment	series_seasonal_adjustment_short
1	Million of Dollars	Mil. of $	Seasonally Adjusted	SA
2	Millions of Dollars	Mil. of $	Seasonally Adjusted	SA
3	Millions of Dollars	Mil. of $	Seasonally Adjusted	SA

Obs	series_last_updated	series_popularity	series_notes
1	2015-03-06 07:56:13-06	0	
2	2014-10-20 09:27:37-05	0	BEA has introduced new table presentations, including a new presentation of services, as part of a comprehensive restructuring of BEA's international economic accounts.For more information see http://www.bea.gov/international/revision-2014.htm.
3	2014-10-20 09:26:44-05	0	BEA has introduced new table presentations, including a new presentation of services, as part of a comprehensive restructuring of BEA's international economic accounts.For more information see http://www.bea.gov/international/revision-2014.htm.

Output 48.8.1 *continued*

Specifying the URL= Option to Create the SERIES Data Set

Obs	seriess_ORDINAL	series_ORDINAL	series_id	series_realtime_start	series_realtime_end
4	1	4	BOMVMPM133S	2015-03-19	2015-03-19
5	1	5	BOMVOMM133S	2015-03-19	2015-03-19
6	1	53	ITXMARM133S	2015-03-19	2015-03-19

Obs	series_title	series_observation_start	series_observation_end	series_frequency	series_frequency_short
4	U.S. Imports of Services - Passenger Fares	1992-01-01	2015-01-01	Monthly	M
5	U.S. Imports of Services - Other Private Services (DISCONTINUED)	1992-01-01	2013-12-01	Monthly	M
6	U.S. Exports of Services: Maintenance and Repair Services, not included elsewhere	1999-01-01	2015-01-01	Monthly	M

Obs	series_units	series_units_short	series_seasonal_adjustment	series_seasonal_adjustment_short
4	Million of Dollars	Mil. of $	Seasonally Adjusted	SA
5	Million of Dollars	Mil. of $	Seasonally Adjusted	SA
6	Millions of Dollars	Mil. of $	Seasonally Adjusted	SA

Obs	series_last_updated	series_popularity	series_notes
4	2015-03-06 07:56:13-06	0	
5	2014-10-20 09:25:54-05	0	BEA has introduced new table presentations, including a new presentation of services, as part of a comprehensive restructuring of BEA's international economic accounts.For more information see http://www.bea.gov/international/revision-2014.htm.
6	2015-03-06 08:16:02-06	0	

Output 48.8.1 *continued*

Specifying the URL= Option to Create the SERIES Data Set

Obs	seriess_ORDINAL	series_ORDINAL	series_id	series_realtime_start	series_realtime_end
7	1	54	ITXOBSM133S	2015-03-19	2015-03-19
8	1	55	ITXTAEM133S	2015-03-19	2015-03-19
9	1	56	ITXTCIM133S	2015-03-19	2015-03-19
10	1	57	ITXTRAM133S	2015-03-19	2015-03-19

Obs	series_title	series_observation_start	series_observation_end	series_frequency	series_frequency_short
7	U.S. Exports of Services: Other Business Services	1999-01-01	2015-01-01	Monthly	M
8	U.S. Exports of Services: Travel (for All Purposes Including Education)	1999-01-01	2015-01-01	Monthly	M
9	U.S. Exports of Services: Telecommunications, Computer, and Information Services	1999-01-01	2015-01-01	Monthly	M
10	U.S. Exports of Services: Transport	1999-01-01	2015-01-01	Monthly	M

Obs	series_units	series_units_short	series_seasonal_adjustment	series_seasonal_adjustment_short
7	Millions of Dollars	Mil. of $	Seasonally Adjusted	SA
8	Millions of Dollars	Mil. of $	Seasonally Adjusted	SA
9	Millions of Dollars	Mil. of $	Seasonally Adjusted	SA
10	Millions of Dollars	Mil. of $	Seasonally Adjusted	SA

Obs	series_last_updated	series_popularity	series_notes
7	2015-03-06 08:16:02-06	0	
8	2015-03-06 08:16:02-06	0	
9	2015-03-06 08:16:02-06	0	
10	2015-03-06 08:16:02-06	0	

Example 48.9: Series for Specific Tags with the URL= Option

The following statements demonstrate how to use the URL= option to obtain the SERIES and SERIESS data sets for specified tag names and how to create a permanent data set named TAGS_SERIES4 in the MyLib SAS library:

```
title 'Specifying the URL= Option to Create the TAGS_SERIES4 Data Set.';
libname _all_ clear;
libname mylib "U:\fred940\doc\";

libname fred4 sasefred "U:\fred940\test\"
   debug=on
   URL="http://api.stlouisfed.org/fred/tags/series?tag_names=slovenia;food;oecd"
   APIKEY='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
   ;

data mylib.tags_series4;
   set fred4.XFREDtpu;
run;


proc print
   data=mylib.tags_series4(obs=2);
run;

proc contents
   data=mylib.tags_series4;
run;
```

The returned data are stored in the XFREDTPU data set and are copied to the permanent data set named *tags_series4.sas7bdat* in the MyLib library. A side effect of the DATA step is the automatic creation of two SAS data sets named *series.sas7bdat* and *seriess.sas7bdat* in the FRED4 library's location. Eighteen series are returned for the specified tag names; the OBS=2 option in the DATA statement in the PROC PRINT step prints only two of them. Output 48.9.1 shows the first two observations of the TAGS_SERIES4 data set.

Output 48.9.1 Specifying the URL= Option to Create the TAGS_SERIES4 Data Set

Specifying the URL= Option to Create the TAGS_SERIES4 Data Set.

Obs	seriess_ORDINAL	series_ORDINAL	series_id	series_realtime_start	series_realtime_end
1	1	1	CPGDFD02SIA657N	2015-03-19	2015-03-19
2	1	2	CPGDFD02SIA659N	2015-03-19	2015-03-19

Obs	series_title	series_observation_start	series_observation_end	series_frequency	series_frequency_short
1	Consumer Price Index: Total Food Excluding Restaurants for Slovenia©	1996-01-01	2014-01-01	Annual	A
2	Consumer Price Index: Total Food Excluding Restaurants for Slovenia©	1996-01-01	2014-01-01	Annual	A

Obs	series_units	series_units_short	series_seasonal_adjustment	series_seasonal_adjustment_short
1	Growth Rate Previous Period	Growth Rate Previous Period	Not Seasonally Adjusted	NSA
2	Growth Rate Same Period Previous Year	Growth Rate Same Period Previous Yr.	Not Seasonally Adjusted	NSA

Obs	series_last_updated	series_popularity	series_notes
1	2015-02-03 15:27:27-06	0	OECD descriptor ID: CPGDFD02 OECD unit ID: GP OECD country ID: SVN All OECD data should be cited as follows: OECD, "Main Economic Indicators - complete database", Main Economic Indicators (database),http://dx.doi.org/10.1787/data-00052-en (Accessed on date) Copyright, 2014, OECD. Reprinted with permission.
2	2015-02-03 15:27:27-06	0	OECD descriptor ID: CPGDFD02 OECD unit ID: GY OECD country ID: SVN All OECD data should be cited as follows: OECD, "Main Economic Indicators - complete database", Main Economic Indicators (database),http://dx.doi.org/10.1787/data-00052-en (Accessed on date) Copyright, 2014, OECD. Reprinted with permission.

Example 48.10: Categories for a Specific Series with the URL= Option

The following statements demonstrate how to use the URL= option to obtain the CATEGORY and CATEGORIES data sets and how to create a permanent data set named SERIES_CAT7 in the MyLib SAS library:

```
title 'Specifying the URL= Option to Create the SERIES_CAT7 Data Set';
libname _all_ clear;
libname mylib "U:\fred940\doc\";

libname fred7 sasefred "U:\fred940\test\"
   debug=on
   URL="http://api.stlouisfed.org/fred/series/categories?series_id=EXJPUS"
   APIKEY='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
   IDLIST='EXJPUS'
   ;

data mylib.series_cat7;
   set fred7.XFREDtpu;
run;

proc print
   data=mylib.series_cat7;
run;

proc contents
   data=mylib.series_cat7;
run;
```

The returned data are stored in the XFREDTPU data set and are copied to the permanent data set named *series_cat7.sas7bdat* in the MyLib library. A side effect of the DATA step is the automatic creation of two SAS data sets named *category.sas7bdat* and *categories.sas7bdat* in the FRED7 library's location. Two categories are returned for the specified series ID, as shown in Output 48.10.1.

Output 48.10.1 Specifying the URL= Option to Create the SERIES_CAT7 Data Set

Specifying the URL= Option to Create the SERIES_CAT7 Data Set

Obs	categories_ORDINAL	category_ORDINAL	category_id	category_name	category_parent_id
1	1	1	95	Monthly Rates	15
2	1	2	275	Japan	158

Example 48.11: Categories for a Specific Source with the URL= Option

The following statements demonstrate how to use the URL= option to obtain the RELEASE and RELEASES data sets for a specific source and how to create a permanent data set named REL8 in the MyLib SAS library:

```
title 'Specifying the URL= Option to Create the REL8 Data Set';
libname _all_ clear;
libname mylib "U:\fred940\doc\";

libname fred8 sasefred "U:\fred940\test\"
   debug=on
   URL="http://api.stlouisfed.org/fred/source/releases?source_id=11"
   APIKEY='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
   ;

data mylib.rel8;
   set fred8.XFREDtpu;
run;

proc print
   data=mylib.rel8;
run;

proc contents
   data=mylib.rel8;
run;
```

Output 48.11.1 Specifying the URL= Option to Create the REL8 Data Set

Specifying the URL= Option to Create the REL8 Data Set

Obs	releases_ORDINAL	release_ORDINAL	release_id	release_realtime_start
1	1	1	72	2015-03-19
2	1	2	102	2015-03-19

Obs	release_realtime_end	release_name	release_press_release	release_link
1	2015-03-19	Daily Treasury Inflation-Indexed Securities	false	
2	2015-03-19	Wall Street Journal	true	http://online.wsj.com/public/us

Example 48.12: Series for a Specific Category with the URL= Option

The following statements demonstrate how to use the URL= option to obtain the SERIES data set for a specific category and how to create a permanent data set named SERIES_CAT5 in the MyLib SAS library:

```
title 'Specifying the URL= Option to Create the SERIES_CAT5 Data Set';
libname _all_ clear;
libname mylib "U:\fred940\doc\";

libname fred5 sasefred "U:\fred940\test\"
   debug=on
   URL="http://api.stlouisfed.org/fred/category/series?category_id=125"
   APIKEY='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
   ;

data mylib.series_cat5;
   set fred5.XFREDtpu;
run;

proc print
   data=mylib.series_cat5;
run;

proc contents
   data=mylib.series_cat5;
run;
```

The returned data are stored in the XFREDTPU data set and are copied to the permanent data set named *series_cat5.sas7bdat* in the MyLib library. A side effect of the DATA step is the automatic creation of two SAS data sets named *series.sas7bdat* and *seriess.sas7bdat* in the FRED5 library's location. Forty-five series are returned for the specified category ID, as shown in Output 48.12.1.

Output 48.12.1 Specifying the URL= Option to Create the SERIES_CAT5 Data Set

Specifying the URL= Option to Create the SERIES_CAT5 Data Set

Obs	seriess_ORDINAL	series_ORDINAL	series_id	series_realtime_start	series_realtime_end	series_title
1	1	1	BOPBCA	2015-03-19	2015-03-19	Balance on Current Account (Discontinued Series)
2	1	2	BOPBCAA	2015-03-19	2015-03-19	Balance on Current Account (Discontinued Series)
3	1	3	BOPBCAN	2015-03-19	2015-03-19	Balance on Current Account (Discontinued Series)
4	1	4	BOPBGS	2015-03-19	2015-03-19	Balance on Goods and Services (Discontinued Series)

Obs	series_observation_start	series_observation_end	series_frequency	series_frequency_short	series_units
1	1960-01-01	2014-01-01	Quarterly	Q	Billions of Dollars
2	1960-01-01	2013-01-01	Annual	A	Billions of Dollars
3	1960-01-01	2014-01-01	Quarterly	Q	Billions of Dollars
4	1960-01-01	2014-01-01	Quarterly	Q	Billions of Dollars

Obs	series_units_short	series_seasonal_adjustment	series_seasonal_adjustment_short	series_last_updated
1	Bil. of $	Seasonally Adjusted	SA	2014-06-18 08:41:28-05
2	Bil. of $	Not Seasonally Adjusted	NSA	2014-06-18 08:41:28-05
3	Bil. of $	Not Seasonally Adjusted	NSA	2014-06-18 08:41:28-05
4	Bil. of $	Seasonally Adjusted	SA	2014-06-18 08:41:28-05

Obs	series_popularity	series_notes
1	57	This series has been discontinued as a result of the comprehensive restructuring of the international economic accounts (http://www.bea.gov/international/modern.htm). For a crosswalk of the old and new series in FRED see: http://research.stlouisfed.org/CompRevisionReleaseID49.xlsx.
2	34	This series has been discontinued as a result of the comprehensive restructuring of the international economic accounts (http://www.bea.gov/international/modern.htm). For a crosswalk of the old and new series in FRED see: http://research.stlouisfed.org/CompRevisionReleaseID49.xlsx.
3	28	This series has been discontinued as a result of the comprehensive restructuring of the international economic accounts (http://www.bea.gov/international/modern.htm). For a crosswalk of the old and new series in FRED see: http://research.stlouisfed.org/CompRevisionReleaseID49.xlsx.
4	28	This series has been discontinued as a result of the comprehensive restructuring of the international economic accounts (http://www.bea.gov/international/modern.htm). For a crosswalk of the old and new series in FRED see: http://research.stlouisfed.org/CompRevisionReleaseID49.xlsx.

Output 48.12.1 *continued*

Specifying the URL= Option to Create the SERIES_CAT5 Data Set

Obs	seriess_ORDINAL	series_ORDINAL	series_id	series_realtime_start	series_realtime_end	series_title
5	1	5	BOPBGSA	2015-03-19	2015-03-19	Balance on Goods and Services (Discontinued Series)
6	1	6	BOPBGSN	2015-03-19	2015-03-19	Balance on Goods and Services (Discontinued Series)
7	1	7	BOPBII	2015-03-19	2015-03-19	Balance on Investment Income (Discontinued Series)
8	1	8	BOPBIIA	2015-03-19	2015-03-19	Balance on Investment Income (Discontinued Series)

Obs	series_observation_start	series_observation_end	series_frequency	series_frequency_short	series_units
5	1960-01-01	2013-01-01	Annual	A	Billions of Dollars
6	1960-01-01	2014-01-01	Quarterly	Q	Billions of Dollars
7	1960-01-01	2014-01-01	Quarterly	Q	Billions of Dollars
8	1960-01-01	2013-01-01	Annual	A	Billions of Dollars

Obs	series_units_short	series_seasonal_adjustment	series_seasonal_adjustment_short	series_last_updated
5	Bil. of $	Not Seasonally Adjusted	NSA	2014-06-18 08:41:28-05
6	Bil. of $	Not Seasonally Adjusted	NSA	2014-06-18 08:41:28-05
7	Bil. of $	Seasonally Adjusted	SA	2014-06-18 08:41:27-05
8	Bil. of $	Not Seasonally Adjusted	NSA	2014-06-18 08:41:27-05

Obs	series_popularity	series_notes
5	14	This series has been discontinued as a result of the comprehensive restructuring of the international economic accounts (http://www.bea.gov/international/modern.htm). For a crosswalk of the old and new series in FRED see: http://research.stlouisfed.org/CompRevisionReleaseID49.xlsx.
6	0	This series has been discontinued as a result of the comprehensive restructuring of the international economic accounts (http://www.bea.gov/international/modern.htm). For a crosswalk of the old and new series in FRED see: http://research.stlouisfed.org/CompRevisionReleaseID49.xlsx.
7	7	This series has been discontinued as a result of the comprehensive restructuring of the international economic accounts (http://www.bea.gov/international/modern.htm). For a crosswalk of the old and new series in FRED see: http://research.stlouisfed.org/CompRevisionReleaseID49.xlsx.
8	0	This series has been discontinued as a result of the comprehensive restructuring of the international economic accounts (http://www.bea.gov/international/modern.htm). For a crosswalk of the old and new series in FRED see: http://research.stlouisfed.org/CompRevisionReleaseID49.xlsx.

Output 48.12.1 *continued*

Specifying the URL= Option to Create the SERIES_CAT5 Data Set

Obs	seriess_ORDINAL	series_ORDINAL	series_id	series_realtime_start	series_realtime_end	series_title
9	1	9	BOPBIIN	2015-03-19	2015-03-19	Balance on Investment Income (Discontinued Series)
10	1	10	BOPBM	2015-03-19	2015-03-19	Balance on Merchandise Trade (Discontinued Series)
11	1	11	BOPBMA	2015-03-19	2015-03-19	Balance on Merchandise Trade (Discontinued Series)
12	1	12	BOPBMN	2015-03-19	2015-03-19	Balance on Merchandise Trade (Discontinued Series)

Obs	series_observation_start	series_observation_end	series_frequency	series_frequency_short	series_units
9	1960-01-01	2014-01-01	Quarterly	Q	Billions of Dollars
10	1960-01-01	2014-01-01	Quarterly	Q	Billions of Dollars
11	1960-01-01	2013-01-01	Annual	A	Billions of Dollars
12	1960-01-01	2014-01-01	Quarterly	Q	Billions of Dollars

Obs	series_units_short	series_seasonal_adjustment	series_seasonal_adjustment_short	series_last_updated
9	Bil. of $	Not Seasonally Adjusted	NSA	2014-06-18 08:41:27-05
10	Bil. of $	Seasonally Adjusted	SA	2014-06-18 08:41:27-05
11	Bil. of $	Not Seasonally Adjusted	NSA	2014-06-18 08:41:27-05
12	Bil. of $	Not Seasonally Adjusted	NSA	2014-06-18 08:41:27-05

Obs	series_popularity	series_notes
9	0	This series has been discontinued as a result of the comprehensive restructuring of the international economic accounts (http://www.bea.gov/international/modern.htm). For a crosswalk of the old and new series in FRED see: http://research.stlouisfed.org/CompRevisionReleaseID49.xlsx.
10	18	This series has been discontinued as a result of the comprehensive restructuring of the international economic accounts (http://www.bea.gov/international/modern.htm). For a crosswalk of the old and new series in FRED see: http://research.stlouisfed.org/CompRevisionReleaseID49.xlsx.
11	7	This series has been discontinued as a result of the comprehensive restructuring of the international economic accounts (http://www.bea.gov/international/modern.htm). For a crosswalk of the old and new series in FRED see: http://research.stlouisfed.org/CompRevisionReleaseID49.xlsx.
12	0	This series has been discontinued as a result of the comprehensive restructuring of the international economic accounts (http://www.bea.gov/international/modern.htm). For a crosswalk of the old and new series in FRED see: http://research.stlouisfed.org/CompRevisionReleaseID49.xlsx.

Output 48.12.1 *continued*

Specifying the URL= Option to Create the SERIES_CAT5 Data Set

Obs	seriess_ORDINAL	series_ORDINAL	series_id	series_realtime_start	series_realtime_end	series_title
13	1	13	BOPBSV	2015-03-19	2015-03-19	Balance on Services (Discontinued Series)
14	1	14	BOPBSVA	2015-03-19	2015-03-19	Balance on Services (Discontinued Series)
15	1	15	BOPBSVN	2015-03-19	2015-03-19	Balance on Services (Discontinued Series)
16	1	16	BOPCAT	2015-03-19	2015-03-19	Capital Account Transactions, Net (Discontinued Series)

Obs	series_observation_start	series_observation_end	series_frequency	series_frequency_short	series_units
13	1960-01-01	2014-01-01	Quarterly	Q	Billions of Dollars
14	1960-01-01	2013-01-01	Annual	A	Billions of Dollars
15	1960-01-01	2014-01-01	Quarterly	Q	Billions of Dollars
16	1989-10-01	2014-01-01	Quarterly	Q	Billions of Dollars

Obs	series_units_short	series_seasonal_adjustment	series_seasonal_adjustment_short	series_last_updated
13	Bil. of $	Seasonally Adjusted	SA	2014-06-18 08:41:27-05
14	Bil. of $	Not Seasonally Adjusted	NSA	2014-06-18 08:41:27-05
15	Bil. of $	Not Seasonally Adjusted	NSA	2014-06-18 08:41:27-05
16	Bil. of $	Seasonally Adjusted	SA	2014-06-18 08:41:26-05

Obs	series_popularity	series_notes
13	0	This series has been discontinued as a result of the comprehensive restructuring of the international economic accounts (http://www.bea.gov/international/modern.htm). For a crosswalk of the old and new series in FRED see: http://research.stlouisfed.org/CompRevisionReleaseID49.xlsx.
14	0	This series has been discontinued as a result of the comprehensive restructuring of the international economic accounts (http://www.bea.gov/international/modern.htm). For a crosswalk of the old and new series in FRED see: http://research.stlouisfed.org/CompRevisionReleaseID49.xlsx.
15	0	This series has been discontinued as a result of the comprehensive restructuring of the international economic accounts (http://www.bea.gov/international/modern.htm). For a crosswalk of the old and new series in FRED see: http://research.stlouisfed.org/CompRevisionReleaseID49.xlsx.
16	23	This series has been discontinued as a result of the comprehensive restructuring of the international economic accounts (http://www.bea.gov/international/modern.htm). For a crosswalk of the old and new series in FRED see: http://research.stlouisfed.org/CompRevisionReleaseID49.xlsx.

Output 48.12.1 *continued*

Specifying the URL= Option to Create the SERIES_CAT5 Data Set

Obs	seriess_ORDINAL	series_ORDINAL	series_id	series_realtime_start	series_realtime_end	series_title
17	1	17	BOPCATA	2015-03-19	2015-03-19	Capital Account Transactions, Net (Discontinued Series)
18	1	18	BOPCATN	2015-03-19	2015-03-19	Capital Account Transactions, Net (Discontinued Series)
19	1	19	BOPG	2015-03-19	2015-03-19	Unilateral Transfers, Net (Discontinued Series)
20	1	20	BOPGA	2015-03-19	2015-03-19	Unilateral Transfers, Net (Discontinued Series)

Obs	series_observation_start	series_observation_end	series_frequency	series_frequency_short	series_units
17	1989-01-01	2013-01-01	Annual	A	Billions of Dollars
18	1989-10-01	2014-01-01	Quarterly	Q	Billions of Dollars
19	1960-01-01	2014-01-01	Quarterly	Q	Billions of Dollars
20	1960-01-01	2013-01-01	Annual	A	Billions of Dollars

Obs	series_units_short	series_seasonal_adjustment	series_seasonal_adjustment_short	series_last_updated
17	Bil. of $	Not Seasonally Adjusted	NSA	2014-06-18 08:41:26-05
18	Bil. of $	Not Seasonally Adjusted	NSA	2014-06-18 08:41:26-05
19	Bil. of $	Seasonally Adjusted	SA	2014-06-18 08:41:26-05
20	Bil. of $	Not Seasonally Adjusted	NSA	2014-06-18 08:41:25-05

Obs	series_popularity	series_notes
17	0	This series has been discontinued as a result of the comprehensive restructuring of the international economic accounts (http://www.bea.gov/international/modern.htm). For a crosswalk of the old and new series in FRED see: http://research.stlouisfed.org/CompRevisionReleaseID49.xlsx.
18	0	This series has been discontinued as a result of the comprehensive restructuring of the international economic accounts (http://www.bea.gov/international/modern.htm). For a crosswalk of the old and new series in FRED see: http://research.stlouisfed.org/CompRevisionReleaseID49.xlsx.
19	27	This series has been discontinued as a result of the comprehensive restructuring of the international economic accounts (http://www.bea.gov/international/modern.htm). For a crosswalk of the old and new series in FRED see: http://research.stlouisfed.org/CompRevisionReleaseID49.xlsx.
20	7	This series has been discontinued as a result of the comprehensive restructuring of the international economic accounts (http://www.bea.gov/international/modern.htm). For a crosswalk of the old and new series in FRED see: http://research.stlouisfed.org/CompRevisionReleaseID49.xlsx.

Output 48.12.1 *continued*

Specifying the URL= Option to Create the SERIES_CAT5 Data Set

Obs	seriess_ORDINAL	series_ORDINAL	series_id	series_realtime_start	series_realtime_end	series_title
21	1	21	BOPGN	2015-03-19	2015-03-19	Unilateral Transfers, Net (Discontinued Series)
22	1	22	BOPGSTB	2015-03-19	2015-03-19	Trade Balance: Goods and Services, Balance of Payments Basis
23	1	23	BOPGTB	2015-03-19	2015-03-19	Trade Balance: Goods, Balance of Payments Basis
24	1	24	BOPSTB	2015-03-19	2015-03-19	Trade Balance: Services, Balance of Payments Basis
25	1	25	IEABC	2015-03-19	2015-03-19	Balance on current account

Obs	series_observation_start	series_observation_end	series_frequency	series_frequency_short	series_units
21	1960-01-01	2014-01-01	Quarterly	Q	Billions of Dollars
22	1992-01-01	2015-01-01	Monthly	M	Millions of Dollars
23	1992-01-01	2015-01-01	Monthly	M	Millions of Dollars
24	1992-01-01	2015-01-01	Monthly	M	Millions of Dollars
25	1999-01-01	2014-10-01	Quarterly	Q	Millions of Dollars

Obs	series_units_short	series_seasonal_adjustment	series_seasonal_adjustment_short	series_last_updated
21	Bil. of $	Not Seasonally Adjusted	NSA	2014-06-18 08:41:25-05
22	Mil. of $	Seasonally Adjusted	SA	2015-03-06 07:56:13-06
23	Mil. of $	Seasonally Adjusted	SA	2015-03-06 07:56:14-06
24	Mil. of $	Seasonally Adjusted	SA	2015-03-06 07:56:14-06
25	Mil. of $	Seasonally Adjusted	SA	2015-03-19 08:36:18-05

Obs	series_popularity	series_notes
21	0	This series has been discontinued as a result of the comprehensive restructuring of the international economic accounts (http://www.bea.gov/international/modern.htm). For a crosswalk of the old and new series in FRED see: http://research.stlouisfed.org/CompRevisionReleaseID49.xlsx.
22	65	
23	40	
24	21	
25	43	Calculated by subtracting the imports of goods and services and income payments (debits) from the exports of goods and services and income receipts (credits)

Output 48.12.1 *continued*

Specifying the URL= Option to Create the SERIES_CAT5 Data Set

Obs	seriess_ORDINAL	series_ORDINAL	series_id	series_realtime_start	series_realtime_end	series_title
26	1	26	IEABCA	2015-03-19	2015-03-19	Balance on current account
27	1	27	IEABCG	2015-03-19	2015-03-19	Balance on goods
28	1	28	IEABCGA	2015-03-19	2015-03-19	Balance on goods
29	1	29	IEABCGN	2015-03-19	2015-03-19	Balance on goods
30	1	30	IEABCGS	2015-03-19	2015-03-19	Balance on goods and services

Obs	series_observation_start	series_observation_end	series_frequency	series_frequency_short	series_units
26	1999-01-01	2014-01-01	Annual	A	Millions of Dollars
27	1999-01-01	2014-10-01	Quarterly	Q	Millions of Dollars
28	1999-01-01	2014-01-01	Annual	A	Millions of Dollars
29	1999-01-01	2014-10-01	Quarterly	Q	Millions of Dollars
30	1999-01-01	2014-10-01	Quarterly	Q	Millions of Dollars

Obs	series_units_short	series_seasonal_adjustment	series_seasonal_adjustment_short	series_last_updated
26	Mil. of $	Not Seasonally Adjusted	NSA	2015-03-19 08:56:09-05
27	Mil. of $	Seasonally Adjusted	SA	2015-03-19 08:36:18-05
28	Mil. of $	Not Seasonally Adjusted	NSA	2015-03-19 08:56:09-05
29	Mil. of $	Not Seasonally Adjusted	NSA	2015-03-19 08:36:18-05
30	Mil. of $	Seasonally Adjusted	SA	2015-03-19 08:36:17-05

Obs	series_popularity	series_notes
26	24	Calculated by subtracting the imports of goods and services and income payments (debits) from the exports of goods and services and income receipts (credits)
27	11	Calculated by subtracting the imports of goods from the exports of goods
28	0	Calculated by subtracting the imports of goods from the exports of goods
29	11	Calculated by subtracting the imports of goods from the exports of goods
30	22	Calculated by subtracting the imports of goods and services from the exports of goods and services

Output 48.12.1 *continued*

Specifying the URL= Option to Create the SERIES_CAT5 Data Set

Obs	seriess_ORDINAL	series_ORDINAL	series_id	series_realtime_start	series_realtime_end	series_title
31	1	31	IEABCGSA	2015-03-19	2015-03-19	Balance on goods and services
32	1	32	IEABCGSN	2015-03-19	2015-03-19	Balance on goods and services
33	1	33	IEABCN	2015-03-19	2015-03-19	Balance on current account
34	1	34	IEABCP	2015-03-19	2015-03-19	Balance on capital account
35	1	35	IEABCPA	2015-03-19	2015-03-19	Balance on capital account

Obs	series_observation_start	series_observation_end	series_frequency	series_frequency_short	series_units
31	1999-01-01	2014-01-01	Annual	A	Millions of Dollars
32	1999-01-01	2014-10-01	Quarterly	Q	Millions of Dollars
33	1999-01-01	2014-10-01	Quarterly	Q	Millions of Dollars
34	1999-01-01	2014-10-01	Quarterly	Q	Millions of Dollars
35	1999-01-01	2014-01-01	Annual	A	Millions of Dollars

Obs	series_units_short	series_seasonal_adjustment	series_seasonal_adjustment_short	series_last_updated
31	Mil. of $	Not Seasonally Adjusted	NSA	2015-03-19 08:56:08-05
32	Mil. of $	Not Seasonally Adjusted	NSA	2015-03-19 08:36:17-05
33	Mil. of $	Not Seasonally Adjusted	NSA	2015-03-19 08:36:17-05
34	Mil. of $	Seasonally Adjusted	SA	2015-03-19 08:36:17-05
35	Mil. of $	Not Seasonally Adjusted	NSA	2015-03-19 08:56:08-05

Obs	series_popularity	series_notes
31	0	Calculated by subtracting the imports of goods and services from the exports of goods and services
32	0	Calculated by subtracting the imports of goods and services from the exports of goods and services
33	24	Calculated by subtracting the imports of goods and services and income payments (debits) from the exports of goods and services and income receipts (credits)
34	35	Calculated by subtracting the capital transfer payments and other debits from the capital transfer receipts and other credits
35	30	Calculated by subtracting the capital transfer payments and other debits from the capital transfer receipts and other credits

Output 48.12.1 *continued*

Specifying the URL= Option to Create the SERIES_CAT5 Data Set

Obs	seriess_ORDINAL	series_ORDINAL	series_id	series_realtime_start	series_realtime_end	series_title
36	1	36	IEABCPI	2015-03-19	2015-03-19	Balance on primary income
37	1	37	IEABCPIA	2015-03-19	2015-03-19	Balance on primary income
38	1	38	IEABCPIN	2015-03-19	2015-03-19	Balance on primary income
39	1	39	IEABCPN	2015-03-19	2015-03-19	Balance on capital account
40	1	40	IEABCS	2015-03-19	2015-03-19	Balance on services

Obs	series_observation_start	series_observation_end	series_frequency	series_frequency_short	series_units
36	1999-01-01	2014-10-01	Quarterly	Q	Millions of Dollars
37	1999-01-01	2014-01-01	Annual	A	Millions of Dollars
38	1999-01-01	2014-10-01	Quarterly	Q	Millions of Dollars
39	1999-01-01	2014-10-01	Quarterly	Q	Millions of Dollars
40	1999-01-01	2014-10-01	Quarterly	Q	Millions of Dollars

Obs	series_units_short	series_seasonal_adjustment	series_seasonal_adjustment_short	series_last_updated
36	Mil. of $	Seasonally Adjusted	SA	2015-03-19 08:36:17-05
37	Mil. of $	Not Seasonally Adjusted	NSA	2015-03-19 08:56:08-05
38	Mil. of $	Not Seasonally Adjusted	NSA	2015-03-19 08:36:17-05
39	Mil. of $	Not Seasonally Adjusted	NSA	2015-03-19 08:36:17-05
40	Mil. of $	Seasonally Adjusted	SA	2015-03-19 08:36:17-05

Obs	series_popularity	series_notes
36	7	Calculated by subtracting the primary income payments from the primary income receipts
37	14	Calculated by subtracting the primary income payments from the primary income receipts
38	7	Calculated by subtracting the primary income payments from the primary income receipts
39	20	Calculated by subtracting the capital transfer payments and other debits from the capital transfer receipts and other credits
40	11	Calculated by subtracting the imports of services from the exports of services

Output 48.12.1 *continued*

Specifying the URL= Option to Create the SERIES_CAT5 Data Set

Obs	seriess_ORDINAL	series_ORDINAL	series_id	series_realtime_start	series_realtime_end	series_title
41	1	41	IEABCSA	2015-03-19	2015-03-19	Balance on services
42	1	42	IEABCSI	2015-03-19	2015-03-19	Balance on secondary income
43	1	43	IEABCSIA	2015-03-19	2015-03-19	Balance on secondary income
44	1	44	IEABCSIN	2015-03-19	2015-03-19	Balance on secondary income
45	1	45	IEABCSN	2015-03-19	2015-03-19	Balance on services

Obs	series_observation_start	series_observation_end	series_frequency	series_frequency_short	series_units
41	1999-01-01	2014-01-01	Annual	A	Millions of Dollars
42	1999-01-01	2014-10-01	Quarterly	Q	Millions of Dollars
43	1999-01-01	2014-01-01	Annual	A	Millions of Dollars
44	1999-01-01	2014-10-01	Quarterly	Q	Millions of Dollars
45	1999-01-01	2014-10-01	Quarterly	Q	Millions of Dollars

Obs	series_units_short	series_seasonal_adjustment	series_seasonal_adjustment_short	series_last_updated
41	Mil. of $	Not Seasonally Adjusted	NSA	2015-03-19 08:56:08-05
42	Mil. of $	Seasonally Adjusted	SA	2015-03-19 08:36:17-05
43	Mil. of $	Not Seasonally Adjusted	NSA	2015-03-19 08:56:08-05
44	Mil. of $	Not Seasonally Adjusted	NSA	2015-03-19 08:36:17-05
45	Mil. of $	Not Seasonally Adjusted	NSA	2015-03-19 08:36:17-05

Obs	series_popularity	series_notes
41	0	Calculated by subtracting the imports of services from the exports of services
42	0	Calculated by subtracting the secondary income (current transfer) payments from the secondary income (current transfer) receipts
43	0	Calculated by subtracting the secondary income (current transfer) payments from the secondary income (current transfer) receipts
44	0	Calculated by subtracting the secondary income (current transfer) payments from the secondary income (current transfer) receipts
45	0	Calculated by subtracting the imports of services from the exports of services

Example 48.13: Sources for Today's Date with the URL= Option

The following statements demonstrate how to use the URL= option to obtain the first 10 sources (LIMIT=10) for the SOURCES6 data set for today's date and how to create a permanent data set named SOURCES6 in the MyLib SAS library:

```
title 'Specifying the URL= Option to Create the SOURCES6 Data Set';
libname _all_ clear;
libname mylib "U:\fred940\doc\";

libname fred6 sasefred "U:\fred940\test\"
   debug=on
   URL="http://api.stlouisfed.org/fred/sources?limit=10"
   APIKEY='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
   ;

data mylib.sources6;
   set fred6.XFREDtpu;
run;

proc print
   data=mylib.sources6;
run;

proc contents
   data=mylib.sources6;
run;
```

The returned data are stored in the XFREDTPU data set and are copied to the permanent data set named *sources6.sas7bdat* in the MyLib library. A side effect of the DATA step is the automatic creation of two SAS data sets named *source.sas7bdat* and *sources.sas7bdat* in the FRED6 library's location. Sixty-eight sources could be returned for today's date, but the LIMIT=10 option obtains only the first 10 sources, as shown in Output 48.13.1.

Output 48.13.1 Specifying the URL= Option to Create the SOURCES6 Data Set

Specifying the URL= Option to Create the SOURCES6 Data Set

Obs	sources_ORDINAL	source_ORDINAL	source_id	source_realtime_start	source_realtime_end
1	1	1	1	2015-03-19	2015-03-19
2	1	2	3	2015-03-19	2015-03-19
3	1	3	4	2015-03-19	2015-03-19
4	1	4	6	2015-03-19	2015-03-19
5	1	5	11	2015-03-19	2015-03-19
6	1	6	13	2015-03-19	2015-03-19
7	1	7	14	2015-03-19	2015-03-19
8	1	8	15	2015-03-19	2015-03-19
9	1	9	16	2015-03-19	2015-03-19
10	1	10	17	2015-03-19	2015-03-19

Obs	source_name	source_link
1	Board of Governors of the Federal Reserve System (US)	http://www.federalreserve.gov/
2	Federal Reserve Bank of Philadelphia	http://www.philadelphiafed.org/
3	Federal Reserve Bank of St. Louis	http://www.stlouisfed.org/
4	Federal Financial Institutions Examination Council (US)	http://www.ffiec.gov/
5	Dow Jones & Company	http://www.dowjones.com
6	Institute for Supply Management	http://www.ism.ws/
7	Thomson Reuters/University of Michigan	https://customers.reuters.com/community/university/default.aspx
8	Council of Economic Advisers (US)	http://www.whitehouse.gov/cea/
9	US. Office of Management and Budget	http://www.whitehouse.gov/omb/
10	US. Congressional Budget Office	http://www.cbo.gov/

Example 48.14: Releases Available for Today's Date with the URL= Option

The following statements demonstrate how to use the URL= option to obtain the first 10 observations (LIMIT=10) of the REL3 data set for today's date and how to create a permanent data set named REL3 in the MyLib SAS library:

```
title 'Specifying the URL= Option to Create the REL3 Data Set';
libname _all_ clear;
libname mylib "U:\fred940\doc\";

libname fred3 sasefred "U:\fred940\test\"
   debug=on
   URL="http://api.stlouisfed.org/fred/releases?limit=10"
   APIKEY='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
   ;

data mylib.rel3;
   set fred3.XFREDtpu;
run;

proc print
   data=mylib.rel3;
run;

proc contents
   data=mylib.rel3;
run;
```

The returned data are stored in the XFREDTPU data set and are copied to the permanent data set named *rel3.sas7bdat* in the MyLib library. A side effect of the DATA step is the automatic creation of two SAS data sets named *release.sas7bdat* and *releases.sas7bdat* in the FRED3 library's location. More than 200 available releases could be returned for today, but the LIMIT=10 option obtains only the first 10 releases, as shown in Output 48.14.1.

Output 48.14.1 Specifying the URL= Option to Create the REL3 Data Set

Specifying the URL= Option to Create the REL3 Data Set

Obs	releases_ORDINAL	release_ORDINAL	release_id	release_realtime_start	release_realtime_end
1	1	1	9	2015-03-19	2015-03-19
2	1	2	10	2015-03-19	2015-03-19
3	1	3	11	2015-03-19	2015-03-19
4	1	4	13	2015-03-19	2015-03-19
5	1	5	14	2015-03-19	2015-03-19
6	1	6	15	2015-03-19	2015-03-19
7	1	7	17	2015-03-19	2015-03-19
8	1	8	18	2015-03-19	2015-03-19
9	1	9	19	2015-03-19	2015-03-19
10	1	10	20	2015-03-19	2015-03-19

Obs	release_name	release_press_release	release_link
1	Advance Monthly Sales for Retail and Food Services	true	http://www.census.gov/retail/
2	Consumer Price Index	true	http://www.bls.gov/cpi/
3	Employment Cost Index	true	http://www.bls.gov/ncs/ect/
4	G.17 Industrial Production and Capacity Utilization	true	http://www.federalreserve.gov/releases/g17/
5	G.19 Consumer Credit	true	http://www.federalreserve.gov/releases/g19/
6	G.5 Foreign Exchange Rates	true	http://www.federalreserve.gov/releases/g5/
7	H.10 Foreign Exchange Rates	true	http://www.federalreserve.gov/releases/h10/
8	H.15 Selected Interest Rates	true	http://www.federalreserve.gov/releases/h15/
9	H.3 Aggregate Reserves of Depository Institutions and the Monetary Base	true	http://www.federalreserve.gov/releases/h3/
10	H.4.1 Factors Affecting Reserve Balances	true	http://www.federalreserve.gov/releases/h41/

References

Federal Reserve Bank of St. Louis (2012). "Economic Research." Accessed November 7, 2012. http://research.stlouisfed.org/.

PostgreSQL Global Development Group (2013). "PostgreSQL 8.4.17 Documentation." http://www.postgresql.org/docs/8.4/static/queries-limit.html.

Chapter 49
The SASEHAVR Interface Engine

Contents

Overview: SASEHAVR Interface Engine **3432**
Getting Started: SASEHAVR Interface Engine **3432**
Setting Up the Haver Analytics DLX Application Programming Interface 3432
Structure of a SAS Data Set That Contains Time Series Data 3433
Reading and Converting Haver DLX Time Series 3433
Using the SAS DATA Step 3433
Using the SAS Windowing Environment 3434
Syntax: SASEHAVR Interface Engine **3434**
LIBNAME *libref* SASEHAVR Statement 3435
Details: SASEHAVR Interface Engine **3439**
SAS Output Data Set 3439
Mapping Haver Frequencies to SAS Time Intervals 3440
Error Recovery for the SASEHAVR Interface Engine 3440
Data Elements Reference: Haver Analytics DLX Database Profile 3444
Examples: SASEHAVR Interface Engine **3452**
Example 49.1: Examining the Contents of a Haver Database 3452
Example 49.2: Viewing Quarterly Time Series from a Haver Database 3453
Example 49.3: Viewing Monthly Time Series from a Haver Database 3454
Example 49.4: Viewing Weekly Time Series from a Haver Database 3455
Example 49.5: Viewing Daily Time Series from a Haver Database 3456
Example 49.6: Limiting the Range of Time Series from a Haver Database 3457
Example 49.7: Using the WHERE Statement to Subset Time Series from a Haver Database 3458
Example 49.8: Using the KEEP Option to Subset Time Series from a Haver Database 3460
Example 49.9: Using the SOURCE Option to Subset Time Series from a Haver Database 3462
Example 49.10: Using the GROUP Option to Subset Time Series from a Haver Database 3464
Example 49.11: Using the OUTSELECT=ON Option to View the Key Selection Variables in a Haver Database 3470
Example 49.12: Selecting Variables Based on Short Source Key Code 3471
Example 49.13: Selecting Variables Based on Geography Key Codes 3473
References **3478**

Overview: SASEHAVR Interface Engine

The SASEHAVR interface engine is a seamless interface between Haver Analytics and SAS data processing that enables SAS users to read economic and financial time series data that reside in a Haver Analytics DLX (Data Link Express) database. The Haver Analytics DLX economic and financial database offerings include U.S. economic indicators, specialized databases, and financial indicators; data about industry, industrial countries, emerging markets, and international organizations; forecasts and as-reported data; and data about U.S. regional services. For more information, see the section "Data Elements Reference: Haver Analytics DLX Database Profile" on page 3444.

The SASEHAVR engine uses the LIBNAME statement to enable you to specify how to subset your Haver data and how to aggregate the selected time series at the same frequency. You can then use the SAS DATA step to perform further subsetting and to store the resulting time series in a SAS data set. You can perform more analysis (if desired) either in the same SAS session or in a later session.

The SASEHAVR engine supports both 32-bit and 64-bit Windows hosts. Haver Analytics supplies two versions of the DLX application programming interface (API), one for 32-bit applications (dlxapi32.dll) and one for 64-bit applications (dlxapi64.dll). Choose the appropriate application, either 32-bit or 64-bit, for your platform. You can follow the instructions for setting up your installation of the Haver API in the section "Setting Up the Haver Analytics DLX Application Programming Interface" on page 3432.

Getting Started: SASEHAVR Interface Engine

Setting Up the Haver Analytics DLX Application Programming Interface

If this is your first time using the SASEHAVR interface engine on your Windows machine, then it is necessary to follow these setup instructions. If you have already used the SASEHAVR interface, then just check the file version number of your already installed dlxapi32.dll (or dlxapi64.dll). For 32-bit installations, the file version is 1.1.9.0, and for 64-bit installations, the file version is 2.0.0.1. The Haver API version number appears in the SAS log the first time you assign a SASEHAVR libref. In Windows Explorer, you can see a file's properties, including its version number, by hovering the mouse pointer over the file icon. Alternatively, you can right-click on the file icon to bring up the properties and click the **Details** tab to see the version number.

To set up the Haver Analytics API on your machine, visit the SAS Technical Support download site at the following URL:

http://ftp.sas.com/techsup/download/base/

First, create a folder on your system drive (usually designated as C:), and name the folder HAVER. Create an environment variable named HAVER as follows:

```
HAVER=C:\HAVER\
```

If your SAS system is 32-bit, then download the files dlxapi32.h, dlxapi32.dll, and dlxapi32.lib to your HAVER folder. If your SAS system is 64-bit, then download the files dlxapi64.h, dlxapi64.dll, and dlxapi64.lib

to your HAVER folder. Second, prepend the location of your HAVER folder to the system environment variable (%PATH%) as follows, so that the SASEHAVR engine can find your downloaded Haver API files:

```
PATH=C:\HAVER\;%PATH%
```

Reboot your system to complete the Haver API setup.

Structure of a SAS Data Set That Contains Time Series Data

SAS represents time series data in a two-dimensional array called a SAS data set whose columns correspond to series variables and whose rows correspond to measurements of these variables at certain time periods. The time periods at which observations are recorded can be included in the data set as time ID variables. The SASEHAVR engine provides a time ID variable called DATE. The DATE variable can be represented in any of the time intervals shown in the section "Mapping Haver Frequencies to SAS Time Intervals" on page 3440.

Reading and Converting Haver DLX Time Series

The SASEHAVR engine supports reading and converting all selected time series that reside in Haver DLX databases. The SASEHAVR engine enables you to limit the range of data by specifying the START= and END= options in the LIBNAME statement. Start dates and end dates are recommended to help save resources when you are processing large databases or a large number of observations.

The SASEHAVR engine enables you to convert or aggregate all selected time series to a desired frequency. By default, the SASEHAVR engine selects the time series variables that match the frequency of the first selected variable. To select variables of one specific frequency, use the FREQ= option. If no selection criteria are specified, the first selected variable is the first physical DLX record read from the Haver database. To force aggregation of all selected variables to the frequency specified by the FREQ= option, use the FORCE=FREQ option. The AGGMODE= option enables you to specify a strict or relaxed aggregation method; by default, AGGMODE=RELAXED. Aggregation is supported only from a more frequent time interval to a less frequent time interval, such as from weekly to monthly. If a conversion to a more frequent frequency is attempted, all missing values are returned by the Haver DLX API. For more information, see the section "Aggregating to Quarterly Frequency Using the FORCE=FREQ Option" on page 3443. The FORCE= option is ignored if the FREQ= option is not specified.

Using the SAS DATA Step

If desired, you can store your selected time series in a SAS data set by using the SAS DATA step. You can further subset your data by using the WHERE, KEEP, or DROP statement in your DATA step.

For more efficient subsetting of time series by Haver variables, Haver groups, Haver sources, Haver short sources, Haver long sources, or Haver geographic codes, you can use the corresponding KEEP=, GROUP=, SOURCE=, SHORTSOURCE=, LONGSOURCE=, GEOGCODE1=, or GEOGCODE2= option in the LIBNAME *libref* SASEHAVR statement. To see the available Haver selection key values, including geographic codes, short sources, and long sources for your database, specify the OUTSELECT=ON option.

From the OUTSELECT= option output, you can use convenient wildcard symbols to create the selection list for your next LIBNAME *libref* SASEHAVR statement.

There are three wildcard symbols: '*', '?' and '#'. The '*' wildcard corresponds to any character string and includes any string pattern that corresponds to that position in the matching variable name. The '?' stands for any single alphanumeric character. Lastly, the '#' wildcard corresponds to a single numeric character.

You can also deselect time series by Haver variables, by Haver groups, by Haver sources, by Haver short sources, by Haver long sources, or by Haver geographic codes, by using the corresponding DROP=, DROPGROUP=, DROPSOURCE=, DROPSHORT=, DROPLONG=, DROPGEOG1=, or DROPGEOG2= option. These options also support wildcards.

After your selected data is stored in a SAS data set, you can use it as you would any other SAS data set.

Using the SAS Windowing Environment

You can see the available data sets in the SAS LIBNAME window of the SAS windowing environment by selecting the SASEHAVR *libref* in the LIBNAME window that you have previously defined in your LIBNAME statement. You can view your SAS output observations by double-clicking on the desired output data set *libref* in the LIBNAME window of the SAS windowing environment. You can type **Viewtable** on the SAS command line to view your SASEHAVR tables, views, or librefs.

Before you use **Viewtable**, it is recommended that you store your output data sets in a physical folder or library that is separate from the folder or library used for your input databases. (The default location for output data sets is the SAS Work library.) If you do not follow this guideline, you will receive the following error message for each input database that does not have the selected options in the SASEHAVR *libref* that you double-clicked:

```
ERROR: No variable selected with current options.
```

Syntax: SASEHAVR Interface Engine

The SASEHAVR engine uses standard engine syntax. Table 49.1 summarizes the options used in the LIBNAME *libref* SASEHAVR statement.

Table 49.1 Summary of LIBNAME *libref* SASEHAVR Statement Options

Option	Description
FREQUENCY=	Specifies the Haver frequency
START=	Specifies a Haver start date to limit the selection of time series to those that begin with the specified date
END=	Specifies a Haver end date to limit the selection of time series to those that end with the specified date
KEEP=	Specifies a list of comma-delimited Haver variables to keep in the output SAS data set

Table 49.1 *continued*

Option	Description
DROP=	Specifies a list of comma-delimited Haver variables to drop from the output SAS data set
GROUP=	Specifies a list of comma-delimited Haver groups to keep in the output SAS data set
DROPGROUP=	Specifies a list of comma-delimited Haver groups to drop from the output SAS data set
SOURCE=	Specifies a list of comma-delimited Haver sources to keep in the output SAS data set
DROPSOURCE=	Specifies a list of comma-delimited Haver sources to drop from the output SAS data set
SHORT=	Specifies a list of comma-delimited Haver short sources to keep in the output SAS data set
DROPSHORT=	Specifies a list of comma-delimited Haver short sources to drop from the output SAS data set
LONG=	Specifies a list of comma-delimited Haver long sources to keep in the output SAS data set
DROPLONG=	Specifies a list of comma-delimited Haver long sources to drop from the output SAS data set
GEOG1=	Specifies a list of comma-delimited Haver geography1 codes to keep in the output SAS data set
DROPGEOG1=	Specifies a list of comma-delimited Haver geography1 codes to drop from the output SAS data set
GEOG2=	Specifies a list of comma-delimited Haver geography2 codes to keep in the output SAS data set
DROPGEOG2=	Specifies a list of comma-delimited Haver geography2 codes to drop from the output SAS data set
OUTSELECT=	Specifies what values the output data is to contain
FORCE=FREQ	Specifies that all selected time series variables be aggregated to the frequency specified in the FREQ= option
AGGMODE=	Specifies the aggregation method used for aggregating time series (STRICT or RELAXED)

LIBNAME *libref* SASEHAVR Statement

LIBNAME *libref* **sasehavr** '*physical name*' *options* **;**

The '*physical name*' specifies the location of the folder where your Haver DLX database resides.

You can use the following options in the LIBNAME *libref* SASEHAVR statement:

FREQ=*haver_frequency*

FREQUENCY=*haver_frequency*

INTERVAL=*haver_frequency*

specifies the Haver frequency. All Haver frequencies are supported by the SASEHAVR engine. Accepted frequency values are annual, year, yearly, quarter, quarterly, qtr, monthly, month, mon, week.1, week.2, week.3, week.4, week.5, week.6, week.7, weekly, week, daily, and day.

START=*start_date*

STARTDATE=*start_date*

STDATE=*start_date*

BEGIN=*start_date*

specifies the start date for the time series in the form YYYYMMDD.

END=*end_date*

ENDDATE=*end_date*

ENDATE=*end_date*

specifies the end date for the time series in the form YYYYMMDD.

KEEP="*haver_variable_list*"

specifies the list of Haver variables to be included in the output SAS data set. This list is comma-delimited and must be surrounded by quotes "".

DROP="*haver_variable_list*"

specifies the list of Haver variables to be excluded from the output SAS data set. This list is comma-delimited and must be surrounded by quotes "".

GROUP="*haver_group_list*"

KEEPGROUP="*haver_group_list*"

specifies the list of Haver groups to be included in the output SAS data set. This list is comma-delimited and must be surrounded by quotes "".

DROPGROUP="*haver_group_list*"

specifies the list of Haver groups to be excluded from the output SAS data set. This list is comma-delimited and must be surrounded by quotes "".

SOURCE="*haver_source_list*"

KEEPSOURCE="*haver_source_list*"

specifies the list of Haver sources to be included in the output SAS data set. This list is comma-delimited and must be surrounded by quotes "".

DROPSOURCE="*haver_source_list*"

specifies the list of Haver sources to be excluded from the output SAS data set. This list is comma-delimited and must be surrounded by quotes "".

SHORT="*haver_shortsource_list*"

KEEPSHORT="*haver_shortsource_list*"

SHORTSOURCE="*haver_shortsource_list*"

specifies the list of Haver short sources to be included in the output SAS data set. This list is comma-delimited and must be surrounded by quotes "".

DROPSHORT="*haver_shortsource_list*"

DROPSHORTSOURCE="*haver_shortsource_list*"

specifies the list of Haver short sources to be excluded from the output SAS data set. This list is comma-delimited and must be surrounded by quotes "".

LONG="*haver_longsource_list*"

KEEPLONG="*haver_longsource_list*"

LONGSOURCE="*haver_longsource_list*"

specifies the list of Haver long sources to be included in the output SAS data set. This list is comma-delimited and must be surrounded by quotes "".

DROPLONG="*haver_longsource_list*"

DROPLONGSOURCE="*haver_longsource_list*"

specifies the list of Haver long sources to be excluded from the output SAS data set. This list is comma-delimited and must be surrounded by quotes "".

GEOG1="*haver_geographycode1_list*"

KEEPGEOG1="*haver_geographycode1_list*"

GEOGCODE1="*haver_geographycode1_list*"

specifies the list of Haver geography1 codes to be included in the output SAS data set. This list is comma-delimited and must be surrounded by quotes "".

DROPGEOG1="*haver_geographycode1_list*"

DROPGEOGCODE1="*haver_geographycode1_list*"

specifies the list of Haver geography1 codes to be excluded from the output SAS data set. This list is comma-delimited and must be surrounded by quotes "".

GEOG2="*haver_geographycode2_list*"

KEEPGEOG2="*haver_geographycode2_list*"

GEOGCODE2="*haver_geographycode2_list*"

specifies the list of Haver geography2 codes to be included in the output SAS data set. This list is comma-delimited and must be surrounded by quotes "".

DROPGEOG2="*haver_geographycode2_list*"

DROPGEOGCODE2="*haver_geographycode2_list*"

specifies the list of Haver geography2 codes to be excluded from the output SAS data set. This list is comma-delimited and must be surrounded by quotes "".

OUTSELECT=ON | OFF

specifies what the output data set shows. OUTSELECT=ON specifies that the output data set show values of selection keys (such as geography codes, groups, sources, short sources, and long sources) for each selected variable name (time series) in the database. OUTSELECT=OFF specifies that the output data set show the observations in the range for all selected time series. The default is OUTSELECT=OFF.

AGGMODE=STRICT | RELAXED

specifies whether the SASEHAVR engine uses a strict or relaxed aggregation method when converting time series from a higher to lower frequency.

A strict aggregation method returns a missing value whenever there is a missing observation in a time period. For instance, if a monthly time series has a missing value for the month of February, 2005, then attempting to aggregate to a quarterly frequency results in a missing value for the first quarter of 2005. The SAS log reports the status of this option.

When a relaxed aggregation method is used, some observations can be missing, but the relaxed method returns an aggregated value calculated from the nonmissing data points according to the series aggregation type (average, sum, or end of period). Average type only needs one valid (nonmissing) data point to calculate the average. Sum type needs all the data points to be available in order to sum the values. End of period type calculates the end of period value if there is at least one valid (nonmissing) data point in the aggregated span. It returns the last available valid data point in the aggregated span. The default is AGGMODE=RELAXED.

FORCE=FREQ

specifies that the selected variables be aggregated to the frequency in the FREQ= option. Aggregation is supported only from a more frequent time interval to a less frequent time interval, such as from weekly to monthly. See the section "Aggregating to Quarterly Frequency Using the FORCE=FREQ Option" on page 3443 for sample output and suggested error recovery from attempting a conversion that yields missing values when a higher frequency conversion is specified. This option is ignored if the FREQ= option is not set. For a more complete discussion of Haver frequencies and SAS time intervals, see the section "Mapping Haver Frequencies to SAS Time Intervals" on page 3440.

Following is an example of the LIBNAME *libref* SASEHAVR statement:

```
LIBNAME libref sasehavr 'physical-name'
        FREQ=MONTHLY;
```

By default, the SASEHAVR engine reads all time series in the Haver database that you reference by *libref*. The *start_date* is specified in the form YYYYMMDD. The start date is used to delimit the data to a specified start date.

For example, to read the time series in the TEST library starting on July 4, 1996, specify the following statement:

```
LIBNAME test sasehavr 'physical-name'
        STARTDATE=19960704;
```

When you use the START= option, you limit the range of observations that are read from the time series and that are converted to the desired frequency. Start dates can help save resources when processing large databases or when processing a large number of observations. It is also possible to select specific variables to be included or excluded from the SAS data set by using the KEEP= or the DROP= option, respectively.

```
LIBNAME test sasehavr 'physical-name'
        KEEP="ABC*, XYZ??";

LIBNAME test sasehavr 'physical-name'
        DROP="*SC*, #T#";
```

When the KEEP= or the DROP= option is used, the resulting SAS data set keeps or drops the variables that you select in that option. Three wildcards are available: '*', '?' and '#'. The '*' wildcard corresponds to any character string and includes any string pattern that corresponds to that position in the matching variable name. The '?' means that any single alphanumeric character is valid. The '#' wildcard corresponds to a single numeric character. You can also select time series in your data by using the GROUP=, SOURCE=, SHORT=, LONG=, GEOG1=, or the GEOG2= option to select on group name, source name, short source name, long source name, geography1 code, or the geography2 code, respectively. Alternatively, you can deselect time series by using the DROPGROUP=, DROPSOURCE=, DROPSHORT=, DROPLONG=, DROPGEOG1=, or the DROPGEOG2= option, respectively.

Following are examples that perform variable selection (or deselection) based on groups or sources:

```
LIBNAME test sasehavr 'physical-name'
        GROUP="CBA, *ZYX";

LIBNAME test sasehavr 'physical-name'
        DROPGROUP="TKN*, XCZ?";

LIBNAME test sasehavr 'physical-name'
        SOURCE="FRB";

LIBNAME test sasehavr 'physical-name'
        DROPSOURCE="NYSE";
```

SASEHAVR selects only the variables that are of the specified frequency in the FREQ= option. If this option is not specified, SASEHAVR selects the variables that match the frequency of the first selected variable. If no other selection criteria are specified, by default the first selected variable is the first physical DLX record read from the Haver database. You can specify the FORCE=FREQ option to force the aggregation of all variables selected to be of the frequency specified in the FREQ= option. Aggregation is supported only from a more frequent time interval to a less frequent time interval, such as from weekly to monthly. See the section "Aggregating to Quarterly Frequency Using the FORCE=FREQ Option" on page 3443 for suggested recovery from using a frequency that does not aggregate the data appropriately. The FORCE= option is ignored if the FREQ= option is not specified. The AGGMODE= STRICT option is used when a strict aggregation method is desired. The default value for AGGMODE is RELAXED, the same method that was used in prior releases of SASEHAVR.

Details: SASEHAVR Interface Engine

SAS Output Data Set

You can use the SAS DATA step to write the Haver converted series to a SAS data set so that you can easily analyze the data using the SAS System. You can specify the name of the output data set in the DATA statement. This causes the engine supervisor to create a SAS data set with the specified name in either the SAS Work library, or if specified, the Sasuser library.

When OUTSELECT=OFF (the default), the contents of the SAS data set include the date of each observation, the name of each series read from the Haver database, and the label or Haver description of each series. Missing values are represented as '.' in the SAS data set. You can use the PRINT procedure and the CONTENTS procedure to print your output data set and its contents. You can use the SQL procedure along with the SASEHAVR engine to create a view of your SAS data set.

The DATE variable in the SAS data set contains the date of the observation. The SASEHAVR engine automatically maps the Haver intervals to the appropriate corresponding SAS intervals.

When OUTSELECT=ON, the OUT= data set does not contain the observations of all time series. Instead, each observation contains the name of the time series, the source of the time series, the geography1 code, the geography2 code, the short source, and the long source for that time series. In addition, the contents of the OUT= data set shows every selected time series name and label. See Output 49.11.1 and Output 49.11.2 for more details about the OUTSELECT=ON option.

A more detailed discussion of how to map Haver frequencies to SAS time intervals follows.

Mapping Haver Frequencies to SAS Time Intervals

Table 49.2 summarizes the mapping of Haver frequencies to SAS time intervals. For more information, see Chapter 5, "Date Intervals, Formats, and Functions."

Table 49.2 Mapping Haver Frequencies to SAS Time Intervals

Haver Frequency	SAS Time Interval	FREQ=
ANNUAL	YEAR	YEARLY
QUARTERLY	QTR	QTRLY
MONTHLY	MONTH	MON
WEEKLY (SUNDAY)	WEEK.1	WEEK.1
WEEKLY (MONDAY)	WEEK.2	WEEK.2
WEEKLY (TUESDAY)	WEEK.3	WEEK.3
WEEKLY (WEDNESDAY)	WEEK.4	WEEK.4
WEEKLY (THURSDAY)	WEEK.5	WEEK.5
WEEKLY (FRIDAY)	WEEK.6	WEEK.6
WEEKLY (SATURDAY)	WEEK.7	WEEK.7
WEEKLY WEEK.1-WEEK.7	WEEKLY	WEEKLY
DAILY	WEEKDAY17W	DAY

Error Recovery for the SASEHAVR Interface Engine

Common errors are easy to avoid by noting the valid dates that are specified in the warning messages in your SAS log. Often you can get rid of errors by removing the date restriction (START= and END= options), by removing the FORCE=FREQ option, or by deleting the FREQ= option so that the frequency defaults to the original frequency rather than attempting a conversion.

Following are some common error scenarios and how to handle them.

Using the Optimum Range for Best Output Results

Suppose you see the following warnings in your SAS log:

```
libname kgs2 sasehavr "%sysget(HAVER_DATA)"
        start= 19550101 end=19600105
        keep="FCSEED, FCSEEI, FCSEEM, BGSX, BGSM, FXDUSBC"
        group="I01, F56, M02, R30"
        source="JPM,CEN,OMB" ;

NOTE: Libref KGS2 was successfully assigned as follows:
      Engine:        SASEHAVR
      Physical Name: C:\haver

data kgse9;
   set kgs2.haver;
NOTE: Defaulting to MONTHLY frequency.
WARNING: Start date (19550101) is not a valid date.
         Engine is ignoring your start date and using
         default. Setting the default Haver start date to 7001.
WARNING: End date (19600105) is not a valid date.
         Engine is ignoring your end date and using
         default. Setting the default Haver end date to 10103.
run;

NOTE: There were 375 observations read from the data set KGS2.HAVER.
NOTE: The data set WORK.KGSE9 has 375 observations and 4 variables.
```

The important diagnostic to note here is the warning message that tells you that the data starts in January of 1970 (Haver date 7001), and ends in March, 2001 (Haver date 10103). Since the specified range falls outside the range of data, no observations are in range. So, the engine uses the default range stated in the warning messages. Change the START= and END= options to overlap the results in data that span from JAN1970 to MAR2001. To view the entire range of selected data, remove the START= and END= options from the LIBNAME statement:

```
libname kgs sasehavr "%sysget(HAVER_DATA)"
        keep="FCSEED, FCSEEI, FCSEEM, BGSX, BGSM, FXDUSBC"
        group="I01, F56, M02, R30"
        source="JPM,CEN,OMB" ;

NOTE: Libref KGS was successfully assigned as follows:
      Engine:        SASEHAVR
      Physical Name: C:\haver

data kgse5;
   set kgs.haver;
NOTE: Defaulting to MONTHLY frequency.
   run;

NOTE: There were 375 observations read from the data set KGS.HAVER.
NOTE: The data set WORK.KGSE5 has 375 observations and 4 variables.
```

Using a Valid Range of Data with START= and END= Options

In this example, an error about an invalid range is issued:

```
libname lib1 sasehavr "%sysget(HAVER_DATA)"  freq=Weekly
   start=20060301 end=20060531;
NOTE: Libref LIB1 was successfully assigned as follows:
      Engine:        SASEHAVR
      Physical Name: C:\haver
libname lib2 "\\dntsrc\usrtmp\saskff" ;
NOTE: Libref LIB2 was successfully assigned as follows:
      Engine:        V9
      Physical Name: \\dntsrc\usrtmp\saskff
data lib2.wweek;
   set lib1.intwkly;
ERROR: No observations found inside RANGE.
       The valid range for HAVER dates is (610104-1050318).
ERROR: No observations found in specified range.
            keep date m11: ;
         run;

WARNING: The variable date in the DROP, KEEP, or RENAME list
         has never been referenced.
WARNING: The variable m11: in the DROP, KEEP, or RENAME list
         has never been referenced.
NOTE: The SAS System stopped processing this step because of errors.
WARNING: The data set LIB2.WWEEK may be incomplete.
         When this step was stopped there were 0
         observations and 0 variables.
WARNING: Data set LIB2.WWEEK was not replaced because this step was stopped.
```

The important diagnostic message is the first error statement which tells you that the range of Haver dates is not valid for the specified frequency. A valid range is one that overlaps the dates (610104–1050318). Removing the range altogether causes the engine to output the entire range of data.

```
libname lib1 sasehavr "%sysget(HAVER_DATA)"  freq=Weekly;

NOTE: Libref LIB1 was successfully assigned as follows:
      Engine:        SASEHAVR
      Physical Name: C:\haver

libname lib2 "\\dntsrc\usrtmp\saskff" ;
NOTE: Libref LIB2 was successfully assigned as follows:
      Engine:        V9
      Physical Name: \\dntsrc\usrtmp\saskff

data lib2.wweek;
   set lib1.intwkly;
   keep date m11: ;
run;

NOTE: There were 2307 observations read from the data set LIB1.INTWKLY.
NOTE: The data set LIB2.WWEEK has 2307 observations and 35 variables.
```

Since the START= and END= options give day-based dates, it is important to use dates that correspond to the FREQ= option when giving a range of dates, especially with weekly frequencies such as WEEK.1–WEEK.7. Since FREQ=WEEK.4 selects weeks that begin on Wednesday, the start and end dates need to be specified as Wednesday dates.

```
libname lib1 sasehavr "%sysget(HAVER_DATA)"  freq=Week.4
   start=20050302 end=20050309;
NOTE: Libref LIB1 was successfully assigned as follows:
      Engine:        SASEHAVR
      Physical Name: \\tappan\crsp1\haver
title2 'Weekly dataset with freq=week.4 range is small';
libname lib2 "\\dntsrc\usrtmp\saskff" ;
NOTE: Libref LIB2 was successfully assigned as follows:
      Engine:        V9
      Physical Name: \\dntsrc\usrtmp\saskff

data lib2.wweek;
   set lib1.intwkly;
   keep date m11: ;
run;

NOTE: There were 2 observations read from the data set LIB1.INTWKLY.
NOTE: The data set LIB2.WWEEK has 2 observations and 25 variables.
```

Giving bad dates (for example, Tuesday dates) for a Wednesday FREQ=WEEK.4 results in the following error:

```
ERROR: Fatal error in GetDate routine.
       Remove the range statement or change the START= date to
       be consistent with the freq=option.
ERROR: No observations found in specified range.
```

Aggregating to Quarterly Frequency Using the FORCE=FREQ Option

In the next example, six time series are selected by the KEEP= option. Their frequencies are annual, monthly, and quarterly, so when the FREQ=WEEKLY and FORCE=FREQ options are used, a diagnostic appears in the log stating that the engine is forcing the frequency to QUARTERLY for better date alignment of observations. The first selected variable is BALO, which is a quarterly time series and causes the default choice of FREQ to be quarterly.

```
title1 '***HAVKWC.SAS: KEEP= option tests with wildcards***';

%setup( ets );

 /*----------------*/
 /* Wildcard: *    */
 /*----------------*/

title2 "keep=B*, G*, I*";
title3 "6 valid variables are: BALO BGSM BGSX BPBCA G IUM";
```

```
libname lib1 sasehavr 'C:\haver\' keep="B*, G*, I*"
   freq=weekly force=freq;
NOTE: Libref LIB1 was successfully assigned as follows:
      Engine:        SASEHAVR
      Physical Name: C:\haver\

data wc;
   set lib1.haver;
WARNING: Earliest Start Date in DLX Database matches QUARTERLY frequency
         better than the specified WEEKLY frequency.
         Engine is forcing the frequency to QUARTERLY for better date
         alignment of observations.
run;

NOTE: There were 221 observations read from the data set LIB1.HAVER.
NOTE: The data set WORK.WC has 221 observations and 7 variables.
```

Note that the time series IUM is an annual frequency. The attempt to convert to a quarterly frequency produces all missing values in the output range because aggregation produces only missing values when forced to go from a lower frequency to a higher frequency.

Data Elements Reference: Haver Analytics DLX Database Profile

The Haver DLX economic and financial database offerings include U.S. economic indicators, specialized databases, financial indicators, industry, industrial countries, emerging markets, international organizations, forecasts and as-reported data, and U.S. regional service. Table 49.3 is a list of available databases and the corresponding description of each.

Table 49.3 Available Data Offerings

Database Name	Offering Type	Description
USECON	U.S. economic indicators	U.S. economic, financial data
USNA	U.S. economic indicators	Complete U.S. NIPA accounts from the Bureau of Economic Analysis (BEA)
SURVEYS	U.S. economic indicators	Business and consumer expectations, surveys
SURVEYW	U.S. economic indicators	Business and consumer expectations, weekly surveys
CPIDATA	U.S. economic indicators	Consumer price indexes (CPI), monthly in CPI detailed report
PPI	U.S. economic indicators	Producer price indexes (PPI), by the Bureau of Labor Statistics (BLS)
PPIR	U.S. economic indicators	Producer price indexes by BLS
LABOR	U.S. economic indicators	Employment and earnings by BLS

Table 49.3 *continued*

Database Name	Offering Type	Description
EMPL	U.S. economic indicators	Household employment survey, monthly by BLS
CEW	U.S. economic indicators	Covered employment and wages, monthly, quarterly
IP	U.S. economic indicators	Industrial production and capacity utilization by Federal Reserve Board (FRB)
FFUNDS	U.S. economic indicators	Flow of funds data by FRB
CAPSTOCK	U.S. economic indicators	Capital stock by the Bureau of Economic Analysis (BEA)
USINT	U.S. economic indicators	U.S. international trade (TIC) data by country and product
CBDB	Specialized databases	Conference Board database, monthly by The Conference Board (TCB)
BCI	Specialized databases	U.S. business cycle indicators, by TCB
UMSCA	Specialized databases	Consumer Sentiment Survey from the University of Michigan
FIBERUS	Specialized databases	U.S. FIBER business cycle indicators from the Foundation of International Business and Economic Research (FIBER)
FIBER	Specialized databases	FIBER business cycle indicators from FIBER
DAILY	Financial indicators	U.S. daily statistics data
INTDAILY	Financial indicators	Country daily statistics
WEEKLY	Financial indicators	U.S. weekly statistics
INTWKLY	Financial indicators	Country weekly statistics
SPD	Financial indicators	Standard and Poor's industry groups, daily
SPW	Financial indicators	Standard and Poor's industry groups, weekly
SPM	Financial indicators	Standard and Poor's industry groups, monthly
SPAH	Financial indicators	Standard and Poor's Analysts' Handbook, yearly
MSCID	Financial indicators	Morgan Stanley Capital International, daily
MSCIW	Financial indicators	Morgan Stanley Capital International, weekly

Table 49.3 *continued*

Database Name	Offering Type	Description
MSCIM	Financial indicators	Morgan Stanley Capital International, monthly
EMBI	Financial indicators	Emerging Markets Bond Index from J.P. Morgan
BONDINDX	Financial indicators	U.S. bond indexes, from Barclays Capital, Citigroup, Merrill Lynch, and Standard and Poors
BONDS	Financial indicators	Citigroup bond performance indexes by Citigroup Global Markets, formerly Salomon Smith Barney
ICI	Financial indicators	Mutual fund activity from the Investment Company Institute
QFR	Financial indicators	Quarterly financial report by FRB
MBAMTG	Financial indicators	Mortgage delinquency rates by the Mortgage Bankers Association
MBAMOS	Financial indicators	Mortgage origination surveys, from two Mortgage Banker Association surveys
MARKIT	Financial indicators	Markit's indexes used to price credit default swaps
DLINQ	Financial indicators	Consumer delinquency rates by American Bankers Association, monthly
FDIC	Financial indicators	FDIC banking statistics TIC data from the Quarterly Banking Profile
GOVFIN	Financial indicators	U.S. government financial statistics by U.S. Treasury
INDUSTRY	Industry	U.S. industry statistics, from Department of Agriculture, trade associations
WARDS	Industry	Automotive statistics, from Ward's Automotive Group
USDA	Industry	World agriculture statistics, from U.S. Department of Agriculture (USDA)
REALTOR	Industry	Home sales from National Association of Realtors
CREALTOR	Industry	Home sales from National Association of Realtors
PREALTOR	Industry	Pending home sales from National Association of Realtors
EEI	Industry	U.S. electric output from the Edison Electric Institute, weekly
ASM	Industry	Annual Survey of Manufactures from the U.S. Census Bureau
RAILSHAR	Industry	Railcar loadings from Association of American Railroads and Atlantic Systems
CHEMWEEK	Industry	Weekly chemical prices from Access Intelligence
BALTIC	Industry	Baltic freight indexes, from the Baltic Exchange in London

Table 49.3 *continued*

Database Name	Offering Type	Description
OGJ	Industry	U.S. and international energy statistics, from Pennwell Publishing's Oil & Gas Journal
OGJANN	Industry	U.S. and international energy statistics, from Pennwell Publishing's Oil & Gas Journal, annual
OILWKLY	Industry	Weekly oil statistics, from Penwell Publishing's Oil & Gas Journal
OMI	Industry	Oil market intelligence, from Energy Intelligence
NGW	Industry	Natural Gas Week, from Energy Intelligence
WGI	Industry	World Gas Intelligence, from Energy Intelligence
G10+	Industrial countries	International Macroeconomic Data by Haver Analytics
PMI	Industrial countries	Purchasing Managers Indexes by Markit Economics
INTSRVYS	Industrial countries	Country surveys
JAPAN	Industrial countries	Japan from Nomura Research Institute
JAPANW	Industrial countries	Japan from Nomura Research Institute, weekly
CANSIM	Industrial countries	Canada from Statistics Canada and the Bank of Canada
CANSIMR	Industrial countries	Canada from Statistics Canada and the Bank of Canada
UK	Industrial countries	United Kingdom, from the Office of National Statistics and the Bank of England
UKSRVYS	Industrial countries	United Kingdom surveys, by NTC Economics, Ltd.
GERMANY	Industrial countries	Germany, from the Deutsche Bundesbank and Statistics Bundesamt
FRANCE	Industrial countries	France, Statistics from INSEE (France's National Statistical Office), the Bank of France, and the Ministry of France
ITALY	Industrial countries	Italy, from Istituto Nazionale di Statistica and Banca d'Italia
SPAIN	Industrial countries	Spain, from the Instituto Nacional de Estadistica and the Banco de Espana
IRELAND	Industrial countries	Ireland, from the Central Statistics Office and Central Bank
NORDIC	Industrial countries	Norway, Sweden, Denmark, Finland
ALPMED	Industrial countries	Austria, Switzerland, Greece, Portugal

Table 49.3 *continued*

Database Name	Offering Type	Description
BENELUX	Industrial countries	Belgium, Netherlands, Luxembourg, monthly
ANZ	Industrial countries	Australia and New Zealand
EMERGELA	Emerging markets	Latin American macroeconomic data
EMERGECW	Emerging markets	Central and Eastern Europe and Western Asia
EMERGEMA	Emerging markets	Middle East and African emerging markets
EMERGEPR	Emerging markets	Asia/Pacific Rim emerging markets
CHINA	Emerging markets	CEIC Premium China Database, from CEIC Data Company Ltd (CEIC)
INDIA	Emerging markets	CEIC Premium India Database, from CEIC
EUROSTAT	International organizations	European Union data from Eurostat, the European Central Bank, and the European Commission
EULABOR	International organizations	European Union regional labor from Eurostat
OECDMEI	International organizations	Organization for Economic Cooperation and Development (OECD) main economic indicators
OECDNAQ	International organizations	OECD Quarterly National Accounts
OECDNA	International organizations	OECD Annual National Accounts
OECDFIN	International organizations	OECD Financial Accounts and Financial Balance Sheets
OECDFEI	International organizations	OECD foreign direct investment data
OUTLOOK	International organizations	OECD Economic Outlook
IFS	International organizations	International Financial Statistics from International Monetary Fund
IFSANN	International organizations	International Financial Statistics, annual from International Monetary Fund
IMFBOP	International organizations	Balance of Payment Statistics from International Monetary Fund
IMFBOPA	International organizations	Annual Balance of Payment Statistics from International Monetary Fund
IMFDOT	International organizations	Direction of Trade Statistics from International Monetary Fund

Table 49.3 *continued*

Database Name	Offering Type	Description
IMFDOTM	International organizations	Direction of Trade Statistics, monthly from International Monetary Fund
IMFWEO	International organizations	Analysis and projections of economic development at global level from International Monetary Fund
BIS	International organizations	International financial claims and liabilities from the Bank for International Settlements
WBPRICES	International organizations	World commodity prices from The World Development Prospects Group (Pinksheets)
WBDEBT	International organizations	Global development finance from The World Bank debt tables
UNPOP	International organizations	United Nations population projections
INTCOMP	International organizations	International comparisons from U.S. Bureau of Labor Statistics
MA4CAST	Forecasts and as-reported data	Short-term U.S. economic forecasts from Macroeconomic Advisers
MA4CSTL	Forecasts and as-reported data	Long-term U.S. economic forecasts from Macroeconomic Advisers
CQM	Forecasts and as-reported data	Canadian quarterly model from Centre for Spatial Economics
CPM	Forecasts and as-reported data	Canadian provincial model from Centre for Spatial Economics
OEFQMACR	Forecasts and as-reported data	Global macroeconomic forecasts from Oxford Economic Forecasting
OEFMAJOR	Forecasts and as-reported data	Global macroeconomic forecasts from Oxford Economic Forecasting
OEFINTER	Forecasts and as-reported data	Global macroeconomic forecasts from Oxford Economic Forecasting
OEFMINOR	Forecasts and as-reported data	Global macroeconomic forecasts from Oxford Economic Forecasting
OEFQIND	Forecasts and as-reported data	Global industry from Oxford Economic Forecasting
EIUIAMER	Forecasts and as-reported data	Market indicators and forecasts (America) from Economist Intelligence Unit
EIUIASIA	Forecasts and as-reported data	Market indicators and forecasts (Asia) from Economist Intelligence Unit
EIUIEEUR	Forecasts and as-reported data	Market indicators and forecasts (Eastern Europe) from Economist Intelligence Unit
EIUIMENA	Forecasts and as-reported data	Market indicators and forecasts from Economist Intelligence Unit
EIUISUBS	Forecasts and as-reported data	Market indicators and forecasts from Economist Intelligence Unit

Table 49.3 *continued*

Database Name	Offering Type	Description
EIUIWEUR	Forecasts and as-reported data	Market indicators and forecasts (Western Europe) from Economist Intelligence Unit
EIUIREGS	Forecasts and as-reported data	Market indicators and forecasts from Economist Intelligence Unit
EIUDAMER	Forecasts and as-reported data	Country data (America) from Economist Intelligence Unit
EIUDASIA	Forecasts and as-reported data	Country data (Asia) from Economist Intelligence Unit
EIUDEEUR	Forecasts and as-reported data	Country data (Eastern Europe) from Economist Intelligence Unit
EIUDMENA	Forecasts and as-reported data	Country data from Economist Intelligence Unit
EIUDSUBS	Forecasts and as-reported data	Country data from Economist Intelligence Unit
EIUDWEUR	Forecasts and as-reported data	Country data (Western Europe) from Economist Intelligence Unit
EIUDOECD	Forecasts and as-reported data	Country data (OECD) from Economist Intelligence Unit
EIUDREGS	Forecasts and as-reported data	Country data from Economist Intelligence Unit
AS1REPNA	Forecasts and as-reported data	Action Economics forecast medians and as reported data
MMSAMER	Forecasts and as-reported data	MMS survey medians and as-first-reported data (America) from MMS International
MMSEUR	Forecasts and as-reported data	MMS survey medians and as-first-reported data (Europe) from MMS International
SURVEYS	Forecasts and as-reported data	Economic survey forecasts
AS4CAST	Forecasts and as-reported data	Historical economic forecasts
ASREPGDP	Forecasts and as-reported data	As-reported U.S. gross domestic product from Bureau of Economic Analysis
LABORR	U.S. regional	Monthly payroll employment from Bureau of Labor Statistics
EMPLR	U.S. regional	Labor force and unemployment from Bureau of Labor Statistics
EMPLC	U.S. regional	Labor force and unemployment from Bureau of Labor Statistics

Table 49.3 *continued*

Database Name	Offering Type	Description
BEAEMPL	U.S. regional	Annual employment by industry
BEAEMPM	U.S. regional	Annual employment by industry
PERMITS	U.S. regional	Residential building permits
PERMITY	U.S. regional	Residential building permits
PERMITP	U.S. regional	Residential building permits
PERMITC	U.S. regional	Residential building permits
PERMITA	U.S. regional	Residential building permits
REGIONAL	U.S. regional	Selected regional indicators
REGIONW	U.S. regional	Selected regional indicators
PIQR	U.S. regional	Personal income
PIR	U.S. regional	Personal income
PIRMSA	U.S. regional	Personal income
PICOUNTY	U.S. regional	Personal income
PIRC1 to 9	U.S. regional	Personal income
MBAMTG	U.S. regional	Mortgage delinquency rates from Mortgage Bankers Association
DLINQR	U.S. regional	Consumer delinquency rates from American Bankers Association
FALOAN	U.S. regional	Real estate and construction delinquency rates by Foresight Analytics
BANKRUPT	U.S. regional	Bankruptcies by county and metropolitan statistical area
GSP	U.S. regional	Gross state product from BEA
GDPMSA	U.S. regional	Gross domestic product by Metropolitan Statistical Areas (MSA)
USPOP	U.S. regional	Population by age and sex
USPOPC	U.S. regional	Population by age and sex
PORTS	U.S. regional	Trade by port
EXPRQ1 to 9	U.S. regional	Exports by industry and country from the World Institute for Strategic Economic Research and the U.S. Census Bureau
EXPORTSR	U.S. regional	Exports by industry and country from the World Institute for Strategic Economic Research and the U.S. Census Bureau
GOVFINR	U.S. regional	Government financial statistics from the U.S. Census Bureau and Rockefeller Institute of Government
FDICR	U.S. regional	FDIC banking statistics

Examples: SASEHAVR Interface Engine

Before running the following sample code, set your HAVER_DATA environment variable to point to the SAS/ETS SASMISC folder that contains sample Haver databases. The provided sample data files are HAVERD.DAT, HAVERD.IDX, HAVERW.IDX, and HAVERW.DAT. In the following example, the Haver database is called haverw and it resides in the directory lib1. The DATA statement names the SAS output data set hwouty, which will reside in the Work library.

Example 49.1: Examining the Contents of a Haver Database

To see which time series are in your Haver database, use the CONTENTS procedure with the SASEHAVR LIBNAME statement to read the contents.

```
libname lib1 sasehavr "%sysget(HAVER_DATA)"
        freq=yearly start=19920101
        end=20041231
        force=freq;

data hwouty;
   set lib1.haverw;
run;

title1 'Haver Analytics Database, HAVERW.DAT';
title2 'PROC CONTENTS for Time Series converted to yearly frequency';
proc contents data=hwouty;
run;
```

All time series in the Haver haverw database are listed alphabetically in Output 49.1.1.

Output 49.1.1 Examining the Contents of Haver Analytics Database, haverw.dat

Haver Analytics Database, HAVERW.DAT
PROC CONTENTS for Time Series converted to yearly frequency

The CONTENTS Procedure

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Label
1	DATE	Num	8	YEAR4.	Date of Observation
2	FA	Num	8		Total Assets: All Commercial Banks (SA, Bil.$)
3	FCM1M	Num	8		1-Month Treasury Bill Market Bid Yield at Constant Maturity (%)
4	FM1	Num	8		Money Stock: M1 (SA, Bil.$)
5	FTA1MA	Num	8		Treasury 4-Week Bill: Total Amount Accepted (Bil$)
6	FTB3	Num	8		3-Month Treasury Bills, Auction (% p.a.)
7	LICN	Num	8		Unemployment Insurance: Initial Claims, State Programs (NSA, Thous)

You could also use the following SAS statements to create a SAS data set named hwouty and to print its contents.

```
libname lib1 sasehavr "%sysget(HAVER_DATA)"
       freq=yearly
       start=19920101
       end=20041231
       force=freq;

data hwouty;
   set lib1.haverw;
run;

title1 'Haver Analytics Database, Frequency=yearly, infile=haverw.dat';
title2 'Define a range inside the data range for OUT= dataset,';
title3 'Using the START=19920101 END=20041231 LIBNAME options.';

proc print data=hwouty;
run;
```

The preceding LIBNAME LIB1 statement specifies that all time series in the haverw database be converted to a yearly frequency but to select only the range of data from January 1, 1992, to December 31, 2004. The resulting SAS data set hwouty is shown in Output 49.1.2.

Output 49.1.2 Defining a Range inside the Data Range for Yearly Time Series

Haver Analytics Database, Frequency=yearly, infile=haverw.dat
Define a range inside the data range for OUT= dataset,
Using the START=19920101 END=20041231 LIBNAME options.

Obs	DATE	FA	FCM1M	FM1	FTA1MA	FTB3	LICN
1	1992	3466.3	.	965.31	.	3.45415	407.340
2	1993	3624.6	.	1077.69	.	3.01654	344.934
3	1994	3875.8	.	1144.85	.	4.28673	340.054
4	1995	4209.3	.	1142.70	.	5.51058	357.038
5	1996	4399.1	.	1106.46	.	5.02096	351.358
6	1997	4820.3	.	1069.23	.	5.06885	321.513
7	1998	5254.8	.	1079.56	.	4.80726	317.077
8	1999	5608.1	.	1101.14	.	4.66154	301.581
9	2000	6115.4	.	1104.07	.	5.84644	301.108
10	2001	6436.2	2.31368	1136.31	11.753	3.44471	402.583
11	2002	7024.9	1.63115	1192.03	18.798	1.61548	402.796
12	2003	7302.9	1.02346	1268.40	16.089	1.01413	399.137
13	2004	7950.5	1.26642	1337.89	13.019	1.37557	345.109

Example 49.2: Viewing Quarterly Time Series from a Haver Database

The following statements specify a quarterly frequency conversion of all time series for the period spanning April 1, 2001, to December 31, 2004:

```
libname lib1 sasehavr "%sysget(HAVER_DATA)"
        freq=quarterly
        start=20010401
```

```
        end=20041231
        force=freq;

data hwoutq;
   set lib1.haverw;
run;

title1 'Haver Analytics Database, Frequency=quarterly, infile=haverw.dat';
title2 '  Define a range inside the data range for OUT= dataset';
title3 '  Using the START=20010401 END=20041231 LIBNAME options.';

proc print data=hwoutq;
run;
```

The resulting SAS data set hwoutq is shown in Output 49.2.1.

Output 49.2.1 Defining a Range inside the Data Range for Quarterly Time Series

Haver Analytics Database, Frequency=quarterly, infile=haverw.dat
Define a range inside the data range for OUT= dataset
Using the START=20010401 END=20041231 LIBNAME options.

Obs	DATE	FA	FCM1M	FM1	FTA1MA	FTB3	LICN
1	2001Q2	6225.4	.	1115.75	.	3.68308	356.577
2	2001Q3	6425.9	2.98167	1157.90	12.077	3.27615	368.408
3	2001Q4	6436.2	2.00538	1169.62	11.753	1.95308	477.685
4	2002Q1	6396.3	1.73077	1186.92	22.309	1.72615	456.292
5	2002Q2	6563.5	1.72769	1183.30	17.126	1.72077	368.592
6	2002Q3	6780.0	1.69231	1189.89	21.076	1.64769	352.892
7	2002Q4	7024.9	1.37385	1207.80	18.798	1.36731	433.408
8	2003Q1	7054.5	1.17846	1231.41	24.299	1.15269	458.746
9	2003Q2	7319.6	1.08000	1262.24	14.356	1.05654	386.185
10	2003Q3	7238.6	0.92000	1286.21	16.472	0.92885	361.346
11	2003Q4	7302.9	0.91538	1293.76	16.089	0.91846	390.269
12	2004Q1	7637.3	0.90231	1312.43	21.818	0.91308	400.585
13	2004Q2	7769.8	0.94692	1332.75	12.547	1.06885	310.508
14	2004Q3	7949.5	1.34923	1343.79	21.549	1.49393	305.862
15	2004Q4	7950.5	1.82429	1362.60	13.019	2.01731	362.171

Example 49.3: Viewing Monthly Time Series from a Haver Database

The following statements convert weekly time series to a monthly frequency:

```
libname lib1 sasehavr "%sysget(HAVER_DATA)"
        freq=monthly
        start=20040401
        end=20041231
        force=freq;

data hwoutm;
   set lib1.haverw;
run;
```

```
title1 'Haver Analytics Database, Frequency=monthly, infile=haverw.dat';
title2 '  Define a range inside the data range for OUT= dataset';
title3 '  Using the START=20040401 END=20041231 LIBNAME options.';

proc print data=hwoutm;
run;
```

The result from using the range of April 1, 2004, to December 31, 2004, is shown in Output 49.3.1.

Output 49.3.1 Defining a Range inside the Data Range for Monthly Time Series

Haver Analytics Database, Frequency=monthly, infile=haverw.dat
Define a range inside the data range for OUT= dataset
Using the START=20040401 END=20041231 LIBNAME options.

Obs	DATE	FA	FCM1M	FM1	FTA1MA	FTB3	LICN
1	APR2004	7703.8	0.9140	1325.73	16.946	0.93900	317.36
2	MAY2004	7704.7	0.9075	1332.96	25.043	1.03375	297.00
3	JUN2004	7769.8	1.0275	1339.50	12.547	1.26625	315.45
4	JUL2004	7859.5	1.1840	1330.13	21.823	1.34900	357.32
5	AUG2004	7890.0	1.3650	1347.84	25.213	1.48000	276.70
6	SEP2004	7949.5	1.5400	1352.40	21.549	1.65000	270.70
7	OCT2004	7967.6	1.6140	1355.28	21.322	1.74750	304.24
8	NOV2004	8053.4	1.9125	1366.06	21.862	2.05625	335.85
9	DEC2004	7950.5	1.9640	1365.60	13.019	2.20200	441.16

Example 49.4: Viewing Weekly Time Series from a Haver Database

The following statements show weekly data that span from September 1, 2004, to December 31, 2004:

```
libname lib1 sasehavr "%sysget(HAVER_DATA)"
        freq=weekly
        start=20040901
        end=20041231;

data hwoutw;
   set lib1.haverw;
run;

title1 'Haver Analytics Database, Frequency=weekly, infile=haverw.dat';
title2 '  Define a range inside the data range for OUT= dataset';
title3 '  Using the START=20040901 END=20041231 LIBNAME options.';

proc print data=hwoutw;
run;
```

Output 49.4.1 shows the output.

Output 49.4.1 Defining a Range inside the Data Range for Weekly Time Series

Haver Analytics Database, Frequency=weekly, infile=haverw.dat
Define a range inside the data range for OUT= dataset
Using the START=20040901 END=20041231 LIBNAME options.

Obs	DATE	FA	FCM1M	FM1	FTA1MA	FTB3	LICN
1	29AUG2004	7890.0	1.39	1360.8	27.342	1.515	275.2
2	05SEP2004	7906.2	1.46	1353.7	25.213	1.580	273.7
3	12SEP2004	7962.7	1.57	1338.3	25.255	1.635	250.6
4	19SEP2004	7982.1	1.57	1345.6	15.292	1.640	275.8
5	26SEP2004	7987.9	1.56	1359.7	15.068	1.685	282.7
6	03OCT2004	7949.5	1.54	1366.0	21.549	1.710	279.6
7	10OCT2004	7932.4	1.56	1362.3	17.183	1.685	338.7
8	17OCT2004	7956.9	1.59	1350.1	17.438	1.680	279.8
9	24OCT2004	7957.3	1.63	1346.0	12.133	1.770	317.6
10	31OCT2004	7967.6	1.75	1362.7	21.322	1.855	305.5
11	07NOV2004	7954.1	1.84	1350.4	22.028	1.950	354.8
12	14NOV2004	8009.7	1.89	1354.8	25.495	2.045	311.9
13	21NOV2004	7938.3	1.93	1364.5	24.000	2.075	356.0
14	28NOV2004	8053.4	1.99	1381.3	24.424	2.155	320.7
15	05DEC2004	8010.7	2.05	1379.3	21.862	2.195	472.7
16	12DEC2004	8054.8	2.08	1355.1	22.178	2.210	370.6
17	19DEC2004	8019.2	1.98	1358.3	12.066	2.200	374.7
18	26DEC2004	7995.5	1.89	1366.3	12.787	2.180	446.6

Example 49.5: Viewing Daily Time Series from a Haver Database

Consider viewing the Haver Analytics daily database named haverd. The contents of this database can be seen by submitting the following DATA step:

```
libname lib1 sasehavr "%sysget(HAVER_DATA)"
        freq=daily
        start=20041201
        end=20041231;

data hwoutd;
   set lib1.haverd;
run;

title1 'Haver Analytics Database, HAVERD.DAT';
title2 'PROC CONTENTS for Time Series converted to daily frequency';
proc contents data=hwoutd;
run;
```

Output 49.5.1 shows the output of PROC CONTENTS with the time ID variable DATE followed by the time series variables FCM10, FCM1M, FFED, FFP1D, FXAUS, and TCC with their corresponding attributes such as type, length, format, and label.

Output 49.5.1 Examining the Contents of a Daily Haver Analytics Database, haverd.dat

Haver Analytics Database, HAVERD.DAT
PROC CONTENTS for Time Series converted to daily frequency

The CONTENTS Procedure

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Label
1	DATE	Num	8	DATE9.	Date of Observation
2	FCM10	Num	8		10-Year Treasury Note Yield at Constant Maturity (Avg, % p.a.)
3	FCM1M	Num	8		1-Month Treasury Bill Market Bid Yield at Constant Maturity (%)
4	FFED	Num	8		Federal Funds [Effective] Rate (% p.a.)
5	FFP1D	Num	8		1-Day AA Financial Commercial Paper (% per annum)
6	FXAUS	Num	8		Foreign Exchange Rate: Australia (US$/Australian$)
7	TCC	Num	8		Treasury: Closing Operating Cash Balance (Today, Mil.$)

Example 49.6: Limiting the Range of Time Series from a Haver Database

The following statements limit the range of data to the month of December:

```
libname lib1 sasehavr "%sysget(HAVER_DATA)"
        freq=daily
        start=20041201
        end=20041231;

data hwoutd;
   set lib1.haverd;
run;

title1 'Haver Analytics Database, Frequency=daily, infile=haverd.dat';
title2 '   Define a range inside the data range for OUT= dataset';
title3 '   Using the START=20041201 END=20041231 LIBNAME options.';

proc print data=hwoutd;
run;
```

Note that Output 49.6.1 for daily conversion shows the frequency as the SAS time interval for WEEKDAY.

Output 49.6.1 Defining a Range inside the Data Range for Daily Time Series

Haver Analytics Database, Frequency=daily, infile=haverd.dat
Define a range inside the data range for OUT= dataset
Using the START=20041201 END=20041231 LIBNAME options.

Obs	DATE	FCM10	FCM1M	FFED	FFP1D	FXAUS	TCC
1	01DEC2004	4.38	2.06	2.04	2.01	0.7754	7564
2	02DEC2004	4.40	2.06	2.00	1.98	0.7769	8502
3	03DEC2004	4.27	2.06	1.98	1.96	0.7778	7405
4	06DEC2004	4.24	2.09	2.04	1.98	0.7748	7019
5	07DEC2004	4.23	2.08	1.99	1.99	0.7754	15520
6	08DEC2004	4.14	2.08	2.01	1.98	0.7545	12329
7	09DEC2004	4.19	2.07	2.05	2.03	0.7532	5441
8	10DEC2004	4.16	2.07	2.09	2.07	0.7495	6368
9	13DEC2004	4.16	2.04	2.18	2.13	0.7592	11395
10	14DEC2004	4.14	2.01	2.24	2.22	0.7566	13695
11	15DEC2004	4.09	1.98	2.31	2.27	0.7652	39765
12	16DEC2004	4.19	1.93	2.26	2.24	0.7563	33640
13	17DEC2004	4.21	1.95	2.23	2.20	0.7607	32764
14	20DEC2004	4.21	1.97	2.26	2.21	0.7644	36216
15	21DEC2004	4.18	1.92	2.24	2.21	0.7660	35056
16	22DEC2004	4.21	1.84	2.25	2.22	0.7656	34599
17	23DEC2004	4.23	1.83	2.34	2.08	0.7654	24467
18	24DEC2004	.	.	2.27	.	0.7689	26898
19	27DEC2004	4.30	1.90	2.24	2.26	0.7777	31874
20	28DEC2004	4.31	1.88	2.24	2.24	0.7787	30513
21	29DEC2004	4.33	1.76	2.23	2.23	0.7709	34754
22	30DEC2004	4.27	1.68	2.24	2.18	0.7785	20045
23	31DEC2004	4.24	1.89	1.97	2.18	0.7805	24690

Example 49.7: Using the WHERE Statement to Subset Time Series from a Haver Database

Using a WHERE statement in the DATA step can be useful for further subsetting.

```
libname lib1 sasehavr "%sysget(HAVER_DATA)"
        freq=daily start=20041101 end=20041231;

data hwoutd;
   set lib1.haverd;
   where date  between '01nov2004'd and '01dec2004'd;
run;

title1 'Haver Analytics Database, Frequency=daily, infile=haverd.dat';
title2 '   Define a range inside the data range for OUT= dataset';
title3 '   Using the START=20041101 END=20041231 LIBNAME options.';
title4 'Subset further: where date between 01nov2004 and 31dec2004.';
proc print data=hwoutd;
run;
```

Output 49.7.1 shows that the time slice of November 1, 2004, to December 31, 2004, is narrowed further by the DATE test in the WHERE statement to stop at December 1, 2004.

Output 49.7.1 Defining a Range Using the WHERE Statement, START=20041101, and END=20041231

Haver Analytics Database, Frequency=daily, infile=haverd.dat
Define a range inside the data range for OUT= dataset
Using the START=20041101 END=20041231 LIBNAME options.
Subset further: where date between 01nov2004 and 31dec2004.

Obs	DATE	FCM10	FCM1M	FFED	FFP1D	FXAUS	TCC
1	01NOV2004	4.11	1.79	1.83	1.80	0.7460	35111
2	02NOV2004	4.10	1.86	1.74	1.74	0.7447	34091
3	03NOV2004	4.09	1.83	1.73	1.73	0.7539	14862
4	04NOV2004	4.10	1.85	1.77	1.75	0.7585	23304
5	05NOV2004	4.21	1.86	1.76	1.75	0.7620	19872
6	08NOV2004	4.22	1.88	1.80	1.84	0.7578	21095
7	09NOV2004	4.22	1.89	1.79	1.81	0.7618	16390
8	10NOV2004	4.25	1.88	1.92	1.85	0.7592	12872
9	11NOV2004	.	.	1.92	.	.	12872
10	12NOV2004	4.20	1.91	2.02	1.96	0.7685	28926
11	15NOV2004	4.20	1.92	2.06	2.03	0.7719	10480
12	16NOV2004	4.21	1.93	1.98	1.95	0.7728	13417
13	17NOV2004	4.14	1.90	1.99	1.93	0.7833	10506
14	18NOV2004	4.12	1.91	1.99	1.94	0.7786	6293
15	19NOV2004	4.20	1.98	1.99	1.93	0.7852	5100
16	22NOV2004	4.18	1.98	2.01	1.96	0.7839	6045
17	23NOV2004	4.19	1.99	2.00	1.95	0.7860	18135
18	24NOV2004	4.20	1.98	2.02	1.89	0.7863	14109
19	25NOV2004	.	.	2.02	.	.	14109
20	26NOV2004	4.24	2.01	2.01	1.97	0.7903	20588
21	29NOV2004	4.34	2.02	2.03	2.00	0.7852	24322
22	30NOV2004	4.36	2.07	2.02	2.04	0.7723	18033
23	01DEC2004	4.38	2.06	2.04	2.01	0.7754	7564

Example 49.8: Using the KEEP Option to Subset Time Series from a Haver Database

To select specific time series, you can use the KEEP= or DROP= option as follows:

```
libname lib1 sasehavr "%sysget(HAVER_DATA)"
        freq=daily
        start=20041101
        end=20041231
        keep="FCM*";

data hwoutd;
   set lib1.haverd;
run;

title1 'Haver Analytics Database, Frequency=daily, infile=haverd.dat';
title2 '   Define a range inside the data range for OUT= dataset';
title3 '   Using the START=20041101 END=20041231 LIBNAME options.';
title4 '   Subset further: Using keep="FCM*" LIBNAME option ';
proc print data=hwoutd;
run;
```

Output 49.8.1 shows two series that are selected by using KEEP="FCM*" in the LIBNAME statement.

Output 49.8.1 Using the KEEP Option and Defining a Range Using START=20041101 and END=20041231

Haver Analytics Database, Frequency=daily, infile=haverd.dat
Define a range inside the data range for OUT= dataset
Using the START=20041101 END=20041231 LIBNAME options.
Subset further: Using keep="FCM*" LIBNAME option

Obs	DATE	FCM10	FCM1M
1	01NOV2004	4.11	1.79
2	02NOV2004	4.10	1.86
3	03NOV2004	4.09	1.83
4	04NOV2004	4.10	1.85
5	05NOV2004	4.21	1.86
6	08NOV2004	4.22	1.88
7	09NOV2004	4.22	1.89
8	10NOV2004	4.25	1.88
9	11NOV2004	.	.
10	12NOV2004	4.20	1.91
11	15NOV2004	4.20	1.92
12	16NOV2004	4.21	1.93
13	17NOV2004	4.14	1.90
14	18NOV2004	4.12	1.91
15	19NOV2004	4.20	1.98
16	22NOV2004	4.18	1.98
17	23NOV2004	4.19	1.99
18	24NOV2004	4.20	1.98
19	25NOV2004	.	.
20	26NOV2004	4.24	2.01
21	29NOV2004	4.34	2.02
22	30NOV2004	4.36	2.07
23	01DEC2004	4.38	2.06
24	02DEC2004	4.40	2.06
25	03DEC2004	4.27	2.06
26	06DEC2004	4.24	2.09
27	07DEC2004	4.23	2.08
28	08DEC2004	4.14	2.08
29	09DEC2004	4.19	2.07
30	10DEC2004	4.16	2.07
31	13DEC2004	4.16	2.04
32	14DEC2004	4.14	2.01
33	15DEC2004	4.09	1.98
34	16DEC2004	4.19	1.93
35	17DEC2004	4.21	1.95
36	20DEC2004	4.21	1.97
37	21DEC2004	4.18	1.92
38	22DEC2004	4.21	1.84
39	23DEC2004	4.23	1.83
40	24DEC2004	.	.
41	27DEC2004	4.30	1.90
42	28DEC2004	4.31	1.88
43	29DEC2004	4.33	1.76

Output 49.8.1 *continued*

Haver Analytics Database, Frequency=daily, infile=haverd.dat
Define a range inside the data range for OUT= dataset
Using the START=20041101 END=20041231 LIBNAME options.
Subset further: Using keep="FCM*" LIBNAME option

Obs	DATE	FCM10	FCM1M
44	30DEC2004	4.27	1.68
45	31DEC2004	4.24	1.89

You can use the DROP option to drop specific variables from a Haver database. To specify this option, use DROP= instead of KEEP=.

Example 49.9: Using the SOURCE Option to Subset Time Series from a Haver Database

You can use the SOURCE= or DROPSOURCE= option to select specific variables that belong to a certain source, similar to the way you use the KEEP= or DROP= option.

```
libname lib1 sasehavr "%sysget(HAVER_DATA)"
        freq=daily
        start=20041101
        end=20041223
        source="FRB";

data hwoutd;
   set lib1.haverd;
run;

title1 'Haver Analytics Database, Frequency=daily, infile=haverd.dat';
title2 '   Define a range inside the data range for OUT= dataset';
title3 '   Using the START=20041101 END=20041223 LIBNAME options.';
title4 '   Subset further: Using source="FRB" LIBNAME option';
proc print data=hwoutd;
run;
```

Output 49.9.1 shows two series that are selected by using SOURCE="FRB" in the LIBNAME statement.

Output 49.9.1 Using the SOURCE Option and Defining a Range Using START=20041101 and END=20041223

Haver Analytics Database, Frequency=daily, infile=haverd.dat
Define a range inside the data range for OUT= dataset
Using the START=20041101 END=20041223 LIBNAME options.
Subset further: Using source="FRB" LIBNAME option

Obs	DATE	FCM10	FFED	FFP1D	FXAUS
1	01NOV2004	4.11	1.83	1.80	0.7460
2	02NOV2004	4.10	1.74	1.74	0.7447
3	03NOV2004	4.09	1.73	1.73	0.7539
4	04NOV2004	4.10	1.77	1.75	0.7585
5	05NOV2004	4.21	1.76	1.75	0.7620
6	08NOV2004	4.22	1.80	1.84	0.7578
7	09NOV2004	4.22	1.79	1.81	0.7618
8	10NOV2004	4.25	1.92	1.85	0.7592
9	11NOV2004	.	1.92	.	.
10	12NOV2004	4.20	2.02	1.96	0.7685
11	15NOV2004	4.20	2.06	2.03	0.7719
12	16NOV2004	4.21	1.98	1.95	0.7728
13	17NOV2004	4.14	1.99	1.93	0.7833
14	18NOV2004	4.12	1.99	1.94	0.7786
15	19NOV2004	4.20	1.99	1.93	0.7852
16	22NOV2004	4.18	2.01	1.96	0.7839
17	23NOV2004	4.19	2.00	1.95	0.7860
18	24NOV2004	4.20	2.02	1.89	0.7863
19	25NOV2004	.	2.02	.	.
20	26NOV2004	4.24	2.01	1.97	0.7903
21	29NOV2004	4.34	2.03	2.00	0.7852
22	30NOV2004	4.36	2.02	2.04	0.7723
23	01DEC2004	4.38	2.04	2.01	0.7754
24	02DEC2004	4.40	2.00	1.98	0.7769
25	03DEC2004	4.27	1.98	1.96	0.7778
26	06DEC2004	4.24	2.04	1.98	0.7748
27	07DEC2004	4.23	1.99	1.99	0.7754
28	08DEC2004	4.14	2.01	1.98	0.7545
29	09DEC2004	4.19	2.05	2.03	0.7532
30	10DEC2004	4.16	2.09	2.07	0.7495
31	13DEC2004	4.16	2.18	2.13	0.7592
32	14DEC2004	4.14	2.24	2.22	0.7566
33	15DEC2004	4.09	2.31	2.27	0.7652
34	16DEC2004	4.19	2.26	2.24	0.7563
35	17DEC2004	4.21	2.23	2.20	0.7607
36	20DEC2004	4.21	2.26	2.21	0.7644
37	21DEC2004	4.18	2.24	2.21	0.7660
38	22DEC2004	4.21	2.25	2.22	0.7656
39	23DEC2004	4.23	2.34	2.08	0.7654

Example 49.10: Using the GROUP Option to Subset Time Series from a Haver Database

You can use the GROUP= or DROPGROUP= option to select specific variables that belong to a certain group, similar to the way you use the KEEP= or DROP= option.

Output 49.10.1, Output 49.10.2, and Output 49.10.3 show three different cross sections of the same database, haverw, by specifying three unique GROUP= options: GROUP="F*" in LIBNAME LIB1, GROUP="M*" in LIBNAME LIB2, and GROUP="E*" in LIBNAME LIB3.

The following statements specify GROUP="F*" in the LIBNAME LIB1 statement:

```
libname lib1 sasehavr "%sysget(HAVER_DATA)"
        freq=week.6
        force=freq
        start=20040102
        end=20041001
        group="F*";

data hwoutwA;
   set lib1.haverw;
run;

title1 'Haver Analytics Database, Frequency=week.6, infile=haverw.dat';
title2 '   Define a range inside the data range for OUT= dataset';
title3 '   Using the START=20040102 END=20041001 LIBNAME options.';
title4 '   Subset further: Using group="F*" LIBNAME option';
proc print data=hwoutwA;
run;
```

Output 49.10.1 shows the output.

Output 49.10.1 Using the GROUP=F* Option and Defining a Range

Haver Analytics Database, Frequency=week.6, infile=haverw.dat
Define a range inside the data range for OUT= dataset
Using the START=20040102 END=20041001 LIBNAME options.
Subset further: Using group="F*" LIBNAME option

Obs	DATE	FCM1M	FTA1MA	FTB3
1	01JAN2004	0.86	16.089	0.885
2	08JAN2004	0.88	12.757	0.920
3	15JAN2004	0.84	12.141	0.870
4	22JAN2004	0.79	12.593	0.875
5	29JAN2004	0.86	17.357	0.890
6	05FEB2004	0.90	21.759	0.920
7	12FEB2004	0.90	21.557	0.920
8	19FEB2004	0.92	21.580	0.915
9	26FEB2004	0.96	21.390	0.930
10	04MAR2004	0.97	24.119	0.940
11	11MAR2004	0.96	24.294	0.930
12	18MAR2004	0.94	23.334	0.945
13	25MAR2004	0.95	21.400	0.930
14	01APR2004	0.95	21.818	0.945
15	08APR2004	0.94	17.255	0.930
16	15APR2004	0.92	14.143	0.915
17	22APR2004	0.89	14.136	0.935
18	29APR2004	0.87	16.946	0.970
19	06MAY2004	0.89	22.772	0.985
20	13MAY2004	0.89	23.113	1.060
21	20MAY2004	0.91	25.407	1.040
22	27MAY2004	0.94	25.043	1.050
23	03JUN2004	0.97	27.847	1.130
24	10JUN2004	1.01	27.240	1.230
25	17JUN2004	1.05	17.969	1.390
26	24JUN2004	1.08	12.159	1.315
27	01JUL2004	1.11	12.547	1.355
28	08JUL2004	1.14	21.303	1.320
29	15JUL2004	1.16	25.024	1.315
30	22JUL2004	1.21	25.327	1.330
31	29JUL2004	1.30	21.823	1.425
32	05AUG2004	1.34	21.631	1.465
33	12AUG2004	1.37	28.237	1.470
34	19AUG2004	1.36	26.070	1.470
35	26AUG2004	1.39	27.342	1.515
36	02SEP2004	1.46	25.213	1.580
37	09SEP2004	1.57	25.255	1.635
38	16SEP2004	1.57	15.292	1.640
39	23SEP2004	1.56	15.068	1.685
40	30SEP2004	1.54	21.549	1.710

The following statements specify GROUP="M*" in the LIBNAME LIB2 statement:

```
libname lib2 sasehavr "%sysget(HAVER_DATA)"
        freq=week.6
        force=freq start=20040102
        end=20041001
        group="M*";

data hwoutwB;
   set lib2.haverw;
run;

title1 'Haver Analytics Database, Frequency=week.6, infile=haverw.dat';
title2 '   Define a range inside the data range for OUT= dataset';
title3 '   Using the START=20040102 END=20041001 LIBNAME options.';
title4 '   Subset further: Using group="M*" LIBNAME option';
proc print data=hwoutwB;
run;
```

Output 49.10.2 shows the output.

Output 49.10.2 Using the GROUP=M* Option and Defining a Range

Haver Analytics Database, Frequency=week.6, infile=haverw.dat
Define a range inside the data range for OUT= dataset
Using the START=20040102 END=20041001 LIBNAME options.
Subset further: Using group="M*" LIBNAME option

Obs	DATE	FA	FM1
1	31DEC2003	7302.9	1298.2
2	07JAN2004	7351.2	1294.3
3	14JAN2004	7378.5	1286.8
4	21JAN2004	7434.7	1296.7
5	28JAN2004	7492.4	1305.1
6	04FEB2004	7510.4	1303.1
7	11FEB2004	7577.8	1309.1
8	18FEB2004	7648.7	1317.0
9	25FEB2004	7530.6	1321.1
10	03MAR2004	7546.7	1316.2
11	10MAR2004	7602.0	1312.7
12	17MAR2004	7603.0	1324.0
13	24MAR2004	7625.5	1337.6
14	31MAR2004	7637.3	1337.9
15	07APR2004	7667.4	1327.3
16	14APR2004	7692.5	1321.8
17	21APR2004	7698.4	1322.2
18	28APR2004	7703.8	1331.6
19	05MAY2004	7686.8	1342.5
20	12MAY2004	7734.6	1325.5
21	19MAY2004	7695.8	1330.1
22	26MAY2004	7704.7	1337.7
23	02JUN2004	7715.1	1329.0
24	09JUN2004	7754.0	1324.4
25	16JUN2004	7753.2	1336.4
26	23JUN2004	7796.2	1345.8
27	30JUN2004	7769.8	1351.4
28	07JUL2004	7852.3	1330.1
29	14JUL2004	7852.8	1326.3
30	21JUL2004	7854.7	1323.5
31	28JUL2004	7859.5	1340.6
32	04AUG2004	7847.9	1337.3
33	11AUG2004	7888.7	1340.1
34	18AUG2004	7851.8	1347.3
35	25AUG2004	7890.0	1360.8
36	01SEP2004	7906.2	1353.7
37	08SEP2004	7962.7	1338.3
38	15SEP2004	7982.1	1345.6
39	22SEP2004	7987.9	1359.7
40	29SEP2004	7949.5	1366.0

The following statements specify GROUP="E*" in the LIBNAME LIB3 statement:

```
libname lib3 sasehavr "%sysget(HAVER_DATA)"
        freq=week.6
        force=freq
        start=20040102
        end=20041001
        group="E*";

data hwoutwC;
   set lib3.haverw;
run;

title1 'Haver Analytics Database, Frequency=week.6, infile=haverw.dat';
title2 '   Define a range inside the data range for OUT= dataset';
title3 '   Using the START=20040102 END=20041001 LIBNAME options.';
title4 '   Subset further: Using group="E*" LIBNAME option';
proc print data=hwoutwC;
run;
```

Output 49.10.3 shows the output.

Output 49.10.3 Using the GROUP=E* Option and Defining a Range

Haver Analytics Database, Frequency=week.6, infile=haverw.dat
Define a range inside the data range for OUT= dataset
Using the START=20040102 END=20041001 LIBNAME options.
Subset further: Using group="E*" LIBNAME option

Obs	DATE	LICN
1	02JAN2004	552.8
2	09JAN2004	677.9
3	16JAN2004	490.8
4	23JAN2004	382.3
5	30JAN2004	406.3
6	06FEB2004	433.2
7	13FEB2004	341.6
8	20FEB2004	328.2
9	27FEB2004	342.1
10	05MAR2004	339.0
11	12MAR2004	312.1
12	19MAR2004	304.5
13	26MAR2004	296.8
14	02APR2004	304.2
15	09APR2004	350.7
16	16APR2004	335.0
17	23APR2004	313.7
18	30APR2004	283.2
19	07MAY2004	292.8
20	14MAY2004	297.1
21	21MAY2004	294.0
22	28MAY2004	304.1
23	04JUN2004	308.2
24	11JUN2004	312.4
25	18JUN2004	322.5
26	25JUN2004	318.7
27	02JUL2004	349.9
28	09JUL2004	444.5
29	16JUL2004	394.4
30	23JUL2004	315.7
31	30JUL2004	282.1
32	06AUG2004	291.5
33	13AUG2004	268.0
34	20AUG2004	272.1
35	27AUG2004	275.2
36	03SEP2004	273.7
37	10SEP2004	250.6
38	17SEP2004	275.8
39	24SEP2004	282.7
40	01OCT2004	279.6

Example 49.11: Using the OUTSELECT=ON Option to View the Key Selection Variables in a Haver Database

Suppose you want to select your time series based on geography codes or source codes. To construct your wildcard for selection, first run with the OUTSELECT=ON option to see the possible values for each selection key.

```
Libname lib1 sasehavr "%sysget(HAVER_DATA)"
        outselect=on ;

data validD1;
   set lib1.haverd;
run;

title1 'OUTSELECT=ON, Print the OUT= Data Set';
title2 'Shows the Values for Key Selection Variables:';
title3 'Name, Source, Geog1, Geog2, Shortsrc, Longsrc';
title4 'OUTSELECT=ON, the CONTENTS Procedure with Variable Names and Labels';
proc print data=validD1;
run;

proc contents data=validD1;
run;
```

Output 49.11.1 shows the output values for each key selection variable.

Output 49.11.1 OUTSELECT=ON Option Shows the Values for Key Selection Variables

OUTSELECT=ON, Print the OUT= Data Set
Shows the Values for Key Selection Variables:
Name, Source, Geog1, Geog2, Shortsrc, Longsrc
OUTSELECT=ON, the CONTENTS Procedure with Variable Names and Labels

Obs	NAME	SOURCE	GEOG1	GEOG2	SHORTSRC	LONGSRC	FCM10	FCM1M	FFED	FFP1D	FXAUS	TCC
1	NAME	SOURCE	GEOG1	GEOG2	SHORTSRC	LONGSRC						
2	FCM10	FRB	0000000		FRB	Federal Reserve Board						
3	FCM1M	UST	0000000		FRB	Federal Reserve Board						
4	FFED	FRB	0000000		FRB	Federal Reserve Board						
5	FFP1D	FRB	0000000		FRB	Federal Reserve Board						
6	FXAUS	FRB	0000000		FRBNY	Federal Reserve Bank of New York						
7	TCC	UST	0000000		TREASURY	U.S. Treasury						

If you also want to see a list of all the variables and their corresponding labels for this OUTSELECT=ON data set, you can run the CONTENTS Procedure.

Output 49.11.2 shows the contents of the output data set.

Output 49.11.2 OUTSELECT=ON Option Shows the Contents of HAVERD.DAT

Alphabetic List of Variables and Attributes				
#	Variable	Type	Len	Label
7	FCM10	Char	8	10-Year Treasury Note Yield at Constant Maturity (Avg, % p.a.)
8	FCM1M	Char	8	1-Month Treasury Bill Market Bid Yield at Constant Maturity (%)
9	FFED	Char	8	Federal Funds [Effective] Rate (% p.a.)
10	FFP1D	Char	8	1-Day AA Financial Commercial Paper (% per annum)
11	FXAUS	Char	8	Foreign Exchange Rate: Australia (US$/Australian$)
3	GEOG1	Char	8	DLXRECORD.Geography1
4	GEOG2	Char	8	DLXRECORD.Geography2
6	LONGSRC	Char	70	DLXRECORD.LongSource
1	NAME	Char	10	DLXRECORD.VarName
5	SHORTSRC	Char	10	DLXRECORD.ShortSourc
2	SOURCE	Char	6	DLXRECORD.Source
12	TCC	Char	8	Treasury: Closing Operating Cash Balance (Today, Mil.$)

Example 49.12: Selecting Variables Based on Short Source Key Code

Using the information from Example 49.11, you can now select time series by using selection keys such as the SHORT= , GEOG1=, or GEOG2= options. Since the short source values are nontrivial in database haverd, it is best in this case to use the SHORT= option. For more information about using geography codes as selection keys, see Output 49.13.1 for the GEOG1= option and Output 49.13.2 for the GEOG2= option.

```
Libname lib1 sasehavr "%sysget(HAVER_DATA)"
        short="GOLDMAN, FRB, CRB";
data validE2;
   set lib1.haverd;
   where date  between '18jan2005'd and '29mar2005'd;
run;

title1 'SHORT= option list: GOLDMAN, FRB, CRB';
title2 'Should contain these time series:';
title3 'FCM10, FCM1M, FFED, FFP1D';
title4 'SHORT= option, Print the OUT= ValidE2 Data Set';
proc print data=validE2;
run;

title4 'SHORT= option, Print the Contents of OUT= ValidE2 Data Set';
proc contents data=validE2;
run;
```

Output 49.12.1 shows the output for the SHORT= option.

Output 49.12.1 SHORT= Option Shows the Selected Variables

SHORT= option list: GOLDMAN, FRB, CRB
Should contain these time series:
FCM10, FCM1M, FFED, FFP1D
SHORT= option, Print the OUT= ValidE2 Data Set

Obs	DATE	FCM10	FCM1M	FFED	FFP1D
1	18JAN2005	4.21	2.05	2.31	2.30
2	19JAN2005	4.20	1.95	2.19	2.22
3	20JAN2005	4.17	1.89	2.25	2.22
4	21JAN2005	4.16	2.02	2.26	2.19
5	24JAN2005	4.14	2.05	2.26	2.22
6	25JAN2005	4.20	2.13	2.29	2.22
7	26JAN2005	4.21	2.16	2.33	2.26
8	27JAN2005	4.22	2.16	2.39	2.30
9	28JAN2005	4.16	2.12	2.48	2.37
10	31JAN2005	4.14	2.06	2.50	2.47
11	01FEB2005	4.15	2.23	2.40	2.47
12	02FEB2005	4.15	2.22	2.29	2.45
13	03FEB2005	4.18	2.18	2.49	2.46
14	04FEB2005	4.09	2.20	2.51	2.45
15	07FEB2005	4.07	2.27	2.50	2.47
16	08FEB2005	4.05	2.34	2.48	2.45
17	09FEB2005	4.00	2.34	2.50	2.45
18	10FEB2005	4.07	2.35	2.51	2.47
19	11FEB2005	4.10	2.36	2.50	2.48
20	14FEB2005	4.08	2.37	2.51	2.50
21	15FEB2005	4.10	2.40	2.53	2.54
22	16FEB2005	4.16	2.39	2.48	2.45
23	17FEB2005	4.19	2.40	2.50	2.47
24	18FEB2005	4.27	2.39	2.51	2.45
25	21FEB2005	.	.	2.51	.
26	22FEB2005	4.29	2.43	2.57	2.49
27	23FEB2005	4.27	2.47	2.53	2.48
28	24FEB2005	4.29	2.48	2.55	2.52
29	25FEB2005	4.27	2.50	2.54	2.52
30	28FEB2005	4.36	2.51	2.52	2.58
31	01MAR2005	4.38	2.55	2.39	2.51
32	02MAR2005	4.38	2.54	2.48	2.44
33	03MAR2005	4.39	2.55	2.51	2.49
34	04MAR2005	4.32	2.56	2.50	2.46
35	07MAR2005	4.31	2.59	2.51	2.49
36	08MAR2005	4.38	2.61	2.49	2.47
37	09MAR2005	4.52	2.60	2.50	2.45
38	10MAR2005	4.48	2.60	2.52	2.49
39	11MAR2005	4.56	2.60	2.51	2.48
40	14MAR2005	4.52	2.62	2.59	2.53
41	15MAR2005	4.54	2.70	2.61	2.60
42	16MAR2005	4.52	2.68	2.57	2.50
43	17MAR2005	4.47	2.68	2.68	2.58
44	18MAR2005	4.51	2.70	2.70	2.68

Output 49.12.1 *continued*

SHORT= option list: GOLDMAN, FRB, CRB
Should contain these time series:
FCM10, FCM1M, FFED, FFP1D
SHORT= option, Print the OUT= ValidE2 Data Set

Obs	DATE	FCM10	FCM1M	FFED	FFP1D
45	21MAR2005	4.53	2.72	2.71	2.72
46	22MAR2005	4.63	2.77	2.72	2.68
47	23MAR2005	4.61	2.72	2.73	2.69
48	24MAR2005	4.60	2.70	2.75	2.62
49	25MAR2005	.	.	2.80	2.59
50	28MAR2005	4.64	2.69	2.79	2.79
51	29MAR2005	.	.	.	2.76

If you also want to see a list of all the variables and their corresponding labels for this data set, you can run the CONTENTS Procedure.

Output 49.12.2 shows the output.

Output 49.12.2 SHORT= Option Shows the Contents of the validE2 Data Set

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Label
1	DATE	Num	8	DATE9.	Date of Observation
2	FCM10	Num	8		10-Year Treasury Note Yield at Constant Maturity (Avg, % p.a.)
3	FCM1M	Num	8		1-Month Treasury Bill Market Bid Yield at Constant Maturity (%)
4	FFED	Num	8		Federal Funds [Effective] Rate (% p.a.)
5	FFP1D	Num	8		1-Day AA Financial Commercial Paper (% per annum)

Example 49.13: Selecting Variables Based on Geography Key Codes

Since the haverd database did not have interesting geography codes, the following statements access the INTWKLY database by using its more complete geography key codes to select the desired time series from the specified geography codes:

```
Libname lib1 sasehavr "%sysget(HAVER_DATA_NEW)"
        outselect=on
   keep="R273RF3,X924USBE,R023DF,R273G1,F023A,F158FBS,F023ACR,X156VEB,F023ACE";

data valid1(keep=NAME SOURCE GEOG1 GEOG2 SHORTSRC LONGSRC);
   set lib1.intwkly;
run;

title1 'OUTSELECT=ON, Print the OUT= Data Set';
title2 'Shows the Values for Key Selection Variables:';
title3 'Name, Source, Geog1, Geog2, Shortsrc, Longsrc';
title4 'OUTSELECT=ON, the CONTENTS Procedure with Variable Names and Labels';
proc print data=valid1;
```

```
run;

Libname lib2 sasehavr "%sysget(HAVER_DATA_NEW)"
   geog1="156";

data valid2(
   keep=date R273RF3 X924USBE R023DF R273G1 F023A F158FBS F023ACR X156VEB F023ACE);
   set lib2.intwkly;
run;

title1 'Only one GEOG1 Code, 156, contains time series X156VEB';
title2 'Select Geography Code 1 Option:';
title3 'GEOG1= option';
title4 'Only Time Series X156VEB has Geog1 = 156';

proc contents
   data=valid2;
run;

Libname lib3 sasehavr "%sysget(HAVER_DATA_NEW)"
   geog2="299";

data valid3(
   keep=date R273RF3 X924USBE R023DF R273G1 F023A F158FBS F023ACR X156VEB F023ACE);
   set lib3.intwkly;
run;

title1 'Only one GEOG2 Code, 299, contains time series X156VEB';
title2 'Select Geography Code 2 Option:';
title3 'GEOG2= option';
title4 'Only Time Series X156VEB has Geog2 = 299';

proc contents
   data=valid3;
run;

title1 'Compare GEOG1 Code 156';
title2 'Over nonmissing values range';
title3 'With GEOG2 Code 299';
title4 'Over nonmissing values range';

proc compare listall briefsummary criterion=1.0e-5
   base=valid2(
   where=( date between '09jan1998'd and '28dec2007'd ))
   compare=valid3(
   where=( date between '09jan1998'd and '28dec2007'd ));
run;
```

Output 49.13.1, Output 49.13.2, Output 49.13.3, and Output 49.13.4 show the output.

Output 49.13.1 OUTSELECT=ON Option Shows the Values for Key Selection Variables

OUTSELECT=ON, Print the OUT= Data Set
Shows the Values for Key Selection Variables:
Name, Source, Geog1, Geog2, Shortsrc, Longsrc
OUTSELECT=ON, the CONTENTS Procedure with Variable Names and Labels

Obs	NAME	SOURCE	GEOG1	GEOG2	SHORTSRC	LONGSRC
1	NAME	SOURCE	GEOG1	GEOG2	SHORTSRC	LONGSRC
2	F023A	STLF	023		ECB	European Central Bank
3	F023ACE	STLF	023		ECB	European Central Bank
4	F023ACR	STLF	023		ECB	European Central Bank
5	F158FBS	---	158		JMoF	Ministry of Finance
6	R023DF	---	023		ECB	European Central Bank
7	X156VEB	STLF	156	299	BOCAN	Bank of Canada
8	X924USBE	STLF	924	111	SAFE	China State Administration of Foreign Exchange

Output 49.13.2 Only One GEOG1 Code, 156, Contains Time Series X156VEB

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Label
1	DATE	Num	8	DATE9.	Date of Observation
2	X156VEB	Num	8		Canada: Venezuelan Bolivar Noon Exchange Rate (C$/Bolivar)

Output 49.13.3 Only One GEOG2 Code, 299, Contains Time Series X156VEB

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Label
1	DATE	Num	8	DATE9.	Date of Observation
2	X156VEB	Num	8		Canada: Venezuelan Bolivar Noon Exchange Rate (C$/Bolivar)

Output 49.13.4 Comparing GEOG1 and GEOG2 Access of INTWKLY Haver DLX Database

OUTSELECT=ON, Print the OUT= Data Set
Shows the Values for Key Selection Variables:
Name, Source, Geog1, Geog2, Shortsrc, Longsrc
OUTSELECT=ON, the CONTENTS Procedure with Variable Names and Labels

Obs	NAME	SOURCE	GEOG1	GEOG2	SHORTSRC	LONGSRC
1	NAME	SOURCE	GEOG1	GEOG2	SHORTSRC	LONGSRC
2	F023A	STLF	023		ECB	European Central Bank
3	F023ACE	STLF	023		ECB	European Central Bank
4	F023ACR	STLF	023		ECB	European Central Bank
5	F158FBS	---	158		JMoF	Ministry of Finance
6	R023DF	---	023		ECB	European Central Bank
7	X156VEB	STLF	156	299	BOCAN	Bank of Canada
8	X924USBE	STLF	924	111	SAFE	China State Administration of Foreign Exchange

Output 49.13.4 *continued*

Only one GEOG1 Code, 156, contains time series X156VEB
Select Geography Code 1 Option:
GEOG1= option
Only Time Series X156VEB has Geog1 = 156

The CONTENTS Procedure

Data Set Name	WORK.VALID2	**Observations**	2404
Member Type	DATA	**Variables**	2
Engine	V9	**Indexes**	0
Created	03/03/2015 16:40:40	**Observation Length**	16
Last Modified	03/03/2015 16:40:40	**Deleted Observations**	0
Protection		**Compressed**	NO
Data Set Type		**Sorted**	NO
Label			
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information

Data Set Page Size	65536
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	4062
Obs in First Data Page	2404
Number of Data Set Repairs	0
ExtendObsCounter	YES
Filename	C:\Users\saskff\AppData\Local\Temp\SAS Temporary Files_TD3696_D79286_\valid2.sas7bdat
Release Created	9.0401M3
Host Created	W32_7PRO

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Label
1	DATE	Num	8	DATE9.	Date of Observation
2	X156VEB	Num	8		Canada: Venezuelan Bolivar Noon Exchange Rate (C$/Bolivar)

Output 49.13.4 *continued*

Only one GEOG2 Code, 299, contains time series X156VEB
Select Geography Code 2 Option:
GEOG2= option
Only Time Series X156VEB has Geog2 = 299

The CONTENTS Procedure

Data Set Name	WORK.VALID3	**Observations**	682
Member Type	DATA	**Variables**	2
Engine	V9	**Indexes**	0
Created	03/03/2015 16:46:54	**Observation Length**	16
Last Modified	03/03/2015 16:46:54	**Deleted Observations**	0
Protection		**Compressed**	NO
Data Set Type		**Sorted**	NO
Label			
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information

Data Set Page Size	65536
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	4062
Obs in First Data Page	682
Number of Data Set Repairs	0
ExtendObsCounter	YES
Filename	C:\Users\saskff\AppData\Local\Temp\SAS Temporary Files_TD3696_D79286_\valid3.sas7bdat
Release Created	9.0401M3
Host Created	W32_7PRO

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Label
1	DATE	Num	8	DATE9.	Date of Observation
2	X156VEB	Num	8		Canada: Venezuelan Bolivar Noon Exchange Rate (C$/Bolivar)

Compare GEOG1 Code 156
Over nonmissing values range
With GEOG2 Code 299
Over nonmissing values range

```
The COMPARE Procedure
Comparison of WORK.VALID2 with WORK.VALID3
(Method=RELATIVE(2.22E-09), Criterion=0.00001)

NOTE: No unequal values were found. All values compared are exactly equal.
```

References

Haver Analytics (2010), *DLX API Programmer's Reference*, New York.

Haver Analytics (2012), *DLX Database Profile*, New York.
http://www.haver.com/databaseprofiles.html

Haver Analytics (2012), *Data Link Express*, New York.
http://www.haver.com/datalink.html

Chapter 50
The SASEQUAN Interface Engine

Contents

Overview: SASEQUAN Interface Engine **3480**
Getting Started: SASEQUAN Interface Engine **3480**
Syntax: SASEQUAN Interface Engine **3482**
The LIBNAME *libref* SASEQUAN Statement 3483
Details: SASEQUAN Interface Engine **3488**
Quandl API Key 3488
Available Sources That Provide Quandl Time Series Data 3488
Useful Lists for Easy Downloading of Quandl Time Series Data 3490
Available Time Series for Each Quandl Code 3491
SAS Output Data Set 3491
SAS OUTXML File 3491
SAS XML Map File 3492
Reading All Financial Ratios Data 3492
Examples: SASEQUAN Interface Engine **3499**
Example 50.1: Retrieving Historical Price Data for Oil India Limited 3499
Example 50.2: Retrieving Data by Using Three Quandl Codes 3501
Example 50.3: Selecting IBM and Google Quandl Codes for All Financial Ratios . . . 3503
Example 50.4: Retrieving Data for the JASDAQ-TOP20 Exchange Traded Fund (ETF) 3510
Example 50.5: Collapsing Data for the JASDAQ-TOP20 Exchange Traded Fund (ETF) 3512
Example 50.6: Transforming Data for the JASDAQ-TOP20 Exchange Traded Fund (ETF) 3513
Example 50.7: Reading Superset Data for Multiple Time Series 3514
References **3517**

Overview: SASEQUAN Interface Engine

The SASEQUAN interface engine enables SAS users to retrieve economic data from the Quandl website, which is hosted by Quandl. The Quandl website offers access to 8 million time series data sets from 400 sources in finance, economics, society, health, energy, demography, and more. These time series are updated at annual, quarterly, monthly, weekly, and daily intervals. The economic time series on the Quandl website contain observation or measurement periods that are associated with data values.

The SASEQUAN interface engine uses the LIBNAME statement to enable you to specify how to subset your Quandl data and how to collapse the selected time series to the same update frequency. You can then use the SAS DATA step to perform further subsetting and to store the resulting time series in a SAS data set. You can perform more analysis (if desired) either in the same SAS session or in a later session.

The SASEQUAN interface engine supports Linux X64 (LAX) and Windows. Although the SASEQUAN engine uses the Quandl API, it is not endorsed or certified by Quandl. By using the SASEQUAN interface engine, you are agreeing to comply with the Quandl terms of use, which are described on the web page at the following URL: **`https://www.quandl.com/about/terms`**.

Getting Started: SASEQUAN Interface Engine

You can query the Quandl data set to retrieve the observations or data values for a list of economic time series by specifying the Quandl code of the data set. The Quandl code consists of a source code and a table code for the data set that contains the time series that you want to read into SAS. You must also specify your unique Quandl API key (authentication token for unlimited access). To obtain your own unique API key, visit the Quandl website at the following URL: **`https://www.quandl.com/users/sign_up`**.

The Quandl API key is a 20-character mixed-case alphanumeric string, such as “abCDefghiJKLMn123456,” and is represented by ’XXXXXXXXXXXXXXXXXXXX’ in the APIKEY= option in the following example. In addition, the example URLs in this section and in the section “Details: SASEQUAN Interface Engine” on page 3488 use the same Quandl API key as the argument *your_quan_apikey*.

After you have your assigned Quandl API key and have agreed to the Quandl terms of use, you are almost ready to download Quandl data. Before you download, make sure you have the necessary rights to work with the data.

Now that your are informed about the terms of use of the Quandl data, you can use your Quandl API key to access the Quandl data, as shown in the following example.

The statements that follow enable you to access the prices for oil from the National Stock Exchange of India’s time series data from September 1, 2013, to November 5, 2013, on a daily basis. The observations are sorted by the time ID variable DATE. The output is shown in Output 50.1.

```
options validvarname=any;
title 'Retrieve Data for Oil India Limited Prices';
libname _all_ clear;

libname quan sasequan "%sysget(QUANDL)"
   OUTXML=oiltd
   XMLMAP="%sysget(QUANDL)oiltd.map"
   APIKEY='XXXXXXXXXXXXXXXXXXX'
   IDLIST='NSE/OIL';

data oil_gsa;
   set quan.oiltd;
run;

proc contents data=oil_gsa; run;
proc print data=oil_gsa(firstobs=1328 obs=1342); run;
```

Figure 50.1 Oil India Limited Prices: Oil_Gsa (FIRSTOBS=1328 OBS=1342)

Retrieve Data for Oil India Limited Prices

Obs	date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
1328	2015-02-02	536.20	540.90	530.25	534.00	533.25	201704	1077.25
1329	2015-02-03	539.80	541.00	526.25	531.50	531.35	923694	4910.35
1330	2015-02-04	541.00	550.45	536.40	545.50	548.75	485793	2644.40
1331	2015-02-05	548.85	549.00	538.25	540.50	540.05	877473	4742.75
1332	2015-02-06	536.50	552.90	536.50	545.35	547.00	358329	1962.28
1333	2015-02-09	545.00	553.75	530.00	540.00	543.00	608323	3332.38
1334	2015-02-10	540.00	546.45	527.00	531.45	530.85	326785	1759.67
1335	2015-02-11	532.00	536.40	529.10	530.30	530.95	116276	618.56
1336	2015-02-12	534.65	536.00	528.00	531.95	531.65	189407	1006.99
1337	2015-02-13	521.00	525.90	495.10	504.00	500.20	895268	4542.81
1338	2015-02-16	505.00	513.90	495.00	495.00	499.00	379163	1909.42
1339	2015-02-18	501.80	506.50	494.40	500.95	501.10	261958	1314.47
1340	2015-02-19	503.30	506.00	494.15	497.00	497.30	161816	806.24
1341	2015-02-20	499.00	502.90	493.00	494.30	494.40	220134	1092.32
1342	2015-02-23	500.00	500.00	485.20	487.80	487.30	194121	952.37

The XML data that the Quandl website returns are placed in a file that is named by the OUTXML= option—in this case, OILTD1.xml. Note that the SASEQUAN engine appends a numeral to the XML filename, and the file extension (.xml) is excluded from the filename that appears in the OUTXML= option. This XML data file resides in the location that is given inside the string enclosed in double quotation marks in the SASEQUAN LIBNAME statement. So, in the preceding example, if the QUANDL environment variable is set to

```
C:\quandata\
```

then the downloaded XML file is located at

```
C:\quandata\OILTD1.xml
```

An equivalent LIBNAME statement that does not use any environment variables could be as follows:

```
libname quan sasequan "C:\quandata\"
   OUTXML=oiltd
   XMLMAP="C:\quandata\oiltd.map"
   APIKEY='XXXXXXXXXXXXXXXXXXXX'
   IDLIST='NSE/OIL';
```

You could also use either a SAS macro variable or a system environment variable to store the value of your Quandl API key so that the key does not appear explicitly in your SAS code. The XML map that is created is assigned the full pathname that the XMLMAP= option specifies. The SASEQUAN engine appends a numeral to the XML filename to indicate the position of the Quandl code in the IDLIST= option.

The IDLIST= option specifies the list of Quandl data sets (that contain time series) that you want to retrieve. This option accepts a string, enclosed in single quotation marks, that denotes a list of one or more Quandl data sets that you select (keep) in the resulting SAS data set. The result, OILTD, is named in the DATA step and is shown in Figure 50.1. The preceding example uses only one Quandl code, which is in the first position of the IDLIST= option, so the numeral 1 is appended to the name of the XML file, resulting in OIL1.xml.

It is more efficient to use the DATA step to store your Quandl data in a SAS data set and then refer to the SAS data set directly in your PROC PRINT or PROC GPLOT statement. You can also refer to the SASEQUAN libref directly, as in the statement

```
proc print data=quan.oiltd; run;
```

This statement uses the member name, OILTD, in the PROC PRINT statement; this usage corresponds to specifying the OUTXML=OILTD option. Although using this statement might seem easier, it is not as efficient, because every time you use the SASEQUAN libref, the Quandl interface engine reads the entire XML file into SAS again. So it is better to refer to the SAS data set repeatedly than to invoke the interface engine repeatedly. For another example that uses more SASEQUAN LIBNAME statement options, see the section "Reading All Financial Ratios Data" on page 3492.

Syntax: SASEQUAN Interface Engine

The SASEQUAN interface engine uses standard engine syntax to read the observations or data values for one or more Quandl data sets that can contain one or more time series in each data set. Table 50.1 summarizes the options that the SASEQUAN engine uses. In addition, there is one required option: API_KEY='*quan_api_key*'.

Table 50.1 Summary of LIBNAME *libref* SASEQUAN Options

Option	Description
APIKEY=	Specifies the required Quandl access key that enables you to access the data that the Quandl website provides.
AUTOMAP=	Specifies whether or not to overwrite the existing XML map file.
COLLAPSE=	Specifies the reporting frequency (lower frequency to collapse the output results to). The valid reporting frequencies are daily, weekly, monthly, quarterly, annual, and none.
FORMAT=	Specifies a file extension that indicates the type of file to retrieve. Only XML is supported for the SASEQUAN interface engine in SAS/ETS 13.2 (SAS 9.4M2).
FREQ=	Specifies the frequency of the selected data: daily, weekly, monthly, quarterly, and annual. When the IDLIST= option contains more than one Quandl code, the FREQ= option aggregates higher-frequency data series to lower-frequency time series (such as converting a monthly time series to an annual time series).
IDLIST=	Specifies a list of Quandl codes for Quandl data set codes for accessing Quandl time series data. To select more than one data set, list the unique Quandl codes, separated by commas. There is a limit of nine Quandl codes in the IDLIST= option.
PROXY=	Specifies the proxy server you want to use (if you have trouble connecting without specifying a proxy)
ROWS=	Specifies the maximum number of observations (rows) to return (integer between 1 and 100,000, optional; the default is 100,000)
SORT=	Specifies the order of the results in ascending or descending observation-date order. The valid sort arguments are *asc* and *desc*; the default is *asc*.
START=	Specifies the start date (trim_start) for the observation period (YYYY-MM-DD formatted string, optional; the default is 9999-12-31 (latest available))
END=	Specifies the end date (trim_end) for the observation period (YYYY-MM-DD formatted string, optional; the default is 1776-07-04 (earliest available))
TRANS=	Specifies the transformation method to be used for data transformation. The valid transformation arguments are DIFF, RDIFF, CUMUL, NORMALIZE, and NONE; the default is NONE. See Table 50.2 for formulas.

The LIBNAME *libref* SASEQUAN Statement

LIBNAME *libref* **SASEQUAN** '*physical-name*' *options* **;**

The LIBNAME statement assigns a SAS library reference (libref) to the physical path of the directory of Quandl data files in which the downloaded Quandl XML data are stored. The required *physical-name* argument specifies the location of the folder where your Quandl XML data reside. It should end with a backslash if you are in a Windows environment and a forward slash if you are in a UNIX environment.

You can specify the following *options* in the LIBNAME *libref* SASEQUAN statement.

APIKEY='*quan_apikey*'

specifies the Quandl authentication token or access key that enables you to access the data that the Quandl website provides. The Quandl access key is a 20-character mixed-case alphanumeric string, and it is required. It must be enclosed on single quotation marks. You can request your *quan_apikey* by visiting the website at the following URL:

https://www.quandl.com/users/sign_up

AUTOMAP=REPLACE | REUSE
specifies whether or not to overwrite the existing XML map file.

REPLACE specifies that the XML map file be overwritten, and ensures that the most current XML map that is generated by the SASEQUAN engine and named by the XMLMAP= option is used.

REUSE specifies that the XML map file not be overwritten, and ensures that a pre-existing XML map file that is named by the XMLMAP= option is used.

By default, AUTOMAP=REPLACE.

MAPREF=*quan_xmlmapref*
specifies the fileref to use for the map assignment. For an example of the SASEQUAN engine that uses the MAPREF= and XMLMAP= options in the FILENAME statement in order to assign a filename, as in the following, see the section "Reading All Financial Ratios Data" on page 3492:

```
FILENAME MyMap "U:\quan950\test\gstart.map";
```

You can use the MAPREF= and XMLMAP= options to control where the map resides, what you name the map, and how you refer to it with a fileref. You can use the OUTXML= option to name your XML data file. It is placed in the folder that is designated by *physical-name,* and you can reference it by using the myLib libref in your SASEQUAN LIBNAME statement. This is shown in the section "Getting Started: SASEQUAN Interface Engine" on page 3480, inside the DATA step in the SET statement. The SET statement reads observations from the input data set myLib.GSTART and stores them in a SAS data set named All_Fin.

OUTXML=*quan_xmlfile*
specifies the name of the file where the XML data that are returned from the Quandl website are stored. Each Quandl code that is listed in the IDLIST= option is given a positional numeral: 1 for the first code in the IDLIST, 2 for the second code in the IDLIST, and so on. The engine appends this numeral to the filename of the XML of each data set that the website returns. When all the XML files are retrieved, the data are merged into a SAS data set. When only one Quandl code is used in the IDLIST= option, the filename has the numeral 1 appended to the OUTXML filename.

XMLMAP=*quan_xmlmapfile*
specifies the fully qualified name of the location where the XML map file is automatically stored.

COLLAPSE=DAILY | WEEKLY | MONTHLY | QUARTERLY | ANNUAL | NONE
specifies the frequency to which you want to collapse the reporting frequency.

DAILY collapses the report to a daily frequency.

WEEKLY collapses the report to a weekly frequency.

MONTHLY collapses the report to a monthly frequency.

QUARTERLY collapses the report to a quarterly frequency.

ANNUAL collapses the report to an annual frequency.

NONE does not collapse the report.

COLLAPSE= is optional. By default, COLLAPSE=NONE when IDLIST=option specifies one Quandl code, but when the IDLIST= option specifies more than one Quandl code, the default for the collapse frequency is set to the same frequency that is specified in the FREQ= option.

The Quandl frequency-collapsing feature reports the native (higher-frequency) time series at a lower-frequency (the collapse frequency). When you collapse the frequency of a data set, Quandl returns the last observation for the given period. So if you collapse a daily data set to monthly, you get a sample of the original data set in which the observation for each month is the last data point available for that month. When you specify more than one Quandl code in the IDLIST= option, it is important to check that the *from* date and *to* date of every selected series use the same fiscal year, so that the reporting interval of the merged date values from all the data sets aligns to the same date for the first observation in the range. For example, if multiple Quandl codes are listed in the IDLIST= option, some annual time series have *from* dates that start in January, and some annual time series have *from* dates that start in June, then the merged data set will have observation dates reported for both January and June (if COLLAPSE=NONE), resulting in a semiannual interval instead of an annual interval in the merged data. To preserve the annual frequency, specify COLLAPSE=ANNUAL so that each annual time series aligns to the appropriate annual date in the merged data set. The COLLAPSE= option is applied to each Quandl data set that is specified in the IDLIST= option, so that when the data sets are merged, the reporting frequency is equal to the COLLAPSE= frequency. The resulting merged SAS data set contains the same data as a Quandl superset that is created from the same Quandl codes in the IDLIST= option. Quandl supersets are being deprecated by QUANDL but already existing supersets are still supported. Supersets are mentioned at the following URL: **https://www.quandl.com/help/supersets**.

NOTE: COLLAPSE=MONTHLY reports the daily, weekly, and monthly native frequencies of the time series at a monthly frequency (collapse frequency). If an annual native frequency time series is specified in the IDLIST= option, then it will not be selected when COLLAPSE=MONTHLY is specified. Only the time series that have native frequencies higher than the reporting frequency specified in the COLLAPSE= option are selected.

NOTE: It is highly recommended that you use the COLLAPSE= option when more than one Quandl code is specified in the IDLIST= option.

CAUTION: If the COLLAPSE= NONE is specified then undesirable time intervals can occur when more than one Quandl code is specified in the IDLIST= option.

FORMAT=XML

specifies the format of the file to be received from the Quandl website. Although Quandl can report data in many formats, the SASEQUAN engine for SAS 9.4M2 (SAS/ETS 13.2) supports only the XML format.

FREQ=DAILY | WEEKLY | MONTHLY | QUARTERLY | ANNUAL
specifies a lower frequency to aggregate values to. The FREQ= option also selects only those time series that aggregate to the specified frequency. In Quandl data, the highest frequency is daily, and the lowest frequency is annual.

DAILY	selects time series that aggregate to a daily frequency.
WEEKLY	selects time series that aggregate to a weekly frequency.
MONTHLY	selects time series that aggregate to a monthly frequency.
QUARTERLY	selects time series that aggregate to a quarterly frequency.
ANNUAL	selects time series that aggregate to an annual frequency.

The FREQ= option is not required, and the default value is the native frequency of the Quandl data set.

NOTE: An error is returned if you specify a frequency higher than the native frequency of the selected series. For example, if a series has the native frequency "Annual," it is not possible to aggregate the series to the higher "Monthly" frequency. To find the native frequency of an economic time series, enter the following URL in your web browser, and click on the Quandl code. The output gives you the list of available time series and their native frequencies and descriptions, including the "Frequency" field, which shows the native frequency of that time series.

https://www.quandl.com/resources/data-sources

You can use the data-sources URL to obtain the necessary information to request time series data for any of the listed data sources that you want to query.

NOTE: When you specify a single Quandl code in the IDLIST= option and the FREQ= option is not specified or is an empty string, the native frequency of the time series in that data set is used as the reporting frequency unless you specify the reporting frequency in the COLLAPSE= option. When you specify multiple data sets (and time series) in the IDLIST= option, the "Annual" frequency is used as the default frequency unless you specify the reporting frequency in the COLLAPSE= option. If any time series in the IDLIST= option have a lower native frequency than the specified frequency, then those time series are dropped from the list and excluded from the output.

IDLIST=*'quan_idlist'*
specifies the list of Quandl codes for the data sets that contain the time series to be included in the output SAS data set. There is a limit of nine Quandl codes in the IDLIST= option. This list is comma-delimited and must be enclosed in single quotation marks.

START=*'quan_startdate'*
specifies the start date for the time series in the format YYYY-MM-DD. START= is optional, and the default is 1776-07-04 (earliest available). The date must be enclosed in single quotation marks.

END=*'quan_enddate'*
specifies the end date for the time series in the format YYYY-MM-DD. END= is optional, and the default is 9999-12-31 (latest available). The date must be enclosed in single quotation marks.

PROXY="*quan_proxyserver*"
specifies the proxy server to use. PROXY= is optional. The specified proxy server is used only when a connection-refused error or a connection-timed-out error occurs. For *quan_proxyserver*, specify the server's HTTP address followed by a colon and the port number, and enclose that string with double quotation marks. For example, PROXY="http://inetgw.unx.sas.com:8118".

ROWS=*quan_rows*
specifies the maximum number of rows (time series observations) to return, which is an integer between 1 and 100,000. ROWS= is optional, and the default is ROWS=100000.

SORT=ASC | DESC
specifies the order in which to sort the date of time series observations.

ASC sorts time series observations in ascending date order.

DESC sorts time series observations in descending date order.

SORT= is optional, and the default is SORT=ASC.

TRANS=CUMUL | DIFF | NORMALIZE | RDIFF | NONE
TRANSFORMATION=CUMUL | DIFF | NORMALIZE | RDIFF | NONE
specifies the data value transformation.

CUMUL performs the cumulative function.

DIFF performs the difference function.

NORMALIZE performs the normalize function.

RDIFF performs the ratio difference function.

NONE does no transformation on the data.

TRANS= is optional, and the default is TRANS=NONE. The details of the arguments and the corresponding function formulas are presented in Table 50.2.

Table 50.2 Quandl Transformation Codes

Trans Code	Description	Formula
cumul	Cumulative	$x_t + x_{t-1} + \cdots + x_{t-N}$
diff	Difference	$x_t - x_{t-1}$
normalize	Normalize	$(\frac{x_t}{x_{t-N}}) \times 100$
rdiff	Ratio difference	$(\frac{x_t - x_{t-1}}{x_{t-1}})$

x_t is the value of series x at time period t. N is the number of observations per year, which differs by frequency: Daily ($N = 260$), Annual ($N = 1$), Monthly ($N = 12$), Quarterly ($N = 4$), and Weekly ($N = 52$).

Details: SASEQUAN Interface Engine

The SASEQUAN interface engine enables SAS users to access time series data that are stored in Quandl data sets that the Quandl website provides. Every Quandl data set is identified by a unique ID. For example, the Prague Stock Index is uniquely identified by the code PRAGUESE/PX. The unique code for any data set is always visible on the data set page beside the words "Quandl Code."

Quandl API Key

The API key that is used in these examples, abCDefghiJKLMn123456, is for demonstration purposes only. To successfully download data from the Quandl website, use your own Quandl API key, which is a 20-character mixed-case alphanumeric string. You can request your own API key by visiting the website at the following URL:

```
https://www.quandl.com/users/sign_up
```

Available Sources That Provide Quandl Time Series Data

To obtain a list of the available sources of Quandl data, enter the following URL in your web browser. Table 50.3 shows some of the financial sources available (in the order shown on the Quandl website), which you can see by clicking on the "Financial Data" category.

```
https://www.quandl.com/resources/data-sources?auth_token=your_quan_apikey
```

Table 50.3 Some Available Sources of Financial Data

Quandl Code (Source)	Name
OFDP	Open Financial Data Project
DMDRN	Damodaran Financial Data
YAHOO	Yahoo Finance
GOOG	Google Finance
PSYCH	PsychSignal
NASDAQ	NASDAQ
NSE	National Stock Exchange of India
NYX	NYSE Euronext
ICE	IntercontinentalExchange
SPDJ	S&P Dow Jones Indices
EUREX	Eurex
Quandl	Quandl
LIFFE	Euronext LIFFE
PXDATA	Price-Data.com
SHFE	Shanghai Futures Exchange
MGEX	Minneapolis Grain Exchange
CBOE	Chicago Board Options Exchange
SGX	Singapore Exchange
MX	Montréal Exchange
EUREKA	Eurekahedge
WFE	World Federation of Exchanges
ASX	Australian Securities Exchange
SIX	Swiss Exchange
WGC	World Gold Council
SANDP	Standard & Poor's
WSJ	The Wall Street Journal
BITCOIN	Bitcoin Charts
MYX	Bursa Malaysia
OSE	Osaka Securities Exchange
ABMI	Asian Bond Market Initiative
ODE	Osaka Dojima Commodity Exchange
AMMANSE	Amman Stock Exchange
BCHAIN	Block Chain
UCR	Unicorn Research Corporation
ML	Merrill Lynch
UIS	University of Stavanger
NIKKEI	Nikkei Group
WREN	Wren Research
MONEYTREE	MoneyTree Report
PUP	Policy Uncertainty Project
NASOMXNORDIC	NASDAQ OMX Nordic
PHILSE	Philippine Stock Exchange
THAISE	Stock Exchange of Thailand
BUDAPESTSE	Budapest Stock Exchange
RENCAP	Renaissance Capital

Useful Lists for Easy Downloading of Quandl Time Series Data

To get a list of Quandl codes for the available time series for a specific source, enter the following URL in your web browser and click on the category or the particular link for that source:

https://www.quandl.com/resources/useful-lists

Table 50.4 shows the list of Quandl codes for the Dow Jones Industrial Average.

Table 50.4 Quandl Codes for Dow Jones Industrial Average Constituents

Ticker Symbol	Quandl Code	Name
AXP	GOOG/NYSE_AXP	American Express
BA	GOOG/NYSE_BA	Boeing
CAT	GOOG/NYSE_CAT	Caterpillar
CSCO	GOOG/NASDAQ_CSCO	Cisco Systems
CVX	GOOG/NYSE_CVX	Chevron
DD	GOOG/NYSE_DD	DuPont
DIS	GOOG/NYSE_DIS	Walt Disney
GE	GOOG/NYSE_GE	General Electric
GS	GOOG/NYSE_GS	Goldman Sachs
HD	GOOG/NYSE_HD	The Home Depot
IBM	GOOG/NYSE_IBM	IBM
INTC	GOOG/NASDAQ_INTC	Intel
JNJ	GOOG/NYSE_JNJ	Johnson & Johnson
JPM	GOOG/NYSE_JPM	JPMorgan Chase
KO	GOOG/NYSE_KO	Coca-Cola
MCD	GOOG/NYSE_MCD	McDonald's
MMM	GOOG/NYSE_MMM	3M
MRK	GOOG/NYSE_MRK	Merck
MSFT	GOOG/NASDAQ_MSFT	Microsoft
NKE	GOOG/NYSE_NKE	Nike
PFE	GOOG/NYSE_PFE	Pfizer
PG	GOOG/NYSE_PG	Procter & Gamble
T	GOOG/NASDAQ_T	AT&T
TRV	GOOG/NYSE_TRV	Travelers
UNH	GOOG/NYSE_UNH	UnitedHealth Group
UTX	GOOG/NYSE_UTX	United Technologies
V	GOOG/NYSE_V	Visa
VZ	GOOG/NYSE_VZ	Verizon
WMT	GOOG/NYSE_WMT	Wal-Mart
XOM	GOOG/NYSE_XOM	ExxonMobil

Available Time Series for Each Quandl Code

Quandl data sets can contain different numbers of time series at various frequencies. To get the list of time series for a specific source, use the Quandl search capability by entering the following URL in your web browser and specify the source ID that corresponds to that Quandl code. For example, the following URL retrieves a list of all time series for the United Nations Office on Drugs and Crime (UNODC, source_ids=692):

```
https://www.quandl.com/search/*?source_ids=692&api_key=your_quan_apikey
```

A shortcut to this source is at the URL **https://www.quandl.com/resources/data-sources**.

Under International Organizations, you can see the UNODC code, which links to the list of available time series and their native frequencies and descriptions. You can use the data-sources URL to obtain the necessary information to request time series data for any of the listed data source that you want to query.

SAS Output Data Set

You can use a SAS DATA step to write the selected Quandl data to a SAS data set. This enables you to use SAS software to easily analyze the data. If you specify the name of the output data set in the DATA statement, the engine supervisor creates a SAS data set that has the specified name in either the SAS Work library or, if specified, the SAS User library.

The contents of the SAS data set include the date of each observation and the series name of each series that is read from the Quandl data source.

The SASEQUAN interface engine maintains the sort order, so the time series are sorted in the resulting SAS data set by the order that is specified in the SORT= option, by date (time ID), and by variable (time series item name).

You can use the PRINT and CONTENTS procedures to print your output data set and its contents. Alternatively, you can view your SAS output observations by opening the desired output data set in a SAS Explorer window. You can also use the SQL procedure with your SASEQUAN libref to create a custom view of your data.

SAS OUTXML File

The SAS XML (XML format) data that are returned from the Quandl website are placed in a file that is named by the OUTXML= option. The SASEQUAN interface engine creates a separate XML file for each Quandl code that is listed in the IDLIST= option. The engine numbers each data set's XML file in the order in which it appears in the IDLIST= option, so the first data set has a 1 concatenated to the filename, the second data set has a 2 concatenated to the filename, and so on. In instances of the IDLIST= option that contain more than one Quandl code, the variable names also have the same numeral concatenated to them. This naming convention enables the engine to merge all the selected time series into one SAS data set while preserving the identity of each time series. The SAS XML data file is placed in the location that is specified by the *physical-name* in your LIBNAME *libref* SASEQUAN statement, which is described in the section "The LIBNAME *libref* SASEQUAN Statement" on page 3483.

SAS XML Map File

The XML map that (by default) is automatically created is assigned the full pathname that is given by the XMLMAP= option in your LIBNAME *libref* SASEQUAN statement. The map file is either reused (not overwritten) if you specify AUTOMAP=REUSE or overwritten by a new map if you specify AUTOMAP=REPLACE (the default). The SASEQUAN interface engine invokes the XMLV2 engine to create the map and to read the data into SAS.

Reading All Financial Ratios Data

The following statements enable you to access the "All Financial Ratios Data" from the source Damodaran Financial Data (Quandl source code = DMDRN) for Google (GOOG) and IBM on an annual basis. The output is shown in Output 50.2.

```
options validvarname=any;
title 'Quandl Data: All Financial Ratios for Google and IBM';
LIBNAME myLib sasequan "%sysget(QUANDL)"
   OUTXML=gstart
   AUTOMAP=replace
   MAPREF=MyMap
   XMLMAP="%sysget(QUANDL)gstart.map"
   APIKEY='XXXXXXXXXXXXXXXXXXXX'
   IDLIST='DMDRN/GOOG_ALLFINANCIALRATIOS,DMDRN/IBM_ALLFINANCIALRATIOS'
   START='2003-12-31'
   END='2012-12-31'
   FORMAT=xml
   ;

data all_fin;
   set myLib.gstart ;
run;

proc contents data=all_fin; run;
proc print data=all_fin; run;
```

Figure 50.2 Quandl Data: All_Fin

Quandl Data: All Financial Ratios for Google and IBM

Obs	date	Number of Shares Outstanding_1	3-Year Regression Beta_1	3-year Standard Deviatio_4_1	Book Debt to Capital Ratio_1	Book Value of Equity_1	Book Value of Assets_1	Capital Expenditures_1	Cash_1
1	2003-12-31	235.0	0.00	0.0000	0.010834	.	871.5	176.8	334.7
2	2004-12-31	277.3	0.00	0.0000	0.000648	.	3313.4	319.0	2132.3
3	2005-12-31	297.2	.	.	0.000000	9419.0	10271.8	838.2	8034.2
4	2006-12-31	310.3	1.21	0.3342	0.000000	17039.8	18473.4	1902.8	11243.9
5	2007-12-31	313.3	1.32	0.4285	0.000000	22689.7	25335.8	2402.8	14218.6
6	2008-12-31	315.1	0.88	0.4044	0.000000	28238.9	31767.6	2358.5	15845.8
7	2009-12-31	317.8	0.98	0.4180	0.000000	36004.2	40496.8	809.9	24484.8
8	2010-12-31	321.3	0.85	0.2957	0.069710	46241.0	57851.0	4018.0	34975.0
9	2011-12-31	325.1	1.23	0.3073	0.067400	58145.0	72574.0	3438.0	44626.0

Quandl Data: All Financial Ratios for Google and IBM

Obs	Cash as Percentage of Fi_10_1	Cash as Percentage of Revenues_1	Cash as Percentage of To_12_1	Change in Non-Cash Worki_13_1	Correlation with the Market_1	Current PE Ratio_1	Depreciation_1	Dividend Yield_1	Dividends_1
1	0.00635	0.12536	.	-4.435	.	499.086	43.9	0	0
2	0.01691	0.40608	0.64354	143.910	.	310.686	128.5	0	0
3	0.05709	1.30880	0.78216	-114.693	.	92.689	256.8	0	0
4	0.05155	1.06026	0.60865	-202.355	0.45482	74.173	571.9	0	0
5	0.14919	0.85685	0.56121	-269.870	0.28005	22.671	807.7	0	0
6	0.08174	0.72702	0.49880	82.169	0.35608	36.583	1212.2	0	0
7	0.12667	1.03527	0.60461	319.377	0.53828	29.644	1240.0	0	0
8	0.16869	1.19283	0.60457	12.300	0.76091	23.971	1067.0	0	0
9	0.18890	1.17730	0.61490	111.130	0.80000	23.830	1396.0	0	0

Quandl Data: All Financial Ratios for Google and IBM

Obs	Earnings Before Interest_19_1	EBIT for Previous Period_1	Earnings Before Interest_21_1	Effective Tax Rate_1	Effective Tax Rate on Income_1	Enterprise Value_1	EV to Invested Capital Ratio_1	EV to Trailing Sales Ratio_1	EV to EBIT Ratio_1	EV to EBITDA Ratio_1
1	615.68	246.27	659.58	0.60000	.	52375.4	155.417	19.6170	85.069	79.4074
2	1120.07	534.05	1248.57	0.52320	.	123977.0	155.223	23.6102	110.687	99.2951
3	2107.30	1433.74	2364.10	0.31963	.	132704.3	95.829	14.2408	62.974	56.1331
4	3550.00	2602.68	4121.90	0.26685	.	206877.5	35.694	13.8169	58.275	50.1898
5	5084.40	3766.93	5892.10	0.25912	.	81085.5	9.572	3.8758	15.948	13.7617
6	6632.00	6004.02	7844.20	0.09469	.	178017.0	14.364	7.8499	26.842	22.6941
7	8312.20	6466.81	9552.20	0.22201	0.22202	168808.2	14.654	6.1263	20.308	17.6722
8	10381.00	8178.05	11448.00	0.21221	0.21221	172359.8	11.700	4.8198	16.603	15.0559
9	11742.00	9275.71	13138.00	0.21000	0.21000	191631.0	10.810	4.0300	16.320	14.5900

Figure 50.2 *continued*

Quandl Data: All Financial Ratios for Google and IBM

Obs	EV To Sales Ratio_1	Expected Growth in Earni_30_1	Expected Growth in Revenues_1	Free Cash Flow to Firm_1	Firm Value_1	Ratio of Fixed Assets to_34_1	Forward Earnings Per Share_1	Forward PE Ratio_1	Growth in Earnings Per Share_1	Previous Year Growth in_38_1	Hi-Lo Risk_1
1	35.7292	.	.	117.81	52710.1	0.21606	2.11	93.5071	.	2.3358	0.35502
2	38.8740	0.635	0.465	199.64	126109.3	0.11435	6.35	65.3323	.	0.8434	0.44223
3	21.6180	0.455	0.385	967.04	140738.5	0.09363	11.26	40.8952	.	0.7960	0.21485
4	19.5077	0.325	0.285	1474.14	218121.4	0.12966	16.06	40.9091	.	0.6549	0.26197
5	4.8864	0.230	0.220	2441.70	95304.1	0.15943	18.87	16.3037	.	0.5434	0.48656
6	8.1676	0.160	0.135	4775.55	193862.8	0.16475	21.52	28.8095	.	0.3058	0.37771
7	7.1376	0.165	0.125	6577.53	193293.0	0.11963	28.41	20.9071	0.845	0.0760	0.18527
8	5.8784	0.200	0.170	5214.75	207334.8	0.13412	38.80	16.6469	0.550	0.2261	0.15516
9	5.0600	0.155	0.185	7122.58	236257.0	0.13230	39.20	18.0500	0.365	0.3350	0.16370

Quandl Data: All Financial Ratios for Google and IBM

Obs	Insider Holdings_1	Institutional Holdings_1	Ratio of Intangible Asse_42_1	Invested Capital_1	Market Capitalization_1	Market Debt to Equity Ratio_1	Market Debt to Capital Ratio_1	Net Income_1	Net Margin_1
1	0.6130	0.1144	0.12117	337.0	52703.5	0.000125	0.000125	105.6	0.07204
2	0.0409	0.4728	0.05852	798.7	126107.4	0.000015	0.000015	405.9	0.12727
3	0.0160	0.5684	0.02704	1384.8	140738.5	0.000000	0.000000	1518.4	0.24735
4	0.0090	0.6124	0.10242	5795.9	218121.4	0.000000	0.000000	2940.7	0.27730
5	0.0100	0.5995	0.09076	8471.1	95304.1	0.000000	0.000000	4203.7	0.25333
6	0.0100	0.6210	0.18372	12393.1	193862.8	0.000000	0.000000	5299.3	0.24314
7	0.0100	0.6024	0.14020	11519.4	193293.0	0.000000	0.000000	6520.4	0.27570
8	0.0100	0.6309	0.12619	14731.0	203869.8	0.016996	0.016712	8505.0	0.29007
9	0.0100	0.8362	0.12300	17723.0	232053.0	0.018100	0.017800	9737.0	0.25690

Quandl Data: All Financial Ratios for Google and IBM

Obs	Non-Cash Working Capital_1	Non-Cash Working Capital_50_1	Payout Ratio_1	Price to Book Value Ratio_1	PE to Growth Ratio_1	Pre-Tax Operating Margin_1	Price to Sales Ratio_1	Reinvestment Amount_1	Reinvestment Rate_1	Revenues_1
1	-5.4	-0.003684	0	97.5629	.	0.42000	35.9530	128.46	0.52164	1465.9
2	222.6	0.069798	0	43.0533	4.89269	0.35121	39.5420	334.41	0.62618	3189.2
3	221.4	0.036067	0	14.9420	2.03711	0.34329	22.9268	466.71	0.32552	6138.6
4	491.3	0.046328	0	12.8007	2.28226	0.33475	20.5680	1128.55	0.43361	10604.9
5	1034.9	0.062366	0	4.2003	0.98572	0.30640	5.7433	1325.23	0.35181	16594.0
6	2030.3	0.093152	0	6.8651	2.28642	0.30428	8.8946	1228.47	0.20461	21795.6
7	1934.7	0.081803	0	5.3686	1.79663	0.35146	8.1729	-110.72	-0.01712	23650.6
8	56.0	0.001910	0	4.4089	1.19853	0.35405	6.9530	2963.30	0.36235	29321.0
9	437.0	0.011500	0	3.9900	1.54000	0.30980	6.1200	2153.13	0.23210	37905.0

Figure 50.2 *continued*

Quandl Data: All Financial Ratios for Google and IBM

Obs	Return on Capital_1	Return on Equity_1	Sales General and Admini_61_1	Stock Price_1	Total Debt_1	Trading Volume_1	Trailing 12-month Revenues_1	Trailing Net Income_1	Trailing PE Ratio_1	Trailing Revenues_1
1	0.73078	0.17524	268.3	197.30	6.6	5627185	2669.9	190.6	276.514	2669.9
2	0.66865	0.13857	611.6	414.86	1.9	7507209	5251.0	0.0	.	5251.0
3	1.03534	0.16121	1459.8	460.48	0.0	2531693	9318.6	2119.8	66.392	9318.6
4	0.44906	0.17258	2829.9	657.00	0.0	5350543	14972.8	3889.6	56.078	14972.8
5	0.44468	0.18527	4860.5	307.65	0.0	2831538	20921.2	4804.8	19.835	20921.2
6	0.48446	0.18766	6542.1	619.98	0.0	3434825	22677.7	6034.9	32.124	22677.7
7	0.56138	0.18110	6494.3	593.97	0.0	2764989	27554.8	7936.0	24.356	27554.8
8	0.55516	0.18393	8523.0	645.90	3465.0	3018131	35761.0	10075.0	20.235	35761.0
9	0.52340	0.16750	12975.0	707.38	4204.0	1992857	47544.0	10517.0	22.060	47544.0

Quandl Data: All Financial Ratios for Google and IBM

Obs	Value Line Beta_1	EV to Book Value Ratio_1	Number of Shares Outstanding_2	3-Year Regression Beta_2	3-year Standard Deviatio_73_2	Book Debt to Capital Ratio_2	Book Value of Equity_2	Book Value of Assets_2	Capital Expenditures_2	Cash_2
1	0.00	86.5235	1694.5	1.70	0.3289	0.45891	.	104457	4393	7647
2	0.00	43.0260	1645.6	1.82	0.3107	0.43526	.	109183	4368	10570
3	1.05	14.9420	1574.0	1.64	0.1858	0.40620	33098	105748	3842	13686
4	0.95	12.8007	1506.5	1.63	0.1988	0.44312	28506	103234	4362	10657
5	0.75	4.2003	1385.2	1.05	0.2235	0.55337	28470	120431	4630	16146
6	0.90	6.8651	1339.1	0.73	0.2266	0.71587	13465	109524	4171	12907
7	0.90	5.3686	1305.3	0.69	0.2214	0.53423	22755	109022	3447	13974
8	0.90	4.1712	1228.0	0.40	0.1460	0.55263	23172	113452	4185	11651
9	0.90	3.7900	1163.2	0.63	0.1459	0.60750	20236	116433	4108	11922

Quandl Data: All Financial Ratios for Google and IBM

Obs	Cash as Percentage of Fi_79_2	Cash as Percentage of Revenues_2	Cash as Percentage of To_81_2	Change in Non-Cash Worki_82_2	Correlation with the Market_2	Current PE Ratio_2	Depreciation_2	Dividend Yield_2	Dividends_2
1	0.04083	0.08237	.	253.577	1.03375	21.4948	4701	0.007336	1220.04
2	0.06860	0.11200	0.09681	-93.490	1.17155	15.1756	4915	0.009735	1316.48
3	0.08112	0.15017	0.12942	68.163	0.80000	17.1465	5188	0.012931	1888.80
4	0.06135	0.11657	0.10323	-146.379	0.76459	16.0391	4983	0.015960	2410.40
5	0.11256	0.16344	0.13407	-336.333	0.64430	10.3835	5201	0.025610	2770.40
6	0.06306	0.12455	0.11785	-382.777	0.63548	13.8453	5450	0.017700	3022.60
7	0.06737	0.14593	0.12818	75.923	0.80000	13.5063	4994	0.018500	3354.47
8	0.04791	0.11666	0.10270	176.586	0.80000	14.4647	4831	0.017000	3647.45
9	0.04790	0.11150	0.10240	-111.350	0.80000	13.7300	4815	0.019000	4129.36

Figure 50.2 *continued*

Quandl Data: All Financial Ratios for Google and IBM

Obs	Earnings Before Interest_88_2	EBIT for Previous Period_2	Earnings Before Interest_90_2	Effective Tax Rate_2	Effective Tax Rate on Income_2	Enterprise Value_2	EV to Invested Capital Ratio_2	EV to Trailing Sales Ratio_2	EV to EBIT Ratio_2	EV to EBITDA Ratio_2
1	14786.83	10352.41	19487.83	0.29989	.	179624.7	4.09644	1.93482	12.1476	9.2173
2	11291.11	7903.21	16206.11	0.30005	.	143520.0	3.40870	1.52069	12.7109	8.8559
3	9376.00	6533.10	14564.00	0.30321	.	155025.9	3.68644	1.73028	16.5343	10.6445
4	12621.00	8923.93	17604.00	0.29293	.	163049.9	4.02274	1.69529	12.9189	9.2621
5	15276.00	10983.90	20477.00	0.28097	.	127303.2	2.67455	1.20678	8.3335	6.2169
6	15938.00	11760.65	21388.00	0.26210	.	191787.3	5.56163	2.00755	12.0333	8.9671
7	17013.00	12592.34	22007.00	0.25984	0.25984	193448.6	5.54596	1.97229	11.3706	8.7903
8	18150.00	13650.07	22981.00	0.24793	0.24793	231528.6	5.76731	2.17504	12.7564	10.0748
9	20286.00	15313.70	25101.00	0.24510	0.24510	237147.2	5.98000	2.27000	11.6900	9.4500

Quandl Data: All Financial Ratios for Google and IBM

Obs	EV To Sales Ratio_2	Expected Growth in Earni_99_2	Expected Growth in Revenues_2	Free Cash Flow to Firm_2	Firm Value_2	Ratio of Fixed Assets to_103_2	Forward Earnings Per Share_2	Forward PE Ratio_2	Growth in Earnings Per Share_2	Previous Year Growth in_107_2	Hi-Lo Risk_2
1	2.01529	0.115	0.095	10406.83	187271.7	0.14062	5.31	18.4840	0.070	0.1159	0.10163
2	1.49045	0.095	0.075	8543.70	154090.0	0.13899	5.33	15.4184	0.060	0.1125	0.15940
3	1.70108	0.095	0.070	7810.94	168711.9	0.13008	6.22	15.6190	0.050	-0.0105	0.14284
4	1.78345	0.130	0.090	9691.31	173706.9	0.13988	7.64	13.2369	0.055	0.0481	0.15550
5	1.28868	0.145	0.110	11891.24	143449.2	0.12523	9.42	8.9342	0.075	0.1751	0.30649
6	1.85069	0.105	0.070	13422.43	204694.3	0.13061	10.18	12.8585	0.120	0.0852	0.23806
7	2.02018	0.130	0.065	14063.42	207422.6	0.12993	12.12	12.1089	0.145	-0.0521	0.11964
8	2.31830	0.120	0.090	14119.48	243179.6	0.12425	13.89	13.2383	0.160	0.1086	0.14358
9	2.22000	0.095	0.065	16132.05	249069.2	0.11920	15.07	12.7100	0.165	0.0700	0.08850

Quandl Data: All Financial Ratios for Google and IBM

Obs	Insider Holdings_2	Institutional Holdings_2	Ratio of Intangible Asse_111_2	Invested Capital_2	Market Capitalization_2	Market Debt to Equity Ratio_2	Market Debt to Capital Ratio_2	Net Income_2	Net Margin_2	Non-Cash Working Capital_2
1	0.01	0.5458	0.07175	43849	163639.7	0.14441	0.12619	7613	0.08541	6097
2	0.01	0.5384	0.09366	42104	131163.0	0.17480	0.14879	8643	0.08976	4701
3	0.01	0.5624	0.10500	42053	146070.9	0.15500	0.13420	8519	0.09348	4039
4	0.01	0.6406	0.14584	40532	151023.9	0.15019	0.13058	9416	0.10299	2815
5	0.01	0.5995	0.13611	47598	108175.2	0.32608	0.24590	10418	0.10546	4956
6	0.01	0.6032	0.19269	34484	170768.3	0.19867	0.16574	12334	0.11902	4899
7	0.01	0.5886	0.20824	34881	181322.6	0.14394	0.12583	13425	0.14020	3127
8	0.01	0.6010	0.25230	40145	214555.6	0.13341	0.11771	14833	0.14852	2681
9	0.01	0.5817	0.25430	39634	217749.2	0.14380	0.12570	15855	0.14830	5346

Figure 50.2 *continued*

Quandl Data: All Financial Ratios for Google and IBM

Obs	Non-Cash Working Capital_119_2	Payout Ratio_2	Price to Book Value Ratio_2	PE to Growth Ratio_2	Pre-Tax Operating Margin_2	Price to Sales Ratio_2	Reinvestment Amount_2	Reinvestment Rate_2	Revenues_2	Return on Capital_2
1	0.068405	0.16026	5.8728	1.86911	0.16590	1.83595	-54.42	-0.00526	89131	0.23609
2	0.048820	0.15232	4.4093	1.59744	0.11726	1.36212	-640.49	-0.08104	96293	0.18771
3	0.044319	0.22172	4.4133	1.80489	0.10288	1.60281	-1277.84	-0.19559	91134	0.15535
4	0.030791	0.25599	5.2980	1.23377	0.13805	1.65191	-767.38	-0.08599	91424	0.22017
5	0.050169	0.26592	3.7996	0.71610	0.15464	1.09505	-907.33	-0.08261	98786	0.23076
6	0.047274	0.24506	12.6824	1.31860	0.15380	1.64787	-1661.78	-0.14130	103630	0.34105
7	0.032655	0.24987	7.9685	1.03895	0.17767	1.89355	-1471.08	-0.11682	95758	0.36101
8	0.026845	0.24590	9.2593	1.20540	0.18174	2.14835	-469.41	-0.03439	99870	0.34002
9	0.050000	0.25400	10.7600	1.45000	0.18970	2.04000	-818.35	-0.05340	106916	0.38640

Quandl Data: All Financial Ratios for Google and IBM

Obs	Return on Equity_2	Sales General and Admini_130_2	Stock Price_2	Total Debt_2	Trading Volume_2	Trailing 12-month Revenues_2	Trailing Net Income_2	Trailing PE Ratio_2	Trailing Revenues_2	Value Line Beta_2	EV to Book Value Ratio_2
1	0.27322	22929	98.15	23632	1872600	92838	0.0	.	92838	1.05	3.63663
2	0.29055	24737	82.18	22927	4456000	94378	8643.0	15.1756	94378	1.10	2.92535
3	0.25739	27156	97.15	22641	4432500	89596	3225.9	45.2807	89596	1.05	3.02682
4	0.33032	26366	101.13	22683	11019149	96178	9860.2	15.3165	96178	0.95	3.39344
5	0.36593	26453	84.16	35274	6537607	105490	11859.0	9.1218	105490	0.90	2.25040
6	0.91600	29723	130.90	33926	8080103	95533	13038.0	13.0977	95533	0.90	4.31927
7	0.58998	26772	146.76	26100	5406125	98083	14389.0	12.6015	98083	0.85	4.24568
8	0.64013	27863	183.88	28624	5608725	106448	15623.0	13.7333	106448	0.85	4.69495
9	0.78350	29852	191.55	31320	4551170	104689	16260.0	13.3900	104689	0.85	4.83000

The SASEQUAN interface engine supports the XML format. The XML data that the Quandl website returns are placed in a file named by the OUTXML= option. The XML map that is automatically created is assigned the full pathname specified by the XMLMAP= option, and the fileref that is used for the map assignment is specified by the MAPREF= option. In the example at the beginning of this section, the SASEQUAN engine uses the MAPREF= and XMLMAP= options in the FILENAME statement to assign a filename:

```
FILENAME MyMap "U:\quan950\test\gstart.map";
```

You can use the MAPREF= and XMLMAP= options to control where the map resides, what you name the map, and how you refer to it with a fileref. You can use the OUTXML= option to name your XML data file; it is described in the section "SAS OUTXML File" on page 3491. The XML data file is placed in the folder that is designated by *physical-name,* which is described in the section "The LIBNAME *libref* SASEQUAN Statement" on page 3483. You can refer to your data by using the myLib libref in your SASEQUAN LIBNAME statement. The myLib libref is shown inside the DATA step in the SET statement. The SET statement reads observations from the input data set myLib.gstart and stores them in a SAS data set named All_Fin, as shown in Figure 50.2. You can also use the SAS DATA step to perform further processing and to store the resulting time series in a SAS data set; this process is described in the section "SAS Output Data Set" on page 3491.

To specify the list of Quandl data sets that you want to retrieve, use the IDLIST= option. This option accepts a string, enclosed in single quotation marks, that denotes a list of Quandl codes that specify data sets where the time series reside that you select for the resulting SAS data set. The Quandl codes are separated by commas, so valid Quandl codes cannot contain embedded commas or quotes. The All_Fin data set contains a total of 139 variables: the DATE variable and the 138 time series variables. The DMDRN/GOOG_ALLFINANCIALRATIOS data set includes 69 time series that are designated by _1 to indicate first order in the IDLIST= option; and the DMDRN/IBM_ALLFINANCIALRATIOS data set has 69 variables designated by _2 to indicate second order in the IDLIST= option. The observation range for both datasets is controlled by the START= and END= options. The All_Fin data set contains observations that range from December 31, 2003, to December 31, 2012, as specified by the START= and END= options. The frequency of the data is annual, which is the default frequency, because the FREQ= option was not specified.

NOTE: The "%20" is a special character for URL encoding of blanks. If the Quandl code that you name in the IDLIST= option contains a blank, you must use "%20" wherever the blank appears in the Quandl code. If the Quandl code contains an underscore, then you must use an underscore in the IDLIST= option. The underscore and the blank are not equivalent in the Quandl databases, so make sure that you use "%20" (URL-encoded space) to designate blank characters.

Examples: SASEQUAN Interface Engine

Example 50.1: Retrieving Historical Price Data for Oil India Limited

This example shows how to use one Quandl code, NSE/OIL, to retrieve historical prices for Oil India Limited, starting September 1, 2013, and ending November 5, 2013, with a daily frequency. The output is shown in Output 50.1.1.

```
options validvarname=any;

title 'Historical Prices for Oil India Limited';
libname _all_ clear;
libname mylib "U:\quan950\doc\";

libname myQoil sasequan "%sysget(QUANDL)"
   apikey='XXXXXXXXXXXXXXXXXXX'
   idlist='NSE/OIL'
   format=XML
   outXml=oil
   automap=replace
   mapref=MyMap
   xmlmap="%sysget(QUANDL)oil.map"
   start='2013-09-01'
   end='2013-11-05'
   freq='daily'
   collapse='daily'
   ;

data mylib.oilall;
   set myQoil.oil;
run;

proc contents data=mylib.oilall; run;
proc print data=mylib.oilall; run;
```

Output 50.1.1 Historical Prices for Oil India Limited

Historical Prices for Oil India Limited

Obs	date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
1	2013-09-02	435.95	441.65	427.20	431.00	431.45	174437	755.45
2	2013-09-03	439.90	439.90	427.00	428.50	429.05	199749	860.41
3	2013-09-04	435.00	435.00	426.15	429.50	429.45	790295	3396.42
4	2013-09-05	430.00	439.95	430.00	435.00	432.60	586678	2539.29
5	2013-09-06	437.00	450.00	433.30	445.25	445.15	543652	2402.79
6	2013-09-10	450.00	465.00	446.10	462.10	460.65	663553	2997.61
7	2013-09-11	462.00	485.00	461.00	466.00	466.70	371647	1733.05
8	2013-09-12	458.05	466.00	446.10	448.70	448.10	211533	959.45
9	2013-09-13	452.50	484.00	448.15	471.05	470.25	826546	3884.01
10	2013-09-16	483.70	484.00	458.80	476.00	467.00	335598	1593.84
11	2013-09-17	467.00	479.20	460.35	473.00	475.55	241830	1148.25
12	2013-09-18	471.20	481.85	471.20	480.00	479.70	182343	868.29
13	2013-09-19	485.00	499.00	476.00	491.10	493.75	457626	2236.70
14	2013-09-20	493.00	493.00	459.00	472.15	466.50	295333	1393.19
15	2013-09-23	466.75	487.00	464.00	480.00	480.40	273803	1302.58
16	2013-09-24	481.90	481.90	464.10	466.00	465.80	314456	1486.22
17	2013-09-25	467.90	473.30	466.10	470.15	470.35	738597	3472.11
18	2013-09-26	471.00	473.70	447.30	453.00	451.95	537088	2434.72
19	2013-09-27	456.70	462.00	450.10	452.00	454.30	345246	1571.16
20	2013-09-30	449.70	457.80	435.00	435.25	437.40	394564	1742.00
21	2013-10-01	437.15	449.35	432.00	449.00	447.90	308033	1367.86
22	2013-10-03	448.00	461.00	444.15	457.10	458.90	197974	898.93
23	2013-10-04	456.95	464.00	455.55	461.50	461.10	227214	1047.43
24	2013-10-07	464.90	471.45	450.00	468.00	464.40	240571	1098.48
25	2013-10-08	467.00	471.65	461.00	463.00	462.25	208627	964.45
26	2013-10-09	462.00	465.80	456.75	465.50	465.10	101852	472.35
27	2013-10-10	465.10	468.50	459.20	460.30	462.25	339738	1578.62
28	2013-10-11	465.00	468.70	457.00	467.50	463.25	213591	983.10
29	2013-10-14	464.65	467.90	461.00	464.10	463.95	125129	580.40
30	2013-10-15	464.00	471.80	456.55	459.30	460.55	407231	1877.01
31	2013-10-17	460.50	465.00	452.50	453.20	454.40	220366	1009.36
32	2013-10-18	457.00	465.95	457.00	465.00	464.55	185891	857.04
33	2013-10-21	465.00	471.90	458.70	468.00	468.85	114130	531.62
34	2013-10-22	468.85	473.20	461.15	465.70	466.65	198435	924.12
35	2013-10-23	463.05	469.50	451.40	456.00	457.65	469852	2152.30
36	2013-10-24	458.00	462.95	452.00	452.00	453.40	246085	1126.66
37	2013-10-25	458.00	460.05	450.00	454.00	454.65	272926	1238.47
38	2013-10-28	455.00	459.70	445.10	457.00	454.10	173547	785.17
39	2013-10-29	457.00	469.30	451.50	464.00	459.95	258106	1179.18
40	2013-10-30	460.20	467.80	453.95	463.50	463.25	301971	1391.67
41	2013-10-31	463.00	481.00	456.00	473.00	473.85	472301	2221.88
42	2013-11-01	470.10	481.00	464.50	480.00	475.05	318091	1495.83
43	2013-11-03	479.00	482.20	475.25	476.00	477.70	34250	163.85
44	2013-11-05	475.05	476.90	465.10	467.05	469.35	190319	894.87

Example 50.2: Retrieving Data by Using Three Quandl Codes

This example shows how to use three Quandl codes of different native frequencies to retrieve quarterly data for corporate profits after tax (FRED/CP), gross domestic product (FRED/GDP), and total consumer credit owned and securitized, outstanding (TOTALSL). The output is shown in Output 50.2.1.

```
title 'Retrieve Data for Three Time Series: FRED/CP, FRED/GDP, FRED/TOTALSL';
libname _all_ clear;
options validvarname=any;
libname mylib "U:\quan950\doc\";

libname myQ3 sasequan "%sysget(QUANDL)"
   OUTXML=fred3
   AUTOMAP=replace
   MAPREF=MyMap
   XMLMAP="%sysget(QUANDl)fred3.map"
   APIKEY='XXXXXXXXXXXXXXXXXXXX'
   IDLIST='FRED/CP,FRED/GDP,FRED/TOTALSL'
   FORMAT=xml
   START='2009-07-01'
   END='2013-07-01'
   FREQ='quarterly'
   COLLAPSE='quarterly'
   ;

data mylib.thrall;
   set myQ3.fred3;
   label Value_1 = "Corporate Profits After Tax";
   label Value_2 = "Gross Domestic Product, 1 Decimal";
   label Value_3 = "Total Consumer Credit Owned and Securitized, Outstanding";
run;

proc contents data=mylib.thrall; run;
proc print data=mylib.thrall label; run;
```

Output 50.2.1 Retrieve Data for Corporate Profits after Tax, Gross Domestic Product, Total Consumer Credit Owned and Securitized, Outstanding

Retrieve Data for Three Time Series: FRED/CP, FRED/GDP, FRED/TOTALSL

Obs	date	Corporate Profits After Tax	Gross Domestic Product, 1 Decimal	Total Consumer Credit Owned and Securitized, Outstanding
1	2009-09-30	1273.2	14384.1	2579.33
2	2009-12-31	1374.4	14566.5	2552.82
3	2010-03-31	1450.2	14681.1	2538.11
4	2010-06-30	1436.8	14888.6	2521.93
5	2010-09-30	1499.1	15057.7	2522.83
6	2010-12-31	1494.5	15230.2	2647.43
7	2011-03-31	1350.2	15238.4	2675.63
8	2011-06-30	1423.1	15460.9	2698.16
9	2011-09-30	1430.4	15587.1	2724.73
10	2011-12-31	1507.1	15785.3	2755.94
11	2012-03-31	1669.7	15956.5	2794.03
12	2012-06-30	1645.9	16094.7	2842.45
13	2012-09-30	1715.1	16268.9	2879.68
14	2012-12-31	1694.6	16332.5	2923.61
15	2013-03-31	1694.1	16502.4	2968.36
16	2013-06-30	1760.7	16619.2	3009.08
17	2013-09-30	1803.4	16872.3	3025.16

Example 50.3: Selecting IBM and Google Quandl Codes for All Financial Ratios

This example shows how to retrieve data for multiple time series that have different Quandl codes. The time series for each Quandl code are named identically in the original Quandl data set, and many exceed the limit of 32 bytes for a valid SAS variable name. Names such as "3-Year Standard Deviation of Stock Price," "Cash as Percentage of Firm Value," "Cash as Percentage of Revenues," and "Sales General and Administration Expenses" are truncated to 30 bytes before appending the suffix _1 to the Google data and _2 to the IBM data. Long variable names are represented using the long-name convention of "literal n variable name n".

The resulting SAS data set contains the merged contents of both Quandl data sets; the first Quandl data set variable names have the appended suffix _1, and the second Quandl data set variable names have the appended suffix _2. The SASEQUAN engine appends the appropriate suffix, _N, for every Quandl code that appears in the IDLIST= option, where N is the positional number of the Quandl code in the IDLIST= option. The Quandl codes are separated by commas in the IDLIST= option. The output is shown in Output 50.3.1.

```
title 'All Financial Ratios for Google and IBM';
libname _all_ clear;
options validvarname=any;
libname mylib "U:\quan950\doc\";

libname myTwo sasequan "%sysget(QUANDL)"
   OUTXML=tallfin
   AUTOMAP=replace
   MAPREF=MyMap
   XMLMAP="%sysget(QUANDL)tallfin.map"
   APIKEY='XXXXXXXXXXXXXXXXXXX'
   IDLIST='DMDRN/GOOG_ALLFINANCIALRATIOS,DMDRN/IBM_ALLFINANCIALRATIOS'
   FORMAT=xml
   START='2003-12-31'
   END='2012-12-31'
   ;

data mylib.tallFin;
   set myTwo.tallfin;
run;

/* Using PROC CONTENTS to obtain alphabetical order of variable names             */
proc contents data=mylib.tallFin out=newa(keep=name); run;

/* Create macro variables for each variable name and count the number of variables */
data _null_;
   set newa end=last;
   k+1;
   call symput('var'||trim(left(put(k,8.))),trim(name));
   if last then call symput('total',trim(left(put(k,8.))));
run;


%macro test;
   %do i=1 %to &total;
```

```
        %put var&i is: &&var&i;
    %end;

    /*-- Use the QUOTELST macro to put " "N around the elements of a variable list --*/

    %macro quotelst(strng,quote1=%str(%"),quote2=%str(%"N),delim=%str( ));
       %local i quotelst;
       %let i=1;
       %do %while(%length(%qscan(&strng,&i,|)) GT 0);
          %if %length(&quotelst) EQ 0
          %then %let quotelst=&quote1.%qscan(&strng,&i,|)&quote2;
          %else %let quotelst=&quotelst.&quote1.%qscan(&strng,&i,|)&quote2;
          %let i=%eval(&i + 1);
       %if %length(%qscan(&strng,&i,|)) GT 0 %then %let quotelst=&quotelst.&delim;
       %end;
       %unquote(&quotelst)
    %mend quotelst;

    /*-- macro variable longname&i holds all the N-literal SAS variable names, --*/
    /*-- which are now in alphabetical order                                   --*/

    %do i=1 %to &total;
        %let longname&i=%quotelst(&&var&i);
        %put LONGNAME&i is: &&longname&i;
    %end;

    %let total=%eval(&total-1); /* handle date variable separately */

    /*-- RETAIN statement gives the preferred order of variables,          --*/
    /*-- with date first, and the rest in alphabetical order....           --*/
    /*-- shuffling to get each MSFT series next to each matching IBM series--*/

    data mylib.orderFin;
       retain date;
       retain %do j=1 %to &total;
           &&longname&j
            %end;;
       set mylib.tallFin;
    run;

    proc print data=mylib.orderFin;
    run;
%mend test;

/* Invoke the macro */

%test;
```

Output 50.3.1 All Financial Ratios for Google and IBM, Side by Side, Shuffled Order

All Financial Ratios for Google and IBM

Obs	date	3-Year Regression Beta_1	3-Year Regression Beta_2	3-year Standard Deviatio_4_1	3-year Standard Deviatio_73_2	Book Debt to Capital Ratio_1	Book Debt to Capital Ratio_2	Book Value of Assets_1	Book Value of Assets_2	Book Value of Equity_1
1	2003-12-31	0.00	1.70	0.0000	0.3289	0.010834	0.45891	871.5	104457	.
2	2004-12-31	0.00	1.82	0.0000	0.3107	0.000648	0.43526	3313.4	109183	.
3	2005-12-31	.	1.64	.	0.1858	0.000000	0.40620	10271.8	105748	9419.0
4	2006-12-31	1.21	1.63	0.3342	0.1988	0.000000	0.44312	18473.4	103234	17039.8
5	2007-12-31	1.32	1.05	0.4285	0.2235	0.000000	0.55337	25335.8	120431	22689.7
6	2008-12-31	0.88	0.73	0.4044	0.2266	0.000000	0.71587	31767.6	109524	28238.9
7	2009-12-31	0.98	0.69	0.4180	0.2214	0.000000	0.53423	40496.8	109022	36004.2
8	2010-12-31	0.85	0.40	0.2957	0.1460	0.069710	0.55263	57851.0	113452	46241.0
9	2011-12-31	1.23	0.63	0.3073	0.1459	0.067400	0.60750	72574.0	116433	58145.0

All Financial Ratios for Google and IBM

Obs	Book Value of Equity_2	Capital Expenditures_1	Capital Expenditures_2	Cash as Percentage of Fi_10_1	Cash as Percentage of Fi_79_2	Cash as Percentage of Revenues_1	Cash as Percentage of Revenues_2	Cash as Percentage of To_12_1
1	.	176.8	4393	0.00635	0.04083	0.12536	0.08237	.
2	.	319.0	4368	0.01691	0.06860	0.40608	0.11200	0.64354
3	33098	838.2	3842	0.05709	0.08112	1.30880	0.15017	0.78216
4	28506	1902.8	4362	0.05155	0.06135	1.06026	0.11657	0.60865
5	28470	2402.8	4630	0.14919	0.11256	0.85685	0.16344	0.56121
6	13465	2358.5	4171	0.08174	0.06306	0.72702	0.12455	0.49880
7	22755	809.9	3447	0.12667	0.06737	1.03527	0.14593	0.60461
8	23172	4018.0	4185	0.16869	0.04791	1.19283	0.11666	0.60457
9	20236	3438.0	4108	0.18890	0.04790	1.17730	0.11150	0.61490

All Financial Ratios for Google and IBM

Obs	Cash as Percentage of To_81_2	Cash_1	Cash_2	Change in Non-Cash Worki_13_1	Change in Non-Cash Worki_82_2	Correlation with the Market_1	Correlation with the Market_2	Current PE Ratio_1	Current PE Ratio_2	Depreciation_1
1	.	334.7	7647	-4.435	253.577	.	1.03375	499.086	21.4948	43.9
2	0.09681	2132.3	10570	143.910	-93.490	.	1.17155	310.686	15.1756	128.5
3	0.12942	8034.2	13686	-114.693	68.163	.	0.80000	92.689	17.1465	256.8
4	0.10323	11243.9	10657	-202.355	-146.379	0.45482	0.76459	74.173	16.0391	571.9
5	0.13407	14218.6	16146	-269.870	-336.333	0.28005	0.64430	22.671	10.3835	807.7
6	0.11785	15845.8	12907	82.169	-382.777	0.35608	0.63548	36.583	13.8453	1212.2
7	0.12818	24484.8	13974	319.377	75.923	0.53828	0.80000	29.644	13.5063	1240.0
8	0.10270	34975.0	11651	12.300	176.586	0.76091	0.80000	23.971	14.4647	1067.0
9	0.10240	44626.0	11922	111.130	-111.350	0.80000	0.80000	23.830	13.7300	1396.0

Output 50.3.1 *continued*

All Financial Ratios for Google and IBM

Obs	Depreciation_2	Dividend Yield_1	Dividend Yield_2	Dividends_1	Dividends_2	EBIT for Previous Period_1	EBIT for Previous Period_2	EV To Sales Ratio_1	EV To Sales Ratio_2	EV to Book Value Ratio_1	EV to Book Value Ratio_2
1	4701	0	0.007336	0	1220.04	246.27	10352.41	35.7292	2.01529	86.5235	3.63663
2	4915	0	0.009735	0	1316.48	534.05	7903.21	38.8740	1.49045	43.0260	2.92535
3	5188	0	0.012931	0	1888.80	1433.74	6533.10	21.6180	1.70108	14.9420	3.02682
4	4983	0	0.015960	0	2410.40	2602.68	8923.93	19.5077	1.78345	12.8007	3.39344
5	5201	0	0.025610	0	2770.40	3766.93	10983.90	4.8864	1.28868	4.2003	2.25040
6	5450	0	0.017700	0	3022.60	6004.02	11760.65	8.1676	1.85069	6.8651	4.31927
7	4994	0	0.018500	0	3354.47	6466.81	12592.34	7.1376	2.02018	5.3686	4.24568
8	4831	0	0.017000	0	3647.45	8178.05	13650.07	5.8784	2.31830	4.1712	4.69495
9	4815	0	0.019000	0	4129.36	9275.71	15313.70	5.0600	2.22000	3.7900	4.83000

All Financial Ratios for Google and IBM

Obs	EV to EBIT Ratio_1	EV to EBIT Ratio_2	EV to EBITDA Ratio_1	EV to EBITDA Ratio_2	EV to Invested Capital Ratio_1	EV to Invested Capital Ratio_2	EV to Trailing Sales Ratio_1	EV to Trailing Sales Ratio_2	Earnings Before Interest_19_1	Earnings Before Interest_21_1	Earnings Before Interest_88_2
1	85.069	12.1476	79.4074	9.2173	155.417	4.09644	19.6170	1.93482	615.68	659.58	14786.83
2	110.687	12.7109	99.2951	8.8559	155.223	3.40870	23.6102	1.52069	1120.07	1248.57	11291.11
3	62.974	16.5343	56.1331	10.6445	95.829	3.68644	14.2408	1.73028	2107.30	2364.10	9376.00
4	58.275	12.9189	50.1898	9.2621	35.694	4.02274	13.8169	1.69529	3550.00	4121.90	12621.00
5	15.948	8.3335	13.7617	6.2169	9.572	2.67455	3.8758	1.20678	5084.40	5892.10	15276.00
6	26.842	12.0333	22.6941	8.9671	14.364	5.56163	7.8499	2.00755	6632.00	7844.20	15938.00
7	20.308	11.3706	17.6722	8.7903	14.654	5.54596	6.1263	1.97229	8312.20	9552.20	17013.00
8	16.603	12.7564	15.0559	10.0748	11.700	5.76731	4.8198	2.17504	10381.00	11448.00	18150.00
9	16.320	11.6900	14.5900	9.4500	10.810	5.98000	4.0300	2.27000	11742.00	13138.00	20286.00

All Financial Ratios for Google and IBM

Obs	Earnings Before Interest_90_2	Effective Tax Rate on Income_1	Effective Tax Rate on Income_2	Effective Tax Rate_1	Effective Tax Rate_2	Enterprise Value_1	Enterprise Value_2	Expected Growth in Earni_30_1	Expected Growth in Earni_99_2	Expected Growth in Revenues_1
1	19487.83	.	.	0.60000	0.29989	52375.4	179624.7	.	0.115	.
2	16206.11	.	.	0.52320	0.30005	123977.0	143520.0	0.635	0.095	0.465
3	14564.00	.	.	0.31963	0.30321	132704.3	155025.9	0.455	0.095	0.385
4	17604.00	.	.	0.26685	0.29293	206877.5	163049.9	0.325	0.130	0.285
5	20477.00	.	.	0.25912	0.28097	81085.5	127303.2	0.230	0.145	0.220
6	21388.00	.	.	0.09469	0.26210	178017.0	191787.3	0.160	0.105	0.135
7	22007.00	0.22202	0.25984	0.22201	0.25984	168808.2	193448.6	0.165	0.130	0.125
8	22981.00	0.21221	0.24793	0.21221	0.24793	172359.8	231528.6	0.200	0.120	0.170
9	25101.00	0.21000	0.24510	0.21000	0.24510	191631.0	237147.2	0.155	0.095	0.185

Output 50.3.1 *continued*

All Financial Ratios for Google and IBM

Obs	Expected Growth in Revenues_2	Firm Value_1	Firm Value_2	Forward Earnings Per Share_1	Forward Earnings Per Share_2	Forward PE Ratio_1	Forward PE Ratio_2	Free Cash Flow to Firm_1	Free Cash Flow to Firm_2	Growth in Earnings Per Share_1	Growth in Earnings Per Share_2
1	0.095	52710.1	187271.7	2.11	5.31	93.5071	18.4840	117.81	10406.83	.	0.070
2	0.075	126109.3	154090.0	6.35	5.33	65.3323	15.4184	199.64	8543.70	.	0.060
3	0.070	140738.5	168711.9	11.26	6.22	40.8952	15.6190	967.04	7810.94	.	0.050
4	0.090	218121.4	173706.9	16.06	7.64	40.9091	13.2369	1474.14	9691.31	.	0.055
5	0.110	95304.1	143449.2	18.87	9.42	16.3037	8.9342	2441.70	11891.24	.	0.075
6	0.070	193862.8	204694.3	21.52	10.18	28.8095	12.8585	4775.55	13422.43	.	0.120
7	0.065	193293.0	207422.6	28.41	12.12	20.9071	12.1089	6577.53	14063.42	0.845	0.145
8	0.090	207334.8	243179.6	38.80	13.89	16.6469	13.2383	5214.75	14119.48	0.550	0.160
9	0.065	236257.0	249069.2	39.20	15.07	18.0500	12.7100	7122.58	16132.05	0.365	0.165

All Financial Ratios for Google and IBM

Obs	Hi-Lo Risk_1	Hi-Lo Risk_2	Insider Holdings_1	Insider Holdings_2	Institutional Holdings_1	Institutional Holdings_2	Invested Capital_1	Invested Capital_2	Market Capitalization_1
1	0.35502	0.10163	0.6130	0.01	0.1144	0.5458	337.0	43849	52703.5
2	0.44223	0.15940	0.0409	0.01	0.4728	0.5384	798.7	42104	126107.4
3	0.21485	0.14284	0.0160	0.01	0.5684	0.5624	1384.8	42053	140738.5
4	0.26197	0.15550	0.0090	0.01	0.6124	0.6406	5795.9	40532	218121.4
5	0.48656	0.30649	0.0100	0.01	0.5995	0.5995	8471.1	47598	95304.1
6	0.37771	0.23806	0.0100	0.01	0.6210	0.6032	12393.1	34484	193862.8
7	0.18527	0.11964	0.0100	0.01	0.6024	0.5886	11519.4	34881	193293.0
8	0.15516	0.14358	0.0100	0.01	0.6309	0.6010	14731.0	40145	203869.8
9	0.16370	0.08850	0.0100	0.01	0.8362	0.5817	17723.0	39634	232053.0

All Financial Ratios for Google and IBM

Obs	Market Capitalization_2	Market Debt to Capital Ratio_1	Market Debt to Capital Ratio_2	Market Debt to Equity Ratio_1	Market Debt to Equity Ratio_2	Net Income_1	Net Income_2	Net Margin_1	Net Margin_2	Non-Cash Working Capital_1	Non-Cash Working Capital_2
1	163639.7	0.000125	0.12619	0.000125	0.14441	105.6	7613	0.07204	0.08541	-5.4	6097
2	131163.0	0.000015	0.14879	0.000015	0.17480	405.9	8643	0.12727	0.08976	222.6	4701
3	146070.9	0.000000	0.13420	0.000000	0.15500	1518.4	8519	0.24735	0.09348	221.4	4039
4	151023.9	0.000000	0.13058	0.000000	0.15019	2940.7	9416	0.27730	0.10299	491.3	2815
5	108175.2	0.000000	0.24590	0.000000	0.32608	4203.7	10418	0.25333	0.10546	1034.9	4956
6	170768.3	0.000000	0.16574	0.000000	0.19867	5299.3	12334	0.24314	0.11902	2030.3	4899
7	181322.6	0.000000	0.12583	0.000000	0.14394	6520.4	13425	0.27570	0.14020	1934.7	3127
8	214555.6	0.016712	0.11771	0.016996	0.13341	8505.0	14833	0.29007	0.14852	56.0	2681
9	217749.2	0.017800	0.12570	0.018100	0.14380	9737.0	15855	0.25690	0.14830	437.0	5346

Output 50.3.1 *continued*

All Financial Ratios for Google and IBM

Obs	Non-Cash Working Capital_119_2	Non-Cash Working Capital_50_1	Number of Shares Outstanding_1	Number of Shares Outstanding_2	PE to Growth Ratio_1	PE to Growth Ratio_2	Payout Ratio_1	Payout Ratio_2	Pre-Tax Operating Margin_1	Pre-Tax Operating Margin_2
1	0.068405	-0.003684	235.0	1694.5	.	1.86911	0	0.16026	0.42000	0.16590
2	0.048820	0.069798	277.3	1645.6	4.89269	1.59744	0	0.15232	0.35121	0.11726
3	0.044319	0.036067	297.2	1574.0	2.03711	1.80489	0	0.22172	0.34329	0.10288
4	0.030791	0.046328	310.3	1506.5	2.28226	1.23377	0	0.25599	0.33475	0.13805
5	0.050169	0.062366	313.3	1385.2	0.98572	0.71610	0	0.26592	0.30640	0.15464
6	0.047274	0.093152	315.1	1339.1	2.28642	1.31860	0	0.24506	0.30428	0.15380
7	0.032655	0.081803	317.8	1305.3	1.79663	1.03895	0	0.24987	0.35146	0.17767
8	0.026845	0.001910	321.3	1228.0	1.19853	1.20540	0	0.24590	0.35405	0.18174
9	0.050000	0.011500	325.1	1163.2	1.54000	1.45000	0	0.25400	0.30980	0.18970

All Financial Ratios for Google and IBM

Obs	Previous Year Growth in_107_2	Previous Year Growth in_38_1	Price to Book Value Ratio_1	Price to Book Value Ratio_2	Price to Sales Ratio_1	Price to Sales Ratio_2	Ratio of Fixed Assets to_103_2	Ratio of Fixed Assets to_34_1	Ratio of Intangible Asse_111_2	Ratio of Intangible Asse_42_1	Reinvestment Amount_1
1	0.1159	2.3358	97.5629	5.8728	35.9530	1.83595	0.14062	0.21606	0.07175	0.12117	128.46
2	0.1125	0.8434	43.0533	4.4093	39.5420	1.36212	0.13899	0.11435	0.09366	0.05852	334.41
3	-0.0105	0.7960	14.9420	4.4133	22.9268	1.60281	0.13008	0.09363	0.10500	0.02704	466.71
4	0.0481	0.6549	12.8007	5.2980	20.5680	1.65191	0.13988	0.12966	0.14584	0.10242	1128.55
5	0.1751	0.5434	4.2003	3.7996	5.7433	1.09505	0.12523	0.15943	0.13611	0.09076	1325.23
6	0.0852	0.3058	6.8651	12.6824	8.8946	1.64787	0.13061	0.16475	0.19269	0.18372	1228.47
7	-0.0521	0.0760	5.3686	7.9685	8.1729	1.89355	0.12993	0.11963	0.20824	0.14020	-110.72
8	0.1086	0.2261	4.4089	9.2593	6.9530	2.14835	0.12425	0.13412	0.25230	0.12619	2963.30
9	0.0700	0.3350	3.9900	10.7600	6.1200	2.04000	0.11920	0.13230	0.25430	0.12300	2153.13

All Financial Ratios for Google and IBM

Obs	Reinvestment Amount_2	Reinvestment Rate_1	Reinvestment Rate_2	Return on Capital_1	Return on Capital_2	Return on Equity_1	Return on Equity_2	Revenues_1	Revenues_2
1	-54.42	0.52164	-0.00526	0.73078	0.23609	0.17524	0.27322	1465.9	89131
2	-640.49	0.62618	-0.08104	0.66865	0.18771	0.13857	0.29055	3189.2	96293
3	-1277.84	0.32552	-0.19559	1.03534	0.15535	0.16121	0.25739	6138.6	91134
4	-767.38	0.43361	-0.08599	0.44906	0.22017	0.17258	0.33032	10604.9	91424
5	-907.33	0.35181	-0.08261	0.44468	0.23076	0.18527	0.36593	16594.0	98786
6	-1661.78	0.20461	-0.14130	0.48446	0.34105	0.18766	0.91600	21795.6	103630
7	-1471.08	-0.01712	-0.11682	0.56138	0.36101	0.18110	0.58998	23650.6	95758
8	-469.41	0.36235	-0.03439	0.55516	0.34002	0.18393	0.64013	29321.0	99870
9	-818.35	0.23210	-0.05340	0.52340	0.38640	0.16750	0.78350	37905.0	106916

Output 50.3.1 *continued*

All Financial Ratios for Google and IBM

Obs	Sales General and Admini_130_2	Sales General and Admini_61_1	Stock Price_1	Stock Price_2	Total Debt_1	Total Debt_2	Trading Volume_1	Trading Volume_2	Trailing 12-month Revenues_1	Trailing 12-month Revenues_2
1	22929	268.3	197.30	98.15	6.6	23632	5627185	1872600	2669.9	92838
2	24737	611.6	414.86	82.18	1.9	22927	7507209	4456000	5251.0	94378
3	27156	1459.8	460.48	97.15	0.0	22641	2531693	4432500	9318.6	89596
4	26366	2829.9	657.00	101.13	0.0	22683	5350543	11019149	14972.8	96178
5	26453	4860.5	307.65	84.16	0.0	35274	2831538	6537607	20921.2	105490
6	29723	6542.1	619.98	130.90	0.0	33926	3434825	8080103	22677.7	95533
7	26772	6494.3	593.97	146.76	0.0	26100	2764989	5406125	27554.8	98083
8	27863	8523.0	645.90	183.88	3465.0	28624	3018131	5608725	35761.0	106448
9	29852	12975.0	707.38	191.55	4204.0	31320	1992857	4551170	47544.0	104689

All Financial Ratios for Google and IBM

Obs	Trailing Net Income_1	Trailing Net Income_2	Trailing PE Ratio_1	Trailing PE Ratio_2	Trailing Revenues_1	Trailing Revenues_2	Value Line Beta_1	Value Line Beta_2
1	190.6	0.0	276.514	.	2669.9	92838	0.00	1.05
2	0.0	8643.0	.	15.1756	5251.0	94378	0.00	1.10
3	2119.8	3225.9	66.392	45.2807	9318.6	89596	1.05	1.05
4	3889.6	9860.2	56.078	15.3165	14972.8	96178	0.95	0.95
5	4804.8	11859.0	19.835	9.1218	20921.2	105490	0.75	0.90
6	6034.9	13038.0	32.124	13.0977	22677.7	95533	0.90	0.90
7	7936.0	14389.0	24.356	12.6015	27554.8	98083	0.90	0.85
8	10075.0	15623.0	20.235	13.7333	35761.0	106448	0.90	0.85
9	10517.0	16260.0	22.060	13.3900	47544.0	104689	0.90	0.85

The results before using the %test macro are shown in Figure 50.2, but you can use the test macro to generate a different order of the results. In Output 50.3.1, the TEST macro shuffles the financial ratios of company 1 (Google) with the ratios for company 2 (IBM) by listing their respective time series side by side. The %QUOTELST macro preserves the embedded blanks within variable names, whereas the RETAIN statement orders the time series for easy side-by-side comparison, as shown in Output 50.3.1.

Example 50.4: Retrieving Data for the JASDAQ-TOP20 Exchange Traded Fund (ETF)

This example shows how to use one Quandl code, GOOG/TYO_1551, to retrieve the price and yield performance of the JASDAQ-TOP20 Exchange Traded Fund (ETF) in Japan, starting January 1, 2014, and ending March 26, 2014, with a daily native frequency. The output is shown in Output 50.4.1.

```
options validvarname=any;

title 'JASDAQ-TOP20 ETF, Five Time Series';
libname _all_ clear;
libname mylib "U:\quan950\doc\";

libname myTOP20 sasequan "%sysget(QUANDL)"
   apikey='XXXXXXXXXXXXXXXXXXXX'
   idlist='GOOG/TYO_1551'
   format=XML
   outXml=jasdaq
   automap=replace
   mapref=MyMap
   xmlmap="%sysget(QUANDL)jasdaq.map"
   start='2014-01-01'
   end='2014-03-26'
   ;

data mylib.jasdaq;
   set myTOP20.jasdaq;
run;

proc contents data=mylib.jasdaq; run;
proc print data=mylib.jasdaq; run;
```

Output 50.4.1 JASDAQ-TOP20 ETF

JASDAQ-TOP20 ETF, Five Time Series

Obs	date	Open	High	Low	Close	Volume
1	2014-01-06	6200	6260	6020	6080	19310
2	2014-01-07	6020	6040	5900	5910	11470
3	2014-01-08	6020	6150	5990	6140	11300
4	2014-01-09	6100	6190	6070	6130	13050
5	2014-01-10	6150	6190	6060	6100	13980
6	2014-01-14	5950	6080	5920	5950	21170
7	2014-01-15	6100	6100	6000	6020	6630
8	2014-01-16	6070	6100	5900	5940	8990
9	2014-01-17	5910	6010	5850	6000	8130
10	2014-01-20	6020	6030	5930	5950	4650
11	2014-01-21	5980	6110	5960	6060	14470
12	2014-01-22	6100	6130	6060	6110	7630
13	2014-01-23	6110	6130	5940	5960	11220
14	2014-01-24	5820	6050	5700	5920	23250
15	2014-01-27	5560	5710	5560	5580	18300
16	2014-01-28	5600	5790	5580	5600	11680
17	2014-01-29	5780	5920	5760	5880	12980
18	2014-01-30	5680	5770	5610	5700	8040
19	2014-01-31	5780	5820	5550	5620	8440
20	2014-02-03	5570	5580	5400	5430	14610
21	2014-02-04	5140	5490	5080	5300	40470
22	2014-02-05	5500	5550	5100	5290	18350
23	2014-02-06	5340	5690	5340	5570	11950
24	2014-02-07	5670	5700	5550	5590	8070
25	2014-02-10	5570	5720	5570	5710	8370
26	2014-02-12	5880	5900	5770	5830	12160
27	2014-02-13	5820	5820	5650	5710	5240
28	2014-02-14	5660	5700	5480	5520	6990
29	2014-02-17	5530	5600	5400	5480	3340
30	2014-02-18	5580	5810	5580	5810	8150
31	2014-02-19	5720	5880	5690	5690	5060
32	2014-02-20	5700	5910	5610	5640	15270
33	2014-02-21	5700	5770	5640	5640	8550
34	2014-02-24	5660	5880	5650	5810	9100
35	2014-02-25	5900	6200	5850	6080	37800
36	2014-02-26	6010	6080	5940	5990	11740
37	2014-02-27	5960	5960	5740	5780	14700
38	2014-02-28	5830	5830	5650	5670	4420
39	2014-03-03	5610	5680	5470	5630	8530
40	2014-03-04	5530	5720	5530	5700	3820
41	2014-03-05	5780	5850	5750	5750	7380
42	2014-03-06	5770	5910	5700	5860	6770
43	2014-03-07	5870	5890	5850	5850	3740
44	2014-03-10	5830	5880	5830	5870	2880
45	2014-03-11	5880	5880	5800	5830	2140
46	2014-03-12	5710	5750	5710	5710	1860
47	2014-03-13	5680	5750	5630	5630	1830

Output 50.4.1 *continued*

JASDAQ-TOP20 ETF, Five Time Series

Obs	date	Open	High	Low	Close	Volume
48	2014-03-14	5550	5620	5420	5470	9030
49	2014-03-17	5470	5490	5330	5370	5090
50	2014-03-18	5500	5530	5380	5430	7630
51	2014-03-19	5460	5460	5250	5370	3130
52	2014-03-20	5300	5340	5130	5140	4780
53	2014-03-24	5170	5260	5140	5170	2300
54	2014-03-25	5100	5100	4885	4885	14080
55	2014-03-26	4860	5010	4850	4950	9100

Example 50.5: Collapsing Data for the JASDAQ-TOP20 Exchange Traded Fund (ETF)

This example shows how to collapse the daily data to a weekly interval by using the same Quandl code as in Example 50.4, GOOG/TYO_1551, to retrieve the price and yield performance of the JASDAQ-TOP20 Exchange Traded Fund (ETF) in Japan, starting January 1, 2014, and ending March 26, 2014, with a daily native frequency, collapsing to a weekly frequency by using the COLLAPSE= option. The output is shown in Output 50.5.1.

```
options validvarname=any;

title 'JASDAQ-TOP20 ETF, COLLAPSE=WEEKLY Option';
libname _all_ clear;
libname mylib "U:\quan950\doc\";

libname myTOP20 sasequan "%sysget(QUANDL)"
   apikey='XXXXXXXXXXXXXXXXXXXX'
   idlist='GOOG/TYO_1551'
   format=XML
   outXml=jasdaqW
   automap=replace
   mapref=MyMap
   xmlmap="%sysget(QUANDL)jasdaqw.map"
   start='2014-01-01'
   end='2014-03-26'
   collapse=weekly
   ;

data mylib.jasdaqW;
   set myTOP20.jasdaqW;
run;

proc contents data=mylib.jasdaqW; run;
proc print data=mylib.jasdaqW; run;
```

Output 50.5.1 JASDAQ-TOP20 ETF, with COLLAPSE=WEEKLY

JASDAQ-TOP20 ETF, COLLAPSE=WEEKLY Option

Obs	date	Open	High	Low	Close	Volume
1	2014-01-12	6150	6190	6060	6100	13980
2	2014-01-19	5910	6010	5850	6000	8130
3	2014-01-26	5820	6050	5700	5920	23250
4	2014-02-02	5780	5820	5550	5620	8440
5	2014-02-09	5670	5700	5550	5590	8070
6	2014-02-16	5660	5700	5480	5520	6990
7	2014-02-23	5700	5770	5640	5640	8550
8	2014-03-02	5830	5830	5650	5670	4420
9	2014-03-09	5870	5890	5850	5850	3740
10	2014-03-16	5550	5620	5420	5470	9030
11	2014-03-23	5300	5340	5130	5140	4780
12	2014-03-30	4860	5010	4850	4950	9100

Example 50.6: Transforming Data for the JASDAQ-TOP20 Exchange Traded Fund (ETF)

This example shows how to transform the daily data by using the diff transformation and the same Quandl code as in Example 50.4 and Example 50.5, GOOG/TYO_1551, to retrieve the price and yield performance of the JASDAQ-TOP20 Exchange Traded Fund (ETF) in Japan. Specify a range by using START='2014-01-01' and END='2014-03-26', a transformation function by using TRANS=DIFF, and a collapse frequency by using COLLAPSE=WEEKLY. The output is shown on Output 50.6.1.

```
options validvarname=any;

title 'JASDAQ-TOP20 ETF, TRANS=DIFF Option';
libname _all_ clear;
libname mylib "U:\quan950\doc\";

libname myTOP20 sasequan "%sysget(QUANDL)"
   apikey='XXXXXXXXXXXXXXXXXXXX'
   idlist='GOOG/TYO_1551'
   format=XML
   outXml=jasdaqX
   automap=replace
   mapref=MyMap
   xmlmap="%sysget(QUANDL)jasdaqX.map"
   start='2014-01-01'
   end='2014-03-26'
   collapse=weekly
   trans=diff
   ;

data mylib.jasdaqX;
   set myTOP20.jasdaqX;
```

```
run;

proc contents data=mylib.jasdaqX; run;
proc print data=mylib.jasdaqX; run;
```

Output 50.6.1 JASDAQ-TOP20 ETF, Weekly Data with TRANS=DIFF

JASDAQ-TOP20 ETF, TRANS=DIFF Option

Obs	date	Open	High	Low	Close	Volume
1	2014-01-19	-240	-180	-210	-100	-5850
2	2014-01-26	-90	40	-150	-80	15120
3	2014-02-02	-40	-230	-150	-300	-14810
4	2014-02-09	-110	-120	0	-30	-370
5	2014-02-16	-10	0	-70	-70	-1080
6	2014-02-23	40	70	160	120	1560
7	2014-03-02	130	60	10	30	-4130
8	2014-03-09	40	60	200	180	-680
9	2014-03-16	-320	-270	-430	-380	5290
10	2014-03-23	-250	-280	-290	-330	-4250
11	2014-03-30	-440	-330	-280	-190	4320

Example 50.7: Reading Superset Data for Multiple Time Series

This example shows how to read data from a Quandl superset by using the Quandl code TAMMER1/6KY to retrieve oil, gold, and stock prices. This superset is described at the following URL: **https://www.quandl.com/TAMMER1/6KY-GoldOil-Gold-and-Stocks**.

There are three superset columns (Oil, Gold, and S&P500), taken from three different Quandl data sets: DOE/_RWTC, BBK01/_WT5511, and YAHOO/_INDEX_GSPC, respectively. Because the Oil, Gold, and S&P500 columns are all from a daily native frequency data set, you can use the collapse frequency "Annual" to minimize the missing values in the output. In the following example, specify a range by using START='1961-12-31' and END='2014-03-27', and specify a collapse frequency by using COLLAPSE=ANNUAL. The output is shown in Output 50.7.1.

```
options validvarname=any;

title 'TAMMER1/6KY Superset with Multiple Time Series Using the COLLAPSE= Option';
libname _all_ clear;
libname mylib "U:\quan950\doc\";

libname myTAM sasequan "%sysget(QUANDL)"
   apikey='XXXXXXXXXXXXXXXXXXX'
   idlist='TAMMER1/6KY'
   format=XML
   outXml=superT
   automap=replace
   mapref=MyMap
   xmlmap="%sysget(QUANDL)superT.map"
   start='1961-12-31'
   end='2014-03-27'
   collapse=annual
   ;

data mylib.superT;
   set myTAM.superT;
run;

proc contents data=mylib.superT; run;
proc print data=mylib.superT; run;
```

Output 50.7.1 Quandl Superset for Multiple Time Series (Oil, Gold, and Stocks) Using COLLAPSE= Option

TAMMER1/6KY Superset with Multiple Time Series Using the COLLAPSE= Option

Obs	date	CAN_NY_GDP_MKTP_KD_ZG - Value	BBK01_WT5511 - Value
1	2015-12-31	.	1195.50
2	2014-12-31	.	1199.25
3	2013-12-31	2.02202	1201.50
4	2012-12-31	1.70901	1664.00
5	2011-12-31	2.52818	1574.50
6	2010-12-31	3.37425	1410.25
7	2009-12-31	-2.71147	1104.00
8	2008-12-31	1.17543	865.00
9	2007-12-31	2.00831	836.50
10	2006-12-31	2.62178	635.70
11	2005-12-31	3.16313	513.00
12	2004-12-31	3.13879	438.00
13	2003-12-31	1.92532	417.25
14	2002-12-31	2.80187	342.75
15	2001-12-31	1.68837	276.50
16	2000-12-31	5.12312	272.65
17	1999-12-31	4.99726	290.85
18	1998-12-31	4.13819	287.45
19	1997-12-31	4.25335	289.20
20	1996-12-31	1.67960	369.55
21	1995-12-31	2.73849	386.70
22	1994-12-31	4.55401	382.50
23	1993-12-31	2.60737	390.65
24	1992-12-31	0.85438	332.90
25	1991-12-31	-2.12024	353.40
26	1990-12-31	0.12898	391.00
27	1989-12-31	2.37549	401.00
28	1988-12-31	4.73765	410.15
29	1987-12-31	4.04228	486.50
30	1986-12-31	2.19816	390.90
31	1985-12-31	4.67925	327.00
32	1984-12-31	5.57187	309.00
33	1983-12-31	2.56606	381.50
34	1982-12-31	-3.02027	448.00
35	1981-12-31	3.50312	400.00
36	1980-12-31	2.16262	589.50
37	1979-12-31	3.80492	524.00
38	1978-12-31	3.95359	224.50
39	1977-12-31	3.45823	165.60
40	1976-12-31	5.19930	134.55
41	1975-12-31	1.82297	140.25
42	1974-12-31	3.69099	187.50
43	1973-12-31	6.96420	112.25
44	1972-12-31	5.44586	64.70
45	1971-12-31	4.11766	43.64
46	1970-12-31	7.65830	37.38
47	1969-12-31	5.26000	35.21

Output 50.7.1 *continued*

TAMMER1/6KY Superset with Multiple Time Series Using the COLLAPSE= Option

Obs	date	CAN_NY_GDP_MKTP_KD_ZG - Value	BBK01_WT5511 - Value
48	1968-12-31	5.29534	41.95
49	1967-12-31	2.91531	.
50	1966-12-31	6.58497	.
51	1965-12-31	6.63655	.
52	1964-12-31	6.69946	.
53	1963-12-31	5.18178	.
54	1962-12-31	7.11678	.
55	1961-12-31	3.16329	.

References

Damodaran, A. (2014). "The Data Page." Accessed March 24, 2014. http://pages.stern.nyu.edu/~adamodar/New_Home_Page/data.html.

Federal Reserve Bank of St. Louis (2012). "Economic Research." Accessed November 7, 2012. http://research.stlouisfed.org/.

Kamel, T. (2014a). "Data Sources." Accessed March 14, 2014. https://www.quandl.com/resources/data-sources/.

Kamel, T. (2014b). "Useful Lists." Accessed March 14, 2014. https://www.quandl.com/resources/useful-lists/.

Kamel, T. (2014c). "Using the Quandl API." Accessed March 14, 2014. https://www.quandl.com/help/api/.

Chapter 51
The SASEXCCM Interface Engine

Contents

Overview: SASEXCCM Interface Engine **3520**
Getting Started: SASEXCCM Interface Engine **3520**
Syntax: SASEXCCM Interface Engine **3522**
The LIBNAME *libref* SASEXCCM Statement 3522
Details: SASEXCCM Interface Engine **3526**
SAS Output Data Set 3526
Missing Values 3526
Data Reference: Introduction 3527
CCM Data Items 3527
CCM Keysets 3528
CCM Data Groups 3530
Daily STK Data Items 3531
Daily STK Data Groups 3532
Monthly STK Data Items 3532
Monthly STK Data Groups 3533
IND Group Data Item Names 3533
Monthly IND Group Data Group Names 3534
Daily IND Group Data Group Names 3534
IND Time Series Data Item Names 3535
Monthly IND Time Series Data Group Names 3535
Daily IND Time Series Data Group Names 3536
Examples: SASEXCCM Interface Engine **3537**
Example 51.1: Retrieving SALE Data for One GVKEY 3537
Example 51.2: Retrieving SALE Data for Multiple Companies 3538
Example 51.3: Retrieving Data from Different Keysets 3539
Example 51.4: Retrieving Items by Using Global Options 3540
Example 51.5: Retrieving All GVKEYs and Company Names 3542
Example 51.6: Retrieving Stock Time Series by PERMNO 3544
Example 51.7: Retrieving Stock and Indices Monthly Time Series by INDNO 3546
Example 51.8: Retrieving Stock and Indices Daily Time Series by INDNO 3548
Example 51.9: Retrieving Information for Availability of Group INDNOs 3550
Example 51.10: Retrieving Daily Group Time Series by the INDNO= Option 3551
Example 51.11: Retrieving Monthly Group Time Series by the INDNO= Option . . . 3553
References **3556**

Overview: SASEXCCM Interface Engine

The SASEXCCM interface engine enables SAS users to access the CRSP/Compustat Merged (CCM) Database, which is created from data delivered via Compustat's Xpressfeed product, the CRSP US Stock (STK) Database, and the CRSP US Stock and Indices (IND) Database. The SASEXCCM engine provides a seamless interface for CRSP, Compustat, and SAS data processing.

The SASEXCCM engine uses the LIBNAME statement to specify which database to open and what parts of the database to access.

To specify the database, you supply the combination of a physical path to indicate the location of the data files (CCM, STK, or IND data) and a set identifier (SETID) to identify the database that you want to access from those available at the physical path. The SASEXCCM engine supports data-item-handling access methods for the SETIDs in Table 51.1.

The SASECRSP engine no longer supports COMPUSTAT access. Instead, use the SASEXCCM engine, SETID 250, to read your CRSP/Compustat Merged data.

Table 51.1 CRSP Database SETIDs

SETID	Data Set
10	CRSP Stock, daily data
20	CRSP Stock, monthly data
250	CRSP/Compustat Merged data
400	CRSP Indices data, monthly index groups
420	CRSP Indices data, monthly index series
440	CRSP Indices data, daily index groups
460	CRSP Indices data, daily index series

Getting Started: SASEXCCM Interface Engine

To specify what parts of the database to access, you supply two things: the appropriate keys for companies or securities that you want to access, and the list of data items that you want to retrieve.

When accessing CCM data, you select the companies that you want to access by specifying the GVKEY for each company. A GVKEY is Compustat's unique identifier and primary key. CRSP uses KYGVKEY to refer to GVKEY in the CCM database. Use the GVKEY= option to specify which GVKEY to include. If no GVKEYs are specified, data for all companies are retrieved. You can use the KEEP= KYGVKEY CONM option to obtain a list of all companies (including their name and GVKEY) in the CCM database, as shown in Example 51.5.

For example, the following statements access the CCM database to retrieve annual sales data for IBM (GVKEY=6066) and Microsoft (GVKEY=12141):

```
LIBNAME myLib sasexccm 'physical-name'
   SETID=250
   GVKEY=6066   /* IBM */
   GVKEY=12141  /* MSFT */
```

```
   ITEMLIST='SALE';
data yrlysale;
   set myLib.annitem;
run;
```

When accessing CRSP US Stock (STK) data, you select the securities you want to access by specifying their PERMNOs. A PERMNO is CRSP's unique permanent issue identification number and the primary key for its stock databases. You specify a PERMNO by using the PERMNO= option. If no PERMNOs are specified, data for all securities in the database are retrieved. You can use this feature to obtain a list of all PERMNOs in the STK database.

For example, the following statements access the STK database to retrieve monthly shares data for IBM (PERMNO=12490) and Microsoft (PERMNO=10107):

```
LIBNAME myLib sasexccm 'physical-name'
   SETID=20
   PERMNO=12490   /* IBM */
   PERMNO=10107  /* MSFT */
   ITEMLIST="MSHROUT.*;MSHRFLG.*";
data mshares_all;
   set myLib.mshares;
run;
```

When accessing CRSP US Stock and Indices (IND) data, you select the security and indices data from the CRSP Daily or Monthly Stock and Indices database by specifying their INDNOs. An INDNO is the primary key for CRSP Indices Databases. You specify an INDNO by using the INDNO= option. If no INDNOs are specified, data for all securities in the database are retrieved. You can use this feature to obtain a list of all INDNOs in the IND database.

For example, the following statements access the IND database to retrieve monthly Consumer Price Index data (INDNO=1000709):

```
LIBNAME myLib sasexccm 'physical-name'
   SETID=420
   INDNO=1000709  /* Consumer Price Index */
   ITEMLIST=
    "MREBAL.*;MRBBEGDT.*;MRBENDDT.*;MRUSDCNT.*;MMINID.*;MMAXID.*;MMINSTAT.*";

data mindts_all;
   set myLib.mindhdr;
   set myLib.mrebal;
run;
```

To specify the list of data items that you want to retrieve, use the ITEMLIST= option. This option accepts a string that denotes a list of requested data items and the reporting format (for example, data format, population source, consolidation level, and so on) in standard CRSP notation by using CRSP's unique mnemonic text name *itm_name* and the mnemonic tag *keyset*. For more information about CRSP notation, see the ITEMLIST= option in the section "The LIBNAME *libref* SASEXCCM Statement" on page 3522.

After the LIBNAME is assigned, the database is opened. The selected data are organized into groups such as ANNITEM for annual time series data or LINK for event-based CRSP/Compustat link data. You can also use the SAS DATA step to perform further subsetting and to store the resulting time series in a SAS data set.

The SASEXCCM engine supports Linux X64 (LAX), Solaris X64 (SAX), Solaris SPARC (S64), and Windows. Windows no longer requires you to install the CRSPAccess API, because it is now distributed automatically by your SAS/ETS installation. Your Windows setup does not require any special environment variables.

Syntax: SASEXCCM Interface Engine

The SASEXCCM interface engine uses standard engine syntax. Options that the SASEXCCM engine uses are summarized in Table 51.2. The SETID= and ITEMLIST= options are required.

Table 51.2 Summary of LIBNAME *libref*

Option	Description
SETID=	Specifies which CRSP database at the physical path to open. See Table 51.1 for the complete list of supported SETIDs.
GVKEY=	Specifies a Compustat GVKEY for accessing CCM data. To select more than one GVKEY, use this option multiple times. See Example 51.1 and Example 51.2.
GVIIDKEY=	Specifies a composite GVKEY.IID for accessing security related items by both GVKEY and IID.
PERMNO=	Specifies a CRSP PERMNO for accessing STK data. To select more than one PERMNO, use this option multiple times.
INDNO=	Specifies a CRSP INDNO for accessing IND data. To select more than one INDNO, use this option multiple times.
ITEMLIST=	Specifies the selected data items to be accessed. This option accepts a string in standard CRSP notation.

The LIBNAME *libref* SASEXCCM Statement

LIBNAME *libref* **SASEXCCM** '*physical-name*' **SETID=***crsp_setidnumber options* **;**

The LIBNAME statement assigns a SAS library reference (libref) to the physical path of the directory of CRSP data files where the CRSP database that you want to open is located. The required *physical-name* argument must end in a slash for UNIX environments and a backslash for Windows environments. The required *SETID=crsp_setidnumber* argument specifies the CRSP database that you want to read from. Choose one SETID from these values: 10, 20, 250, 400, 420, 440, and 460. For example, the following statement accesses the CCM database and retrieves the annual sales data for IBM (GVKEY=6066):

```
LIBNAME myLib SASEXCCM 'physical-name' SETID=250 GVKEY=6066 ITEMLIST='SALE';
```

You can specify the following *options*:

GVKEY=*crsp_gvkey*
selects the companies or issues whose data you want to retrieve. Specify the GVKEY (Compustat's permanent SPC identifier) for the *crsp_gvkey*. There is no limit to the number of GVKEY= options that you can specify. If no GVKEY= options are specified, all GVKEYs in the database are selected.

For example, the following statement accesses the CCM database to retrieve annual sales data for IBM (GVKEY=6066) and Microsoft (GVKEY=12141):

```
LIBNAME myLib sasexccm 'physical-name'
   SETID=250
   GVKEY=6066   /* IBM */
   GVKEY=12141  /* MSFT */
   ITEMLIST='SALE';
```

GVIIDKEY='*crsp_gviidkey*'
selects the companies and issues whose data you want to retrieve. Specify both the GVKEY and the IID (Compustat's permanent issue identifier) by concatenating the two with a '.' and enclosing them in double quotation marks. There is no limit to the number of GVIIDKEY= options that you can specify. The following members use GVIIDKEY access: IDXCST_HIS, MTHSEC, SECHIST, SECURITY, SEC_MDIVFN, SEC_MSPTFN, SEC_MTHSPT, SEC_SPIND, SEC_TS_ITM, and SPIDX_CST.

For example, the following statements access the CCM database to retrieve the security member that gives security header information for Microsoft issue ID=01, IBM issue ID=01, and some other companies' issues shown in the GVIIDKEY= options:

```
LIBNAME crsp sasexccm 'physical-name'
   SETID=250
   GVIIDKEY="12141.01" /* MSFT issue id 01 */
   GVIIDKEY="6066.01"  /* IBM issue id 01 */
   GVIIDKEY="6008.01"  /* INTC issue id 01 */
   GVIIDKEY="12142.01" /* ORCL issue id 01 */
   GVIIDKEY="62634.01" /* YHOO issue id 01 */
   GVIIDKEY="5047.01"  /* GE issue id 01 */
   GVIIDKEY="7866.01"  /* NYT issue id 01 */
   GVIIDKEY="7866.02"  /* NYTAB issue id 02 */
   ITEMLIST="DLDTEI;DLRSNI;DSCI;EPF;EXCHG;IID;IID_SEQ_NUM;ISIN;SBEGDT;SENDDT;SCUSIP;
            !SEDOL;SSECSTAT;TIC;TPCI";
 data headersecurity;
    set crsp.security;
 run;
```

PERMNO=*crsp_permno*
selects the companies or issues whose data you want to retrieve. Specify a CRSP company issue's PERMNO for the *crsp_permno*. There is no limit to the number of PERMNO= options that you can specify. If no PERMNO= options are specified, all PERMNOs in the database are selected.

For example, the following statements access the STK database to retrieve monthly shares data for IBM (PERMNO=12490) and Microsoft (PERMNO=10107):

```
LIBNAME myLib sasexccm 'physical-name'
   SETID=20
   PERMNO=12490   /* IBM */
   PERMNO=10107  /* MSFT */
   ITEMLIST="MSHROUT.*;MSHRFLG.*";
data mshares_all;
   set myLib.mshares;
run;
```

INDNO=*crsp_indno*

selects the time series or the group data from the index whose data you want to retrieve. Specify a CRSP Index's INDNO for the *crsp_indno*. There is no limit to the number of INDNO= options that you can specify. If no INDNO= options are specified, all INDNOs in the database are selected.

For example, the following statements access the IND database to retrieve monthly consumer price index data (INDNO=1000709):

```
LIBNAME myLib sasexccm 'physical-name'
   SETID=420
   INDNO=1000709  /* Consumer Price Index */
   ITEMLIST=
    "MREBAL.*;MRBBEGDT.*;MRBENDDT.*;MRUSDCNT.*;MMINID.*;MMAXID.*;MMINSTAT.*";

data mindts_all;
   set myLib.mindhdr;
   set myLib.mrebal;
run;
```

ITEMLIST="*crsp_itemlist***"**

specifies the items and groups of interest for selection based on keysets, which define the reporting format that you want. Specify a string in CRSP standard notation for *crsp_itemlist*. For an overview of items, groups, and reporting formats, see the section "Data Reference: Introduction" on page 3527. Reference sections that are based on CRSP documentation follow the overview. For more information, see the *CRSPAccess User Guide for the CRSP/Compustat Merged Database*, the *CRSP US Stock and Indices Database*, and the *CRSP US Treasury Database*.

The CRSP standard notation has the form

```
[global_section:]list_section
```

The *list_section* consists of a semicolon-delimited string of list elements in the form

```
list_element[;list_element]
```

Each *list_element* can be an item or group name. You can also specify a particular keyset for the item or group by appending a period and its keyset number. For example, "sale.2" selects the sales item for keyset 2, which contains the industrial format, consolidated information, and standardized summary data from the latest annual filing.

The optional *global_section* holds flags that modify all elements in the list section. The following flags are recognized:

f adds applicable and populated footnote items for every item selected. For example, "f:sale;at;ceq" selects sales, total assets, and common equity items with default keysets and available footnotes for the selected items.

d adds applicable and populated data code items for every item selected. For example, "d:sale;at;ceq" selects sales, total assets, and common equity items with default keysets and available data codes for the selected items.

k.list applies the list of keysets to all items in the list that do not have a specified keyset. The list can be either `*` to select all available keysets or `#-#,#...` to select keysets by their number. For example, "k.1:sale;at;ceq" selects the default keyset, keyset 1, for all items.

The following LIBNAME statement shows how to access the CCM database to retrieve the annual sales data and quarterly total assets data for IBM (GVKEY=6066) and Microsoft (GVKEY=12141):

```
LIBNAME myLib sasexccm 'physical-name'
   SETID=250
   GVKEY=6066   /* IBM */
   GVKEY=12141  /* MSFT */
   ITEMLIST='f:sale;actq';
```

After the libref is assigned, you can access any of the available groups (members) within the opened database:

STK daily See the section "Daily STK Data Groups" on page 3532 for more information about groups in the Daily Stock Database, SETID 10.

STK mthly See the section "Monthly STK Data Groups" on page 3533 for more information about groups in the Monthly Stock Database, SETID 20.

CCM See the section "CCM Data Groups" on page 3530 for more information about groups in the CRSP/Compustat Merged Databases, SETID 250.

IND mthly grp See the section "Monthly IND Group Data Group Names" on page 3534 for more information about groups in the Monthly Indices Group Data Database, SETID 400.

IND mthly ts See the section "Monthly IND Time Series Data Group Names" on page 3535 for more information about groups in the Monthly Indices Time Series Database, SETID 420.

IND daily grp See the section "Daily IND Group Data Group Names" on page 3534 for more information about groups in the Daily Indices Group Data Database, SETID 440.

IND daily ts See the section "Daily IND Time Series Data Group Names" on page 3536 for more information about groups in the Daily Indices Time Series Database, SETID 460.

Details: SASEXCCM Interface Engine

SAS Output Data Set

You can use the SAS DATA step to write the selected CRSP or Compustat data to a SAS data set. This enables you to easily analyze the data by using SAS software. If you specify the name of the output data set in the DATA statement, the engine supervisor creates a SAS data set that has the specified name in either the SAS Work library or, if specified, the User library.

The contents of the SAS data set include the DATE of each observation, the series name of each series read from the CRSPAccess database, event variables, and the label or description of each series, event, or array.

You can use the PRINT and CONTENTS procedures to print your output data set and its contents. Alternatively, you can view your SAS output observations by opening the desired output data set in a SAS Explorer window. You can also use the SQL procedure with your SASEXCCM *libref* to create a custom view of your data.

Missing Values

In general, CRSP missing values are represented as '.' in the SAS data set. When accessing the CCM database, the SASEXCCM engine interprets missing values according to the conditions and codes defined by Compustat and represents them as SAS missing codes, as shown in Table 51.3.

Table 51.3 Mapping of Compustat and SAS Missing Codes

Missing Value	Missing Code	Condition
0.0001	.	No data for data item
0.0002	.S	Data are available only on a semi-annual basis
0.0003	.A	Data are available only on an annual basis
0.0004	.C	Combined into other item
0.0007	.N	Data are not meaningful
0.0008	.I	Reported as insignificant

Missing value codes conform with Compustat's Strategic Insight and binary conventions for missing values. For more information about how CRSP handles Compustat missing codes, see the section "Notes on Missing Values" in the second chapter of the *CRSP/Compustat Merged Database Guide*.

Data Reference: Introduction

Data reference details are presented for items, keysets, and groups available from four CRSPAccess databases in this order: CCM database, STK databases, and IND databases. In addition to summary tables, sample SAS statements show how to generate a customized list of item names available from each group for a particular database.

CCM Data Items

The CRSP/Compustat Merged (CCM) database is organized by company and security according to Compustat's Permanent SPC Identifier (GVKEY) and Compustat's Permanent Issue Identifier (IID). An identifying relationship exists between IID and GVKEY. The two identifiers must be accessed as a pair to properly identify a Compustat security. One GVKEY can have multiple IIDs. The SASEXCCM interface engine provides the GVIIDKEY= option to provide access to Compustat securities through the composite key designated by "GVKEY.IID".

CCM data are broken down into items, and items can be further qualified by a set of secondary keys. CRSP calls these known collections of keys and values a *keyset*, and it assigns a numeric code and mnemonic tag to each unique collection. Each keyset represents different output series. Items are also organized into groups for selection and presentation. A group can include other groups, or a group can include items. Items can belong to more than one group. Sometimes groups are also called members.

For example, the Compustat data item SALE has secondary keys for industry format, data format, population source, and consolidation level. A different value of company sales can be available for any combination of these keys, such as a combination that represents the originally reported sales or the final restated sales from a later filing. The SALE data item is a part of the ANNITEM (Annual Time Series Items, including footnotes and data codes) group.

The CCM database contains data items provided by Compustat in addition to structures and supplementary data items provided by CRSP. All data items include a mnemonic and a field name. This section provides a summary of Compustat data items whose mnemonic differs in the CCM database, and a summary of the supplementary data items provided by CRSP. For more information about the Compustat data items, refer to your Compustat data documentation or see **`www.compustatresources.com/support/index.html`**. For more information about the supplementary CRSP data items, see your CCM Database Guide.

Table 51.4 Items with Different CRSP and Compustat Names

Compustat Mnemonic	CRSP itm_name	Description	Definition
BETA	XPFBETA	Data item	Xpressfeed beta
DVPSXM	XDVPXSM	Data item	Index monthly dividend
PRC	XPFPRC	Data item	Participation rights certificates
PRCCM	XPRCCM	Data item	Index price close monthly
PRCHM	XPRCHM	Data item	Index price high monthly
PRCLM	XPRCLM	Data item	Index price low monthly
PRC_DC	XPFPRC_DC	Data code	Participation rights certificates data code
PRC_FN	XPFPRC_FN	Footnote	Participation rights certificates footnote
RET	XPFRET	Data item	Total real estate property
RET_DC	XPFRET_DC	Data code	Total real estate property data code
RET_FN	XPFRET_FN	Footnote	Total real estate property footnote
YEAR	YEARQ	Data item	Year quarterly

Supplemental CRSP data items are organized into groups. For a list of the supplemental data groups, see the section "CCM Data Groups" on page 3530.

CCM Keysets

Compustat items can be qualified by a set of secondary keys. This collection of secondary keys and values creates a keyset that assigns a numeric code and mnemonic tag to each unique collection. Each keyset represents different output series. For example, one keyset might represent originally reported sales, and another might represent the final restated sales from a later filing. Full details about keysets can be found in the *CRSP/Compustat Merged Database Guide*. For your convenience, Table 51.5 summarizes the keysets.

Table 51.5 Summary of CCM Keysets

Keyset	Tag	Keyset Description
0		Indices
1	STD	Industrial format, consolidated information, standardized presentation
2	SUMM	Industrial format, standardized summary data (StdSumData) from the latest annual filing
3	PRES	Industrial format, StdSumData collected prior to company amendment
4	FS	Financial services format, consolidated information, standardized presentation
5	PFO	Industrial format, pro forma reporting, standardized presentation
6	PFAS	Pre-FASB reporting
7	SFAS	Industrial format, pre-FASB reporting, standardized presentation
8	PRE	Industrial format, StdSumData collected from the latest annual filing
10	PDIV	Industrial format, pre-divestiture reporting, standardized presentation
11	DOM	Domestic
12	SUPF	Industrial format, pre-FASB reporting, StdSumData from the latest annual filing
14	STD1	Industrial format, consolidated information, standardized presentation, rank 1
15	FSFO	Financial services format, pro forma reporting, standardized presentation
16	FS1	Financial services format, consolidated information, standardized presentation, rank 1
17	FS2	Financial services format, consolidated information, standardized presentation, rank 2
18	SUFS	Industrial format, pro forma reporting, StdSumData from the latest annual filing
19	PDI1	Industrial format, pre-divestiture reporting, standardized presentation, rank 1
20	PFA1	Industrial format, pre-FASB reporting, standardized presentation, rank 1
21	SUPD	Industrial format, pre-divestiture reporting, StdSumData from the latest annual filing
22	FS3	Financial services format, consolidated information, standardized presentation, rank 3
23	PDI2	Industrial format, consolidated information, standardized presentation, rank 2
24	CONS	Consolidated information
25	STD2	Industrial format, consolidated information, standardized presentation, rank 2
26	STD3	Industrial format, consolidated information, standardized presentation, rank 3
27	STD4	Industrial format, consolidated information, standardized presentation, rank 4
28	STD5	Industrial format, consolidated information, standardized presentation, rank 5
29	PFA2	Industrial format, pre-FASB reporting, standardized presentation, rank 2
30	PFA3	Industrial format, pre-FASB reporting, standardized presentation, rank 3
31	CUSD	Calendar-based reporting in US dollars
32	FUSD	Fiscal-based reporting in US dollars
33	CCAD	Calendar-based reporting in Canadian dollars
34	FCAD	Fiscal-based reporting in Canadian dollars
35	PFA4	Industrial format, pre-FASB reporting, standardized presentation, rank 4
36	PFO2	Industrial format, pro forma reporting, standardized presentation, rank 2
37	PFO1	Industrial format, pro forma reporting, standardized presentation, rank 1
38	PRE1	Industrial format, standardized data collected before company amendment, rank 1
39	FFO1	Financial services format, pro forma reporting, standardized presentation, rank 1
40	FS4	Financial services format, consolidated information, standardized presentation, rank 4
41	GICS	Industry code type Global Industry Classification Standard
43	FORD	Pro forma reporting
44	BSTD	Bank format, consolidated information, standardized presentation
45	BSUMM	Bank format, consolidated information, StdSumData from the latest annual filing
46	BPFO	Bank format, pro forma reporting, standard presentation
47	BASTD	Bank format, consolidated information, average standardized presentation
48	BASUMM	Bank format, average standardized summary presentation from the latest annual filing
49	BAPFO	Bank format, pro forma reporting, average standardized presentation

CCM Data Groups

CCM items are organized into groups for ease of selection and presentation. Each group is given a group name. These names are unique and do not overlap with item names. A group can be made up of either items or other groups. Items can belong to more than one group. Table 51.6 provides a summary of some groups. See your *CCM Database Guide* for more information about CCM data groups.

Table 51.6 Selected Xpressfeed Primary and CRSP Supplemental Groups

Item Name	Description
MASTER	CCM company ID and range data
COMPANY	CCM company header information
IDX_INDEX	CCM idx_index header information
SPIND	Standard & Poor's (S&P) index header (pre-GICS)
COMPHIST	CCM company header history
CSTHIST	CST header history
LINK	Link history
LINKUSED	CCM company CRSP link used data
LINKRNG	CCM company CRSP link range data
ADJFACT	CCM company adjustment factor history
HGIC	CCM company GICS code history
OFFTITL	CCM company officer title data
CCM_FILEDATE	CCM company filing date data
CCM_IPCD	CCM industry presentation code data
SECURITY	CCM security header information
SECHIST	CCM security header history
SEC_MTHSPT	CCM security monthly split events
SEC_MSPT_FN	CCM security monthly split event footnotes
SEC_MDIV_FN	CCM security monthly dividend event footnotes
SEC_SPIND	CCM security S&P information events
IDXCST_HIS	CCM security historical index constituents
SPIDX_CST	CCM security S&P index constituent events
CCM_SEGCUR	CCM operating segment currency rate data
CCM_SEGSRC	CCM operating segment source data
CCM_SEGPROD	CCM operating segment product data
CCM_SEGCUST	CCM operating segment customer data
CCM_SEGDTL	CCM operating segment detail data
CCM_SEGITM	CCM operating segment item data
CCM_SEGNAICS	CCM operating segment NAICS data
CCM_SEGGEO	CCM operating segment geographic data

Daily STK Data Items

You can generate a customized list of item names available in the daily stock database (SETID=10) by running the following sample statements for each group name in Table 51.8:

```
libname dstock sasexccm
   "\\bb04smb01\thirdparty\lnx\crspdata\DIZ201006\"
   setid=10 permno=12490
   itemlist="group_name.*";

proc contents data=dstock.group_name; run;
```

The following statements generate an item list of all the item names available in the group named STKHDR_ID:

```
libname crsp sasexccm
   "\\bb04smb01\thirdparty\lnx\crspdata\DIZ201006\"
   setid=10 permno=12490
   itemlist="STKHDR_ID.*";

proc contents data=crsp.stkhdr_id; run;
```

The item names in group STKHDR_ID are listed in Table 51.7.

Table 51.7 US Daily Stock Items in Group STKHDR_ID

Item Name	Description
BEGDT	Begdt
COMPNO	COMPNO
CUSIP	CUSIP
ENDDT	Enddt
HCOMNAM	Latest company name
HDLSTCD	DEL
HEXCD	EX
HPRIMEXCH	Ex1
HSECSTAT	Sst
HSHRCD	SH
HSICCD	SIC
HSNAICS	Naics
HSUBEXCH	Ex2
HTICK	Htick
HTRDSTAT	Tst
HTSYMBOL	Symbol
ISSUNO	Issuno
KYPERMNO	PERMNO
PERMCO	PERMCO
PERMNO	PERMNO

Daily STK Data Groups

Daily stock groups are shown in Table 51.8.

Table 51.8 US Daily Stock Group Names

Group Name	Description
STKHDR_ID	Stock header (summary)
STKHDR_ALL	All stock headers
STKHDR_RNG	Stock header plus ranges
LSTKHDR_RNG	Stock header plus calendar index ranges
NAMES_SHORT	Name history (short list)
NAMES	Name history
NAMES_ALL	All names
DISTS	Distribution events
ADJDISTS	Daily adjusted distribution events
SHARES	Shares outstanding observations
RSHARES	Raw shares outstanding observations
ADJSHARES	Daily adjusted shares outstanding observations
DELIST	Delisting history
ADJDELIST	Adjusted delisting events
NASDIN	NASDAQ information history
DLY_DATA	Daily price summary time series
DLY_ADJDATA	Daily adjusted price summary time series
DSTK_TS	Daily time series
DLY_WGT	Daily price, shares, and returns
DLY_ADJ_WGT	Daily adjusted price, shares, and returns
DLY_LVL	Daily index level
DLY_RET	Daily returns
PORTF	Portfolio data
GROUP	Group membership data
DLY_TS_NAT	Daily time series (native only)
DSTK_VOLUME	Volume
DSTK_CAP	Capitalization

Monthly STK Data Items

You can generate a customized list of item names available in the monthly stock database by running the following sample statements for each group name in Table 51.9:

```
libname crsp sasexccm
   "\\tappan\crsp1\data201008_little\MIZ201006\"
   setid=20 permno=12490
   itemlist="group_name.*";

proc contents data=crsp.group_name; run;
```

Monthly STK Data Groups

Monthly stock groups are shown in Table 51.9.

Table 51.9 US Monthly Stock Group Names

Group Name	Description
MSTKHDR_ID	Stock header (summary)
MSTKHDR_RNG	Stock header plus ranges
LMSTKHDR_RNG	Stock header plus calendar index ranges
MNAMES_SHORT	Name history (short list)
MNAMES	Name history
MNAMES_ALL	All MNAMES (monthly name histories)
MDISTS	Distribution events
MADJDISTS	Monthly adjusted distribution events
MSHARES	Shares outstanding observations
RMSHARES	Raw shares outstanding observations
MADJSHARES	Monthly adjusted shares outstanding observations
MDELIST	Delisting history
MADJDELIST	Adjusted delisting events
MNASDIN	NASDAQ information history
MTH_DATA	Monthly price summary time series
MTH_ADJDATA	Monthly adjusted price summary time series
MTH_TS	Monthly time series
MTH_WGT	Monthly price, shares, and returns
MTH_ADJ_WGT	Monthly adjusted price, shares, and returns
MTH_LVL	Monthly index level
MTH_RET	Monthly returns
MPORTF	Portfolio data
MGROUP	Group membership data
MTH_TS_NAT	Monthly time series (native only)
MSTK_VOLUME	Volume
MSTK_CAP	Capitalization

IND Group Data Item Names

You can generate a customized list of available indices group data item names by running the following sample statements for each daily or monthly group name from Table 51.10 or Table 51.11 and substituting the corresponding SETID, data path, and actual daily (or monthly) group name for the **group_name**:

```
libname crsp sasexccm
   "\\bb04smb01\thirdparty\lnx\crspdata\DIZ201006\"
   setid=440
   indno=1000040
   itemlist="group_name.*";

proc contents data=crsp.group_name; run;
```

Monthly IND Group Data Group Names

The monthly group indices data consist of the groups listed in Table 51.10.

Table 51.10 US IND Monthly Group Data Group Names

Group Name	Description
MINDHDRG	Monthly index group header
MINDSUMMG	Monthly index group summary
MLISTG	Monthly index group list history
MREBALG	Monthly index group rebalancing history
MREBALG_ALL	Monthly index group rebalancing
MTHGIND_LVL	Monthly index group levels
MTHGIND_RET	Monthly index group returns
MTHGIND_TS	Monthly index group series
MTHGIND_VAL	Monthly index group values

Daily IND Group Data Group Names

The daily group indices data consist of the groups listed in Table 51.11.

Table 51.11 US IND Daily Group Data Group Names

Group Name	Description
INDHDRG	Index group header
INDSUMMG	Index group summary
LISTG	Index group list history
REBALG	Index group rebalancing history
REBALG_ALL	Index group rebalancing
DLYGIND_LVL	Index group levels
DLYGIND_RET	Index group returns
DLYGIND_TS	Index group series
DLYGIND_VAL	Index group values

IND Time Series Data Item Names

You can generate a customized list of available item names by running the following sample statements for each daily or monthly time series group name from Table 51.12 or Table 51.13 and substituting the corresponding SETID, data path, and actual daily (or monthly) time series group name for the **group_name**:

```
libname daycrsp sasexccm
   "\\bb04smb01\thirdparty\lnx\crspdata\DIZ201006\"
   setid=460
   indno=1000040
   itemlist="group_name.*";

proc contents data=daycrsp.group_name; run;
```

Monthly IND Time Series Data Group Names

The monthly indices data consist of the groups listed in Table 51.12.

Table 51.12 US IND Monthly Series Data Group Names

Group Name	Description
MINDHDR	Monthly index header
MINDSUMM	Monthly index summary
MLIST	Monthly index list history
MREBAL	Monthly index rebalancing history
MREBAL_ALL	Monthly index rebalancing
MTHIND_LVL	Monthly index levels
MTHIND_RET	Monthly index returns
MTHIND_TS	Monthly index series
MTHIND_VAL	Monthly index values

Daily IND Time Series Data Group Names

The daily indices data consist of the groups listed in Table 51.13.

Table 51.13 US IND Daily Time Series Data Group Names

Group Name	Description
INDHDR	Index header
INDSUMM	Index summary
LIST	Index list history
REBAL	Index rebalancing history
REBAL_ALL	Index rebalancing
DLYIND_LVL	Index levels
DLYIND_RET	Index returns
DLYIND_TS	Index series
DLYIND_VAL	Index values

Examples: SASEXCCM Interface Engine

Example 51.1: Retrieving SALE Data for One GVKEY

This simple example shows how to retrieve SALE data for one particular GVKEY=6066 (IBM). Because the ITEMLIST= option does not specify a keyset, the default (standard) keyset, KEYSET_TAG=STD, is selected. For brevity, a subset of the data that contains the most recent figures is specified by the WHERE statement.

```
title 'Retrieve SALE data for IBM';
libname _all_ clear;

libname crsp sasexccm "\\bb04smb01\thirdparty\lnx\crspdata\cmz201201\"
   setid=250
   gvkey=6066
   itemlist="sale";

data recentsales;
    set crsp.annitem;
    where datadate >= '1jan2000'd;

proc print data=recentsales;
run;
```

Output 51.1.1 SALE Data for GVKEY=6066

Retrieve SALE data for IBM

Obs	KYGVKEY	KEYSET_TAG	DATADATE	SALE
1	6066	STD	20001229	88396.0000
2	6066	STD	20011231	85866.0000
3	6066	STD	20021231	81186.0000
4	6066	STD	20031231	89131.0000
5	6066	STD	20041231	96293.0000
6	6066	STD	20051230	91134.0000
7	6066	STD	20061229	91424.0000
8	6066	STD	20071231	98786.0000
9	6066	STD	20081231	103630.0000
10	6066	STD	20091231	95758.0000
11	6066	STD	20101231	99871.0000

Example 51.2: Retrieving SALE Data for Multiple Companies

This example shows how to retrieve several data items for several GVKEYs. Note how the item **offtitl** is not an annual item and is stored in its own member. The default (standard) keyset is used for all items. For brevity, a subset of the data that contains the most recent figures is specified by the WHERE statement.

```
title 'Retrieve Sales, Revenue, Liabilities, and Officer data for IBM and MSFT';
libname _all_ clear;

libname crsp sasexccm "\\bb04smb01\thirdparty\lnx\crspdata\cmz201201\"
   setid=250
   gvkey=6066
   gvkey=12141
   itemlist="sale;revt;lct;offtitl";

data recentannitems;
    set crsp.annitem;
    where datadate >= '1jan2000'd;

proc print data=recentannitems;
proc print data=crsp.offtitl;
run;
```

Output 51.2.1 Data Items for IBM and Microsoft

Retrieve Sales, Revenue, Liabilities, and Officer data for IBM and MSFT

Obs	KYGVKEY	KEYSET_TAG	DATADATE	SALE	REVT	LCT
1	6066	STD	20001229	88396.0000	88396.0000	36406.0000
2	6066	STD	20011231	85866.0000	85866.0000	35119.0000
3	6066	STD	20021231	81186.0000	81186.0000	34550.0000
4	6066	STD	20031231	89131.0000	89131.0000	37900.0000
5	6066	STD	20041231	96293.0000	96293.0000	39798.0000
6	6066	STD	20051230	91134.0000	91134.0000	35152.0000
7	6066	STD	20061229	91424.0000	91424.0000	40091.0000
8	6066	STD	20071231	98786.0000	98786.0000	44310.0000
9	6066	STD	20081231	103630.0000	103630.0000	42435.0000
10	6066	STD	20091231	95758.0000	95758.0000	36002.0000
11	6066	STD	20101231	99871.0000	99871.0000	40562.0000
12	12141	STD	20000630	22956.0000	22956.0000	9755.0000
13	12141	STD	20010629	25296.0000	25296.0000	11132.0000
14	12141	STD	20020628	28365.0000	28365.0000	12744.0000
15	12141	STD	20030630	32187.0000	32187.0000	13974.0000
16	12141	STD	20040630	36835.0000	36835.0000	14969.0000
17	12141	STD	20050630	39788.0000	39788.0000	16877.0000
18	12141	STD	20060630	44282.0000	44282.0000	22442.0000
19	12141	STD	20070629	51122.0000	51122.0000	23754.0000
20	12141	STD	20080630	60420.0000	60420.0000	29886.0000
21	12141	STD	20090630	58437.0000	58437.0000	27034.0000
22	12141	STD	20100630	62484.0000	62484.0000	26147.0000
23	12141	STD	20110630	69943.0000	69943.0000	28774.0000

Output 51.2.1 *continued*

Retrieve Sales, Revenue, Liabilities, and Officer data for IBM and MSFT

Obs	KYGVKEY	OFID	OFCD	OFNM
1	6066	1911113	CE	Ms. Virginia M. Rometty
2	6066	1911113	DI	Ms. Virginia M. Rometty
3	6066	1911113	PR	Ms. Virginia M. Rometty
4	6066	1911114	CB	Mr. Samuel J. Palmisano
5	6066	1911114	DI	Mr. Samuel J. Palmisano
6	6066	1911115	CF	Mr. Mark Loughridge
7	6066	1911116	TO	Mr. Rodney C. Adkins
8	12141	1873964	CE	Mr. Steven A. Ballmer
9	12141	1873964	DI	Mr. Steven A. Ballmer
10	12141	1873965	CB	Mr. William Henry Gates III
11	12141	1873965	DI	Mr. William Henry Gates III
12	12141	1873966	CO	Mr. Brain Kevin Turner
13	12141	1873967	CF	Mr. Peter S. Klein

Example 51.3: Retrieving Data from Different Keysets

This example shows how to retrieve several data items from different keysets. You request data about research and development (R&D) expenses (XRD) and net income (NI) over all available keysets by using the **itm_name.*** syntax in the ITEMLIST= option. Note that data are not available for all items in all keysets. For brevity, a subset of the data that contains the most recent figures is specified by the WHERE statement.

```
title 'Retrieve R&D Expenses and Net Income for IBM';
libname _all_ clear;

libname crsp sasexccm "\\bb04smb01\thirdparty\lnx\crspdata\cmz201201\"
   setid=250
   gvkey=6066
   itemlist="xrd.*;ni.*";

data recent;
    set crsp.annitem;
    where datadate >= '1jan2001'd;

proc print data=recent;
run;
```

Output 51.3.1 R&D Expenses and Net Income for GVKEY=6066

Retrieve R&D Expenses and Net Income for IBM

Obs	KYGVKEY	KEYSET_TAG	DATADATE	XRD	NI
1	6066	STD	20011231	4620.0000	7723.0000
2	6066	STD	20021231	4754.0000	3579.0000
3	6066	STD	20031231	5077.0000	7583.0000
4	6066	STD	20041231	5167.0000	8430.0000
5	6066	STD	20051230	5379.0000	7934.0000
6	6066	STD	20061229	5682.0000	9492.0000
7	6066	STD	20071231	5754.0000	10418.0000
8	6066	STD	20081231	6015.0000	12334.0000
9	6066	STD	20091231	5523.0000	13425.0000
10	6066	STD	20101231	5720.0000	14833.0000
11	6066	SUMM	20011231	.	6484.0000
12	6066	SUMM	20021231	.	2376.0000
13	6066	SUMM	20031231	.	6558.0000
14	6066	SUMM	20041231	.	7479.0000
15	6066	SUMM	20051230	.	7934.0000
16	6066	SUMM	20061229	.	9492.0000
17	6066	SUMM	20071231	.	10418.0000
18	6066	SUMM	20081231	.	12334.0000
19	6066	SUMM	20091231	.	13425.0000
20	6066	SUMM	20101231	.	14833.0000

Example 51.4: Retrieving Items by Using Global Options

This example shows how to retrieve data on total assets (ATQ) and after tax gain or loss (GLAQ) by using the global option for turning on footnote items, which uses the following syntax:

```
ITEMLIST="f:itm_name1;itm_name2;....itm_nameN"
```

The default (standard) keyset is used for all items. For brevity, a subset of the data that contains the most recent figures is specified by the WHERE statement.

```
title 'Retrieve data for IBM with Footnotes';
libname _all_ clear;

libname crsp sasexccm "\\bb04smb01\thirdparty\lnx\crspdata\cmz201201\"
  setid=250
  gvkey=6066
  itemlist="f:atq;glaq";

data recent;
    set crsp.qtritem;
    where datadate >= '1jan2004'd;

proc print data=recent;
run;
```

Output 51.4.1 Data Items with Footnotes

Retrieve data for IBM with Footnotes

Obs	KYGVKEY	KEYSET_TAG	DATADATE	ATQ	ATQ_FN1	GLAQ	GLAQ_FN
1	6066	STD	20040331	101825.0000	JR	.	
2	6066	STD	20040630	99582.0000	JR	.	
3	6066	STD	20040930	100676.0000	JR	.	
4	6066	STD	20041231	109183.0000	JR	.	
5	6066	STD	20050331	104899.0000		.	
6	6066	STD	20050630	103388.0000		732.5550	NC
7	6066	STD	20050930	101009.0000		0.0000	NC
8	6066	STD	20051230	105748.0000		0.0000	NC
9	6066	STD	20060331	102468.0000		.	
10	6066	STD	20060630	103377.0000		.	
11	6066	STD	20060929	104155.0000		.	
12	6066	STD	20061229	103234.0000		29.2500	NR
13	6066	STD	20070330	101619.0000		.	
14	6066	STD	20070629	102548.0000		81.0000	
15	6066	STD	20070928	108609.0000		0.0000	
16	6066	STD	20071231	120431.0000		0.0000	
17	6066	STD	20080331	121823.0000		.	
18	6066	STD	20080630	120928.0000		.	
19	6066	STD	20080930	115910.0000		.	
20	6066	STD	20081231	109524.0000		.	
21	6066	STD	20090331	101944.0000		193.7000	NR
22	6066	STD	20090630	103655.0000		0.0000	NR
23	6066	STD	20090930	103675.0000		0.0000	NR
24	6066	STD	20091231	109022.0000		0.0000	NR
25	6066	STD	20100331	105208.0000		390.3600	NC
26	6066	STD	20100630	103420.0000		0.0000	NR
27	6066	STD	20100930	107174.0000		0.0000	NR
28	6066	STD	20101231	113452.0000		0.0000	NR
29	6066	STD	20110331	112960.0000		.	
30	6066	STD	20110630	113474.0000		.	
31	6066	STD	20110930	110158.0000		.	

Example 51.5: Retrieving All GVKEYs and Company Names

This example shows how to retrieve the GVKEY and name for every company in the CCM database.

```
title 'Retrieve All GVKEYs and Company Names';
libname _all_ clear;

libname crsp sasexccm "\\bb04smb01\thirdparty\lnx\crspdata\cmz201201\"
  setid=250
  itemlist="company";

proc contents data=crsp.company;
proc print data=crsp.company(keep=kygvkey conm obs=20);
run;
```

For brevity, only the first 20 observations are shown, and only KYGVKEY and CONM are kept in Output 51.5.1.

Output 51.5.1 First 20 GVKEYS and Company Names

Retrieve All GVKEYs and Company Names

The CONTENTS Procedure

Data Set Name	CRSP.COMPANY	**Observations**	.
Member Type	DATA	**Variables**	38
Engine	CRSPCCM	**Indexes**	0
Created	03/02/2015 16:30:33	**Observation Length**	3232
Last Modified	03/02/2015 16:30:33	**Deleted Observations**	0
Protection		**Compressed**	NO
Data Set Type		**Sorted**	NO
Label			
Data Representation	Default		
Encoding	Default		

Output 51.5.1 *continued*

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Informat	Label
33	ADD1	Char	68	65.	65.	ADD1
34	ADD2	Char	68	65.	65.	ADD2
35	ADD3	Char	68	65.	65.	ADD3
36	ADD4	Char	68	65.	65.	ADD4
37	ADDZIP	Char	24	24.	24.	ADDZIP
38	BUSDESC	Char	2000	2000.	2000.	BUSDESC
2	CIK	Char	12	10.	10.	CIK
20	CITY	Char	104	104.	104.	CITY
5	CONM	Char	256	255.	255.	CONM
29	CONML	Char	104	100.	100.	CONML
7	COSTAT	Char	4	1.	1.	COSTAT
19	COUNTY	Char	104	100.	100.	COUNTY
9	DLDTE	Num	8	8.	8.	DLDTE
10	DLRSN	Char	12	8.	8.	DLRSN
3	EIN	Char	12	10.	10.	EIN
32	FAX	Char	24	18.	18.	FAX
15	FIC	Char	16	3.	3.	FIC
6	FYRC	Num	8	2.	2.	FYRC
24	GGROUP	Char	12	4.	4.	GGROUP
25	GIND	Char	12	6.	6.	GIND
23	GSECTOR	Char	12	2.	2.	GSECTOR
26	GSUBIND	Char	12	8.	8.	GSUBIND
14	IDBFLAG	Char	12	1.	1.	IDBFLAG
17	INCORP	Char	12	8.	8.	INCORP
8	IPODATE	Num	8	8.	8.	IPODATE
1	KYGVKEY	Num	8	6.	6.	GVKEY
16	LOC	Char	4	3.	3.	LOC
22	NAICS	Char	8	6.	6.	NAICS
31	PHONE	Char	24	18.	18.	PHONE
12	PRICAN	Char	12	8.	8.	PRICAN
13	PRIROW	Char	12	8.	8.	PRIROW
11	PRIUSA	Char	12	8.	8.	PRIUSA
21	SIC	Num	8	4.	4.	SIC
27	SPCINDCD	Num	8	4.	4.	SPCINDCD
28	SPCSECCD	Num	8	4.	4.	SPCSECCD
18	STATE	Char	12	8.	8.	STATE
4	STKO	Num	8	1.	1.	STKO
30	WEBURL	Char	68	60.	60.	WEBURL

Output 51.5.1 *continued*

Retrieve All GVKEYs and Company Names

Obs	KYGVKEY	CONM
1	1000	A & E PLASTIK PAK INC
2	1001	A & M FOOD SERVICES INC
3	1002	AAI CORP
4	1003	A.A. IMPORTING CO INC
5	1004	AAR CORP
6	1005	A.B.A. INDUSTRIES INC
7	1006	ABC INDS INC
8	1007	ABKCO INDUSTRIES INC
9	1008	ABM COMPUTER SYSTEMS INC
10	1009	ABS INDUSTRIES INC
11	1010	ACF INDUSTRIES HOLDING CORP
12	1011	ACS ENTERPRISES INC
13	1012	ACS INDUSTRIES INC
14	1013	ADC TELECOMMUNICATIONS INC
15	1014	ADDSCO INDUSTRIES INC
16	1015	ADI ELECTRONICS INC
17	1016	AEC INC
18	1017	AEL INDUSTRIES -CL A
19	1018	AES TECHNOLOGY SYSTEMS INC
20	1019	AFA PROTECTIVE SYSTEMS INC

Example 51.6: Retrieving Stock Time Series by PERMNO

This example shows how to retrieve the MPRC, MASK, and MBID time series by using PERMNO key access in the STK database. For brevity, the WHERE= option in the DATA step selects a range of MCALDT for 25 observations.

```
title 'Retrieve IBM Monthly PRC, ASK, and BID by PERMNO Access';
libname _all_ clear;

libname crsp sasexccm
   "\\tappan\crsp1\data201008_little\MIZ201006\"
   setid=20
   permno=12490
   itemlist="MPRC;MASK;MBID";

data mstkts_all( where=( mcaldt >= '30jun2008'd) ) ;
   set crsp.mstk_ts;
run;
proc contents data=mstkts_all;
run;
proc print data=mstkts_all;
run;
```

Output 51.6.1 IBM's Monthly PRC, ASK, and BID by PERMNO

Retrieve IBM Monthly PRC, ASK, and BID by PERMNO Access

The CONTENTS Procedure

Data Set Name	WORK.MSTKTS_ALL	**Observations**	25
Member Type	DATA	**Variables**	5
Engine	V9	**Indexes**	0
Created	03/02/2015 16:30:37	**Observation Length**	40
Last Modified	03/02/2015 16:30:37	**Deleted Observations**	0
Protection		**Compressed**	NO
Data Set Type		**Sorted**	NO
Label			
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information

Data Set Page Size	65536
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	1632
Obs in First Data Page	25
Number of Data Set Repairs	0
ExtendObsCounter	YES
Filename	C:\Users\saskff\AppData\Local\Temp\SAS Temporary Files_TD7388_D79286_\mstkts_all.sas7bdat
Release Created	9.0401M3
Host Created	W32_7PRO

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Informat	Label
1	KYPERMNO	Num	8	6.	6.	PERMNO
4	MASK	Num	8	12.5	12.5	Ask
5	MBID	Num	8	12.5	12.5	Bid
2	MCALDT	Num	8	YYMMDDN8.	YYMMDD8.	Caldt
3	MPRC	Num	8	12.5	12.5	Prc

Output 51.6.1 *continued*

Retrieve IBM Monthly PRC, ASK, and BID by PERMNO Access

Obs	KYPERMNO	MCALDT	MPRC	MASK	MBID
1	12490	20080630	118.53000	118.63000	118.28000
2	12490	20080731	127.98000	127.93000	127.99000
3	12490	20080829	121.73000	121.77000	121.74000
4	12490	20080930	116.96000	116.19000	116.09000
5	12490	20081031	92.97000	93.54000	92.92000
6	12490	20081128	81.60000	81.72000	81.60000
7	12490	20081231	84.16000	84.22000	84.09000
8	12490	20090130	91.65000	91.73000	91.71000
9	12490	20090227	92.03000	92.18000	92.16000
10	12490	20090331	96.89000	97.10000	97.10000
11	12490	20090430	103.21000	103.39000	103.31000
12	12490	20090529	106.28000	106.38000	106.40000
13	12490	20090630	104.42000	104.38000	104.34000
14	12490	20090731	117.93000	118.01000	117.93000
15	12490	20090831	118.05000	118.06000	118.04000
16	12490	20090930	119.61000	119.56000	119.56000
17	12490	20091030	120.61000	120.63000	120.62000
18	12490	20091130	126.35000	126.43000	126.42000
19	12490	20091231	130.89999	130.89000	130.84000
20	12490	20100129	122.39000	122.32000	122.30000
21	12490	20100226	127.16000	127.22000	127.21000
22	12490	20100331	128.25000	128.30000	128.25999
23	12490	20100430	129.00000	128.89000	128.86000
24	12490	20100528	125.26000	125.17000	125.08000
25	12490	20100630	123.48000	123.41000	123.38000

Example 51.7: Retrieving Stock and Indices Monthly Time Series by INDNO

This example shows how to retrieve monthly time series by using INDNO key access in the IND database. For brevity, the WHERE= option in the DATA step selects a recent range of MCALDT.

```
title 'Retrieve Several Monthly Time Series by INDNO Access';
libname _all_ clear;

libname crsp sasexccm
   "\\tappan\crsp1\data201008_little\MIZ201006\"
   setid=420
   indno=1000040 indno=1000060 indno=1000080
   itemlist="MAIND;MARET;MIIND";

data mindts_all ( where=( mcaldt >= '30jun2009'd) );
   set crsp.mthind_ts;
run;

proc print data=mindts_all; run;
```

Output 51.7.1 Monthly Time Series by INDNO

Retrieve Several Monthly Time Series by INDNO Access

Obs	KYINDNO	MCALDT	MAIND	MARET	MIIND
1	1000040	20090630	806.96	-0.015017	333.67
2	1000040	20090731	873.04	0.081896	334.25
3	1000040	20090831	902.23	0.033436	335.08
4	1000040	20090930	938.68	0.040394	335.72
5	1000040	20091030	912.56	-0.027819	336.24
6	1000040	20091130	964.47	0.056883	337.16
7	1000040	20091231	982.11	0.018287	337.88
8	1000040	20100129	949.37	-0.033339	338.31
9	1000040	20100226	978.72	0.030920	339.13
10	1000040	20100331	1037.53	0.060087	339.79
11	1000040	20100430	1054.85	0.016696	340.25
12	1000040	20100528	968.99	-0.081403	341.07
13	1000040	20100630	920.85	-0.049681	341.73
14	1000060	20090630	1258.17	0.034083	180.28
15	1000060	20090731	1356.35	0.078031	180.36
16	1000060	20090831	1378.50	0.016334	180.59
17	1000060	20090930	1454.80	0.055349	180.68
18	1000060	20091030	1401.98	-0.036304	180.75
19	1000060	20091130	1468.60	0.047513	181.08
20	1000060	20091231	1555.54	0.059201	181.19
21	1000060	20100129	1470.06	-0.054949	181.25
22	1000060	20100226	1530.69	0.041239	181.49
23	1000060	20100331	1641.63	0.072481	181.61
24	1000060	20100430	1686.83	0.027533	181.69
25	1000060	20100528	1544.38	-0.084447	181.93
26	1000060	20100630	1440.70	-0.067137	182.02
27	1000080	20090630	823.64	-0.004569	305.33
28	1000080	20090731	890.39	0.081042	305.78
29	1000080	20090831	916.85	0.029715	306.46
30	1000080	20090930	956.86	0.043638	306.94
31	1000080	20091030	928.44	-0.029699	307.34
32	1000080	20091130	979.35	0.054839	308.12
33	1000080	20091231	1006.14	0.027352	308.68
34	1000080	20100129	967.73	-0.038177	309.01
35	1000080	20100226	999.85	0.033197	309.68
36	1000080	20100331	1062.61	0.062762	310.20
37	1000080	20100430	1082.91	0.019104	310.55
38	1000080	20100528	994.02	-0.082082	311.22
39	1000080	20100630	940.73	-0.053606	311.73

Example 51.8: Retrieving Stock and Indices Daily Time Series by INDNO

This example shows how to retrieve daily time series by using INDNO key access of the IND database. For brevity, the WHERE= option in the DATA step selects a recent range of CALDT.

```
title 'Retrieve Several Daily Time Series by INDNO Access';
libname _all_ clear;

libname crsp sasexccm
   "\\bb04smb01\thirdparty\lnx\crspdata\DIZ201006\"
   setid=460
   indno=1000040 indno=1000060 indno=1000080
   itemlist="TOTCNT;TOTVAL;TRET";

data dindts_all ( where=( caldt >= '15jun2010'd) );
   set crsp.dlyind_ts;
run;

proc print data=dindts_all; run;
```

Output 51.8.1 Daily Time Series by INDNO

Retrieve Several Daily Time Series by INDNO Access

Obs	KYINDNO	CALDT	TOTCNT	TOTVAL	TRET
1	1000040	20100615	2668	11859837753.20	0.023096
2	1000040	20100616	2669	11841138158.33	-0.001545
3	1000040	20100617	2670	11852139584.84	0.000916
4	1000040	20100618	2671	11873916422.35	0.001838
5	1000040	20100621	2672	11839008747.57	-0.003009
6	1000040	20100622	2672	11627382000.00	-0.017709
7	1000040	20100623	2672	11594378306.30	-0.002811
8	1000040	20100624	2673	11402808548.54	-0.016525
9	1000040	20100625	2675	11479692502.15	0.006645
10	1000040	20100628	2674	11414419071.77	-0.003311
11	1000040	20100629	2674	11051115070.31	-0.031777
12	1000040	20100630	2674	10987117493.51	-0.008165
13	1000060	20100615	2740	3408075602.72	0.027565
14	1000060	20100616	2741	3408111567.48	-0.000031
15	1000060	20100617	2741	3410846831.87	0.000734
16	1000060	20100618	2741	3414496996.34	0.001222
17	1000060	20100621	2738	3383308305.95	-0.009517
18	1000060	20100622	2741	3343876276.79	-0.011656
19	1000060	20100623	2741	3331677101.72	-0.003784
20	1000060	20100624	2741	3277142757.63	-0.016425
21	1000060	20100625	2740	3289059489.28	0.003588
22	1000060	20100628	2739	3281646245.81	-0.002116
23	1000060	20100629	2741	3156605732.63	-0.038661
24	1000060	20100630	2741	3117034325.97	-0.012478
25	1000080	20100615	5408	15267913355.93	0.024090
26	1000080	20100616	5410	15249249725.81	-0.001207
27	1000080	20100617	5411	15262986416.71	0.000875
28	1000080	20100618	5412	15288413418.68	0.001701
29	1000080	20100621	5410	15222317053.52	-0.004462
30	1000080	20100622	5413	14971258276.79	-0.016364
31	1000080	20100623	5413	14926055408.02	-0.003029
32	1000080	20100624	5414	14679951306.17	-0.016503
33	1000080	20100625	5415	14768751991.42	0.005962
34	1000080	20100628	5413	14696065317.58	-0.003044
35	1000080	20100629	5415	14207720802.94	-0.033314
36	1000080	20100630	5415	14104151819.47	-0.009123

Example 51.9: Retrieving Information for Availability of Group INDNOs

This example shows how to retrieve header information about group data and how to obtain a list of all the available INDNO keys in the IND database. The INDNO= option is intentionally omitted so that a default list is generated of all INDNOs in the database that are available for SETID 440.

```
title 'Retrieve Header Information for a Complete INDNO list';
libname _all_ clear;

libname crsp sasexccm
   "\\bb04smb01\thirdparty\lnx\crspdata\DIZ201006\"
   setid=440
   itemlist="INDNOG.*;INDCOG.*;INDNAMEG.*;GROUPNAMEG.*";

data dgindts_all;
   set crsp.indhdrg;
run;

proc print data=dgindts_all(keep=kyindno indnameg); run;
```

Output 51.9.1 Daily Group Indices Header by INDNO

Retrieve Header Information for a Complete INDNO list

Obs	KYINDNO	INDNAMEG
1	1000012	CRSP NYSE Market Capitalization Deciles
2	1000032	CRSP Amex Market Capitalization Deciles
3	1000052	CRSP NYSE/Amex Market Capitalization Deciles
4	1000072	CRSP Nasdaq Market Capitalization Deciles
5	1000092	CRSP NYSE/Amex/Nasdaq Market Capitalization Deciles
6	1000112	CRSP NYSE/Amex Beta Deciles
7	1000132	CRSP NYSE/Amex Standard Deviation Deciles
8	1000152	CRSP Nasdaq Beta Deciles
9	1000172	CRSP Nasdaq Standard Deviation Deciles

Example 51.10: Retrieving Daily Group Time Series by the INDNO= Option

This example shows how to retrieve daily group time series by using the INDNO keys in the IND database that were found in Example 51.9.

```
title 'Retrieve Daily Group Time Series by INDNO';
libname _all_ clear;

libname crsp sasexccm
   "\\bb04smb01\thirdparty\lnx\crspdata\DIZ201006\"
   setid=440
   indno=1000012 indno=1000032
   itemlist="AINDG.*;ARETG.*;USDCNTG.*;USDVALG.*";

 data dgindts_all ( where=( caldt >= '29jun2010'd) );
    set crsp.dlygind_ts;
 run;

 proc print data=dgindts_all; run;
```

Output 51.10.1 Daily Group Indices Time Series by INDNO

Retrieve Daily Group Time Series by INDNO

Obs	KYINDNO	KEYSET_TAG	CALDT	AINDG	ARETG	USDCNTG	USDVALG
1	1000012	1	20100629	7830.32	-0.027881	212	18621519.46
2	1000012	1	20100630	7818.16	-0.001553	212	18102340.54
3	1000012	2	20100629	2152.53	-0.029494	218	41076860.87
4	1000012	2	20100630	2138.40	-0.006568	218	39865355.36
5	1000012	3	20100629	2062.74	-0.030899	221	68226374.39
6	1000012	3	20100630	2050.86	-0.005761	221	66118281.02
7	1000012	4	20100629	1889.98	-0.034272	220	109562324.72
8	1000012	4	20100630	1876.58	-0.007090	220	105767265.49
9	1000012	5	20100629	2452.63	-0.037792	219	176873704.06
10	1000012	5	20100630	2426.61	-0.010609	219	170189222.39
11	1000012	6	20100629	2352.25	-0.036727	218	273518451.00
12	1000012	6	20100630	2326.55	-0.010923	218	263473022.01
13	1000012	7	20100629	1744.62	-0.036671	221	441993909.25
14	1000012	7	20100630	1728.12	-0.009454	221	425785496.41
15	1000012	8	20100629	1837.74	-0.034768	217	728682180.02
16	1000012	8	20100630	1823.06	-0.007989	217	703347314.23
17	1000012	9	20100629	1539.83	-0.037301	217	1551328686.76
18	1000012	9	20100630	1528.09	-0.007619	217	1493462838.15
19	1000012	10	20100629	687.77	-0.029865	219	7887915836.89
20	1000012	10	20100630	682.07	-0.008285	219	7652344301.36
21	1000032	1	20100629	50180.52	-0.028314	46	421298.09
22	1000032	1	20100630	49647.30	-0.010626	46	409369.47
23	1000032	2	20100629	9074.88	-0.018115	53	1027381.77
24	1000032	2	20100630	8999.49	-0.008307	53	1008770.94
25	1000032	3	20100629	7605.46	-0.021016	49	1357042.08
26	1000032	3	20100630	7647.77	0.005564	49	1328521.95
27	1000032	4	20100629	4864.93	-0.019350	49	1882198.23
28	1000032	4	20100630	4862.58	-0.000483	49	1845777.95
29	1000032	5	20100629	3830.96	-0.019162	52	2935672.46
30	1000032	5	20100630	3836.25	0.001382	52	2879419.36
31	1000032	6	20100629	1985.95	-0.023069	49	3471137.55
32	1000032	6	20100630	1994.63	0.004371	49	3391061.96
33	1000032	7	20100629	1737.15	-0.026127	50	5429238.78
34	1000032	7	20100630	1723.08	-0.008097	50	5287388.16
35	1000032	8	20100629	985.90	-0.043221	49	8751647.84
36	1000032	8	20100630	982.62	-0.003332	49	8373396.28
37	1000032	9	20100629	2012.69	-0.035989	47	13946079.35
38	1000032	9	20100630	2008.03	-0.002314	47	13444178.17
39	1000032	10	20100629	580.82	-0.034960	48	77397528.18
40	1000032	10	20100630	574.67	-0.010592	48	74691749.11

Example 51.11: Retrieving Monthly Group Time Series by the INDNO= Option

This example shows how to retrieve monthly group time series by using the INDNO= option.

```
title 'Retrieve Monthly Group Time Series by INDNO';
libname _all_ clear;

libname crsp sasexccm
   "\\tappan\crsp1\data201008_little\MIZ201006\"
   setid=400
   indno=1000357
   itemlist="MTRETG.*;MUSDCNTG.*;MUSDVALG.*";

 data mgindts_all ( where=( mcaldt >= '01apr2010'd) );
    set crsp.mthgind_ts;
 run;

 proc contents data=mgindts_all; run;
 proc print data=mgindts_all; run;
```

Output 51.11.1 Monthly Group Indices Time Series by INDNO

Retrieve Monthly Group Time Series by INDNO

The CONTENTS Procedure

Data Set Name	WORK.MGINDTS_ALL	**Observations**	51
Member Type	DATA	**Variables**	6
Engine	V9	**Indexes**	0
Created	03/02/2015 16:30:56	**Observation Length**	64
Last Modified	03/02/2015 16:30:56	**Deleted Observations**	0
Protection		**Compressed**	NO
Data Set Type		**Sorted**	NO
Label			
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information

Data Set Page Size	65536
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	1021
Obs in First Data Page	51
Number of Data Set Repairs	0
ExtendObsCounter	YES
Filename	C:\Users\saskff\AppData\Local\Temp\SAS Temporary Files_TD7388_D79286_\mgindts_all.sas7bdat
Release Created	9.0401M3
Host Created	W32_7PRO

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Informat	Label
2	KEYSET_TAG	Char	24	6.	6.	KEYSET
1	KYINDNO	Num	8	7.	7.	Indno
3	MCALDT	Num	8	YYMMDDN8.	YYMMDD8.	Caldt
4	MTRETG	Num	8	11.6	11.6	Tret
5	MUSDCNTG	Num	8	8.	8.	Usdcnt
6	MUSDVALG	Num	8	15.2	15.2	Usdval

Output 51.11.1 *continued*

Retrieve Monthly Group Time Series by INDNO

Obs	KYINDNO	KEYSET_TAG	MCALDT	MTRETG	MUSDCNTG	MUSDVALG
1	1000357	1	20100430	0.008174	175	8536273485.00
2	1000357	1	20100528	-0.080409	175	8591982134.00
3	1000357	1	20100630	-0.049665	175	7875937725.00
4	1000357	2	20100430	0.027565	185	1788689604.00
5	1000357	2	20100528	-0.075078	185	1840767431.00
6	1000357	2	20100630	-0.050972	185	1704852934.00
7	1000357	3	20100430	0.040804	190	932695901.00
8	1000357	3	20100528	-0.069991	188	969616974.00
9	1000357	3	20100630	-0.060083	187	895650676.00
10	1000357	4	20100430	0.037353	184	564387964.00
11	1000357	4	20100528	-0.079216	183	582823225.00
12	1000357	4	20100630	-0.070191	183	537053252.00
13	1000357	5	20100430	0.042394	227	478255773.00
14	1000357	5	20100528	-0.081422	228	500248663.00
15	1000357	5	20100630	-0.057584	227	457590748.00
16	1000357	6	20100430	0.057838	222	322738641.00
17	1000357	6	20100528	-0.078382	222	343700243.00
18	1000357	6	20100630	-0.081953	222	317307907.00
19	1000357	7	20100430	0.050365	292	287694406.00
20	1000357	7	20100528	-0.066656	291	301676691.00
21	1000357	7	20100630	-0.071452	292	283193837.00
22	1000357	8	20100430	0.065176	358	220136974.00
23	1000357	8	20100528	-0.075101	355	233544237.00
24	1000357	8	20100630	-0.071157	354	215763356.00
25	1000357	9	20100430	0.069557	514	180205516.00
26	1000357	9	20100528	-0.083660	514	193398849.00
27	1000357	9	20100630	-0.078070	520	180564497.00
28	1000357	10	20100430	0.096078	1374	134793898.00
29	1000357	10	20100528	-0.087666	1371	149100089.00
30	1000357	10	20100630	-0.079349	1368	137743847.00
31	1000357	11	20100430	0.011533	360	10324963089.00
32	1000357	11	20100528	-0.079468	360	10432749564.00
33	1000357	11	20100630	-0.049897	360	9580790659.00
34	1000357	12	20100430	0.040203	601	1975339638.00
35	1000357	12	20100528	-0.075396	599	2052688862.00
36	1000357	12	20100630	-0.062350	597	1890294676.00
37	1000357	13	20100430	0.057194	872	830570020.00
38	1000357	13	20100528	-0.073486	868	878921171.00
39	1000357	13	20100630	-0.075456	868	816265100.00
40	1000357	14	20100430	0.080906	1888	314999414.00
41	1000357	14	20100528	-0.085404	1885	342498939.00
42	1000357	14	20100630	-0.078624	1888	318308344.00
43	1000357	15	20100430	0.016137	961	12300302728.00
44	1000357	15	20100528	-0.078799	959	12485438427.00
45	1000357	15	20100630	-0.051949	957	11471085335.00
46	1000357	16	20100430	0.063714	2760	1145569435.00
47	1000357	16	20100528	-0.076828	2753	1221420110.00

Output 51.11.1 *continued*

Retrieve Monthly Group Time Series by INDNO

Obs	KYINDNO	KEYSET_TAG	MCALDT	MTRETG	MUSDCNTG	MUSDVALG
48	1000357	16	20100630	-0.076345	2756	1134573445.00
49	1000357	17	20100430	0.020191	3721	13445872162.00
50	1000357	17	20100528	-0.078623	3712	13706858536.00
51	1000357	17	20100630	-0.054145	3713	12605658779.00

References

Center for Research in Security Prices (2002a). *CRSP Programmer's Guide*. Chicago: CRSP, Chicago Booth. http://www.crsp.chicagobooth.edu/documentation/.

Center for Research in Security Prices (2002b). *CRSP SFA Guide*. Chicago: CRSP, Chicago Booth. http://www.crsp.chicagobooth.edu/documentation/.

Center for Research in Security Prices (2003a). *CRSP Data Description Guide*. Chicago: CRSP, Chicago Booth. http://www.crsp.chicagobooth.edu/documentation/.

Center for Research in Security Prices (2003b). *CRSP Utilities Guide*. Chicago: CRSP, Chicago Booth. http://www.crsp.chicagobooth.edu/documentation/.

Center for Research in Security Prices (2003c). *CRSPAccess Database Format Release Notes*. Chicago: CRSP, Chicago Booth. http://www.crsp.chicagobooth.edu/documentation/.

Center for Research in Security Prices (2011). *CRSP/Compustat Merged Database Guide*. Chicago: CRSP, Chicago Booth.

Center for Research in Security Prices (2012). *CRSP Utilities and Program Libraries Guide: CRSP US Stock & US Indices Databases and CRSP/Compustat Merged Database*. Chicago: CRSP, Chicago Booth.

Chapter 52
The SASEXFSD Interface Engine

Contents

Overview: SASEXFSD Interface Engine **3558**
Getting Started: SASEXFSD Interface Engine **3560**
Syntax: SASEXFSD Interface Engine **3561**
The LIBNAME *libref* SASEXFSD Statement 3563
The ExtractEconData Factlet 3567
The ExtractFormulaHistory Factlet 3571
The ExtractDataSnapshot Factlet 3572
The ExtractBenchmarkDetail Factlet 3573
The ExtractOFDBItem Factlet 3575
The ExtractOFDBUniverse Factlet 3576
The ExtractScreenUniverse Factlet 3576
Details: SASEXFSD Interface Engine **3577**
FactSet Data and FactSet Sourced Data 3577
SAS Output Data Set 3578
SAS OUTXML File 3579
SAS XML Map File 3579
Specifying Date Ranges and Frequency Codes 3579
Specifying Currency Codes 3581
Examples: SASEXFSD Interface Engine **3584**
Example 52.1: Retrieving Standardized Economic Items for Multiple Countries 3584
Example 52.2: Retrieving Economic Items by Using the FQL Syntax for Function *Z* Score 3585
Example 52.3: Using ECON_EXPR_DATA with the FQL Syntax for Function Returns 3587
Example 52.4: Using SPEC_ID_DATA with the FQL Economic Download Syntax . . 3589
Example 52.5: Using Multiple Database Sources with the FQL Syntax 3591
Example 52.6: Retrieving Price Data for One Company 3592
Example 52.7: Retrieving Price and Sales Data for Multiple Companies 3593
Example 52.8: Retrieving Book Value Data for One Company by Using Relative Dates 3594
Example 52.9: Retrieving Multiple Screen Items for Multiple Companies 3595
Example 52.10: Retrieving Data by Using the ISON= and ISONPARAMS= Options . 3596
Example 52.11: Retrieving Benchmark Data by Using the CUTOFF= option 3597
Example 52.12: Retrieving Benchmark Data by Using the MATCHDATE= Option . . 3598
Example 52.13: Retrieving Multiple Items for Multiple Companies from an OFDB File 3600
Example 52.14: Retrieving a List of Securities from an OFDB file 3601
Example 52.15: Retrieving a List of CUSIPs from a Screen File 3602
References **3603**

Overview: SASEXFSD Interface Engine

The SASEXFSD interface engine enables SAS users to access both FactSet data and FactSet-sourced data that are provided by the FactSet OnDemand service (formerly known as FASTFetch). This service provides access to many FactSet Data Sources and to many other databases. This chapter focuses on accessing the FactSet Fundamentals database, but additional databases and data types are available for use with the SASEXFSD interface engine. For a more comprehensive list of available data, enter the following URL in your web browser:

`http://www.factset.com/data/data`

For detailed descriptions of other databases that you can access, refer to the FactSet workstation Online Assistant.

The SASEXFSD engine uses the LIBNAME statement to specify which factlet (provided by FactSet) to use to open a database and what parts of the database to access. Factlets are functions that encapsulate business logic and data collection procedures. The technology is capable of cross referencing and dealing with time series for a large amount of data.

To specify the factlet, name one of the supported factlets that are listed in Table 52.1. Table 52.5 shows where to find a summary of each factlet's options.

Table 52.1 Supported FactSet OnDemand Factlets

Supported Factlet	Example
ExtractEconData	Example 52.1, Example 52.2
ExtractEconData (cont.)	Example 52.3, Example 52.4, Example 52.5
ExtractFormulaHistory	Example 52.6, Example 52.7, Example 52.8
ExtractDataSnapshot	Example 52.9, Example 52.10
ExtractBenchmarkDetail	Example 52.11, Example 52.12
ExtractOFDBItem	Example 52.13
ExtractOFDBUniverse	Example 52.14
ExtractScreenUniverse	Example 52.15

The Prefix column in Table 52.2 contains the parameters that you are most likely to refer to when requesting data, but each factlet has its own set of optional parameters and default settings. Often the items that you select use a prefix (see the Prefix column) to designate the database where the item resides. Because the availability of data libraries and their contents are constantly changing, Table 52.2 is included for instructional purposes only.

Table 52.2 Sample Databases Available through FactSet

Prefix	Database
ff	FactSet Fundamentals
fe	FactSet Estimates
fg	FactSet Global
p	FactSet Prices—Security Price Data

Table 52.2 shows only a subset of the available FactSet databases. For a more comprehensive list that also includes third-party databases available through FactSet, refer to the FactSet Online Assistant, page ID 2014.

To specify the data library, specify both a physical path to indicate the location of the data files (XML data returned from FactSet OnDemand) and the LIBNAME statement options to specify which factlet to use to request data items and the desired key IDs (identifiers, such as tickers or country codes) for your selection. The orientation of the data that are returned is ETI, entity-time-item, and is kept with sorted keys (entities or BY groups); each observation is indexed by time interval, and each time series data item is organized in columns by item name (time series variable name). The SASEXFSD engine supports the parameters that are required for each supported factlet.

Use the SASEXFSD engine to access all available data library items. To get started, look at the FactSet Fundamentals data items in Table 52.3.

For a complete list of data items for every category, refer to the FactSet Online Assistant, page ID 16331. FactSet workstation user name and serial number credentials are necessary to launch the FactSet Online Assistant from the FactSet workstation. A FactSet representative can provide these credentials.

Because the availability of data libraries and their contents are constantly changing, Table 52.3 is included for instructional purposes only. At the time of this writing, the available data list and items for the FactSet Fundamentals database are as shown in Table 52.3.

Table 52.3 FactSet Fundamentals Database Sets

Formulas by Category	Data Items (SAS Variable Names) in Category
Identifiers	FF_CUSIP, FF_DISCL_ID, FF_ISIN, FF_SEDOL, FF_TICKER, FF_WS_ID
Balance sheet	FF_ASSETS, FF_BDEBT, FF_GW, FF_INVEN_FG, FF_PPE_DEP, FF_PPE_GROSS, FF_PAY_ACCT
Income statement	FF_COGS, FF_DEP_EXP, FF_DIL_ADJ, FF_EBIT, FF_EQ_AFF_INC, FF_EXP_OPER, FF_GROSS_INC
Cash flow	FF_DEBT_CF, FF_DIV_CF, FF_FIN_CF, FF_CAPEX, FF_INVEST_CF, FF_INVEST_PURCH_CF, FF_SALE_ASSAETS_BUS_CF
Ratios	FF_ASSETS_EQ, FF_DEBT_EQ, FF_LIFE_INS, FF_LOAN_ASSETS, FF_NET_CAP_REQUIRE, FF_RD_SALES
Market data	FF_ACQ_DATE, FF_DIV_RATE, FF_ENTITY_TYPE, FF_PRICE_CLOSE, FF_PRICE_HIGH_52WK
Corporate data and classifications	FF_GEN_IND, FF_IND_GRP, FF_MAJOR_SUBIND, FF_SIC_CODE, FF_EMP_NUM
Financial records	FF_ACTG_STANDARD, FF_COVERAGE, FF_CURN_DOC, FF_DEPS_BK, FF_FREQ_CODE, FF_US_GAAP_AVAIL

For a comprehensive list of FactSet Fundamentals data items, refer to the FactSet Online Assistant, page ID 15099. If the page is not found, enter “FactSet Fundamentals” in the search box near the top of the page.

Getting Started: SASEXFSD Interface Engine

To specify the parts of the database that you want to access, you supply two things: the list of IDs for the companies or securities that you want to access, and the list of data items that you want to retrieve.

When accessing company or security data, use the ExtractFormulaHistory factlet. Use the IDS= option to specify the list of IDs that identify the companies that you want to access by specifying the entity ID (such as the ticker symbol) for each company.

For example, the following statements access the FactSet Fundamentals database for monthly sales data (ff_sales) and the Prices database for pricing data (p_price) for IBM (ID='ibm') and for FactSet Research Systems (ID='fds'). To include both IDs in the same request, specify IDS='ibm,fds'; to include both data items in the same request, specify ITEMS='p_price,ff_sales', as follows:

```
LIBNAME myLib sasexfsd "%sysget(FACTSET)"
   FACTLET=ExtractFormulaHistory
   FORMAT=sml
   OUTXML=gstart1
   AUTOMAP=replace
   MAPREF=MyMap
   XMLMAP="%sysget(FACTSET)gstart1.map"
   IDS='ibm,fds'
   ITEMS='p_price,ff_sales'
   DATES='20110130:20110631:m'
   ORIENTATION=eti
   user='XXXXXXXXXXXXXXX'
   pass='XXXXXXXXXXXXXXX';
   ;

data company_pvol;
   set myLib.gstart1;
run;

proc contents data=company_pvol; run;
proc print data=company_pvol; run;
```

Figure 52.1 Getting Started with ExtractFormulaHistory: Company_pvol

Obs	FQL_ENTITY	date	p_price	ff_sales
1	ibm	01-31-2011	162.000	99870.00
2	ibm	02-28-2011	161.880	99870.00
3	ibm	03-31-2011	163.070	99870.00
4	ibm	04-30-2011	170.580	99870.00
5	ibm	05-31-2011	168.930	99870.00
6	ibm	06-30-2011	171.550	99870.00
7	fds	01-31-2011	100.800	641.06
8	fds	02-28-2011	104.880	641.06
9	fds	03-31-2011	104.730	641.06
10	fds	04-30-2011	109.410	641.06
11	fds	05-31-2011	110.860	641.06
12	fds	06-30-2011	102.320	641.06

The SASEXFSD engine supports only the SAS XML (SML) format and the ETI orientation. The XML data that are returned from the FactSet OnDemand service are placed in a file specified by the OUTXML= option. The XML map that is automatically created is assigned the full pathname specified by the XMLMAP= option, and the fileref that is used for the map assignment is specified by the MAPREF= option. In the preceding example, the SASEXFSD engine uses the MAPREF= and XMLMAP= options in the FILENAME statement to assign a filename:

```
FILENAME MyMap "U:\factset\test\gstart1.map";
```

You can use the MAPREF= and XMLMAP= options to control where the map resides, what you name the map, and how you refer to it with a fileref. You can use the OUTXML= option to name your XML data file; this is described in the section "SAS OUTXML File" on page 3579. This data file is placed in the folder designated by 'physical-name', which is described in the section "The LIBNAME *libref* SASEXFSD Statement" on page 3563. You can refer to your data by using the myLib libref in your SASEXFSD LIBNAME statement. In the preceding program, this statement appears inside the DATA step in the SET statement, which names the input data set myLib.gstart1 and causes the reading of the GSTART1.xml file to be input and stored in the SAS data set Company_pvol.

The Company_pvol data set contains two time series variables (data items), p_price and ff_sales, as specified in the ITEMS= option, and the observation range is controlled by the DATES= option. The prefixes, ff_ and p_, are the database designators for the FactSet Fundamentals and Prices databases, respectively, as shown in Table 52.2. The Company_pvol data set contains observations that range from January 30, 2011, to June 31, 2011, as specified in the DATES= option. The frequency of the data is monthly, as specified by the 'm' at the end of the DATES= option. Figure 52.1 shows the results.

To specify the list of data items that you want to retrieve, use the ITEMS= option. This option accepts a string, enclosed in single quotation marks, that denotes a list of data items that you are selecting for the resulting SAS data set. The data item names are separated by commas, so valid item names cannot contain embedded commas or quotation marks. The prefix in each data item name designates the data source as defined in the Prefix column of Table 52.2.

After the libref is assigned by the LIBNAME statement, the database is opened. The selected data are organized into group entities (BY groups) that are sorted by date. In Figure 52.1, the tickers are the BY groups, and within each ticker, the observations are sorted by the time ID variable DATE.

You can also use the SAS DATA step to perform further subsetting and to store the resulting time series in a SAS data set.

The SASEXFSD engine is supported on 64-bit Windows and on Linux X64 (LAX) platforms.

Syntax: SASEXFSD Interface Engine

The SASEXFSD interface engine uses standard engine syntax. Table 52.4 summarizes the options that the SASEXFSD engine supports. In addition, there are two required options: USERNAME='*fact_username*' and PASS='*fact_password*'.

Table 52.4 Summary of LIBNAME *libref* SASEXFSD Options

Option	Description
FACTLET=	Specifies which factlet you want to use. See Table 52.5 for the complete list.
IDS=	Specifies a list of FactSet keys or entity identifiers for accessing FactSet OnDemand data. To select more than one ID, list the unique entity identifiers separated by commas.
ITEMS=	Specifies a list of FactSet data items for accessing FactSet data sources. To select more than one item, list the data item names, separated by commas.
ITEM=	Specifies one FactSet data item name (time series name)
DATES=	Specifies a list with the start date, end date, and frequency, separated by colons (:)
DATE=	Specifies one date in YYYYMMDD format (default is 0B, for today's date)
CURRENCY=	Specifies a currency in which the data are returned, using a three-character ISO code, such as USD for US dollars or EUR for euros
FORMAT=	Specifies a FactSet format. Only SAS XML format (SML format) is supported.
ORIENTATION=	Specifies the layout of the selected data items for access. Only the ETI (entity-time-item) orientation is supported.
PROXY=	Specifies the proxy server you want to use (if you have trouble connecting without specifying a proxy)
CONNECT=	Specifies whether to establish a secure connection that uses the proxy server that is specified in the PROXY= option
ISON=	Specifies whether the company (security), such as SP500 or MSCI_WORLD, *is on* the specified database or index. For a list of additional ISON= option examples, refer to the FactSet Online Assistant, page ID 2014.
ISONPARAMS=	Specifies the parameters used by the ISON code
START=	Specifies the start date for the selected data range (in YYYYMMDD format)
END=	Specifies the end date for the selected data range (in YYYYMMDD format)
FREQ=	Specifies the reporting frequency of the selected data, such as M for monthly and D for daily. See Table 52.20 for the complete list of frequencies.
PERIOD=	Specifies the time interval between the data points (observations) in a time series. The valid period parameters are ANN, QTR, SEMI, MON, YTD, YTD_SEMI, LTM, LTM_SEMI, and SEMI-ANN. The default is ANN.
CAL=	Specifies the calendar that replicates the PSETCAL function
DBSOURCE=	Specifies a standardized database source name, such as EWIN_ECON, GI_ECON, OECD_MEI, FDS_ECON, FDS_COM, or the default, a null string, which uses the standardized economic data (EWIN_ECON_RGDPR_Y)
DIST=	Specifies the distribution technique for spreading over periods when a data series of a lower frequency is distributed to a higher frequency; for example, distributing an annual series to a quarterly series (such as STEP, EVEN, or NONE)
CONV=	Specifies the conversion technique for aggregating periods, when a data series of a higher frequency is converted to a lower frequency; for example, converting a quarterly series to an annual series (such as SUM, AVERAGE, or AVERAGENP)
NFB=	Specifies the "no-feel-back" option in FQL codes. If you do not use the NFB= option, the returned data series contains NAs where the data are not available (default is NFB=1).
FUNCTION=	Adds the FQL function property to change data value; for example, FUNCTION=ZSCORE
OFDB=	Specifies the OFDB filename
UNIVERSE=	Specifies the one account or benchmark. Use this instead of the IDS= option.
UNIVERSEGROUP=	Default value is EQUITY; for DEBT securities, use UNIVERSEGROUP=DEBT
SCREEN=	Specifies the screen file that contains a single user-defined screen for viewing CUSIPs
NAME=	Specifies whether to see the names of each security along with the CUSIP
FQLFLAG=	Sets dates to use FQL instead of screening; FQLFLAG=N (default) or Y

The LIBNAME *libref* SASEXFSD Statement

LIBNAME *libref* **SASEXFSD** '*physical-name*' **FACTLET=***fact_factletname options* **;**

The LIBNAME statement assigns a SAS library reference (libref) to the physical path of the directory of FactSet data files where the downloaded FactSet XML data are stored. Because the required '*physical name*' argument specifies the location of the folder where your FactSet XML data reside, it should end in a backslash if you are in a Windows environment and a forward slash if you are in a UNIX environment.

FACTLET=*fact_factletname* specifies the FactSet factlet that you want to use to download your data. Choose one factlet from these possible values: ExtractEconData, ExtractFormulaHistory, ExtractDataSnapshot, ExtractBenchmarkDetail, ExtractOFDBItem, ExtractOFDBUniverse, and ExtractScreenUniverse. (See Table 52.1.)

For example, the following statements access the FactSet database for daily dividend yield data for IBM:

```
LIBNAME myLib SASEXFSD 'physical-name' FACTLET==ExtractFormulaHistory
   IDS='ibm'
   ITEMS='FG_DIV_YLD'
   FREQ=d
   USER='username'
   PASS='password';
```

After the libref is assigned, you can access the data items for the IDs (keys) from the requested factlet.

You can specify the following *options* in the LIBNAME *libref* SASEXFSD statement.

FACTLET=*fact_factletname*

Each factlet type has its own set of parameters (shown in Table 52.5 in the Factlet Options Table column), allowing flexibility and easy access to FactSet data. For more details about each factlet, refer first to the Factlet Description Section listed in Table 52.5. If you need more information, refer to the Online Assistant, page ID 16948. If the factlet is not listed on that page, then enter the factlet name in the search window of the Online Assistant to retrieve additional information about using the factlet.

Table 52.5 Summary of Factlet Options

Factlet Name	Factlet Description Section	Factlet Options Table
ExtractEconData	"The ExtractEconData Factlet" on page 3567	Table 52.10
ExtractFormulaHistory	"The ExtractFormulaHistory Factlet" on page 3571	Table 52.12
ExtractDataSnapshot	"The ExtractDataSnapshot Factlet" on page 3572	Table 52.13
ExtractBenchmarkDetail	"The ExtractBenchmarkDetail Factlet" on page 3573	Table 52.15
ExtractOFDBItem	"The ExtractOFDBItem Factlet" on page 3575	Table 52.16
ExtractOFDUniverse	"The ExtractOFDBUniverse Factlet" on page 3576	Table 52.17
ExtractScreenUniverse	"The ExtractScreenUniverse Factlet" on page 3576	Table 52.18

IDS=*fact_ids*

specifies a list of FactSet IDs (entity identifiers or keys) for accessing FactSet OnDemand data. To select more than one ID, list the unique entity identifiers, separated by commas (as shown in the following statements). Examples of FactSet IDs include CUSIPs, tickers, SEDOLs, Quick Code, and CINS (CUSIP International Numbering System). For more information, see Example 52.6 and Example 52.7.

```
LIBNAME myLib sasexfsd 'physical-name'
   ids='IBM,MSFT'
   ITEMS='p_price,p_volume,ca_sales';
```

UNIVERSE=*fact_uni*

specifies the universe of securities that passes the specified screening criteria. Up to 500 securities can be returned when this option is specified together with the FACTLET=ExtractFormulaHistory option. This limit is due to the US_UNIV function that is used within the ExtractFormulaHistory factlet to fetch the universe. You can also specify this option together with the FACTLET=ExtractDataSnapshot option, but because the data that are returned as of one specified date, there is no limit on the number of securities that can be returned.

```
LIBNAME myLib sasexfsd 'physical-name'
   factlet=ExtractFormulaHistory
   universe="URANKX((FS_PARENT_EQUITY=CUSIP AND EC_MKT_CAP(0,'CUR=USD')>10 AND P_PRICE(
   items='p_price(0,-4,M)';
```

ITEMS='*fact_itemlist*'

specifies the items and groups of interest for selection based on IDs (keys). Use FactSet's Formula Lookup for a complete list of data items, which is described in the FactSet Online Assistant.

Because the availability of data libraries and their contents are constantly changing, the following tables are included for instructional purposes only. Many other databases are available that are not shown in Table 52.6 to Table 52.9.

Table 52.6 Some FactSet Data Items

Data Source	Table Reference	Online Assistant Page ID
FactSet Fundamentals Data Items	Table 52.7	Page ID 15099
FactSet Global Formula Library	Also see Online Assistant Sidebar	Page IDs 13299, 16664
FactSet Global Indices Formulas	Table 52.8	Page ID 14336
Global Constituents Formulas	Table 52.9	Page ID 15086

Table 52.7 Some FactSet Fundamentals Data Items

Data Source	Online Assistant Page ID
Consolidated Items (FF_)	Page ID 16331
Debt Capital Structure	Page ID 16235
Enhancements to Legacy Formulas	Page ID 16248
Annual Items (FA_)	
Balance Sheet	Page ID 15120
Income Statement	Page ID 15121
Funds Flow Statement	Page ID 15122
Financial Ratios	Page ID 15123
Per Share and Valuation	Page ID 15124
Multiple Share Information	Page ID 15125
Accounting Policies and Methods	Page ID 15126
Segment Data	Page ID 15127
Monthly Items (FM_)	
Monthly Data	Page ID 15128

Table 52.8 FactSet Global Indices Formulas

Data Source	Online Assistant Page ID
Using FG Indices Formulas	Page ID 14337
Database Descriptions for Global Indices	Page ID 14338

Table 52.9 Global Constituents Formulas

Global Constituents Formula	Items
Benchmark Constituent Classification	FG_CONST_CLASS
Benchmark Constituent Country	FG_CONST_COUNTRY
Benchmark Constituent Currency	FG_CONST_CURRENCY
Benchmark Constituent Date	FG_CONST_DATE
Benchmark Constituent Float Factor	FG_CONST_FLOAT_FACTOR
Benchmark Constituent Identifier	FG_CONST_IDENTIFIER
Benchmark Constituent Latest Update	FG_CONST_UPDATE
Benchmark Constituent Market Value	FG_CONST_MCAP
Benchmark Constituent Name	FG_CONST_NAME
Benchmark Constituent Price	FG_CONST_PRICE
Benchmark Constituent Shares	FG_CONST_SHARES
Benchmark Constituent Style Factor	FG_CONST_STYLE_FACTOR
Benchmark Constituent Total Return - 1 Day	FG_CONST_TRET_1D
Benchmark Constituent Valuation	FG_CONST_VALUATION
Benchmark Constituent Weights	FG_CONST_WEIGHT
Benchmark Constituents	FG_CONSTITUENTS

For more information, see the FactSet Online Assistant, page ID 1931. To see each data source's list of available data items, use the search feature of the FactSet Online Assistant. You can open any page in the FactSet Online Assistant by entering the appropriate page ID number in the page ID window, which is located below the search window.

DATES=*'fact_startdate:fact_enddate:fact_freqcode'*
specifies the start date, end date, and frequency, separated by colons (:). For more information, see the section "Specifying Date Ranges and Frequency Codes" on page 3579. An alternative to using the DATES= option is to use the START=, END=, and FREQ= options.

PERIOD=*fact_period*
specifies the periodic frequencies of the actual data points (observations) in a time series. The valid period parameters are ANN, QTR, SEMI, MON, YTD, YTD_SEMI, LTM, LTM_SEMI, and SEMI-ANN. The default is ANN.

OUTXML=*fact_xmlfile*
specifies the filename of the data file where the XML data that are returned from the FactSet OnDemand service are stored. You can use the OUTXML= option to name your XML data file, which is placed in the folder designated by 'physical-name', as shown in the section "The LIBNAME *libref* SASEXFSD Statement" on page 3563.

AUTOMAP=*fact_automap*
specifies whether to overwrite the existing XML map file (AUTOMAP=REPLACE) or whether not to overwrite the existing XML map file (AUTOMAP=REUSE). You can set *fact_automap* to REUSE so that a pre-existing XML map named by the XMLMAP= option is used. You can set *fact_automap* to REPLACE so that the most current XML map generated by the SASEXFSD engine and named by the XMLMAP= option is used.

XMLMAP=*fact_xmlmapfile*
specifies the fully qualified name of the file where the XML map is automatically stored.

MAPREF=*fact_xmlmapref*
specifies the fileref to be used for the map assignment.

You can use the MAPREF= and XMLMAP= options to control where the map resides, what you name the map, and how you refer to it with a fileref. You can use the OUTXML= option to name your XML data file. This file is placed in the folder designated by 'physical-name', and you can reference it by using the myLib libref in your SASEXFSD LIBNAME statement. This is shown in the section "Getting Started: SASEXFSD Interface Engine" on page 3560. The following filename statement is generated by the SASEXFSD interface engine by using the fileref, MyMap, from the MAPREF=MyMap option and from the fully designated filename in the XMLMAP= option:

```
FILENAME MyMap "U:\factset\test\gstart1.map";
```

FORMAT=*fact_xmlformat*
specifies the SAS XML (SML) format, which is the only format that the SASEXFSD engine supports.

ORIENTATION=*fact_xmlorient*
specifies the ETI orientation, which is the only orientation that the SASEXFSD engine supports. The ETI orientation means that the data are returned and stored in entity-time-item logical layout.

PROXY="*fact_proxyserver***"**
specifies the proxy server to use when a connection-refused error or a connection-timed-out error occurs and CONNECT=ON. This option is optional. For *fact_proxyserver*, specify the server's HTTP address followed by a colon and the port number, and enclose that string in double quotation marks. For example, PROXY="http://inetgw.unx.sas.com:8118".

CONNECT=ON | OFF
specifies whether to establish a secure connection that uses the proxy server specified in the PROXY= option when a connection-refused error or a connection-timed-out error occurs.

USERNAME='*fact_username***'**
specifies the FactSet user name that enables you to access the data provided by the FactSet OnDemand service.

PASS='*fact_password***'**
specifies the password that is paired with the user name to enable you to access the data provided by the FactSet OnDemand service. **NOTE:** These FactSet OnDemand user name and password credentials are different from your FactSet workstation login credentials. A FactSet representative can provide these credentials.

The ExtractEconData Factlet

The ExtractEconData factlet provides access to a broad array of macroeconomic content, interest rates and yields, country indices, and various exchange rate measures from both the FactSet Economics and the Standardized Economic databases. The ExtractEconData factlet uses the options listed in Table 52.10 to extract Economic data items for a list of country IDs or for no country IDs over time.

The DBSOURCE= option specifies a standardized database source name, such as EWIN_ECON, GI_ECON, OECD_MEI, FDS_ECON, FDS_COM, or the default, a null string, which uses the Standardized Economic data (EWIN_ECON_RGDPR_Y). Use the IDS= option to specify one or more country IDs based on the database source. See Table 52.11 for the complete list of country IDs that work with the standardized codes for the FactSet Economics database.

Use the ITEM= option to specify an FQL item based on the database source. You can also use the ITEM= option with the downloading syntax for Economic Request Codes described in the FactSet Online Assistant, page ID 11794, and shown in Example 52.2. Example 52.2 uses the shorthand FQL syntax in the ITEM= option to retrieve the same time series data items.

The SASEXFSD engine supports retrieval of the following FQL_entities: ECON_EXPR_DATA, SPEC_ID_DATA, FDS_ECON_DATA, EIU_ECON_DATA, CNS_ECON_DATA, EURO_STAT_DATA, IBJ_NIKKO_DATA, IMF_IFS_DATA, NTCS_ECON_DATA, OECD_OTLK_DATA, OECD_MEI_DATA, ONS_ECON_DATA, TCB_BCI_DATA, TCB_CCI_DATA, CEIC_ECON_DATA, and CEIC_CHINA_DATA.

A FactSet representative can provide you with permissions to access the databases that contain the time series data that you are interested in. See Example 52.3 for an example of using ECON_EXPR_DATA. For more information about using ECON_EXPR_DATA and SPEC_ID_DATA, see the section "Downloading Economic Function Codes to Excel" in the FactSet Online Assistant, page ID 12308. You can replace options such as START=, END=, FREQ=, CONV=, DIST=, NFB=, and FUNCTION= by using the corresponding placement of each option's value in the FQL downloading syntax. In this example, ITEM="FDS_ECON_DATA('FRBIPSB50001', -121,-1,m,step,average)" gives the database source name (FDS_ECON), time series name (FRBIPSB50001), start and end dates, monthly frequency, distribution (step), and conversion (average); no-feel-back defaults to 1.

See the ExtractEconData appendix, linked in the FactSet Online Assistant, page ID 16948, for the complete list of the data series codes available with the standardized FactSet Economics database. The data are available at monthly, quarterly, and annual frequencies, as denoted by the _M, _Q, and _Y suffixes in the code.

Use the FREQ= option to specify the frequency of the data, and use the DATES= option to specify a date range for selecting time series data. For more information, see the section "Frequency" in the FactSet Online Assistant, page ID 11794.

You can use the CONV= option to specify CONV=SUM, AVERAGE, AVERAGENP, or none (the default is none).

You can use the DIST= option to specify DIST=STEP (step distribution), EVEN, or none (the default).

You can use the NFB= option to specify your no-feel-back setting (0, 1, or 2) to indicate full feel-back, no feel-back (the default), or feel-back until most recent data point, respectively (do not fill in data past the last available data point).

You can use the FUNCTION= option to apply an economic function to the data values of the time series (such as FUNCTION=ZSCORE). For a list of economic functions, see the FactSet Online Assistant, page ID 12308.

The ORIENTATION= option supports only ETI, which is the default.

Table 52.10 ExtractEconData Factlet Options

Option	Description
IDS=	Specifies a string array with a list of the country identifiers from the Standardized Economic database
ITEMS=	Specifies the economic series mnemonic (for example, US GDP database source [mnemonic] is FDS_ECON[BEANIPAA191RL1@US])
DATES=	Specifies a date string such as 'YYYYMMDD:YYYYMMDD:F' or relative dates '-1b:-4b:m'
START=	Specifies the numeric start date in YYYYMMDD format
END=	Specifies the numeric end date in YYYYMMDD format
FREQ=	Specifies the valid FQL frequencies, such as M, D, W, Q, and Y. Note: For economic request codes, a frequency argument is necessary to retrieve the data.
DBSOURCE=	Specifies a standardized database source name, such as EWIN_ECON, GI_ECON, OECD_MEI, FDS_ECON, FDS_COM, or the default, a null string, which uses the Standardized Economic data (EWIN_ECON_RGDPR_Y)
CONV=	Specifies the conversion technique for aggregating periods, when a data series of a higher frequency is converted to a lower frequency; for example, you can use this option to convert a quarterly series to an annual series (for example, SUM, AVERAGE, or AVERAGENP, which excludes NAs).
DIST=	Specifies the distribution technique for spreading over periods when a data series of a lower frequency is distributed to a higher frequency; for example, you can use it to distribute an annual series to a quarterly series (such as STEP, EVEN, or NONE).
NFB=	Specifies the optional "no-feel-back" argument in FQL codes. If you do not specify the NFB= option, the returned data series contains NAs where the data are not available (default is NFB=1). If you want the data to "feel back" over NAs to find the last actual data point and carry these data forward, specify either NFB=0 or NFB=2.
FUNCTION=	Adds FQL function property to change data value; for example, FUNCTION=ZSCORE
ORIENTATION=	Specifies an optional orientation (default is ETI, entity-time-item)

Table 52.11 Country Identifiers

Country	Country ID	Country	Country ID
Argentina	CC_AR	Lithuania	CC_LT
Australia	CC_AU	Luxembourg	CC_LU
Austria	CC_AT	Malaysia	CC_MY
Azerbaijan	CC_AZ	Malta	CC_MT
Bangladesh	CC_BD	Mexico	CC_MX
Belarus	CC_BY	Morocco	CC_MA
Belgium	CC_BE	Netherlands	CC_NL
Bolivia	CC_BO	New Zealand	CC_NZ
Brazil	CC_BR	Nigeria	CC_NG
Bulgaria	CC_BG	Norway	CC_NO
Canada	CC_CA	Pakistan	CC_PK
Chile	CC_CL	Panama	CC_PA
China	CC_CN	Paraguay	CC_PY
Colombia	CC_CO	Peru	CC_PE
Costa Rica	CC_CR	Philippines	CC_PH
Croatia	CC_HR	Poland	CC_PL
Cyprus	CC_CY	Portugal	CC_PT
Czech Republic	CC_CZ	Romania	CC_RO
Denmark	CC_DK	Russia	CC_RU
Dominican Republic	CC_DO	Saudi Arabia	CC_SA
Ecuador	CC_EC	Singapore	CC_SG
Egypt	CC_EG	Slovakia	CC_SK
Estonia	CC_EE	Slovenia	CC_SI
Finland	CC_FI	South Africa	CC_ZA
France	CC_FR	South Korea	CC_KR
Germany	CC_DE	Spain	CC_ES
Greece	CC_GR	Sri Lanka	CC_LK
Hong Kong	CC_HK	Sweden	CC_SE
Hungary	CC_HU	Switzerland	CC_CH
Iceland	CC_IS	Taiwan	CC_TW
India	CC_IN	Thailand	CC_TH
Indonesia	CC_ID	Turkey	CC_TR
Ireland	CC_IE	Ukraine	CC_UA
Israel	CC_IL	United Kingdom	CC_GB
Italy	CC_IT	United States	CC_US
Japan	CC_JP	Uruguay	CC_UY
Jordan	CC_JO	Uzbekistan	CC_UZ
Kazakhstan	CC_KK	Venezuela	CC_VE
Latvia	CC_LV	Vietnam	CC_VN

The ExtractFormulaHistory Factlet

The ExtractFormulaHistory factlet is used for extracting one or more items for one security, for an index, or for a list of securities over time. ExtractFormulaHistory uses the FactSet Query Language (FQL). The ExtractFormulaHistory factlet uses the options listed in Table 52.12, such as the IDS= option, which specifies the IDs for one or more securities, or the ISON= and ISONPARAMS= options, which specify an FQL formula that extracts the universe along with any ISON parameters necessary for the ISON code. You can use the START=, END=, and FREQ= options or the DATES= option to specify a date range for selecting time series data. You can select data items by using the ITEMS= option, but only the name/value pairs syntax (not the standard FQL syntax) is supported. The ITEMS= option designates multiple shortcut items or item/statistic combinations. You can use any instance of the Formula Library for the ITEMS= option.

The PERIOD= option is used for FactSet Fundamentals database codes to specify the estimate period of the data that you want to select. The CAL= option enables you to set your calendar in the same way that the PSETCAL function in FQL works. You can specify the CAL= option to be LOCAL, FIVEDAY, FIVEDAYEOM, SEVENDAY, or an exchange code. The list of exchange codes is available in the FactSet Online Assistant, page ID 16610. The ORIENTATION= option is supported only for ETI (entity-time-item), so that your SAS output data set is organized by key entities such as CUSIP or ticker, by date so that observations are kept in time series order, and by item. ETI is the default setting for orientation.

Table 52.12 ExtractFormulaHistory Factlet Options

Option	Description
IDS=	Specifies one or more securities; for example, IDS=IBM,MSFT,FDS
ISON=	Specifies the FQL value that extracts the universe; for example, ISON_SP500 is entered as ISON=SP500, and ISON_MSCI_WORLD(0,1) is written as ISON=MSCI_WORLD.
ISONPARAMS=	Specifies the ISON codes that use parameters; for example, ISON_MSCI_WORLD(0,1) is written as ISONPARAMS=0,1.
UNIVERSE=	Specifies the universe.
DATES=	Specifies a date string such as YYYYMMDD:YYYYMMDD:F or relative dates -1b:-4b:m
START=	Specifies a valid FQL date; START=0 is the default
END=	Specifies a valid FQL date; END=0 is the default
FREQ=	Specifies a valid FQL frequencies; for example, M, D, Y. See Table 52.20.
ITEMS=	Specifies one or more FQL items (only the name/value pair syntax is supported; for example, ff_sales, p_price)
PERIOD=	Specifies valid time intervals between the data points (observations) in a time series; for example, ANN, QTR, and SEMI-ANN; PERIOD=ANN is the default.
ORIENTATION=	Specifies an optional orientation (default is currently ETI)
CAL=	Specifies the calendar setting that replicates the PSETCAL function; for example, LOCAL, FIVEDAY, FIVEDAYEOM, and SEVENDAY, for exchange code CAL=AAM (for a list of exchange codes, see the FactSet Online Assistant, page ID 16610)

The ExtractDataSnapshot Factlet

The ExtractDataSnapshot factlet is used for efficiently extracting multiple items as of a single date, for a universe of both equity and fixed income securities. It uses the FactSet Screening Language to extract data for a large universe of securities as of a single date. The ExtractDataSnapshot factlet uses the options listed in Table 52.13, such as the IDS= option, which specifies the IDS for one or more securities, or you can specify fixed securities by using the UNIVERSEGROUP= option. If you want to access only current constituents, use the ISON= option to specify your ISON codes instead of using the IDS= option. If your ISON code uses parameters, then use the ISONPARAMS= option to specify the parameters for the code that you use in your ISON= option. Use DATE=YYYYMMDD to specify the day that your snapshot is for, or use the START=, END=, and FREQ= options for the FQL scalar data item date that you are interested in. Use the ITEMS= option to specify one or more screening items. The SASEXFSD engine does not support the standard screening syntax, so use the name/value pair syntax instead. For example, instead of using ITEMS='FF_SALES(QTR,20110401)', use ITEMS='FF_SALES' PERIOD=QTR REL_DATE=20110401 in your LIBNAME *libref* SASEXFSD statement. Specify the UNIVERSEGROUP= option to choose between the EQUITY group and the DEBT group. The CAL= option enables you to set your calendar in the same way that the PSETCAL function in FQL works. You can specify the CAL= option to be LOCAL, FIVEDAY, FIVEDAYEOM, SEVENDAY, or an exchange code. The list of exchange codes is available in the FactSet Online Assistant, page ID 16610. The ORIENTATION= option is supported only for ETI (entity-time-item), which is also the default.

Table 52.13 ExtractDataSnapshot Factlet Options

Option	Description
IDS=	Specifies one or more securities; for example, IDS='IBM,MSFT'. Fixed securities are used in conjunction with UNIVERSEGROUP=DEBT (for example, IDS=88579EAE).
ISON=	Specifies a screening code that extracts the universe; for example, ISON_SP500 is entered as ISON=SP500, and ISON_MSCI_WORLD(0,1) is entered as ISON=MSCI_WORLD.
ISONPARAMS=	Specifies ISON codes that use parameters; for example, ISON_MSCI_WORLD(0,1) is written as ISONPARAMS=0,1.
DATE=	Specifies one date in the format YYYYMMDD (default is 0B, for today's date)
START=	Specifies a valid start date; START=0 is the default.
END=	Specifies a valid end date; END=0 is the default.
FREQ=	Specifies a valid frequency; for example, M, D, Y. See Table 52.20.
ITEMS=	Specifies one or more screening items (only the name/value pair syntax is supported)
PERIOD=	Specifies a valid time interval between the data points (observations) in a time series; for example, ANN, QTR, SEMI-ANN; PERIOD=ANN is the default.
UNIVERSEGROUP=	Specifies the universe group. The default value is EQUITY; for DEBT securities, specify UNIVERSEGROUP=DEBT.
ORIENTATION=	Specifies an optional orientation (default is ETI)
CAL=	Specifies a calendar setting that replicates the PSETCAL function; for example, LOCAL, FIVEDAY, FIVEDAYEOM, and SEVENDAY, for exchange code CAL=AAM (for a list of exchange codes, refer to the FactSet Online Assistant, page ID 16610)

The ExtractBenchmarkDetail Factlet

The ExtractBenchmarkDetail factlet is used for retrieving a more comprehensive overview of the index constituent data for a benchmark, without requiring the additional codes and calculations that are needed with the ExtractFormulaHistory factlet. ExtractBenchmarkDetail uses default output in which the identifiers are sorted in descending order by weight in the index, and each row shows the index ID, company ID, date, ticker, and weight. Any additional items are displayed at the end of each row.

The ExtractBenchmarkDetail factlet uses the options that are listed in Table 52.15, such as the IDS= option, which specifies the IDs for one or more benchmarks (indexes). Use DATES='YYYYMMDD:YYYYMMDD:freq' to specify the range of dates in start:end:freq format.

You can designate dates in absolute or relative form, as shown in Table 52.14. Absolute dates specify a day in 'MM/DD/YYYY' format (such as 7/11/1999), a month end in 'MM/YYYY' format (such as 6/1999), a fiscal quarter end in 'YY/FQ' or 'YYY/FQ' format (such as 1999/1F, 2000/3F, or 2001/2F), a calendar quarter end in 'YY/CQ' or 'YYYY/CQ' format (such as 1999/1C, 00/3C, or 2001/1C), or a fiscal year end in 'YY' or 'YYYY' format (such as 2000, 01, or 1999).

Table 52.14 ExtractBenchmarkDetail Factlet Relative Date Arguments

Relative Date Argument	Description
D	0D is the most recent trading day; –1D is one trading day prior.
AW	0AW is the most recent trading day; –1AW is the one actual week (7 days) prior to the most recent trading day.
W	0W is the last day of the most recent trading week (usually Friday); –1W is the last trading day of the prior week.
AM	0AM is the most recent trading day; –1AM is the same day, one actual month prior.
M	0M is the last trading day of the most recent month; –1M is the last trading day of the prior month.
AQ	0AQ is the most recent trading day; –1AQ is the same day 3 months prior.
Q	0Q is the last trading day of the company's most recent fiscal quarter; –1Q is the last day of the prior fiscal quarter.
CQ	0CQ is the last trading day of the most recent calendar quarter (March, June, September, or December); –1CQ is the last trading day of the prior calendar quarter.
AY	0AY is the most recent trading day; –1AY is one actual year (365 days) prior.
Y	0Y is the last trading day of the company's most recent fiscal year; –1Y is the last trading day of the prior fiscal year.
CY	0CY is the last trading day of the most recent calendar year (the last trading day in December); –1CY is the last trading day of the prior calendar year.

Use the ITEMS= option to specify one or more screening items, such as p_price or ca_sales. The CUTOFF= option specifies the number of holdings to show. The default is to show all instances. The optional MATCHDATE= option is used to limit the output to not repeat the dates that "feel back" to a holiday. The default behavior (no MATCHDATE= option) repeats the dates that "feel back" to a holiday. The MATCHDATE= option is always used with a frequency argument set to B, which indicates business days. Use MATCHDATE=ON for better response time and to limit the large amount of data being returned. The UNIVERSEGROUP= option is necessary only when you specify UNIVERSEGROUP=DEBT. By default, UNIVERSEGROUP=EQUITY.

Table 52.15 ExtractBenchmarkDetail Factlet Options

Option	Description
IDS=	Specifies one or more benchmarks; for example, IDS=SP50, R.3000
DATES=	Specifies one or more dates entered in start:end:freq format, '20101215:20100115:d'
ITEMS=	Specifies one or more screening items (such as ITEMS='p_price, ca_sales'). FQL items are named with a preceding underscore.
CUTOFF=	Specifies an optional number of holdings to show; default (no cutoff) displays all instances.
MATCHDATE=	Specifies an optional argument that turns off the default behavior, in which dates are repeated when "feeling back" to a holiday. It is always used with a frequency argument set to B (indicating business days).
UNIVERSEGROUP=	Specifies the universe group. The default value is EQUITY. For fixed income indices, specify UNIVERSEGROUP=DEBT.

The ExtractOFDBItem Factlet

The ExtractOFDBItem factlet provides access to a list of securities and multiple data items for a range of dates uploaded into a single Open FactSet Database (OFDB). An OFDB is a high-performance multidimensional database system that securely stores proprietary numeric and textual data in FactSet.The ExtractOFDBItem factlet uses the options listed in Table 52.16, such as the OFDB= option, which specifies the OFDB file. Use either the IDS= option, which specifies the IDs for one or more securities, or the ISON= and ISONPARAMS= options, which specify an FQL formula (ISON code) that extracts the universe along with any ISON parameters necessary for the ISON code. Use the ITEMS= option to specify one or more items in the OFDB file. Use the DATES= option to specify a date range in YYYYMMDD:YYYYMMDD:freq format, or use FQL dates when FQLFLAG=Y (yes). By default, FQLFLAG=N (no). The DATESONLY= option specifies whether only the dates in the OFDB file are reported. By default, DATESONLY=N. The ORIENTATION= option supports only ETI, which is the default.

Table 52.16 ExtractOFDBItem Factlet Options

Option	Description
OFDB=	Specifies the OFDB file
IDS=	Specifies one or more securities; for example, IDS=IBM, GM
ISON=	Specifies the FQL value that extracts the universe; for example, ISON_SP500 is entered as ISON=SP500, and ISON_MSCI_WORLD(0,1) is written as ISON=MSCI_WORLD.
ISONPARAMS=	Specifies the parameters for the ISON codes that use parameters; for example, ISON_MSCI_WORLD(0,1) is written as ISONPARAMS=0,1.
ITEMS=	Specifies one or more data items from the OFDB file
DATES=	Specifies dates in the format YYYYMMDD:YYYYMMDD:F or relative dates in relative format, such as -1b:-4b:m
DATE=	Specifies one date in the format YYYYMMDD (default is 0B, for today's date)
DATESONLY=	Specifies that dates are reported from the OFDB file only when the option is set to Y (default is N)
FQLFLAG=	Sets dates to use FQL instead of screening; FQLFLAG=N (default) or Y
ORIENTATION=	Specifies an optional orientation (default is currently ETI)

The ExtractOFDBUniverse Factlet

The ExtractOFDBUniverse factlet uses the options that are listed in Table 52.17 to extract a list of CUSIPs that belong to a single OFDB file or ISON code. Use the OFDB= option to specify a OFDB file, and use the DATE= option to specify the date for showing the list of CUSIPs.

Table 52.17 ExtractOFDBUniverse Factlet Options

Option	Description
OFDB=	Specifies the OFDB file
DATE=	Specifies one date in YYYYMMDD format only; DATE="" specifies the most recent date.

The ExtractScreenUniverse Factlet

The ExtractScreenUniverse factlet is used for extracting a list of CUSIPs stored in a single FactSet screen. On the FactSet workstation, a user can screen for equity and fixed income securities based on specified criteria and store a list of companies by using FactSet Universal Screening for equity or debt securities. The ExtractScreenUniverse factlet uses the options that are listed in Table 52.18 to extract a list of CUSIPs that belong to a single user-defined screen. Use the SCREEN= option to specify a screen file, and use the NAME= option to specify whether or not to make the names of the corresponding securities visible. Specify NAME=Y (yes) to view the security names for each CUSIP in the screen. NAME=N (no) is the default, for which security names are not shown with the CUSIP list. Because screen filenames can contain blanks and special characters, enclose the screen filename with single quotation marks:

```
options validvarname=any;
LIBNAME myFast sasexfsd "%sysget(FACTSET)"
    debug=on
    factlet=ExtractScreenUniverse
    screen='factset:1 Week EPS Estimate Revisions - FactSet Consensus'
    name=y
    format=sml
    outXml=sasscrn4
    automap=replace
    mapref=MyMap
    xmlmap="%sysget(FACTSET)sasscrn4.map"
    period=QTR
    user='XXXXXXXXXXXXXXXX'
    pass='XXXXXXXXXXXXXXXX'
    ;
```

Table 52.18 ExtractScreenUniverse Factlet Options

Option	Description
SCREEN=	Specifies a screen file
NAME=	Specifies whether security names are to be visible. NAME=Y indicates yes to show names; default is N, indicating that no names are shown.

Details: SASEXFSD Interface Engine

FactSet Data and FactSet Sourced Data

The SASEXFSD interface engine enables SAS users to access both FactSet data and FactSet sourced data that are provided by the FactSet OnDemand service. FactSet OnDemand offerings can provide access to many databases. Because the list of available data is constantly changing, Table 52.19 is included for instructional purposes only. Many other data offerings are available that are not shown in Table 52.19.

Table 52.19 Sample FactSet Data Types

Pricing and IPO Data
Estimates
Broker Research
Commodity Benchmarks
Equity Benchmarks
Fixed Income Benchmarks
Mutual Fund/Account Return Data
Economic Data
Financial News and Events/Corporate Information
Quantitative Data
Options Data
Investment Banking Data
Fixed Income Data
Deal Data
Other Databases

SAS Output Data Set

You can use the SAS DATA step to write the selected FactSet data to a SAS data set. This enables you to use SAS software to easily analyze the data. If you specify the name of the output data set in the DATA step, the engine supervisor creates a SAS data set that has the specified name in either the SAS Work library or, if specified, the User library.

The contents of the SAS data set include the BY groups, the date of each observation, and the series name of each series that is read from the FactSet data source.

The SASEXFSD interface engine sorts the IDs into keys or BY groups, so that the time series are sorted in the resulting SAS data set by key (entity identifier such as a ticker), by date (time ID), and by variable (time series item name).

You can use the PRINT and CONTENTS procedures to print your output data set and its contents. Alternatively, you can view your SAS output observations by opening the desired output data set in a SAS Explorer window. You can also use the SQL procedure with your SASEXFSD libref to create a custom view of your data.

SAS OUTXML File

The SAS XML (SML format) data that are returned from the FactSet OnDemand service are placed in a file named by the OUTXML= option. The SAS XML data file is placed in the location that is specified by the *physical-name* in your LIBNAME *libref* SASEXFSD statement.

SAS XML Map File

The XML map that is automatically created is assigned the full pathname specified by the XMLMAP= option in your LIBNAME *libref* SASEXFSD statement. The XML map file is either reused (not overwritten) if you specify AUTOMAP=REUSE or overwritten by a new map if you specify AUTOMAP=REPLACE. The SASEXFSD interface engine invokes the XMLV2 engine to create the map and to read the data into SAS.

Specifying Date Ranges and Frequency Codes

When you specify a range of dates for selecting your time series observations, you can specify the range in either absolute or relative dates. The absolute start and end dates are given in YYYYMMDD format and separated by a colon (:). The frequency is given along with the date range and can be any one of the codes shown in Table 52.20. The code frequency indicates the frequency with which you want to display data. Relative dates are relative to the most recently updated period (0). A minus sign (as in –1) represents the period prior to the most recently updated period. The zero date is determined by the natural frequency of the time series data, so a 0 for monthly data represents the most recent month end. Annual data use –1 to represent the fiscal year prior to the most recently updated fiscal year.

Table 52.20 FactSet Frequency Codes

Freq. Code	Description
AD	Displays data on an actual daily basis (that is, all days, not just trading days)
D	Displays data on a daily basis
AW	Displays data weekly, based on the day of the week of the start date
W	Displays data weekly, based on the last day of the completed trading week (usually Friday)
WTD	For a range item (such as price change), displays the week-to-date value. For other items, displays the latest daily value. For the remainder of the time series, displays data weekly, based on the last day of the completed trading week (usually Friday).
AM	Displays data monthly, based on the start date (for example, if the start date is June 16, data are displayed for June 16, May 16, April 16, and so on)
M	Displays data monthly, based on the last trading day of the month
MTD	For a range item (such as price change), displays the month-to-date value. For other items, displays the latest daily value. For the remainder of the time series, displays data monthly, based on the last trading day of the month.
QTD	For a range item (such as price change), displays the calendar quarter-to-date value. For other items, displays the latest daily value. For the remainder of the time series, displays data quarterly, based on the last trading day of the quarter.
CQTD	For a range item (such as price change), displays the calendar quarter-to-date value. For other items, displays the latest daily value. For the remainder of the time series, displays data quarterly, based on the last trading day of the calendar quarter.
FQTD	For a range item (such as price change), displays the fiscal quarter-to-date value. For other items, displays the latest daily value. For the remainder of the time series, displays data quarterly, based on the last trading day of the fiscal quarter.
AQ	Displays data in three-month periods, based on the start date (for example, if the start date is April 7, data are displayed for April 7, January 7, October 7, July 7, and so on)
Q	Displays data quarterly, based on the last trading day of the company's fiscal quarter
CQ	Displays data quarterly, based on the last trading day of the calendar quarter (March, June, September, or December)
FSA	Displays data semiannually, based on the last trading day of the fiscal semiannual period
CSA	Displays data semiannually, based on the last trading day of the calendar semiannual period
ASA	Displays data in six-month periods, based on the start date (for example, if the start date is June, data are displayed for June, January, June (prior), January (prior), and so on)
YTD	For a range item (such as price change), displays the calendar year-to-date value. For other items, displays the latest daily value. For the remainder of the time series, displays data annually, based on the last trading day of the year.
CYTD	For a range item (such as price change), displays the calendar year-to-date value. For other items, displays the latest daily value. For the remainder of the time series, displays data annually, based on the last trading day of the calendar year.
FYTD	For a range item (such as price change), displays the fiscal year-to-date value. For other items, displays the latest daily value. For the remainder of the time series, displays data annually, based on the last trading day of the fiscal year.
AY	Displays data annually, based on the start date (for example, if the start date is October 31, 1995, data are displayed for October 31, 1995, October 31, 1994, October 31, 1993, and so on)
Y	Displays data annually, based on the last trading day of the company's fiscal year
CY	Displays data annually, based on the last trading day of the calendar year

Specifying Currency Codes

Currency is represented by three-character ISO (International Organization for Standardization) codes, such as USD for US dollars or EUR for euros. For a complete list of currency codes, see Table 52.21 and Table 52.22.

Table 52.21 ISO Currency Codes

Currency	ISO Code	Currency	ISO Code
Afghanistan Afghani	AFN	Djibouti Franc	DJF
Albanian Lek	ALL	Dominican Rep. Peso	DOP
Algerian Dinar	DZD	East Caribbean Dollar	XCD
Angolan Kwanza	AOA	East German Ostmark	DDM
Argentine Peso	ARS	Ecuador US Dollar	USD
Armenia Dram	AMD	Egyptian Pound	EGP
Aruban Guilder	AWG	El Salvador Colon	SVC
Australian Dollar	AUD	Estonian Euro	EUR
Austrian Schilling*	ATS	Ethiopian Birr	ETB
Azerbaijan New Manat	AZN	Euro	EUR
Bahamas Dollar	BSD	Euro Floating Rate	EUX
Bahraini Dinar	BHD	European Currency Unit	XEU
Bangladesh Taka	BDT	Falkland Is. Pound	FKP
Barbados Dollar	BBD	Fiji Dollar	FJD
Belarus Rouble	BYR	Finnish Markka*	FIM
Belgian Franc*	BEF	French Euro	EUR
Belize Dollar	BZD	Gambia Dalasi	GMD
Bermuda Dollar	BMD	Georgian Lari	GEL
Bhutan Ngultrum	BTN	German Euro	EUR
Bolivian Boliviano	BOB	Ghana Cedi	GHS
Botswana Pula	BWP	Gibraltar Pound	GIP
Brazilian Real	BRL	Greek Drachma*	GRD
British Pence	GBX	Guatemala Quetzal	GTQ
British Pound	GBP	Guinea Franc	GNF
Brunei Dollar	BND	Guinea-Bissau Peso	XOF
Bulgarian Lev	BGN	Guyana Dollar	GYD
Burundi Franc	BIF	Haiti Gourde	HTG
Cambodian Riel	KHR	Honduras Lempira	HNL
Canadian Dollar	CAD	Hong Kong Dollar	HKD
Cape Verde Is. Escudo	CVE	Hungarian Forint	HUF
Cayman Islands Dollar	KYD	Icelandic Krona	ISK
CFA Franc (C. African)	XAF	Indian Rupee	INR
CFA Franc (W. African)	XOF	Indonesian Rupiah	IDR
CFP Franc	XPF	Iran Rial	IRR
Chile UF	CLF	Iraqi Dinar	IQD
Chilean Peso	CLP	Irish Punt*	IEP
China Yuan Renminbi	CNY	Israeli Shekel	ILS
Colombian Peso	COP	Italian Lira*	ITL
Comoros Franc	KMF	Jamaican Dollar	JMD
Costa Rica Colon	CRC	Japanese Yen	JPY
Croatian Kuna	HRK	Jordanian Dinar	JOD
Cuban Peso	CUP	Kazakhstan Tenge	KZT
Cyprus Pound*	CYP	Kenya Shilling	KES
Czech Koruna	CZK	Kuwait Dinar	KWD
Danish Krone	DKK	Kyrgyzstan Som	KGS

* The local currency and currency code are euro and EUR, respectively.

Table 52.22 Continued

Currency	ISO Code	Currency	ISO Code
Laos New Kip	LAK	Sao Tome and Principe Dobra	STD
Latvian Lats	LVL	Saudi Arabian Riyal	SAR
Lebanese Pound	LBP	Serbian Dinar	RSD
Lesotho Loti	LSL	Seychelles Rupee	SCR
Liberian Dollar	LRD	Sierra Leone Leone	SLL
Libyan Dinar	LYD	Singapore Dollar	SGD
Lithuanian Litas	LTL	Slovakia Koruna*	SKK
Luxembourg Franc*	LUF	Slovenian Tolar*	SIT
Macau Pataca	MOP	Solomon Is. Dollar	SBD
Macedonian Denar	MKD	Somali Shilling	SOS
Malagasy Ariary	MGA	South African Rand	ZAR
Malawi Kwacha	MWK	South Korean Won	KRW
Malaysian Ringgit	MYR	Spanish Peseta*	ESP
Maldive Is. Rufiyaa	MVR	Sri Lanka Rupee	LKR
Maltese Lira*	MTL	St. Helena Pound	SHP
Mauritania Ouguiya	MRO	Sudanese Dinar	SDG
Mauritian Rupee	MUR	Surinam Dollar	SRD
Mexican Peso	MXN	Swaziland Lilangeni	SZL
Moldovan Leu	MDL	Swedish Krona	SEK
Mongolian Tugrik	MNT	Swiss Franc	CHF
Moroccan Dirham	MAD	Syrian Pound	SYP
Mozambique New Metical	MZN	Taiwan Dollar	TWD
Myanmar (Burma) Kyat	MMK	Tajikistan Somoni	TJS
Namibian Dollar	NAD	Tanzania Shilling	TZS
Nepalese Rupee	NPR	Thailand Baht	THB
Netherlands Antilles Guilder	ANG	Tonga Pa'anga	TOP
Netherlands Guilder*	NLG	Trinidad and Tobago Dollar	TTD
New Zealand Dollar	NZD	Tunisian Dinar	TND
Nicaragua Cordoba Oro	NIO	Turkish Lira	TRY
Nigerian Naira	NGN	Turkmenistan Manat	TMT
North Korean Won	KPW	UAE Dirham	AED
Norwegian Krone	NOK	Uganda Shilling	UGX
Oman Rial	OMR	Ukraine Hryvnia	UAH
Pakistan Rupee	PKR	Uruguay Peso	UYU
Panama Balboa	PAB	US Dollar	USD
Papua New Guinea Kina	PGK	Uzbekistan Sum	UZS
Paraguay Guarani	PYG	Vanuatu Vatu	VUV
Peruvian New Sol	PEN	Venezuelan Bolivar Fuerte	VEF
Philippines Peso	PHP	Vietnam Dong	VND
Polish Zloty	PLN	Western Samoa Tala	WST
Portuguese Escudo*	PTE	Yemeni Rial	YER
Qatari Rial	QAR	Zaire Zaire	ZRN
Romanian New Leu	RON	Zambian Kwacha	ZMK
Russian Rouble	RUB	Zimbabwe Dollar	ZWL
Rwanda Franc	RWF		

* The local currency and currency code are euro and EUR, respectively.

Examples: SASEXFSD Interface Engine

Example 52.1: Retrieving Standardized Economic Items for Multiple Countries

This example shows how to use the ExtractEconData factlet to retrieve the standardized government debt values, reflecting debt in billions of dollars at year end for the United States and Greece and using the country identifiers CC_US and CC_GR and the standardized FactSet economic code FDS_ECON_GDP_USD_Y.

```
title 'Retrieve Standardized Economic Items for Multiple Countries (US,GR)';
libname _all_ clear;

libname xfsd sasexfsd "%sysget(FACTSET)"
   factlet=ExtractEconData
   ids='CC_US,CC_GR'
   items='FDS_ECON_GDP_USD_Y'
   dates='-6:-1:y'
   period=QTR
   format=sml
   outXml=fsdex06
   automap=replace
   mapref=MyMap
   xmlmap="%sysget(FACTSET)fsdex06.map"
   orientation=eti
   user='XXXXXXXXXXXXXXXX'
   pass='XXXXXXXXXXXXXXXX';

data econStnd; set xfsd.fsdex06; run;
proc print data=econStnd; run;
```

Output 52.1.1 Standardized Economic Items for Multiple Countries

Retrieve Standardized Economic Items for Multiple Countries (US,GR)

Obs	FQL_Entity	date	fds_econ_gdp_usd_y
1	CC_US	12-31-2006	13377.20
2	CC_US	12-31-2007	14028.70
3	CC_US	12-31-2008	14291.50
4	CC_US	12-31-2009	13973.70
5	CC_US	12-31-2010	14498.90
6	CC_US	12-31-2011	15075.70
7	CC_GR	12-31-2006	261.85
8	CC_GR	12-31-2007	305.54
9	CC_GR	12-31-2008	341.24
10	CC_GR	12-31-2009	321.34
11	CC_GR	12-31-2010	294.03
12	CC_GR	12-31-2011	290.04

Example 52.2: Retrieving Economic Items by Using the FQL Syntax for Function *Z* Score

This example shows how to use the ExtractEconData factlet to retrieve the *Z* score for the GRLM0347861 monthly time series for Greece over the last three years by using the FQL economic download syntax. It is not necessary to use the IDS= option, because all necessary information is contained in the ITEMS= option. The PRINT procedure uses the LABEL option to allow the time series name to be in the column heading in the output. Without the LABEL option, the column heading would be 'ECON_EXPR_DATA'.

```
option validvarname=any;

title 'Retrieve the Z Score Using ECON_EXPR_DATA for the US CFTNCLOI%ALLNQ100CMEF Series';
libname _all_ clear;

libname xfsd sasexfsd "%sysget(FACTSET)"
   factlet=ExtractEconData
   items="ECON_EXPR_DATA('ZSCORE(FDS_ECON[GRLM0347861],
   -5AY,step,average)',0,0/0/-3,m)"
   format=sml
   outXml=fsdecon8z
   automap=replace
   mapref=MyMap
   xmlmap="%sysget(FACTSET)fsdecon8z.map"
   orientation=eti
   user='XXXXXXXXXXXXXXX'
   pass='XXXXXXXXXXXXXXX';

data FQLeconFunc; set xfsd.fsdecon8z; run;
proc print data=FQLeconFunc(firstobs=1 obs=34) label; run;
```

Output 52.2.1 *Z* Score Using ECON_EXPR_DATA for the GRLM0347861 Monthly Series for Greece

Retrieve the Z Score Using ECON_EXPR_DATA for the US CFTNCLOI%ALLNQ100CMEF Series

Obs	FQL_Entity	date	ZSCORE(FDS_ECON[GRLM0347861],-5AY,step,average)
1	econ_expr_data	04-30-2010	-1.34718
2	econ_expr_data	05-28-2010	-1.33648
3	econ_expr_data	06-30-2010	-1.42249
4	econ_expr_data	07-30-2010	-1.32352
5	econ_expr_data	08-31-2010	-1.28179
6	econ_expr_data	09-30-2010	-1.21514
7	econ_expr_data	10-29-2010	-1.01707
8	econ_expr_data	11-30-2010	-0.95096
9	econ_expr_data	12-31-2010	-0.76515
10	econ_expr_data	01-31-2011	-0.70250
11	econ_expr_data	02-28-2011	-0.54235
12	econ_expr_data	03-31-2011	-0.46460
13	econ_expr_data	04-29-2011	-0.54778
14	econ_expr_data	05-31-2011	-0.38856
15	econ_expr_data	06-30-2011	-0.51014
16	econ_expr_data	07-29-2011	-0.40330
17	econ_expr_data	08-31-2011	-0.03023
18	econ_expr_data	09-30-2011	-0.20943
19	econ_expr_data	10-31-2011	-0.06770
20	econ_expr_data	11-30-2011	0.48282
21	econ_expr_data	12-30-2011	0.49980
22	econ_expr_data	01-31-2012	0.68252
23	econ_expr_data	02-29-2012	0.82152
24	econ_expr_data	03-30-2012	0.84469
25	econ_expr_data	04-30-2012	0.81819
26	econ_expr_data	05-31-2012	0.88345
27	econ_expr_data	06-29-2012	1.09286
28	econ_expr_data	07-31-2012	1.02826
29	econ_expr_data	08-31-2012	1.19474
30	econ_expr_data	09-28-2012	1.45434
31	econ_expr_data	10-31-2012	1.43013
32	econ_expr_data	11-30-2012	1.69463
33	econ_expr_data	12-31-2012	1.59840
34	econ_expr_data	01-31-2013	.

Example 52.3: Using ECON_EXPR_DATA with the FQL Syntax for Function Returns

This example shows how to use the ExtractEconData factlet to retrieve the Returns for the GDPEURNS quarterly time series for Greece over the last 10 years by using the FQL economic download syntax. It is not necessary to use the IDS= option, because the FQL downloading syntax allows '\@@GR' to be appended to the series name in the ITEMS= option. The PRINT procedure uses the LABEL option to allow the time series name to be in the column heading in the output. Without the LABEL option, the column heading would be 'ECON_EXPR_DATA'.

```
option validvarname=any;

title 'Retrieve Returns Using ECON_EXPR_DATA for the GDPEURNS@GR Series';
libname _all_ clear;

libname xfsd sasexfsd "%sysget(FACTSET)"
   factlet=ExtractEconData
   items="ECON_EXPR_DATA('RETURNS(EURO_STAT[GDPEURNS@GR],-1AQ,4)',0,0/0/-10,q)"
   format=sml
   outXml=fsdecon20
   automap=replace
   mapref=MyMap
   xmlmap="%sysget(FACTSET)fsdecon20.map"
   orientation=eti
   user='XXXXXXXXXXXXXXX'
   pass='XXXXXXXXXXXXXXX';

data EUROFunc; set xfsd.fsdecon20; run;
proc print data=EUROFunc label; run;
```

Output 52.3.1 Quarterly Returns of Series GDPEURNS@GR Using ECON_EXPR_DATA for Database Source Name EURO_STAT

Retrieve Returns Using ECON_EXPR_DATA for the GDPEURNS@GR Series

Obs	FQL_Entity	date	RETURNS(EURO_STAT[GDPEURNS@GR],-1AQ,4)
1	econ_expr_data	06-30-2003	39.9255
2	econ_expr_data	09-30-2003	24.1206
3	econ_expr_data	12-31-2003	-2.9307
4	econ_expr_data	03-31-2004	-21.9838
5	econ_expr_data	06-30-2004	44.4186
6	econ_expr_data	09-30-2004	21.6794
7	econ_expr_data	12-31-2004	-3.2521
8	econ_expr_data	03-31-2005	-28.4104
9	econ_expr_data	06-30-2005	32.7385
10	econ_expr_data	09-30-2005	31.6561
11	econ_expr_data	12-30-2005	-6.1976
12	econ_expr_data	03-31-2006	-17.0596
13	econ_expr_data	06-30-2006	40.5935
14	econ_expr_data	09-29-2006	19.2535
15	econ_expr_data	12-29-2006	-0.3652
16	econ_expr_data	03-30-2007	-21.9820
17	econ_expr_data	06-29-2007	39.1451
18	econ_expr_data	09-28-2007	23.8728
19	econ_expr_data	12-31-2007	-7.5605
20	econ_expr_data	03-31-2008	-20.5704
21	econ_expr_data	06-30-2008	37.8653
22	econ_expr_data	09-30-2008	20.0536
23	econ_expr_data	12-31-2008	-19.1250
24	econ_expr_data	03-31-2009	-38.6740
25	econ_expr_data	06-30-2009	65.9728
26	econ_expr_data	09-30-2009	11.6375
27	econ_expr_data	12-31-2009	6.0947
28	econ_expr_data	03-31-2010	-39.8820
29	econ_expr_data	06-30-2010	35.5932
30	econ_expr_data	09-30-2010	10.4348
31	econ_expr_data	12-31-2010	-22.8435
32	econ_expr_data	03-31-2011	-33.6109
33	econ_expr_data	06-30-2011	43.5848
34	econ_expr_data	09-30-2011	15.9831
35	econ_expr_data	12-30-2011	-30.2577
36	econ_expr_data	03-30-2012	-33.0359
37	econ_expr_data	06-29-2012	41.0717
38	econ_expr_data	09-28-2012	.
39	econ_expr_data	12-31-2012	.
40	econ_expr_data	03-28-2013	.

Example 52.4: Using SPEC_ID_DATA with the FQL Economic Download Syntax

This example shows how to use the ExtractEconData factlet and SPEC_ID_DATA to retrieve the WTI-FDS:FG_PRICE daily time series for the last 30 days by using the FQL economic download syntax. For more information about real-time data and specifying identifiers for commodity spot prices, see the FactSet Online Assistant, page ID 16992.

```
option validvarname=any;

title 'Retrieve WTI-FDS:FG_PRICE Using SPEC_ID_DATA for the Last 30 Days';
libname _all_ clear;

libname xfsd sasexfsd "%sysget(FACTSET)"
   factlet=ExtractEconData
   item="SPEC_ID_DATA('WTI-FDS:FG_PRICE',0,-30,d)"
   format=sml
   outXml=fsdecon14
   automap=replace
   mapref=MyMap
   xmlmap="%sysget(FACTSET)fsdecon14.map"
   orientation=eti
   user='XXXXXXXXXXXXXXX'
   pass='XXXXXXXXXXXXXXX';

data WTIPrice; set xfsd.fsdecon14; run;
proc print data=WTIPrice; run;
```

Output 52.4.1 Daily Series Using SPEC_ID_DATA for West Texas Intermediate Crude Oil Spot Price

Retrieve WTI-FDS:FG_PRICE Using SPEC_ID_DATA for the Last 30 Days

Obs	FQL_Entity	date	spec_id_data WTI-FDS:FG_PRICE
1	spec_id_data	02-20-2013	94.4600
2	spec_id_data	02-21-2013	92.8400
3	spec_id_data	02-22-2013	93.1300
4	spec_id_data	02-25-2013	93.1100
5	spec_id_data	02-26-2013	92.6300
6	spec_id_data	02-27-2013	92.7600
7	spec_id_data	02-28-2013	92.0500
8	spec_id_data	03-01-2013	90.6800
9	spec_id_data	03-04-2013	90.1200
10	spec_id_data	03-05-2013	90.8200
11	spec_id_data	03-06-2013	90.4300
12	spec_id_data	03-07-2013	91.5600
13	spec_id_data	03-08-2013	91.9500
14	spec_id_data	03-11-2013	92.0600
15	spec_id_data	03-12-2013	92.5400
16	spec_id_data	03-13-2013	92.5200
17	spec_id_data	03-14-2013	93.0300
18	spec_id_data	03-15-2013	93.4500
19	spec_id_data	03-18-2013	93.7400
20	spec_id_data	03-19-2013	92.1600
21	spec_id_data	03-20-2013	92.9600
22	spec_id_data	03-21-2013	92.4500
23	spec_id_data	03-22-2013	93.7100
24	spec_id_data	03-25-2013	94.8100
25	spec_id_data	03-26-2013	96.3400
26	spec_id_data	03-27-2013	96.5800
27	spec_id_data	03-28-2013	97.2300
28	spec_id_data	03-29-2013	97.2300
29	spec_id_data	04-01-2013	97.0700
30	spec_id_data	04-02-2013	97.1900
31	spec_id_data	04-03-2013	94.4500

Example 52.5: Using Multiple Database Sources with the FQL Syntax

This example shows how to use the ExtractEconData factlet and two database sources, FDS_ECON_DATA and EURO_STAT_DATA, to retrieve two time series for the last 11 months by using the FQL economic download syntax. The database source name is prepended to the time series name to retain the integrity of the name of the database source (dbsource). Only the same frequency and same range of observations for multiple series can be requested concurrently.

```
option validvarname=any;

title 'Retrieve Monthly Data from Two Database Sources: FDS_ECON_DATA and EURO_STAT';
libname _all_ clear;

libname xfsd sasexfsd "%sysget(FACTSET)"
   factlet=ExtractEconData
   items="FDS_ECON_DATA('FRBIPSB50001',-11,-1,M,STEP,AVERAGE,1),
   EURO_STAT_DATA('CONSCONFBAL@EUZ',-11,-1,M)"
   format=sml
   outXml=fsdecon17
   automap=replace
   mapref=MyMap
   xmlmap="%sysget(FACTSET)fsdecon17.map"
   orientation=eti
   user='XXXXXXXXXXXXXXX'
   pass='XXXXXXXXXXXXXXX';

data TwoSources; set xfsd.fsdecon17; run;
proc print data=TwoSources; run;
```

Output 52.5.1 Two Monthly Series Using Two Database Sources: FDS_ECON_DATA and EURO_STAT_DATA

Retrieve Monthly Data from Two Database Sources: FDS_ECON_DATA and EURO_STAT

Obs	FQL_ENTITY	date	fds_econ_data FRBIPSB50001	euro_stat_data CONSCONFBAL@EUZ
1	fds_econ_data	04-30-2012	96.8572	-19.7000
2	fds_econ_data	05-31-2012	97.1042	-19.1000
3	fds_econ_data	06-29-2012	97.1322	-19.6000
4	fds_econ_data	07-31-2012	97.5571	-21.3000
5	fds_econ_data	08-31-2012	96.7850	-24.4000
6	fds_econ_data	09-28-2012	96.9549	-25.7000
7	fds_econ_data	10-31-2012	96.8281	-25.5000
8	fds_econ_data	11-30-2012	98.0201	-26.7000
9	fds_econ_data	12-31-2012	98.1594	-26.3000
10	fds_econ_data	01-31-2013	98.3042	-23.9000
11	fds_econ_data	02-28-2013	99.0446	-23.6000

Example 52.6: Retrieving Price Data for One Company

This simple example shows how to use the ExtractFormulaHistory factlet to retrieve price data for one company (in this case IBM).

```
title 'Retrieve Price Data for IBM';
libname _all_ clear;

libname xfsd sasexfsd "%sysget(FACTSET)"
   factlet=ExtractFormulaHistory
   ids='ibm'
   items='p_price'
   dates='20110130:20111231:m'
   format=sml
   outXml=fsdex01
   automap=replace
   mapref=MyMap
   xmlmap="%sysget(FACTSET)fsdex01.map"
   orientation=eti
   user='XXXXXXXXXXXXXXX'
   pass='XXXXXXXXXXXXXXX';

data recentprice; set xfsd.fsdex01; run;
proc print data=recentprice; run;
```

Output 52.6.1 Price Data for IBM

Retrieve Price Data for IBM

Obs	FQL_Entity	date	p_price
1	ibm	01-31-2011	162.000
2	ibm	02-28-2011	161.880
3	ibm	03-31-2011	163.070
4	ibm	04-30-2011	170.580
5	ibm	05-31-2011	168.930
6	ibm	06-30-2011	171.550
7	ibm	07-31-2011	181.850
8	ibm	08-31-2011	171.910
9	ibm	09-30-2011	174.870
10	ibm	10-31-2011	184.630
11	ibm	11-30-2011	188.000
12	ibm	12-31-2011	183.880

Example 52.7: Retrieving Price and Sales Data for Multiple Companies

This example shows how to use the ExtractFormulaHistory factlet to retrieve several data items for several companies. The data items are price and sales, and the companies are IBM and FactSet (FDS).

```
title 'Retrieve Price and Sales Data for IBM and FactSet(FDS)';
libname _all_ clear;

libname xfsd sasexfsd "%sysget(FACTSET)"
   factlet=ExtractFormulaHistory
   ids='ibm,fds'
   items='p_price,ff_sales'
   dates='20110130:20110631:m'
   format=sml
   outXml=fsdex02
   automap=replace
   mapref=MyMap
   xmlmap="%sysget(FACTSET)fsdex02.map"
   orientation=eti
   user='XXXXXXXXXXXXXXX'
   pass='XXXXXXXXXXXXXXX';

data priceSale; set xfsd.fsdex02; run;
proc print data=priceSale; run;
```

Output 52.7.1 Multiple Data Items for IBM and FactSet

Retrieve Price and Sales Data for IBM and FactSet(FDS)

Obs	FQL_ENTITY	date	p_price	ff_sales
1	ibm	01-31-2011	162.000	99870.00
2	ibm	02-28-2011	161.880	99870.00
3	ibm	03-31-2011	163.070	99870.00
4	ibm	04-30-2011	170.580	99870.00
5	ibm	05-31-2011	168.930	99870.00
6	ibm	06-30-2011	171.550	99870.00
7	fds	01-31-2011	100.800	641.06
8	fds	02-28-2011	104.880	641.06
9	fds	03-31-2011	104.730	641.06
10	fds	04-30-2011	109.410	641.06
11	fds	05-31-2011	110.860	641.06
12	fds	06-30-2011	102.320	641.06

Example 52.8: Retrieving Book Value Data for One Company by Using Relative Dates

This example shows how to use the ExtractFormulaHistory factlet to retrieve book value data for one company (in this case Exxon Mobil, or XOM) by using relative dates. The book value represents the proportional common equity divided by outstanding shares at the end of the company's fiscal year. The relative date specifies the date as *n* periods ago based on the frequency (specified or implied); for example, DATES=0:-8:y returns data for the nine years prior to the most recently updated year.

```
title 'Retrieve Book Value Data for Exxon Mobil (XOM) for the Last 9 Years';
libname _all_ clear;

libname xfsd sasexfsd "%sysget(FACTSET)"
   factlet=ExtractFormulaHistory
   ids='xom'
   items='ff_bps'
   dates='0:-8:y'
   format=sml
   outXml=fsdex03
   automap=replace
   mapref=MyMap
   xmlmap="%sysget(FACTSET)fsdex03.map"
   orientation=eti
   user='XXXXXXXXXXXXXXX'
   pass='XXXXXXXXXXXXXXX';

data bookRelative; set xfsd.fsdex03; run;
proc print data=bookRelative; run;
```

Output 52.8.1 Book Value Data for Exxon Mobil for the Last 9 Years

Retrieve Book Value Data for Exxon Mobil (XOM) for the Last 9 Years

Obs	FQL_Entity	date	ff_bps
1	xom	12-31-2004	15.8969
2	xom	12-31-2005	18.1291
3	xom	12-31-2006	19.8715
4	xom	12-31-2007	22.6239
5	xom	12-31-2008	22.7020
6	xom	12-31-2009	23.3910
7	xom	12-31-2010	29.4917
8	xom	12-31-2011	32.6143
9	xom	12-31-2012	36.8421

Example 52.9: Retrieving Multiple Screen Items for Multiple Companies

This example shows how to use the ExtractDataSnapshot factlet to extract multiple screen items (price and sales) as of a single date for multiple companies (in this case IBM and Microsoft) for the quarterly estimate period (PERIOD=QTR).

```
title 'Retrieve Multiple Screen Items for Multiple Companies';
libname _all_ clear;

libname xfsd sasexfsd "%sysget(FACTSET)"
   factlet=ExtractDataSnapshot
   ids='ibm,msft'
   items='p_price,ff_sales'
   dates='20110401'
   period=QTR
   format=sml
   outXml=fsdex05
   automap=replace
   mapref=MyMap
   xmlmap="%sysget(FACTSET)fsdex05.map"
   orientation=eti
   user='XXXXXXXXXXXXXXX'
   pass='XXXXXXXXXXXXXXX';

data snapshot; set xfsd.fsdex05; run;
proc print data=snapshot; run;
```

Output 52.9.1 Multiple Screen Items for Multiple Companies

Retrieve Multiple Screen Items for Multiple Companies

Obs	FQL_ENTITY	date	p_price	ff_sales
1	ibm	04-01-2011	164.270	24607
2	msft	04-01-2011	25.480	16428

Example 52.10: Retrieving Data by Using the ISON= and ISONPARAMS= Options

This example shows how to use the ExtractDataSnapshot factlet to retrieve price-to-earnings (PE) data for the quarterly estimate period by using the ISON= and ISONPARAMS= options. For brevity, only a subset of the output (the first 10 CUSIPs) is displayed.

```
title 'Retrieve Price-to-Earnings Data by Using ISON/ISONPARAMS';
libname _all_ clear;

libname xfsd sasexfsd "%sysget(FACTSET)"
   factlet=ExtractDataSnapshot
   ison='sp500'
   isonparams='0,1'
   items='ff_pe'
   dates='20110401'
   period=QTR
   format=sml
   outXml=fsdex10
   automap=replace
   mapref=MyMap
   xmlmap="%sysget(FACTSET)fsdex10.map"
   orientation=eti
   user='XXXXXXXXXXXXXXX'
   pass='XXXXXXXXXXXXXXX';

data snapIson; set xfsd.fsdex10; run;
proc print data=snapIson(firstobs=1 obs=10); run;
```

Output 52.10.1 Retrieving Price-to-Earnings Data by Using the ISON= and ISONPARAMS= Options

Retrieve Price-to-Earnings Data by Using ISON/ISONPARAMS

Obs	FQL_Entity	date	ff_pe
1	17290810	04-01-2011	18.1419
2	41308610	04-01-2011	13.2635
3	80589M10	04-01-2011	13.3007
4	50242410	04-01-2011	9.5995
5	91301710	04-01-2011	17.2053
6	97665710	04-01-2011	14.5238
7	00130H10	04-01-2011	.
8	31190010	04-01-2011	33.0765
9	20911510	04-01-2011	13.5979
10	53983010	04-01-2011	10.0212

Example 52.11: Retrieving Benchmark Data by Using the CUTOFF= option

This example shows how to use the ExtractBenchmarkDetail factlet to retrieve the holdings for the Standard & Poor's (S&P) 500 (ID='sp50') and display the P_PRICE data that correspond to each holding. For brevity, only a subset of the output (the first 10 holdings) is displayed.

```
title 'Retrieve Benchmark Data for Top Ten Holdings, CUTOFF=10';
libname _all_ clear;

libname fsd sasexfsd "%sysget(FACTSET)"
   factlet=ExtractBenchmarkDetail
   ids='sp50'
   items='p_price,proper_name'
   dates='20130320'
   cutoff=10
   format=sml
   outXml=fsdex12
   automap=replace
   mapref=MyMap
   xmlmap="%sysget(FACTSET)fsdex12.map"
   orientation=eti
   user='XXXXXXXXXXXXXXX'
   pass='XXXXXXXXXXXXXXX';

data bench;
   set fsd.fsdex12;
run;
proc print
   data=bench;
run;
```

The CUTOFF= option limits the output to the number of holdings that are specified. This example uses CUTOFF=10 to print the top 10 holdings. If you omit the CUTOFF= option, all 500 holdings are reported.

Output 52.11.1 Retrieving Benchmark Data for Top 10 Holdings in the S&P 500 Index

Retrieve Benchmark Data for Top Ten Holdings, CUTOFF=10

Obs	FQL_ENTITY	date	SECURITY_ID	Weight	p_price	proper_name
1	SP50	03-20-2013	03783310	3.05742	452.080	Apple Inc.
2	SP50	03-20-2013	30231G10	2.85989	88.630	Exxon Mobil Corp.
3	SP50	03-20-2013	36960410	1.75686	23.460	General Electric Co.
4	SP50	03-20-2013	16676410	1.68383	120.350	Chevron Corp.
5	SP50	03-20-2013	45920010	1.62263	215.060	International Business Machines Corp.
6	SP50	03-20-2013	47816010	1.59946	79.450	Johnson & Johnson
7	SP50	03-20-2013	38259P50	1.56568	814.708	Google Inc. Cl A
8	SP50	03-20-2013	59491810	1.53648	28.315	Microsoft Corp.
9	SP50	03-20-2013	74271810	1.52624	77.580	Procter & Gamble Co.
10	SP50	03-20-2013	71708110	1.46471	28.290	Pfizer Inc.

Example 52.12: Retrieving Benchmark Data by Using the MATCHDATE= Option

This example shows how to use the ExtractBenchmarkDetail factlet to retrieve data from the Prices database for the Russell 1000 constituents (R.1000).

```
title 'Retrieve Benchmark Data for R.1000 with MATCHDATE=ON';
libname _all_ clear;

libname fsd sasexfsd "%sysget(FACTSET)"
   factlet=ExtractBenchmarkDetail
   ids='r.1000'
   items='p_price'
   dates='20120118:20120113:b'
   matchDate=on
   format=sml
   outXml=fsdex13
   automap=replace
   mapref=MyMap
   xmlmap="%sysget(FACTSET)fsdex13.map"
   orientation=eti
   user='XXXXXXXXXXXXXXX'
   pass='XXXXXXXXXXXXXXX';

data benchmatch;
   set fsd.fsdex13;
run;
proc print
   data=benchmatch(firstobs=1 obs=50);
run;
```

If the frequency argument were not set to B (indicating business days) and the MATCHDATE= option were not turned on, the output would contain repetitive dates because of feel-back, resulting in unnecessarily lengthy output.

Output 52.12.1 Retrieving Benchmark Data for the Russell 1000 Index by Using the MATCHDATE=ON Option

Retrieve Benchmark Data for R.1000 with MATCHDATE=ON

Obs	FQL_ENTITY	date	SECURITY_ID	Weight	p_price
1	R.1000	01-17-2012	30231G10	3.17717	85.690
2	R.1000	01-17-2012	03783310	2.95608	424.700
3	R.1000	01-17-2012	45920010	1.64095	180.000
4	R.1000	01-17-2012	16676410	1.61489	106.720
5	R.1000	01-17-2012	59491810	1.57846	28.255
6	R.1000	01-17-2012	36960410	1.49591	18.740
7	R.1000	01-17-2012	74271810	1.39208	66.260
8	R.1000	01-17-2012	00206R10	1.34835	30.250
9	R.1000	01-17-2012	47816010	1.34355	65.120
10	R.1000	01-17-2012	71708110	1.30536	21.935
11	R.1000	01-17-2012	38259P50	1.19073	628.580
12	R.1000	01-17-2012	94974610	1.10366	29.825
13	R.1000	01-17-2012	46625H10	1.04412	34.910
14	R.1000	01-17-2012	08467070	1.02732	77.970
15	R.1000	01-17-2012	71817210	1.01597	75.900
16	R.1000	01-17-2012	19121610	1.00150	33.675
17	R.1000	01-17-2012	45814010	0.99967	25.040
18	R.1000	01-17-2012	58933Y10	0.90210	38.820
19	R.1000	01-17-2012	92343V10	0.83088	39.020
20	R.1000	01-17-2012	68389X10	0.79993	27.660
21	R.1000	01-17-2012	17275R10	0.79941	19.305
22	R.1000	01-17-2012	93114210	0.79937	59.850
23	R.1000	01-17-2012	58013510	0.78525	100.550
24	R.1000	01-17-2012	71344810	0.76915	64.650
25	R.1000	01-17-2012	74752510	0.71803	57.140
26	R.1000	01-17-2012	20825C10	0.70754	70.800
27	R.1000	01-17-2012	80685710	0.69083	67.640
28	R.1000	01-17-2012	00282410	0.65197	55.710
29	R.1000	01-17-2012	17296742	0.61791	28.215
30	R.1000	01-17-2012	67459910	0.60755	99.300
31	R.1000	01-17-2012	25468710	0.54744	38.480
32	R.1000	01-17-2012	91301710	0.53072	77.040
33	R.1000	01-17-2012	20030N10	0.52974	25.535
34	R.1000	01-17-2012	43707610	0.52612	43.740
35	R.1000	01-17-2012	14912310	0.50160	103.370
36	R.1000	01-17-2012	02313510	0.49686	181.660
37	R.1000	01-17-2012	06050510	0.49422	6.480
38	R.1000	01-17-2012	60920710	0.47423	38.130
39	R.1000	01-17-2012	02209S10	0.45520	28.900
40	R.1000	01-17-2012	88579Y10	0.45049	84.230
41	R.1000	01-17-2012	03116210	0.45029	68.070
42	R.1000	01-17-2012	91324P10	0.43685	53.570
43	R.1000	01-17-2012	12665010	0.43416	42.540
44	R.1000	01-17-2012	11012210	0.43298	33.720
45	R.1000	01-17-2012	09702310	0.41809	75.240
46	R.1000	01-17-2012	90297330	0.41721	28.770
47	R.1000	01-17-2012	91131210	0.41034	74.200

Output 52.12.1 *continued*

Retrieve Benchmark Data for R.1000 with MATCHDATE=ON

Obs	FQL_ENTITY	date	SECURITY_ID	Weight	p_price
48	R.1000	01-17-2012	90781810	0.40428	109.500
49	R.1000	01-17-2012	92826C83	0.40357	102.530
50	R.1000	01-17-2012	02581610	0.39715	50.220

Example 52.13: Retrieving Multiple Items for Multiple Companies from an OFDB File

This example shows how to use the ExtractOFDBItem factlet to retrieve the uploaded share and price data for IBM and Microsoft from an OFDB file named SASTESTING for an absolute date range, starting February 27, 2012, and ending February 28, 2012, with a monthly frequency.

```
title 'Retrieve Shares and Price Data for IBM and MSFT from an OFDB File';
libname _all_ clear;

libname xfsd sasexfsd "%sysget(FACTSET)"
   factlet=ExtractOFDBItem
   ofdb='SASTESTING.OFDB'
   ids='ibm,msft'
   items='shares,price'
   dates='20120227:20120228:d'
   period=QTR
   format=sml
   outXml=fsdex07
   automap=replace
   mapref=MyMap
   xmlmap="%sysget(FACTSET)fsdex07.map"
   orientation=eti
   user='XXXXXXXXXXXXXXX'
   pass='XXXXXXXXXXXXXXX';

data shareOFDB; set xfsd.fsdex07; run;
proc print data=shareOFDB; run;
```

Output 52.13.1 Multiple Items for Multiple Companies from an OFDB File

Retrieve Shares and Price Data for IBM and MSFT from an OFDB File

Obs	FQL_ENTITY	date	ofdb_shares	ofdb_price
1	ibm	02-27-2012	1178.60	1.000
2	ibm	02-28-2012	1178.60	197.980
3	msft	02-27-2012	8412.20	1.000
4	msft	02-28-2012	8412.20	31.870

Example 52.14: Retrieving a List of Securities from an OFDB file

This example shows how to use the ExtractOFDBUniverse factlet to retrieve a list of securities that belong to a single OFDB file named SASTESTING for February 27, 2012. For brevity, only a subset of the output (the first 15 securities) is displayed.

```
title 'Retrieve List of Securities Belonging to a Single OFDB File';
libname _all_ clear;

libname xfsd sasexfsd "%sysget(FACTSET)"
   factlet=ExtractOFDBUniverse
   ofdb='SASTESTING.OFDB'
   dates='20120227'
   format=sml
   outXml=fsdex08
   automap=replace
   mapref=MyMap
   xmlmap="%sysget(FACTSET)fsdex08.map"
   user='XXXXXXXXXXXXXXX'
   pass='XXXXXXXXXXXXXXX';

data ofdbUniv; set xfsd.fsdex08; run;
proc print data=ofdbUniv(firstobs=1 obs=15); run;
```

Output 52.14.1 List of Securities from a Single OFDB File

Retrieve List of Securities Belonging to a Single OFDB File

Obs	CUSIP
1	00105510
2	00120410
3	00130H10
4	00206R10
5	00282410
6	00289620
7	00507K10
8	00724F10
9	00790310
10	00817Y10
11	00846U10
12	00915810
13	00936310
14	00971T10
15	01381710

Example 52.15: Retrieving a List of CUSIPs from a Screen File

This example shows how to use the ExtractScreenUniverse factlet to retrieve a list of CUSIPs and names that belong to a single user-defined screen file. For brevity, only a subset of the output (the first 15 securities) is displayed.

```
title 'Retrieve List of Securities Belonging to a Single Screen File';
libname _all_ clear;

libname xfsd sasexfsd "%sysget(FACTSET)"
   factlet=ExtractScreenUniverse
   screen='factset:bankruptcy'
   name=y
   format=sml
   outXml=fsdex09
   automap=replace
   mapref=MyMap
   xmlmap="%sysget(FACTSET)fsdex09.map"
   user='XXXXXXXXXXXXXXX'
   pass='XXXXXXXXXXXXXXX';

data screenUniv; set xfsd.fsdex09; run;
proc print data=screenUniv(firstobs=1 obs=15); run;
```

Output 52.15.1 List of CUSIPs and Names from a Screen File

Retrieve List of Securities Belonging to a Single Screen File

Obs	Id	Name
1	00081T10	ACCO BRANDS CORP
2	00258J10	ABAKAN INC
3	00439710	ACCURAY INC
4	00439T20	ACCURIDE CORP
5	00520810	ADA-ES INC
6	00752K10	ADVANCED CELL TECHNOLOGY INC
7	00847J10	AGILYSYS INC
8	02051Q10	ALON HOLDINGS BLUE SQUARE IS
9	02152V10	ALTEROLA BIOTECH INC
10	02153D10	ALTERRA POWER CORP
11	03011110	AMERICAN SUPERCONDUCTOR CP
12	03236M10	AMYRIS INC
13	03242010	ANACOR PHARMACEUTICALS INC
14	04269E10	ARQULE INC
15	04544X30	ASSISTED LIVING CONCEPTS INC

References

FactSet Research Systems (2012). "Online Assistant." Accessed November 7, 2012. http://www.factset.com/.

International Organization of Standardization (2012). Currency Codes. Accessed November 7, 2012. http://www.iso.org/.

Part IV

Time Series Forecasting System

Chapter 53
Overview of the Time Series Forecasting System

Contents

Introduction 3607
Using the Time Series Forecasting System 3608
SAS Software Products Needed 3609

Introduction

The Time Series Forecasting system forecasts future values of time series variables by extrapolating trends and patterns in the past values of the series or by extrapolating the effect of other variables on the series. The system provides convenient point-and-click windows to control the time series analysis and forecasting tools of SAS/ETS software.

You can use the system in a fully automatic mode, or you can use the system's diagnostic features and time series modeling tools interactively to develop forecasting models customized to best predict your time series. The system provides both graphical and statistical features to help you choose the best forecasting method for each series.

The following is a brief summary of the features of the Time Series Forecasting system. You can use the system in the following ways:

- use a wide variety of forecasting methods, including several kinds of exponential smoothing models, Winters method, and ARIMA (Box-Jenkins) models. You can also produce forecasts by combining the forecasts from several models.
- use predictor variables in forecasting models. Forecasting models can include time trend curves, regressors, intervention effects (dummy variables), adjustments you specify, and dynamic regression (transfer function) models.
- view plots of the data, predicted versus actual values, prediction errors, and forecasts with confidence limits, as well as autocorrelations and results of white noise and stationarity tests. Any of these plots can be zoomed and can represent raw or transformed series.
- use hold-out samples to select the best forecasting method
- compare goodness-of-fit measures for any two forecasting models side by side or list all models sorted by a particular fit statistic

- view the predictions and errors for each model in a spreadsheet or compare the fit of any two models in a spreadsheet
- examine the fitted parameters of each forecasting model and their statistical significance
- control the automatic model selection process: the set of forecasting models considered, the goodness-of-fit measure used to select the best model, and the time period used to fit and evaluate models
- customize the system by adding forecasting models for the automatic model selection process and for point-and-click manual selection
- save your work in a project catalog
- print an audit trail of the forecasting process
- show source statements for PROC ARIMA code
- save and print system output including spreadsheets and graphs

Using the Time Series Forecasting System

Chapters starting from Chapter 54, "Getting Started with Time Series Forecasting," through Chapter 58, "Using Predictor Variables," contain a series of example sessions that show the major features of the system. Chapters from Chapter 59, "Command Reference," through Chapter 61, "Forecasting Process Details," serve as reference and provide more details about how the system operates. The reference chapters contain a complete list of system features.

To get started using the Time Series Forecasting system, it is a good idea to work through a few of the example sessions. Start with Chapter 54, "Getting Started with Time Series Forecasting," and use the system to reproduce the steps shown in the examples. Continue with the other chapters when you feel comfortable using the system.

The example sessions make use of time series data sets contained in the SASHELP library: `air`, `citimon`, `citiqtr`, `citiyr`, `citiwk`, `citiday`, `gnp`, `retail`, `usecon`, and `workers`. You can use these data sets to work through the example sessions or to experiment further with the system.

Once you are familiar with how the system operates, start working with your own data to build your own forecasting models. When you have questions, consult the reference chapters mentioned above for more information about particular features.

The Time Series Forecasting system forecasts *time series*, that is, variables that consist of ordered observations taken at regular intervals over time. Since the Time Series Forecasting system is a part of the SAS software system, time series values must be stored as variables in a SAS data set or data view, with the observations representing the time periods. The data can also be stored in an external spreadsheet or data base if you license SAS/ACCESS software.

The Time Series Forecasting System chapters refer to *series* and *variables*. Since time series are stored as variables in SAS data sets or data views, these terms are used interchangeably. However, the term *series* is preferred when attention is focused on the sequence of data values, and the term *variable* is preferred when attention is focused on the data set.

SAS Software Products Needed

The Time Series Forecasting system is part of SAS/ETS software. To use it, you must have a license for SAS/ETS. To use the graphical display features of the system, you must also license SAS/GRAPH software.

Chapter 54
Getting Started with Time Series Forecasting

Contents

The Time Series Forecasting Window **3612**
Outline of the Forecasting Process **3617**
Specify the Input Data Set 3617
Provide a Valid Time ID Variable 3617
Select and Fit a Forecasting Model for Each Series 3618
Produce the Forecasts 3618
Save Your Work 3618
Summary 3618
The Input Data Set **3619**
The Data Set Selection Window 3619
Time Series Data Sets, ID Variables, and Time Intervals 3622
Automatic Model Fitting Window **3623**
Produce Forecasts Window **3631**
The Forecast Data Set 3633
Forecasting Projects **3637**
Saving and Restoring Project Information 3639
Sharing Projects 3643
Develop Models Window **3644**
Introduction 3644
Fitting Models 3647
Model List and Statistics of Fit 3652
Model Viewer **3654**
Prediction Error Plots 3656
Autocorrelation Plots 3657
White Noise and Stationarity Plots 3658
Parameter Estimates Table 3660
Statistics of Fit Table 3661
Changing to a Different Model 3662
Forecasts and Confidence Limits Plots 3663
Data Table 3664
Closing the Model Viewer 3665

This chapter outlines the forecasting process and introduces the major windows of the system through three example sessions.

The first example, beginning with the section "The Time Series Forecasting Window," shows how to use the system for fully automated forecasting of a set of time series. This example also introduces the system's features for viewing data and forecasts through tables and interactive graphs. It also shows how to save and restore forecasting work in SAS catalogs.

The second example, beginning with the section "Develop Models Window," introduces the features for developing the best forecasting models for individual time series. The chapter concludes with an example showing how to create dating variables for your data in the form expected by the system.

After working through the examples in this chapter, you should be able to do the following:

- select a data set of time series to work with and specify its periodicity and time ID variable
- use the automatic forecasting model selection feature to create forecasting models for the variables in a data set
- produce and save forecasts of variables in a data set
- examine your data and forecasts as tables of values and through interactive graphs
- save and restore your forecasting models by using project files in a SAS catalog and edit project information
- use some of the model development features to fit and select forecasting models for individual time series variables

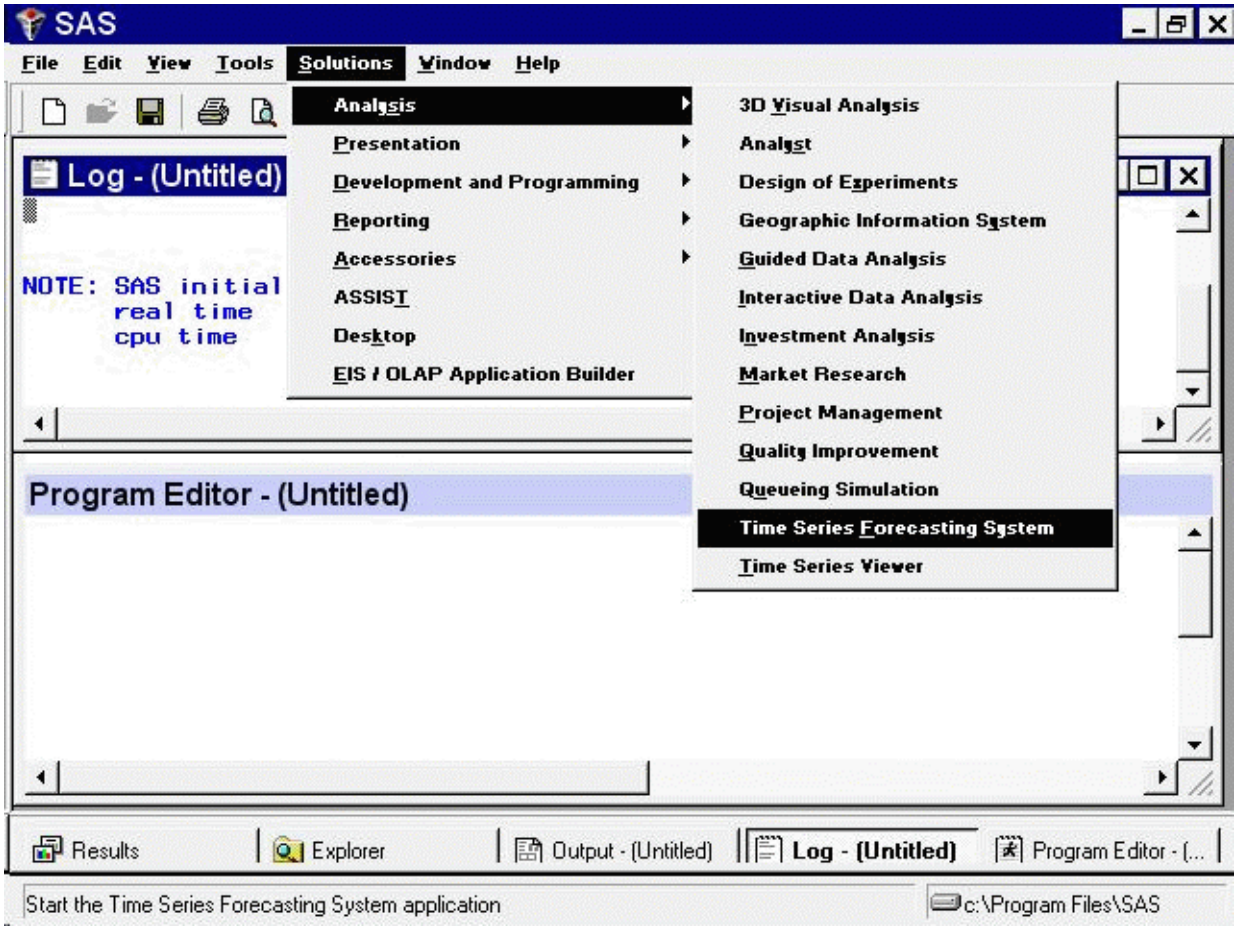
This chapter introduces these topics and helps you get started using the system. Later chapters present these topics in greater detail and document more advanced features and options.

The Time Series Forecasting Window

There are several ways to get to the Time Series Forecasting System. If you prefer to use commands, invoke the system by entering `forecast` on the command line. You can optionally specify additional information on the command line; see Chapter 59, "Command Reference," for details.

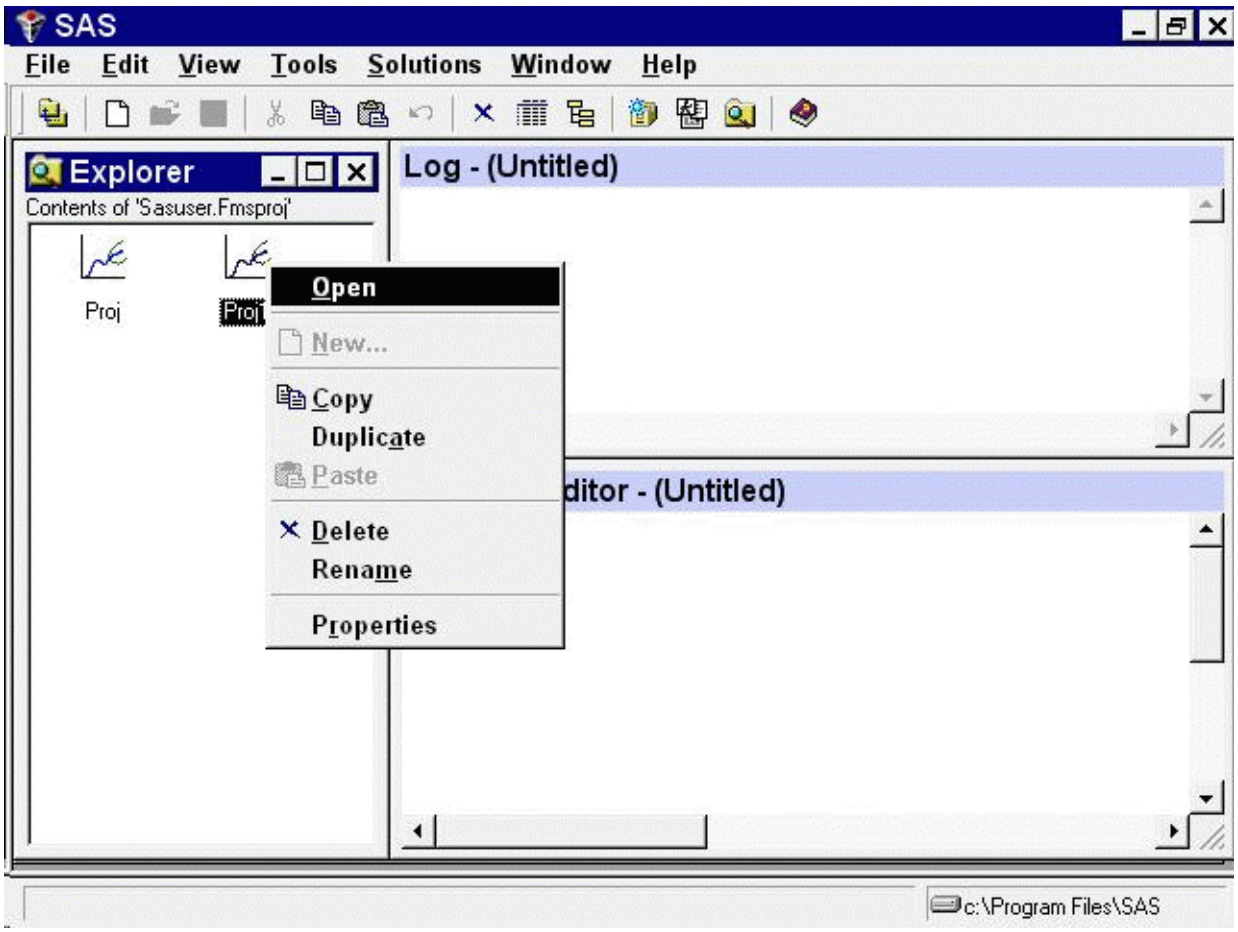
If you are using the SAS windowing environment with pull-down menus, select the Solutions menu from the menu bar, select the Analysis item, and then select `Time Series Forecasting System`, as shown in Figure 54.1.

Figure 54.1 Time Series Forecasting System Menu Selection



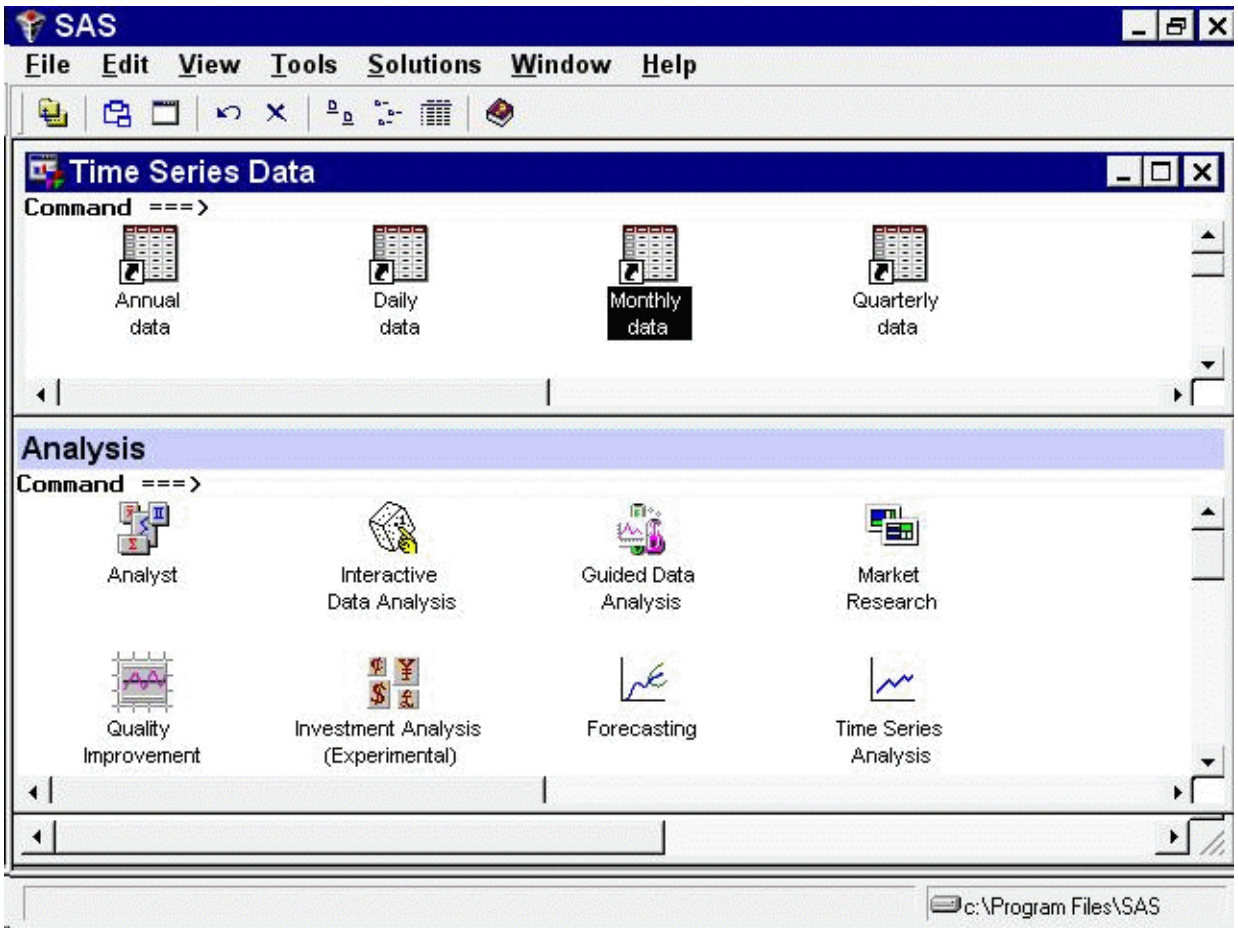
You can invoke the Forecasting System from the SAS Explorer window by opening an existing forecasting project. By default these projects are stored in the FMSPROJ catalog in the SASUSER library. Select SASUSER in the Explorer to display its contents. Then select FMSPROJ. This catalog is created the first time you use the Forecasting System. If you have saved projects, they appear in the Explorer with the forecasting graph icon, as shown in Figure 54.2. Double-click one of the projects, or select it with the right mouse button and then select Open from the pop-up menu, as shown in the figure. This opens the Forecasting System and opens the selected project.

Figure 54.2 Opening a Project from the Explorer



To invoke the Forecasting System in the SAS desktop environment, select the Solutions menu from the menu bar, select `Desktop`, and then open the Analysis folder. You can run the Time Series Forecasting System or the Time Series Viewer directly, or you can drag and drop. Figure 54.3 illustrates dragging a data set (known as a table in the Desktop environment) and dropping it on the Forecasting icon. In this example, the tables reside in a user-defined folder called *Time Series Data*.

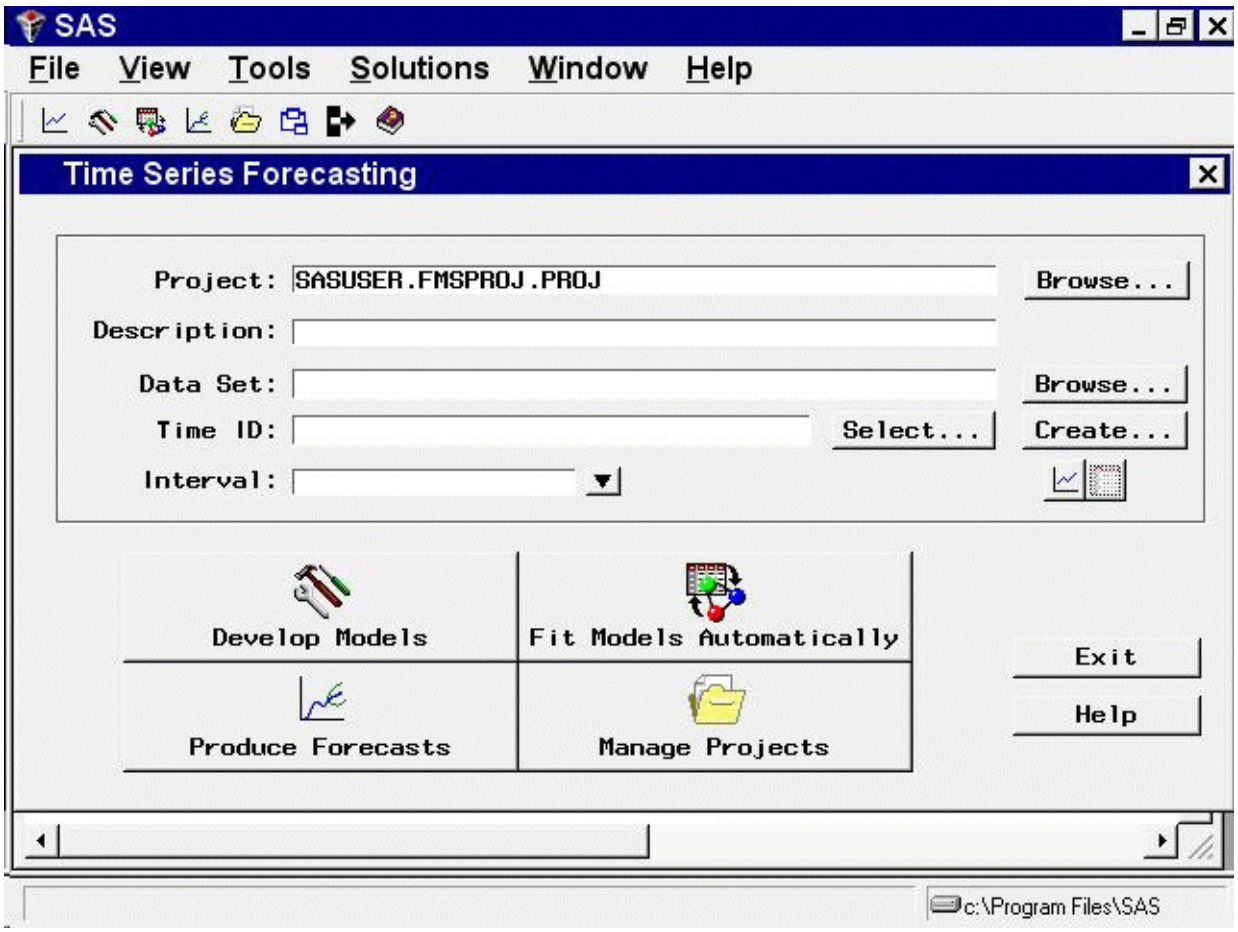
Figure 54.3 Drag and Drop on the SAS Desktop



If you are using SAS/ASSIST software, select the Planning button and then select `Forecasting` from the pop-up menu.

Any of these methods takes you to the Time Series Forecasting window, as shown in Figure 54.4.

Figure 54.4 Time Series Forecasting Window



At the top of the window is a data selection area for specifying a project file and the input data set containing historical data (the known past values) for the time series variables that you want to forecast. This area also contains buttons for opening viewers to explore your input data either graphically, one series at a time, or as a table, one data set at a time.

The Project and Description fields are used to specify a project file for saving and restoring forecasting models created by the system. Using project files is discussed later, and these fields are ignored for now.

The lower part of the window contains six buttons:

`Develop Models`
: opens the Develop Models window, which you use to develop and fit forecasting models interactively for individual time series.

`Fit Models Automatically`
: opens the Automatic Model Fitting window, which you use to search automatically for the best forecasting model for multiple series in the input data set.

`Produce Forecasts`
: opens the Produce Forecasts window, which you use to compute forecasts for all the variables in the input data set for which forecasting models have been fit.

`Manage Projects`
: opens the Manage Forecasting Project window, which lists the time series for which you have fit forecasting models. You can drill down on a series to see the models that have been fit. You can delete series or models from the project, re-evaluate or refit models, and explore models and forecasts graphically or in tabular form.

`Exit`
: exits the Forecasting System.

`Help`
: displays information about the Forecasting System.

Outline of the Forecasting Process

The examples shown in the following sections illustrate the basic process you use with the Forecasting System.

Specify the Input Data Set

Suppose you have a number of *time series*, variables recorded over time, for which you want to forecast future values. The past values of these time series are stored as variables in a SAS data set or data view. The observations of this data set correspond to regular time periods, such as days, weeks, or months. The first step in the forecasting process is to tell the system to use this data set by setting the Data Set field.

If your time series are not in a SAS data set, you must provide a way for the SAS System to access the data. You can use SAS features to read your data into a SAS data set; refer to *SAS Language Reference*. You can use a SAS/ACCESS product to establish a view of data in a database management system; refer to SAS/ACCESS documentation. You can use PROC SQL to create a SAS data view. You can use PROC DATASOURCE to read data from files supplied by supported data vendors; refer to Chapter 13, "The DATASOURCE Procedure," for more details.

Provide a Valid Time ID Variable

To use the Forecasting System, your data set must be dated: the data set must contain a *time ID variable* that gives the date of each observation. The time ID variable must represent the observation dates with *SAS date values* or with *SAS datetime values* (for hourly data or other frequencies less than a day), or you can use a simple time index.

When SAS date values are used, the ID variable contains dates within the time periods corresponding to the observations. For example, for monthly data, the values for the time ID variable can be the date of the first day of the month corresponding to each observation, or the time ID variable can contain the date of the last day in the month. (Any date within the period serves as the time ID for the observation.)

If your data set already contains a valid time ID variable with SAS date or datetime values, the next step is to specify this time ID variable in the Time ID field. If the time ID variable is named DATE, the system fills in the Time ID field automatically.

If your data set does not contain a time ID, you must add a valid time ID variable before beginning the forecasting process. The Forecasting System provides features that make this easy to do. See Chapter 55, "Creating Time ID Variables," for details.

Select and Fit a Forecasting Model for Each Series

If you are using the automated model selection feature, the system performs this step for you and chooses a forecasting model for each series automatically. All you need to do is select the Fit Models Automatically button and then select the variables to fit models for.

If you want more control over forecasting model selection, you can select the Develop Models button, select the series you want to forecast, and use the Develop Models window to specify a forecasting model. As part of this process, you can use the Time Series Viewer and Model Viewer graphical tools. Once you have selected a model for the first series, you can select a different series to work with and repeat the model development process until you have created forecasting models for all the series you want to forecast.

The system provides many features to help you choose the best forecasting model for each series. The features of the Develop Models window and graphical viewer tools are introduced in later sections.

Produce the Forecasts

Once a forecasting model has been fit for each series, select the Produce Forecasts button and use the Produce Forecasts window to compute forecast values and store them in a SAS data set.

Save Your Work

If you want only a single forecast, your task is now complete. But you might want to produce updated forecasts later, as more data becomes available. In this case, you want to save the forecasting models you have created, so that you do not need to repeat the model selection and fitting process.

To save your work, fill in the Project field with the name of a SAS catalog member in which the system will store the model information when you exit the system. Later, you will select the same catalog member name when you first enter the Forecasting System, and the model information will be reloaded.

Note that any number of people can work with the same project file. If you are working on a forecasting project as part of a team, you should take care to avoid conflicting updates to the project file by different team members.

Summary

This is the basic outline of how the Forecasting System works. The system offers many other features and options that you might need to use (for example, the time range of the data used to fit models and how far into the future to forecast). These options will become apparent as you work with the Forecasting System.

As an introductory example, the following sections use the Automatic Model Fitting and Produce Forecasts windows to perform automated forecasting of the series in an example data set.

The Input Data Set

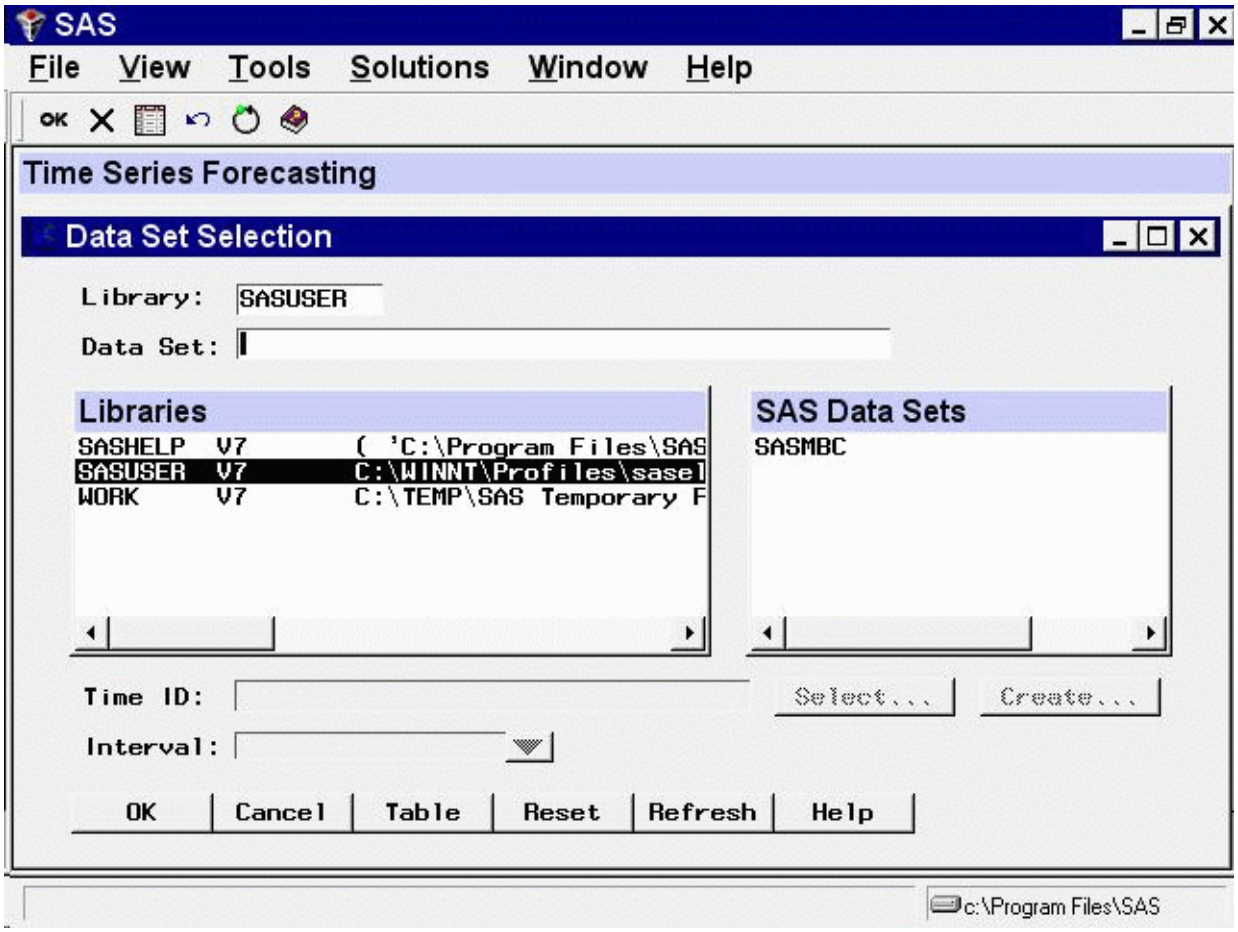
As the first step, you must specify the input data set.

The Data Set field in the Time Series Forecasting window gives the name of the input data set containing the time series to forecast. Initially, this field is blank. You can specify the input data set by typing the data set name in this field. Alternatively, you can select the Browse button at the right of the Data Set field to select the data set from a list, as shown in the following section.

The Data Set Selection Window

Select the Browse button to the right of the Data Set field. This opens the Data Set Selection window, as shown in Figure 54.5.

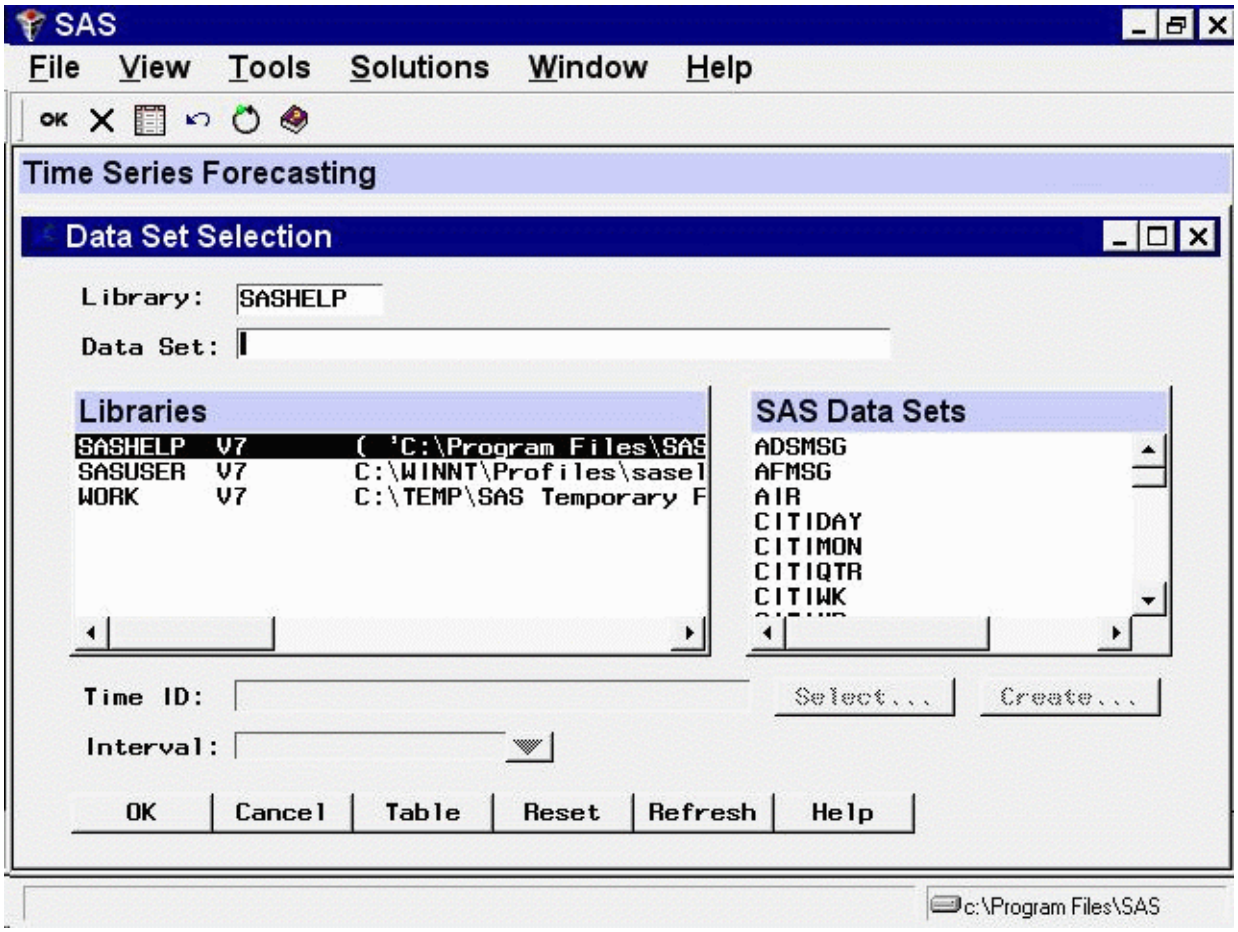
Figure 54.5 Data Set Selection Window



The `Libraries` list shows the SAS librefs that are currently allocated in your SAS session. Initially, the SASUSER library is selected, and the `SAS Data Sets` list shows the data sets available in your SASUSER library.

In the `Libraries` list, select the row that starts with SASHELP. The Data Set Selection window now lists the data sets in the SASHELP library, as shown in Figure 54.6.

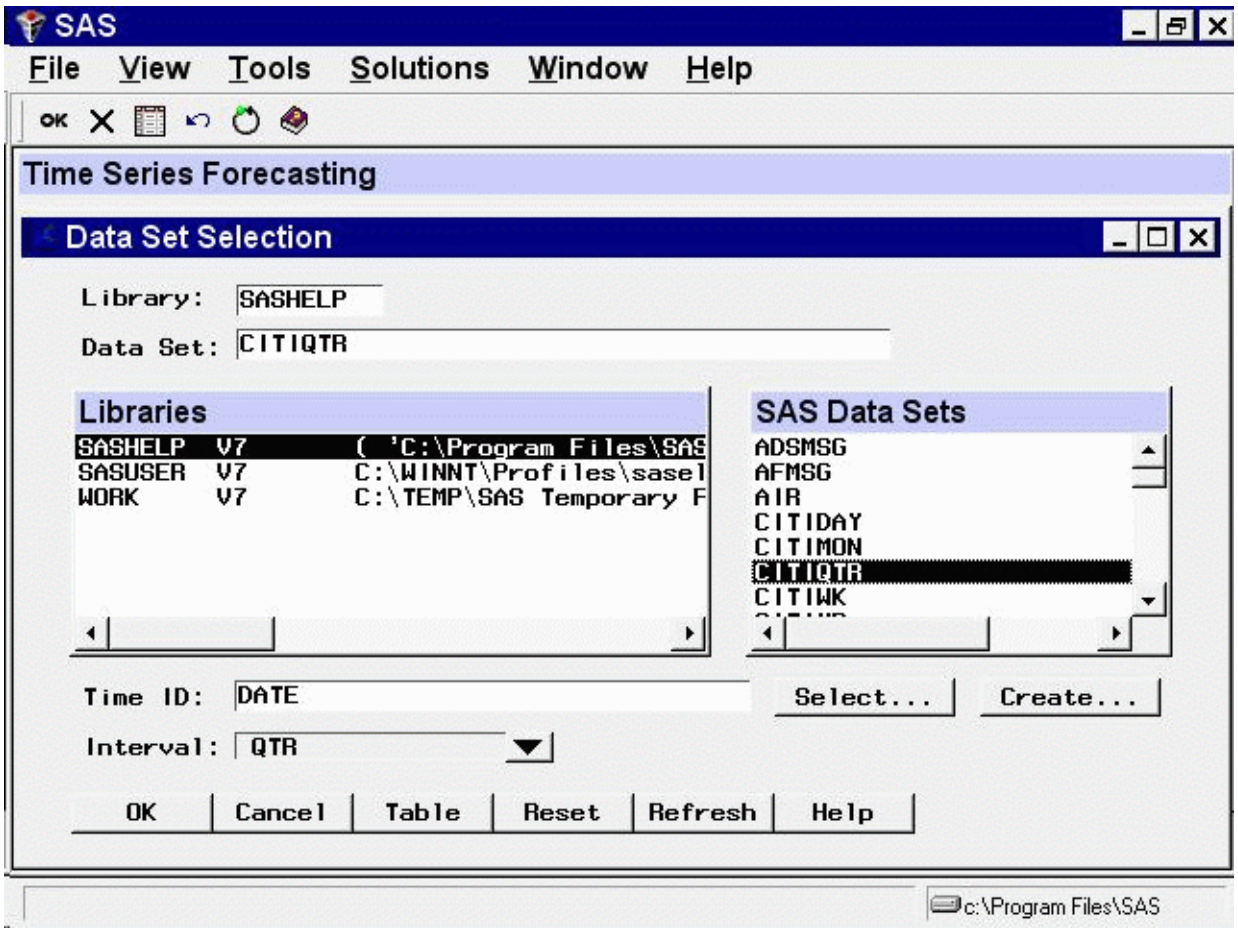
Figure 54.6 SASHELP Library



Use the vertical scroll bar on the `SAS Data Sets` list to scroll down the list until the data set CITIQTR appears. Then select the CITIQTR row. This selects the data set SASHELP.CITIQTR as the input data set.

Figure 54.7 shows the Data Set Selection window after selection of CITIQTR from the SAS Data Sets list.

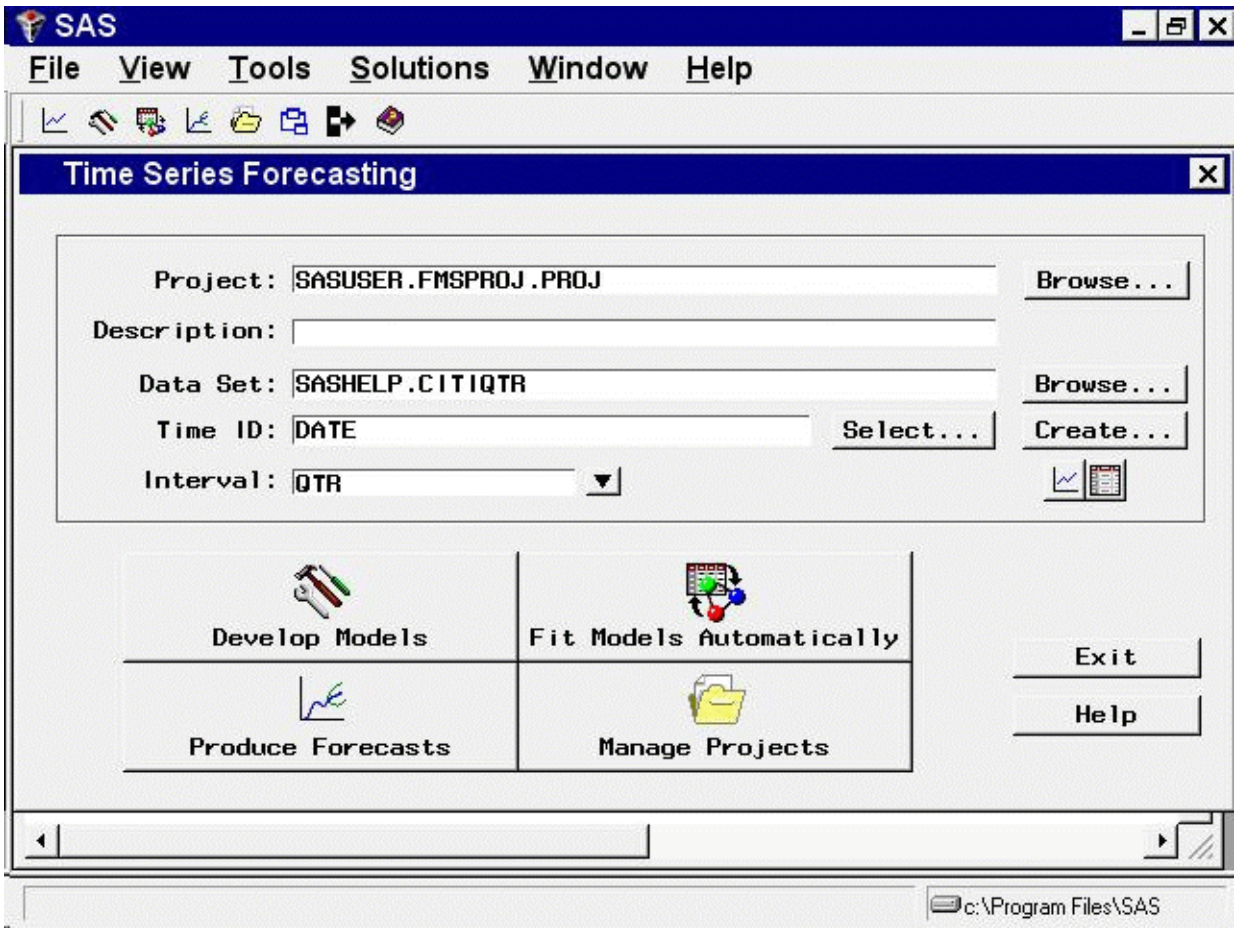
Figure 54.7 CITIQTR Data Set Selected



Note that the Time ID field is now set to DATE and the Interval field is set to QTR. These fields are explained in the following section.

Now select the OK button to complete selection of the CITIQTR data set. This closes the Data Set Selection window and returns to the Time Series Forecasting window, as shown in Figure 54.8.

Figure 54.8 Time Series Forecasting Window



Time Series Data Sets, ID Variables, and Time Intervals

Before you continue with the example, it is worthwhile to consider how the system determined the values for the Time ID and Interval fields in the Data Set Selection window.

The Forecasting System requires that the input data set contain time series observations, with one observation for each time period. The observations must be sorted in increasing time order, and there must be no gaps in the sequence of observations. The time period of each observation must be identified by an ID variable, which is shown in the Time ID field.

If the data set contains a variable named DATE, TIME, or DATETIME, the system assumes that this variable is the SAS date or datetime valued ID variable, and the Time ID field is filled in automatically. The time ID variable for the SASHELP.CITIQTR data set is named DATE, and therefore the system set the Time ID field to DATE.

If the time ID variable for a data set is not named DATE, TIME, or DATETIME, you must specify the time ID variable name. You can specify the time ID variable either by typing the ID variable name in the Time ID field or by clicking the Select button.

If your data set does not contain a time ID variable with SAS date values, you can add a time ID variable using one of the windows described in Chapter 55, "Creating Time ID Variables."

Once the time ID variable is known, the Forecasting System examines the ID values to determine the time interval between observations. The data set SASHELP.CITIQTR contains quarterly observations. Therefore, the system determined that the data have a quarterly interval, and set the Interval field to QTR.

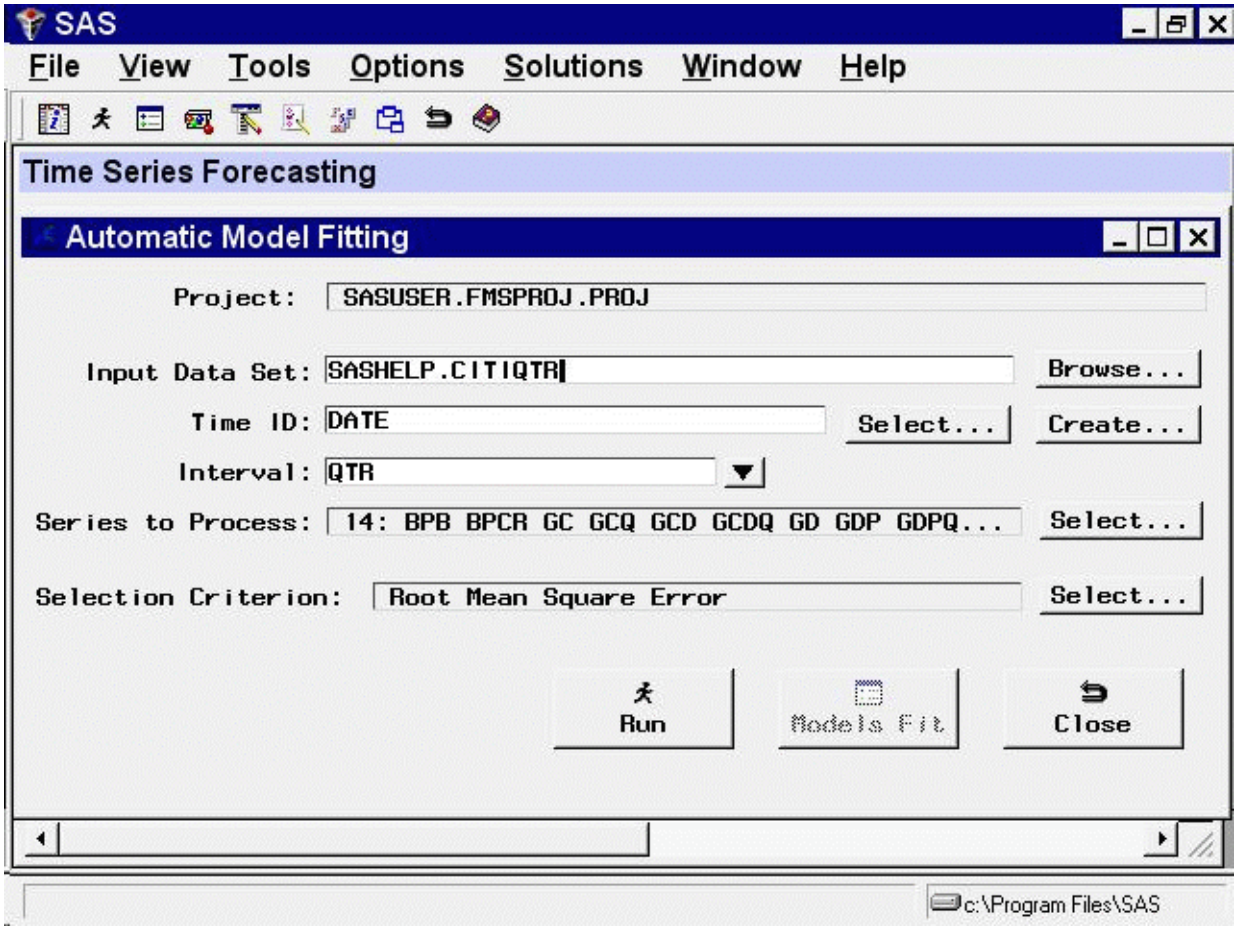
If the system cannot determine the data frequency from the values of the time ID variable, you must specify the time interval between observations. You can specify the time interval by using the `Interval` combo box. In addition to the interval names provided in the pop-up list, you can type in more complex interval names to specify an interval that is a multiple of other intervals or that has date values in the middle of the interval (such as monthly data with time ID values falling on the 10th day of the month).

See Chapter 4, "Working with Time Series Data," and Chapter 5, "Date Intervals, Formats, and Functions," for more information about time intervals, SAS date values, and ID variables for time series data sets.

Automatic Model Fitting Window

Before you can produce forecasts, you must fit forecasting models to the time series. Select the Fit Models Automatically button. This opens the Automatic Model Fitting window, as shown in Figure 54.9.

Figure 54.9 Automatic Model Fitting Window

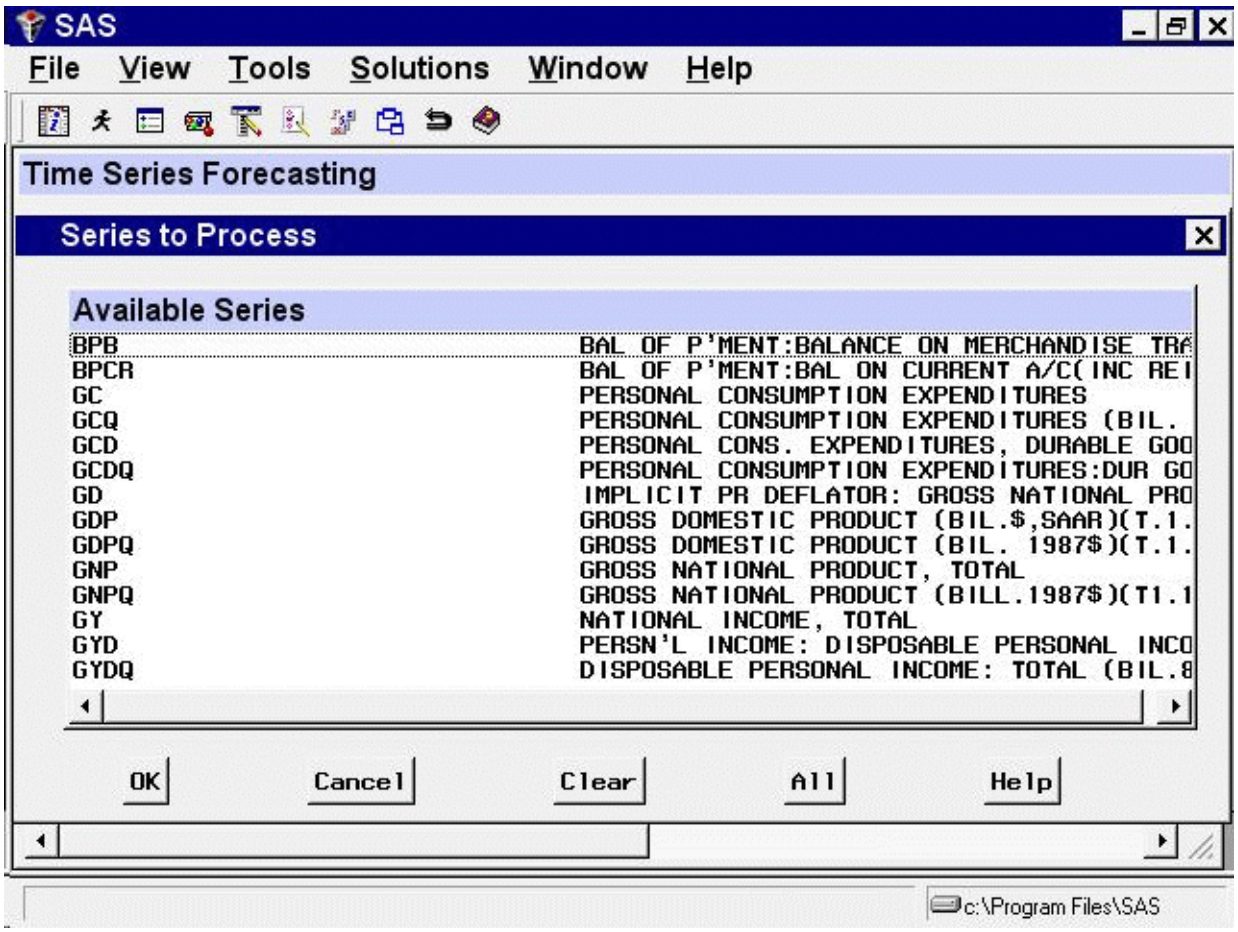


The first part of the Automatic Model Fitting window confirms the project filename and the input data set name.

The Series to Process field shows the number and lists the names of the variables in the input data set to which the Automatic Model Fitting process will be applied. By default, all numeric variables (except the time ID variable) are processed. However, you can specify that models be generated for only a select subset of these variables.

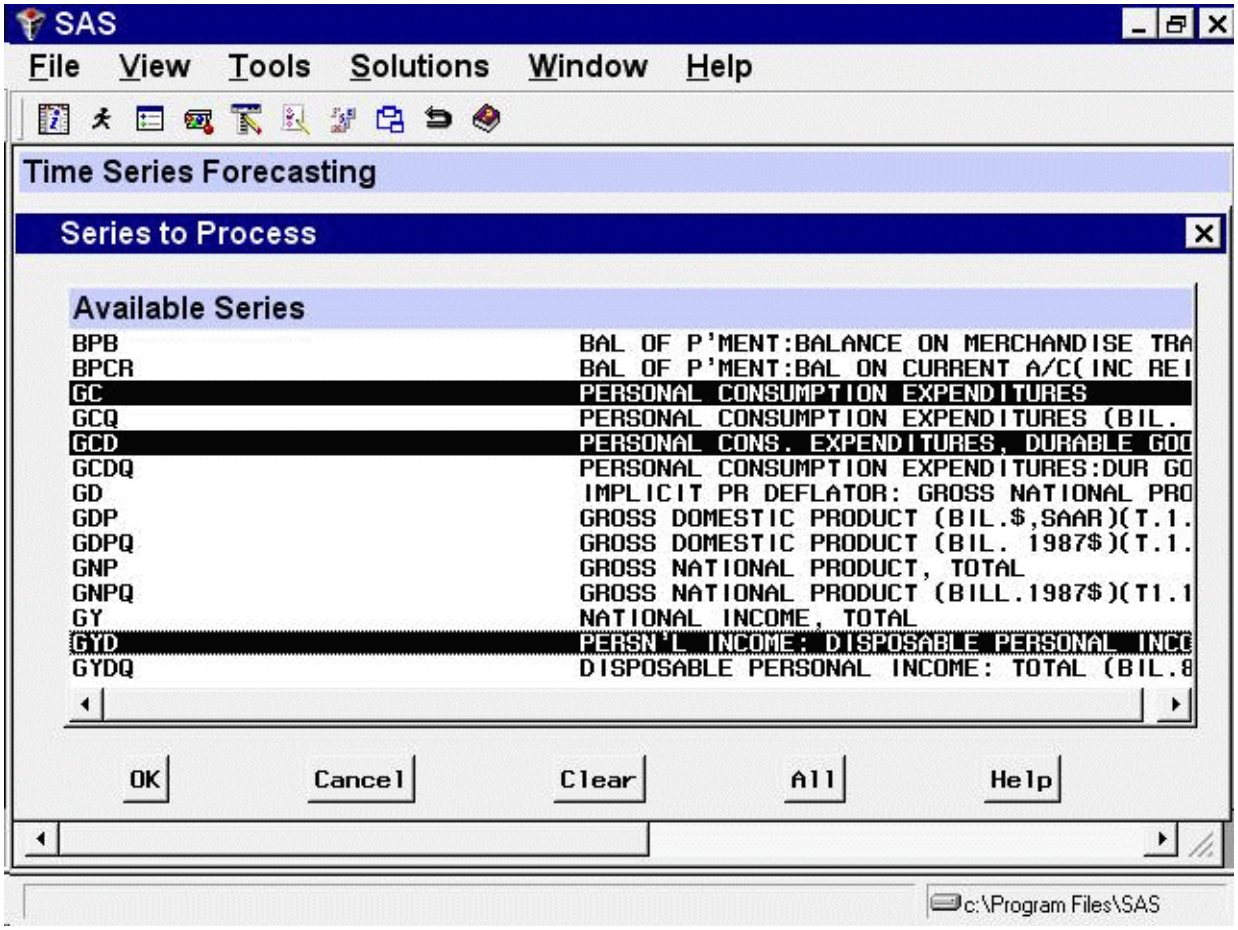
Click the Select button to the right of the Series to Process field. This opens the Series to Process window, as shown in Figure 54.10.

Figure 54.10 Series to Process Window



Use the mouse and the CTRL key to select the personal consumption expenditures series (GC), the personal consumption expenditures for durable goods series (GCD), and the disposable personal income series (GYD), as shown in Figure 54.11. (Remember to hold down the CTRL key as you make the selections; otherwise, selecting a second series will deselect the first.)

Figure 54.11 Selecting Series for Automatic Model Fitting

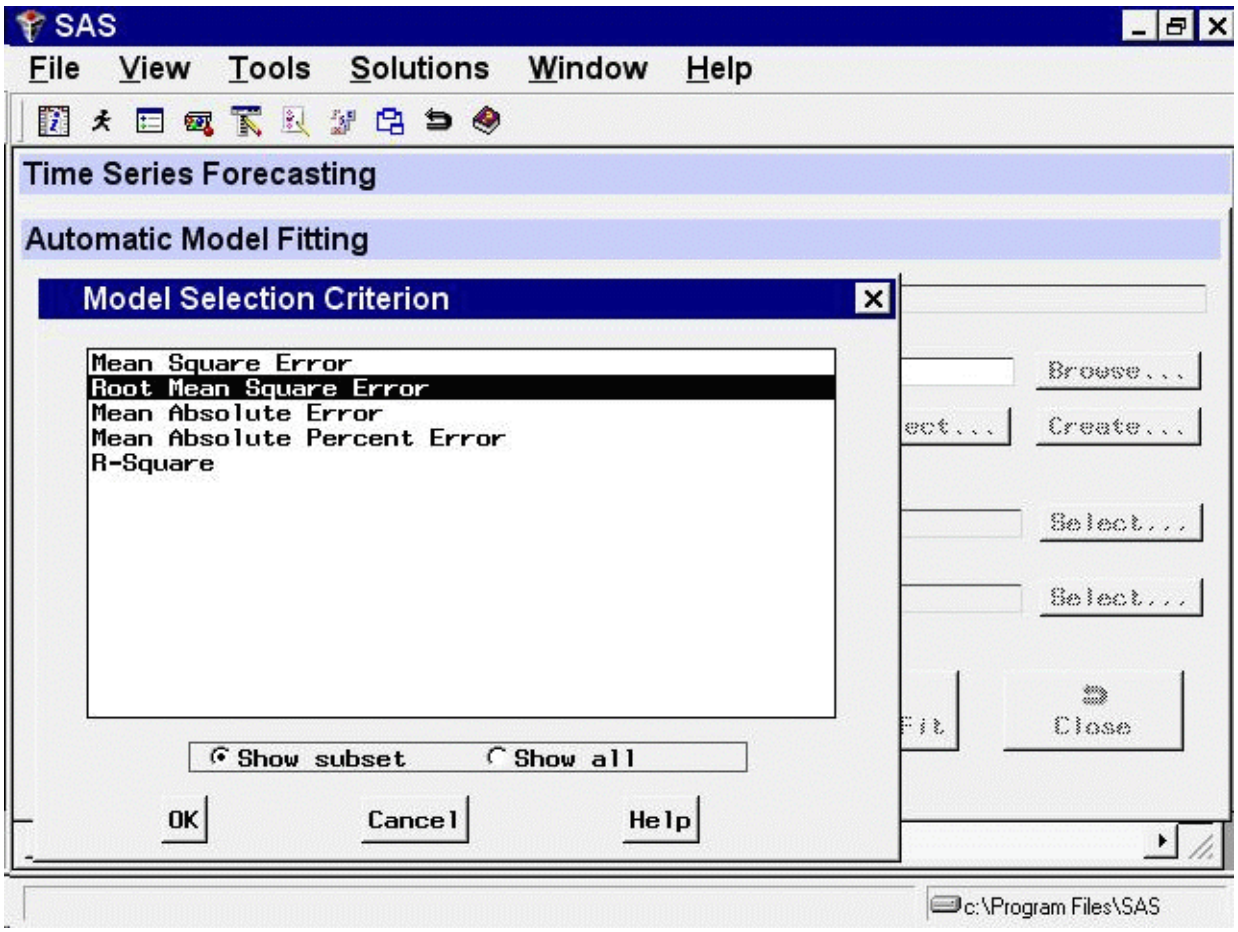


Now select the OK button. This returns you to the Automatic Model Fitting window. The Series to Process field now shows the selected variables.

The Selection Criterion field shows the goodness-of-fit measure that the Forecasting System will use to select the best fitting model for each series. By default, the selection criterion is the root mean squared error. To illustrate how you can control the selection criterion, this example uses the mean absolute percent error to select the best fitting models.

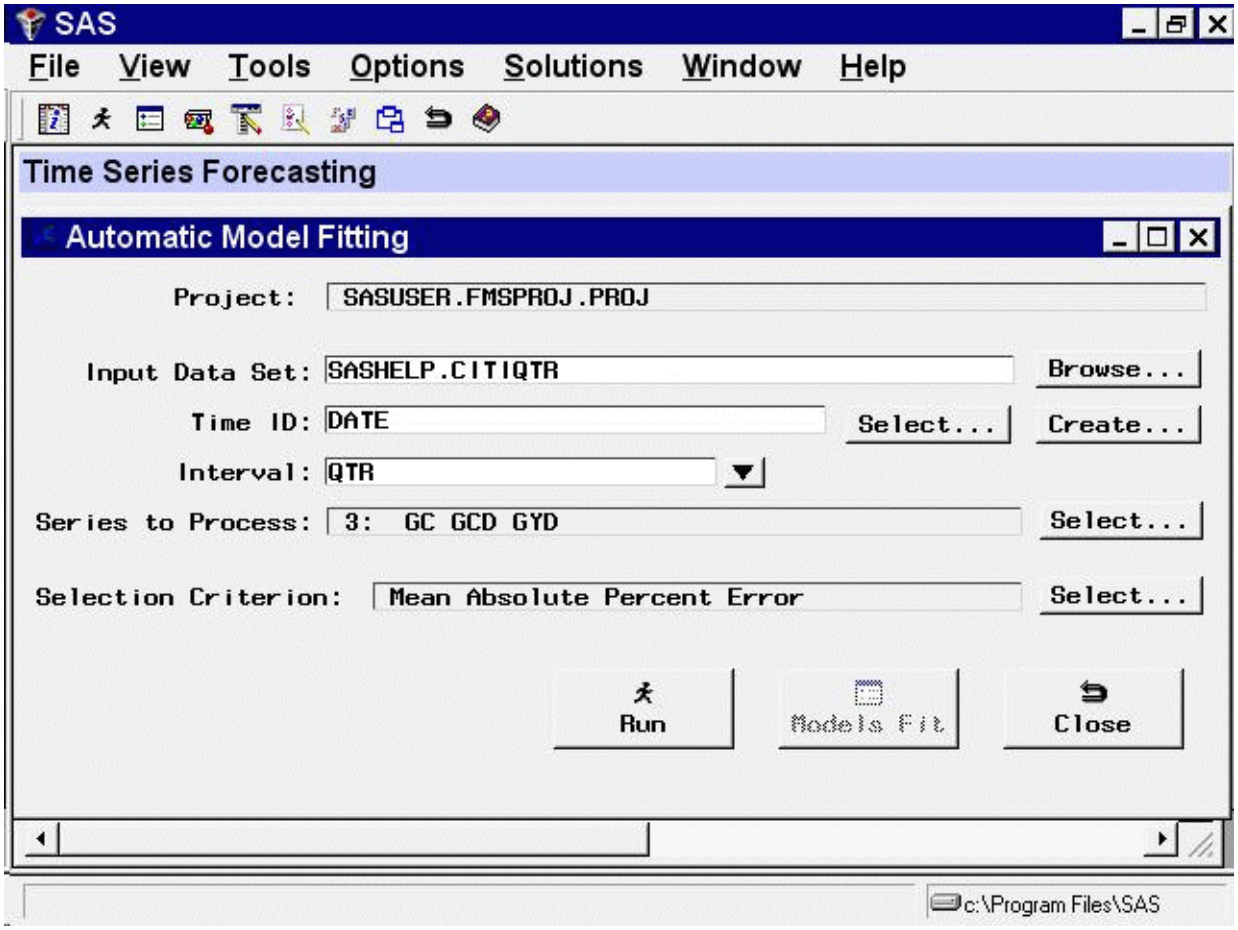
Click the Select button to the right of the Selection Criterion field. This opens a list of statistics of fit, as shown in Figure 54.12.

Figure 54.12 Choosing the Model Selection Criterion



Select *Mean Absolute Percent Error* and then select the OK button. The Automatic Model Fitting window now appears as shown in Figure 54.13.

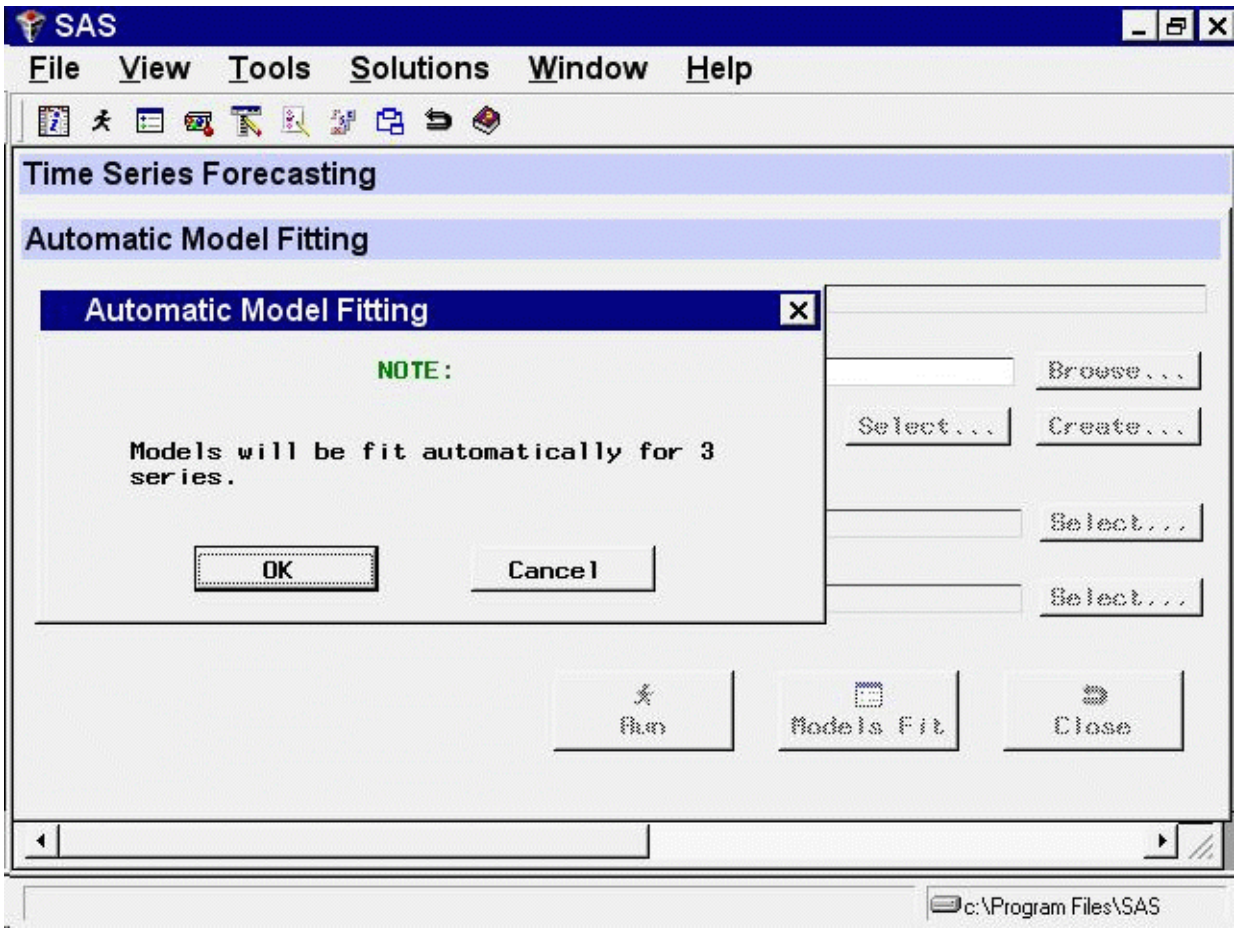
Figure 54.13 Automatic Model Fitting Window



Now that all the options are set appropriately, select the Run button.

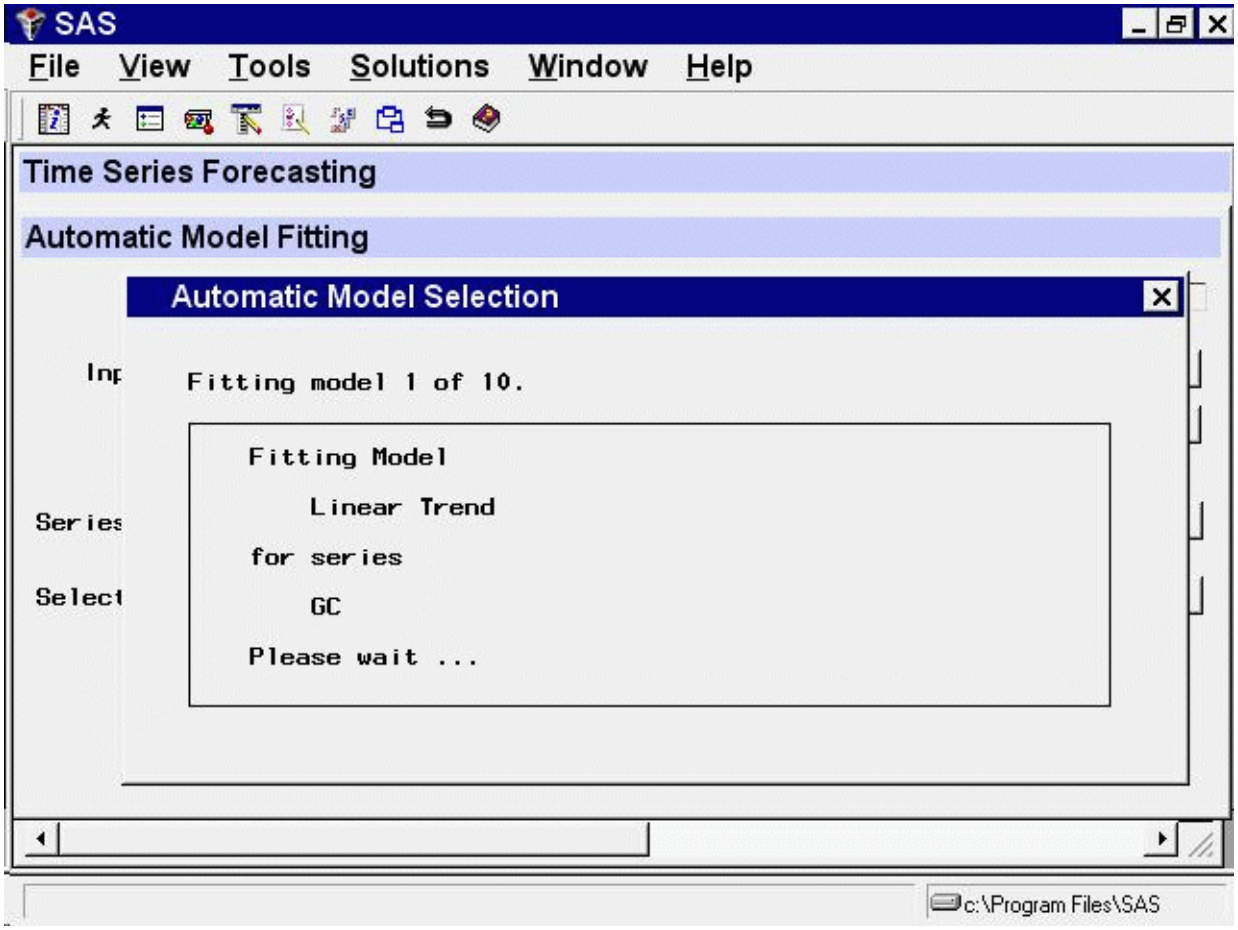
The Forecasting System now displays a notice, shown in Figure 54.14, confirming that models will be fit for three series using the automatic forecasting model search feature. This prompt is displayed because it is possible to fit models for a large number of series at once, which might take a lot of time. So the system gives you a chance to cancel if you accidentally ask to fit models for more series than you intended. Select the OK button.

Figure 54.14 Automatic Model Fitting Note



The system now fits several forecasting models to each of the three series you selected. While the models are being fit, the Forecasting System displays notices indicating what it is doing so that you can observe its progress, as shown in Figure 54.15.

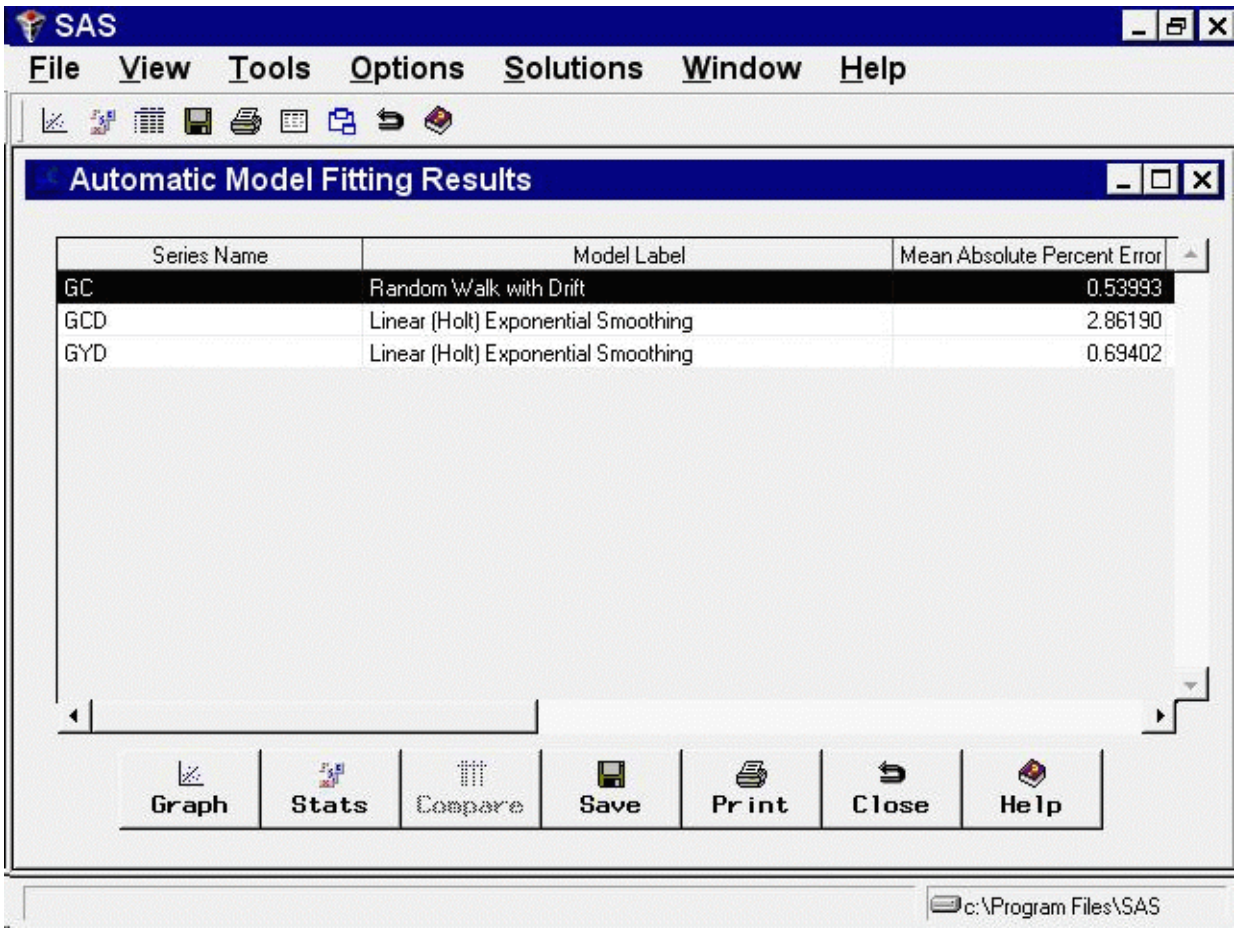
Figure 54.15 "Working" Notice



For each series, the system saves the model that produces the smallest mean absolute percent error. You can have the system save all the models fit by selecting `Automatic Fit` from the Options menu.

After the Automatic Model Fitting process has completed, the results are displayed in the Automatic Model Fitting Results window, as shown in Figure 54.16.

Figure 54.16 Automatic Model Fitting Results



This resizable window shows the list of series names and descriptive labels for the forecasting models chosen for them, as well as the values of the model selection criterion and other statistics of fit. Select the Close button.

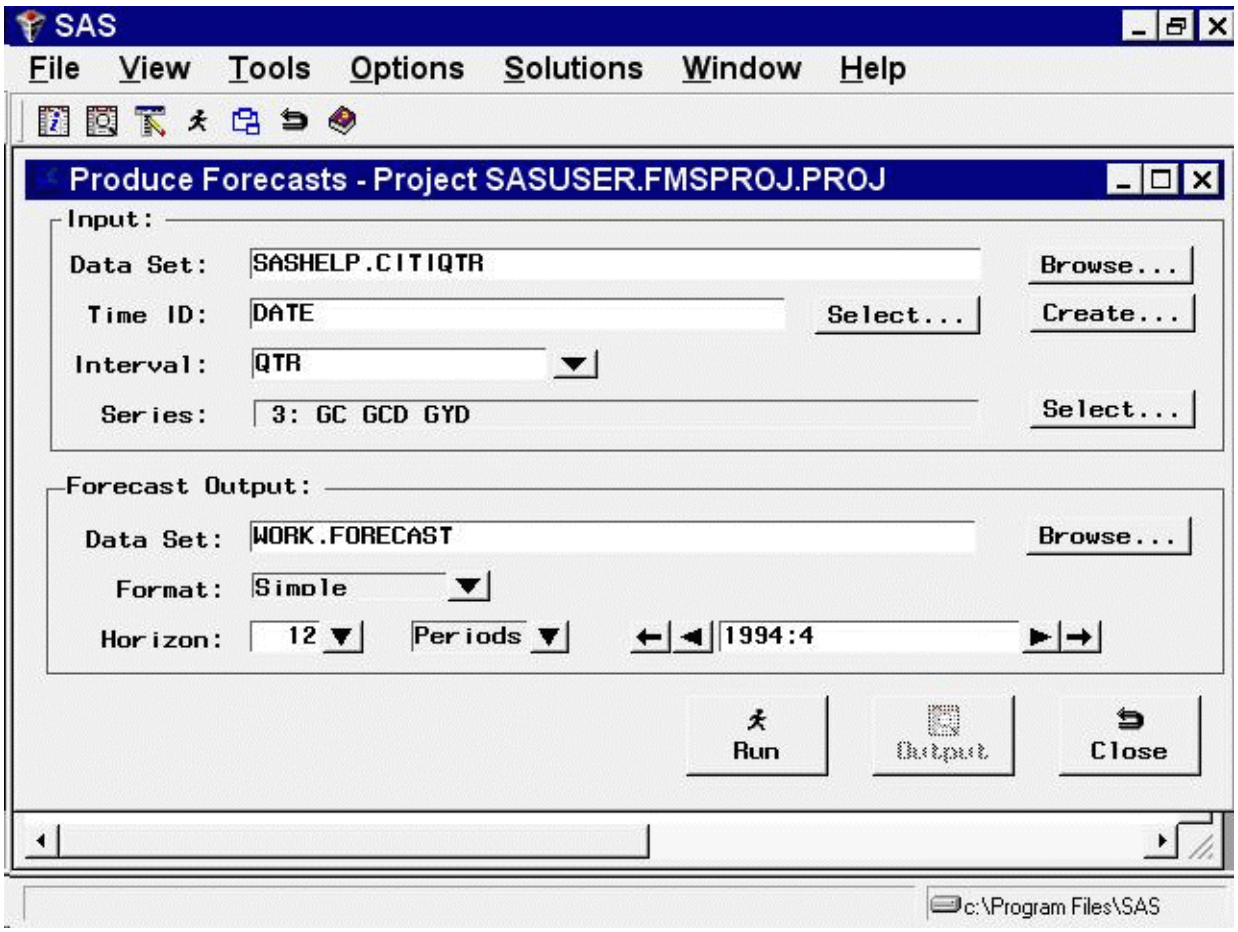
This returns you to the Automatic Model Fitting window. You can now fit models for other series in this data set or change to a different data set and fit models for series in the new data set.

Select the Close button to return to the Time Series Forecasting window.

Produce Forecasts Window

Now that you have forecasting models for these three series, you are ready to produce forecasts. Select the Produce Forecasts button. This opens the Produce Forecasts window, as shown in Figure 54.17.

Figure 54.17 Produce Forecasts Window

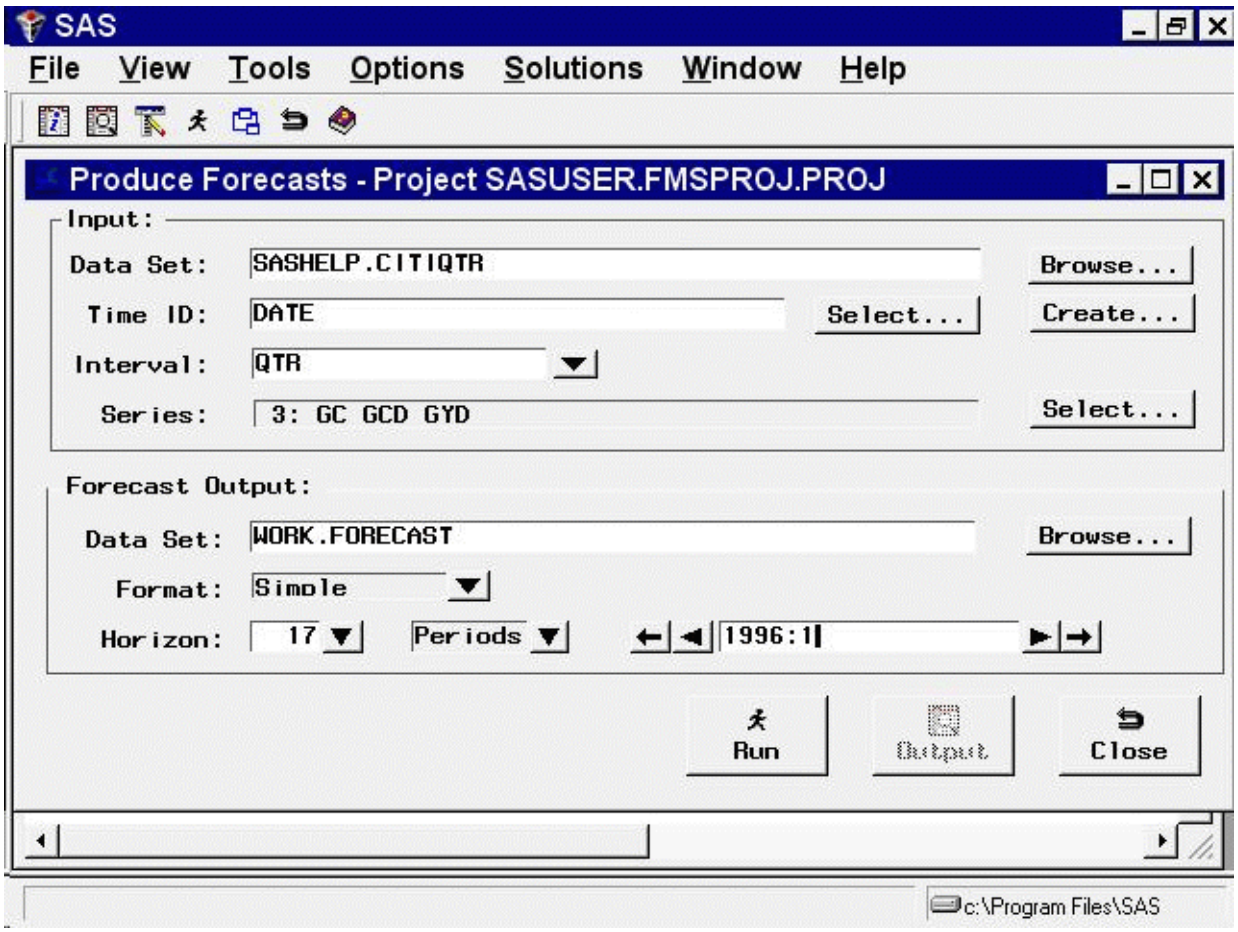


The Produce Forecasts window shows the input data set information and indicates the variables in the input data set for which forecasting models exist. Forecasts will be produced for these series. If you want to produce forecasts for only some of these series, use the Select button at the right of the Series field to select the series to forecast. The Data Set field in the `Forecast Output` box contains the name of the SAS data set in which the system will store the forecasts. The default output data set is WORK.FORECAST.

You can set the forecast horizon by using the controls on the line labeled `Horizon`. The default horizon is 12 periods. You can change it by specifying the number of periods, number of years, or the date of the last forecast period. Position the cursor in the date field and change the forecast ending date to 1 January 1996 by typing `jan1996` and pressing the ENTER key.

The window now appears as shown in Figure 54.18.

Figure 54.18 Produce Forecasts Window



Now select the Run button to produce the forecasts. The system indicates that the forecasts have been stored in the output data set. Select `OK` to dismiss the notice.

The Forecast Data Set

The Forecasting System can save the forecasts to a SAS data set in three different formats. Depending on your needs, you might find one of these output formats more convenient. The output data set format is controlled by the `Format` combo box. You can select the following output formats. The simple format is the default.

`Simple` The data set contains time ID variables and the forecast variables, and it contains one observation per time period. Observations for earlier time periods contain actual values copied from the input data set; later observations contain the forecasts.

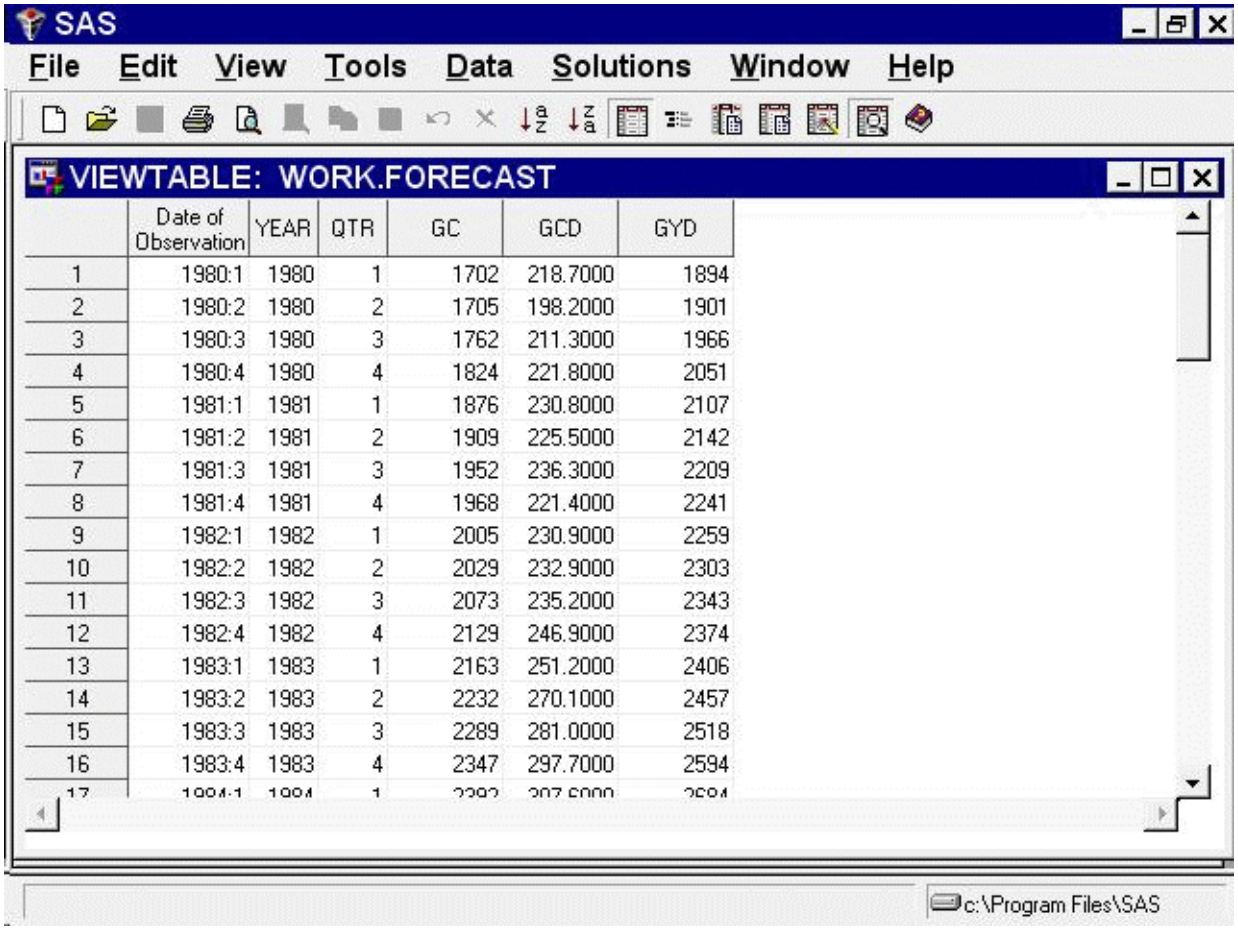
`Interleaved` The data set contains time ID variables, the variable TYPE, and the forecast variables. There are several observations per time period, with the meaning of each observation identified by the TYPE variable.

`Concatenated` The data set contains the variable SERIES, time ID variables, and the variables ACTUAL, PREDICT, ERROR, UPPER, LOWER, and STD. There is one observation per time period per forecast series. The variable SERIES contains the name of the forecast series, and the data set is sorted by SERIES and DATE.

Simple Format Forecast Data Set

To see the simple format forecast data set that the system created, select the Output button. This opens a VIEWTABLE window to display the data set, as shown in Figure 54.19.

Figure 54.19 Forecast Data Set—Simple Format



	Date of Observation	YEAR	QTR	GC	GCD	GYD
1	1980:1	1980	1	1702	218.7000	1894
2	1980:2	1980	2	1705	198.2000	1901
3	1980:3	1980	3	1762	211.3000	1966
4	1980:4	1980	4	1824	221.8000	2051
5	1981:1	1981	1	1876	230.8000	2107
6	1981:2	1981	2	1909	225.5000	2142
7	1981:3	1981	3	1952	236.3000	2209
8	1981:4	1981	4	1968	221.4000	2241
9	1982:1	1982	1	2005	230.9000	2259
10	1982:2	1982	2	2029	232.9000	2303
11	1982:3	1982	3	2073	235.2000	2343
12	1982:4	1982	4	2129	246.9000	2374
13	1983:1	1983	1	2163	251.2000	2406
14	1983:2	1983	2	2232	270.1000	2457
15	1983:3	1983	3	2289	281.0000	2518
16	1983:4	1983	4	2347	297.7000	2594

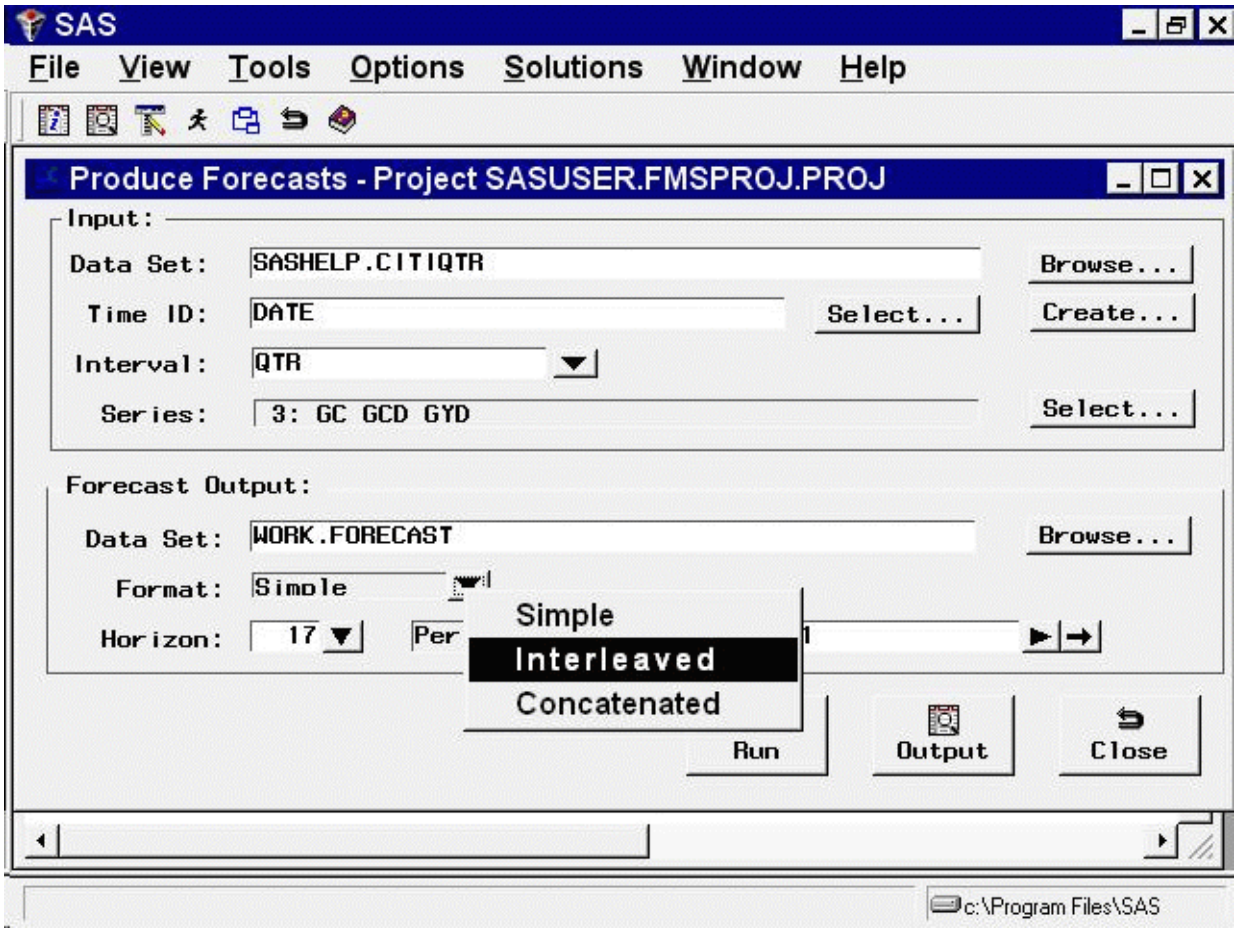
Figure 54.19 shows the default simple format. This form of the forecast data set contains time ID variables and the variables that you forecast. The forecast variables contain actual values or predicted values, depending on whether the date of the observation is within the range of data supplied in the input data set.

Select `File` and `Close` to close the Viewtable window.

Interleaved Format Forecast Data Set

From the Produce Forecasts window, use the list to select the `Interleaved` format, as shown in Figure 54.20.

Figure 54.20 Forecast Data Set Options

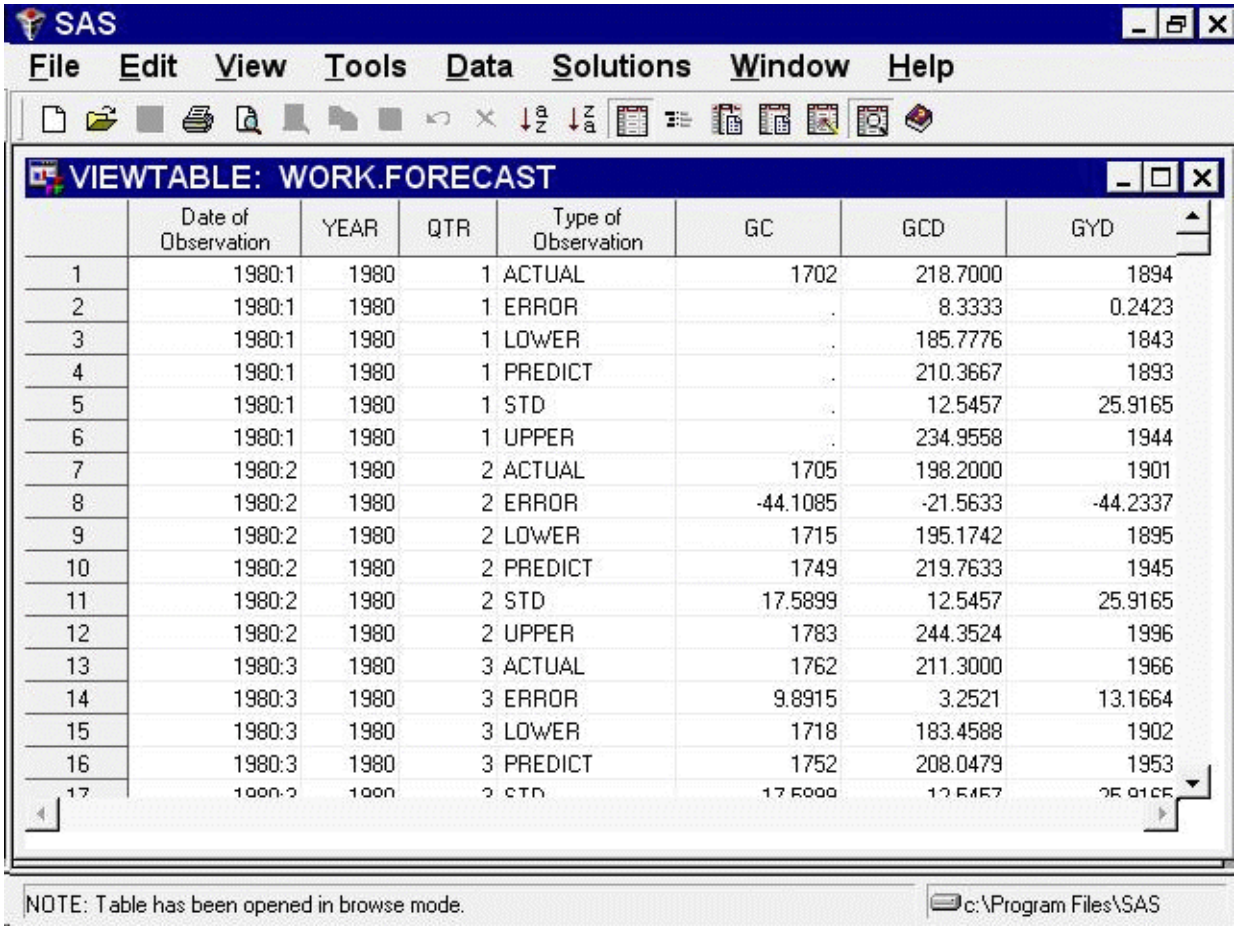


Now select the Run button again. The system presents a warning notice reminding you that the data set WORK.FORECAST already exists and asking if you want to replace it. Select `Replace`.

The forecasts are stored in the data set WORK.FORECAST again, this time in the *Interleaved* format. Dismiss the notice that the forecast was stored.

Now select the Output button again. This opens a Viewtable window to display the data set, as shown in Figure 54.21.

Figure 54.21 Forecast Data Set—Interleaved Format



SAS

File Edit View Tools Data Solutions Window Help

VIEWTABLE: WORK.FORECAST

	Date of Observation	YEAR	QTR	Type of Observation	GC	GCD	GYD
1	1980:1	1980	1	ACTUAL	1702	218.7000	1894
2	1980:1	1980	1	ERROR	.	8.3333	0.2423
3	1980:1	1980	1	LOWER	.	185.7776	1843
4	1980:1	1980	1	PREDICT	.	210.3667	1893
5	1980:1	1980	1	STD	.	12.5457	25.9165
6	1980:1	1980	1	UPPER	.	234.9558	1944
7	1980:2	1980	2	ACTUAL	1705	198.2000	1901
8	1980:2	1980	2	ERROR	-44.1085	-21.5633	-44.2337
9	1980:2	1980	2	LOWER	1715	195.1742	1895
10	1980:2	1980	2	PREDICT	1749	219.7633	1945
11	1980:2	1980	2	STD	17.5899	12.5457	25.9165
12	1980:2	1980	2	UPPER	1783	244.3524	1996
13	1980:3	1980	3	ACTUAL	1762	211.3000	1966
14	1980:3	1980	3	ERROR	9.8915	3.2521	13.1664
15	1980:3	1980	3	LOWER	1718	183.4588	1902
16	1980:3	1980	3	PREDICT	1752	208.0479	1953
17	1980:3	1980	3	STD	17.5899	12.5457	25.9165

NOTE: Table has been opened in browse mode.

c:\Program Files\SAS

In the interleaved format, there are several output observations for each input observation, identified by the TYPE variable. The values of the forecast variables for observations with different TYPE values are as follows.

`ACTUAL` actual values copied from the input data set

`ERROR` the difference between the actual and predicted values

`LOWER` the lower confidence limits

`PREDICT` the predicted values from the forecasting model These are within-sample, one-step-ahead predictions for observations within the historical period, or multistep predictions for observations within the forecast period

`STD` the estimated standard deviations of the prediction errors

`UPPER` the upper confidence limits

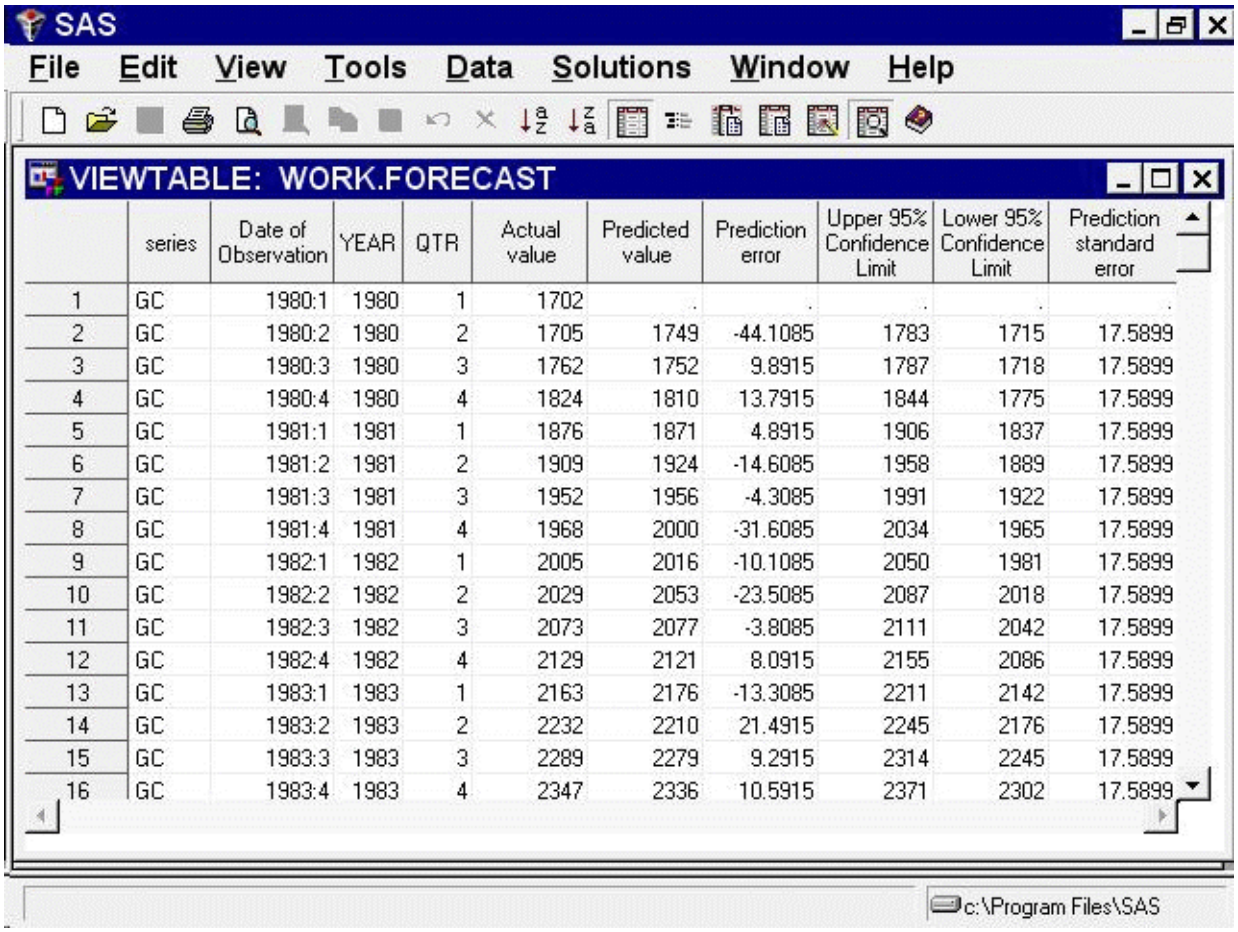
Select `File` and `Close` to close the VIEWTABLE window.

Concatenated Format Forecast Data Set

Use the list to select the `Concatenated` format. Create the forecast data set again, and then select the Output button.

The Viewtable window showing the concatenated format of the forecast data set appears, as shown in Figure 54.22.

Figure 54.22 Forecast Data Set—Concatenated Format



	series	Date of Observation	YEAR	QTR	Actual value	Predicted value	Prediction error	Upper 95% Confidence Limit	Lower 95% Confidence Limit	Prediction standard error
1	GC	1980:1	1980	1	1702	.	.	.	.	.
2	GC	1980:2	1980	2	1705	1749	-44.1085	1783	1715	17.5899
3	GC	1980:3	1980	3	1762	1752	9.8915	1787	1718	17.5899
4	GC	1980:4	1980	4	1824	1810	13.7915	1844	1775	17.5899
5	GC	1981:1	1981	1	1876	1871	4.8915	1906	1837	17.5899
6	GC	1981:2	1981	2	1909	1924	-14.6085	1958	1889	17.5899
7	GC	1981:3	1981	3	1952	1956	-4.3085	1991	1922	17.5899
8	GC	1981:4	1981	4	1968	2000	-31.6085	2034	1965	17.5899
9	GC	1982:1	1982	1	2005	2016	-10.1085	2050	1981	17.5899
10	GC	1982:2	1982	2	2029	2053	-23.5085	2087	2018	17.5899
11	GC	1982:3	1982	3	2073	2077	-3.8085	2111	2042	17.5899
12	GC	1982:4	1982	4	2129	2121	8.0915	2155	2086	17.5899
13	GC	1983:1	1983	1	2163	2176	-13.3085	2211	2142	17.5899
14	GC	1983:2	1983	2	2232	2210	21.4915	2245	2176	17.5899
15	GC	1983:3	1983	3	2289	2279	9.2915	2314	2245	17.5899
16	GC	1983:4	1983	4	2347	2336	10.5915	2371	2302	17.5899

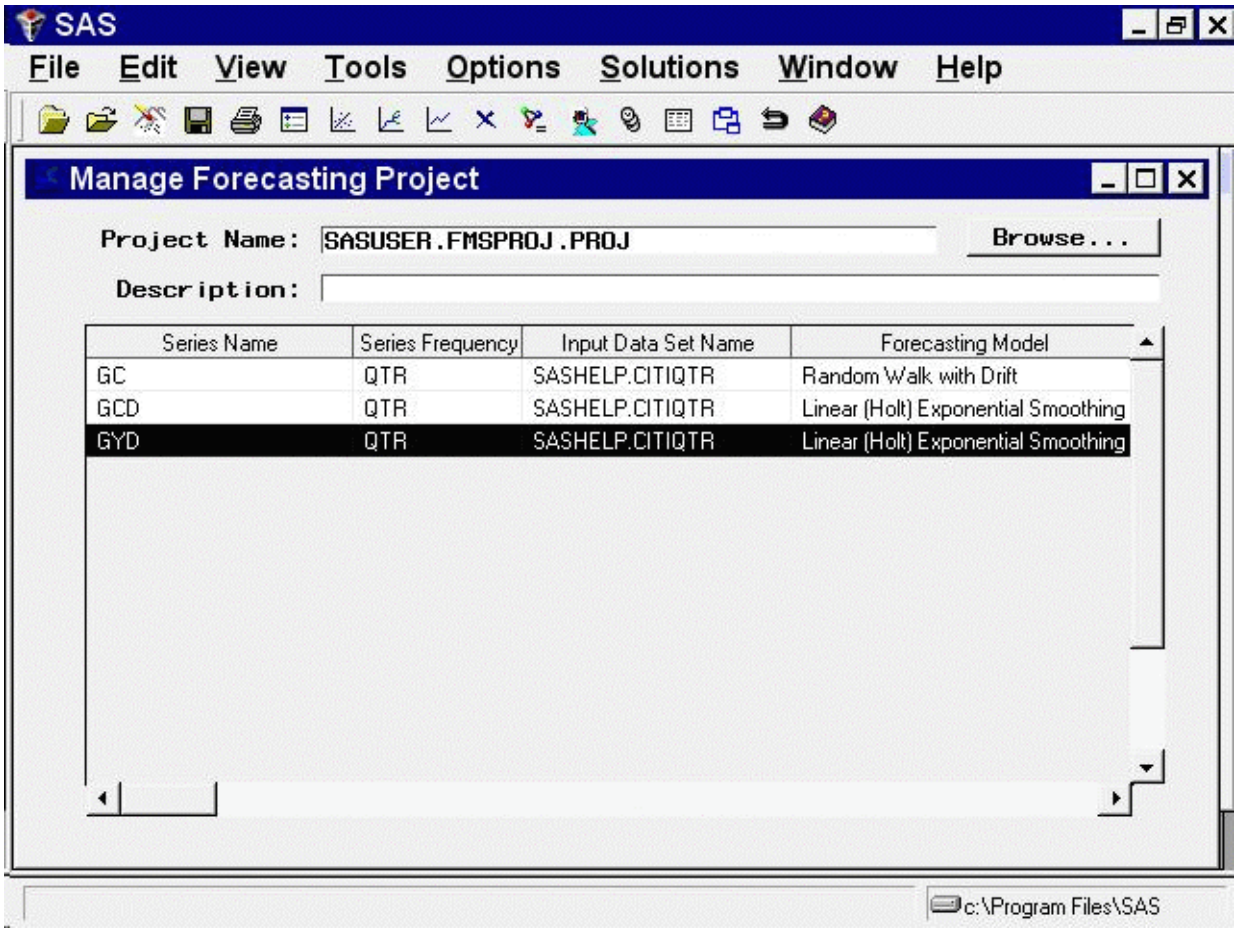
This completes the example of how to use the Produce Forecasts window. Select `File` and `Close` to close the Viewtable window. Select the Close button to return to the Time Series Forecasting window.

Forecasting Projects

The system collects all the forecasting models you create, together with the options you set, into a package called a *forecasting project*. You can save this information in a SAS catalog entry and restore your work in later forecasting sessions. You can store any number of forecasting projects under different catalog entry names.

To see how this works, select the Manage Projects button. This opens the Manage Forecasting Project window, as shown in Figure 54.23.

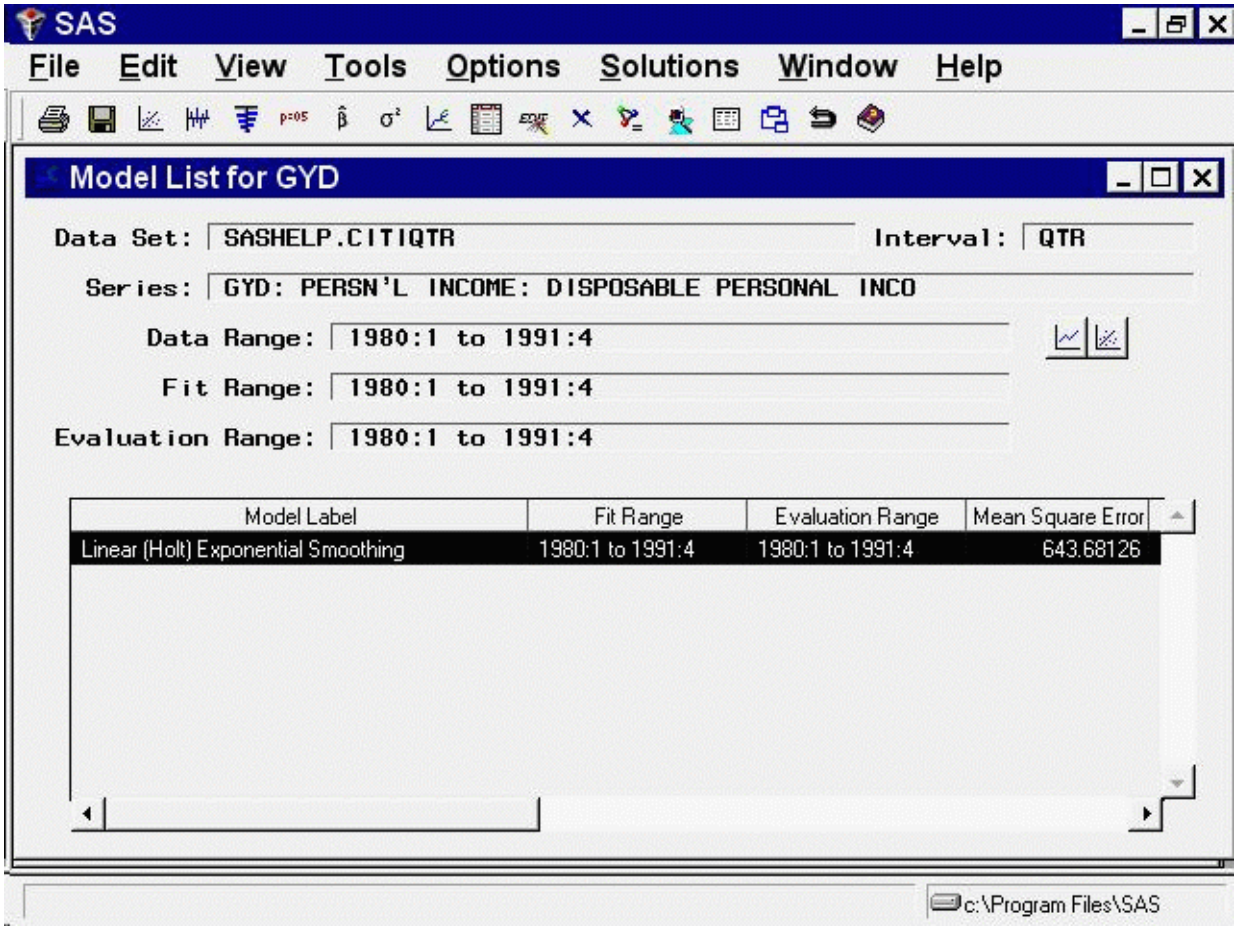
Figure 54.23 Manage Forecasting Project Window



The table in this window lists the series for which forecasting models have been fit, and it shows for each series the forecasting model used to produce the forecasts. This window provides several features that allow you to manage the information in your forecasting project.

You can select a row of the table to drill down to the list of models fit to the series. Select the GYD row of the table, either by double-clicking with the mouse or by clicking once to highlight the table row and then selecting `List Models` from the toolbar or from the Tools menu. This opens the Model List window for this series, as shown in Figure 54.24.

Figure 54.24 Model List Window



Because the Automatic Model Fitting process kept only the best fitting model, only one model appears in the model list. You can fit and retain any number of models for each series, and all the models fit and kept will appear in the series' model list.

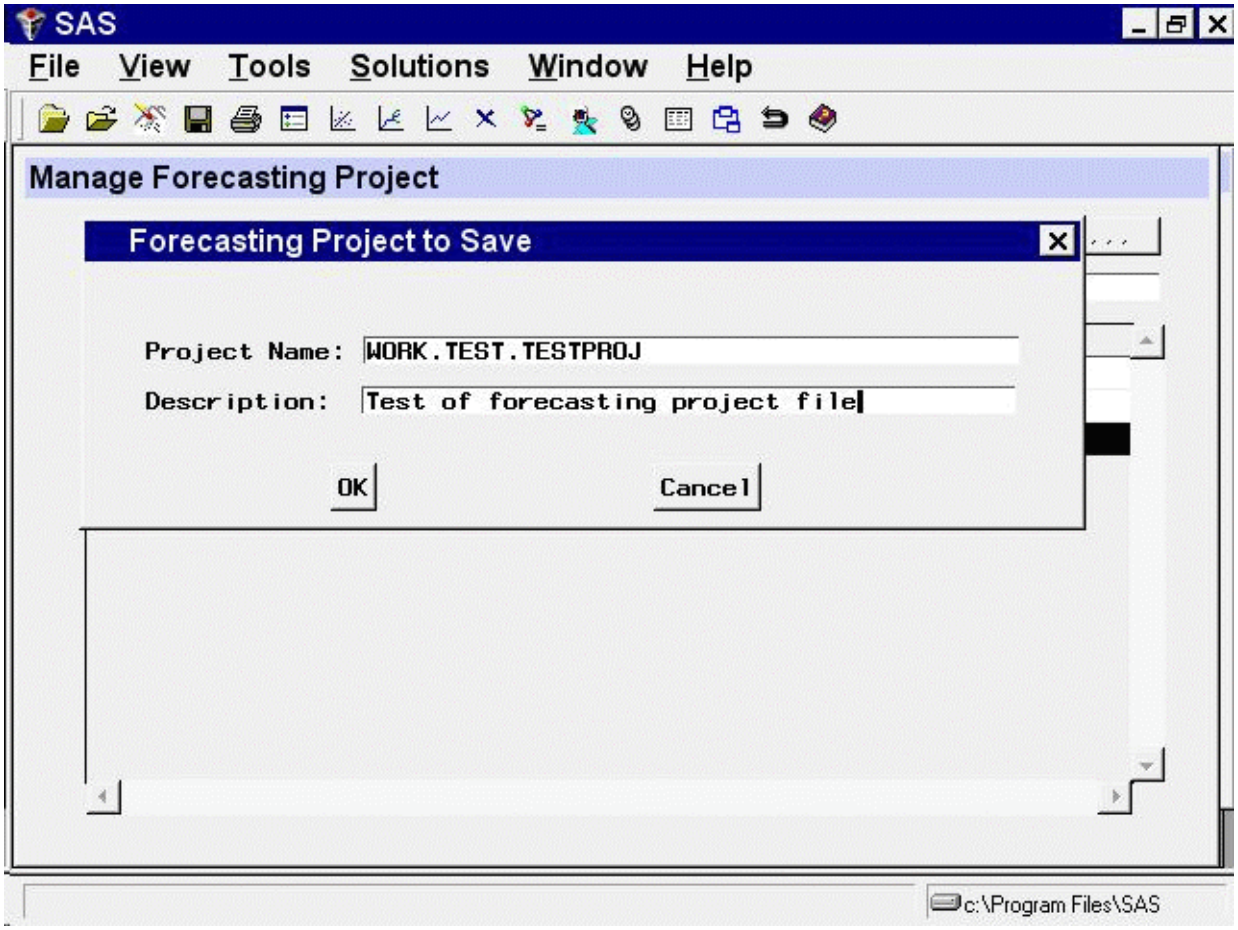
Select `Close` from the toolbar or from the File menu to return to the Manage Forecasting Project window.

Saving and Restoring Project Information

To illustrate how you can save your work between sessions, in this section you will exit and then re-enter the Forecasting System.

From the Manage Forecasting Project window, select `File` and `Save as`. This opens the Forecasting Project to Save window. In the Project Name field, type the name WORK.TEST.TESTPROJ. In the Description field, type "Test of forecasting project file." The window should now appear as shown in Figure 54.25.

Figure 54.25 Project to Save Name and Description



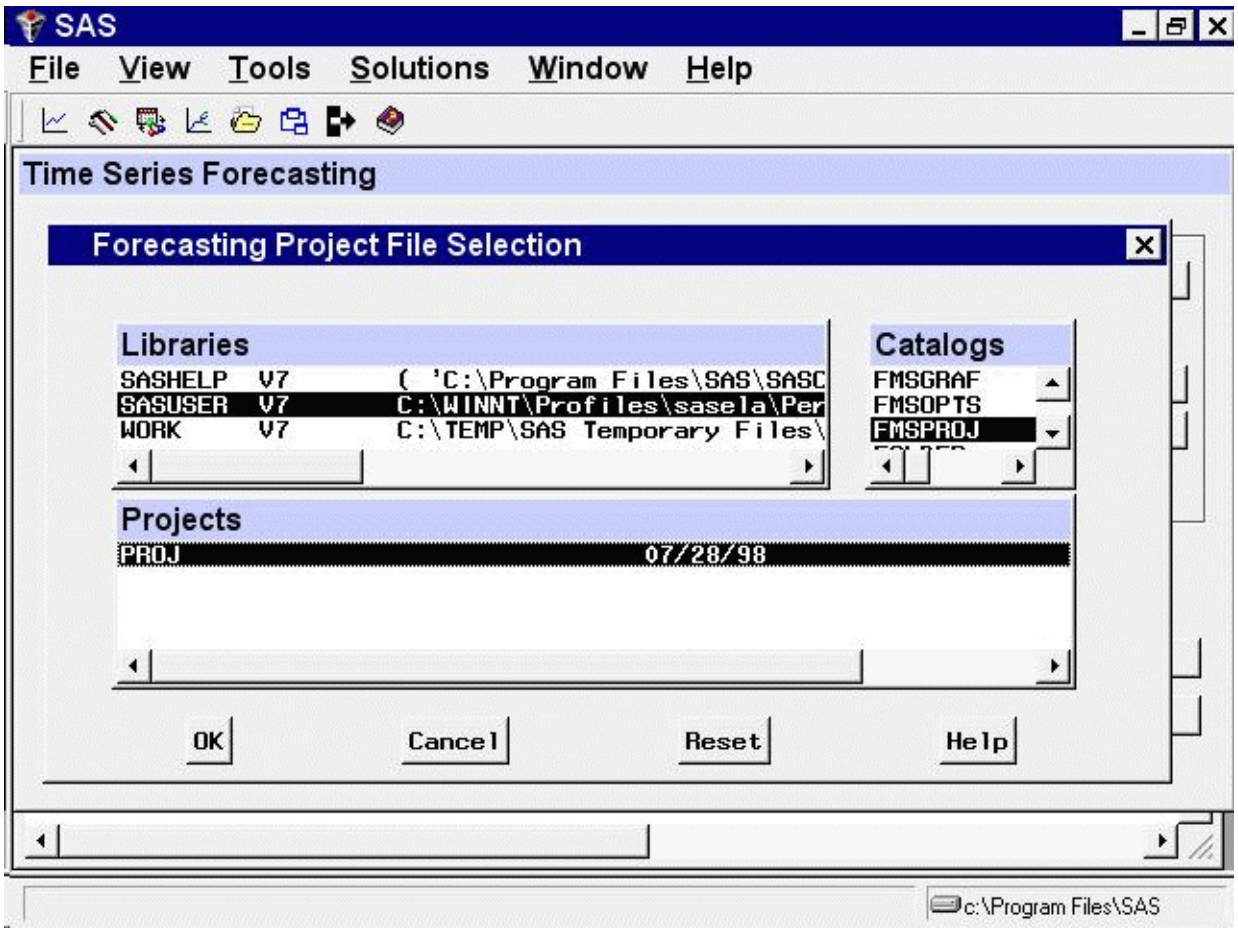
Select the OK button. This returns you to the Project Management window and displays a message indicating that the project was saved.

Select `Close` from the toolbar or from the File menu to return to the Time Series Forecasting window. Now select the Exit button. The system asks if you are sure you want to exit the system; select `Yes`. The forecasting application now terminates.

Open the forecasting application again. A new project name is displayed by default.

Now restore the forecasting project you saved previously. Select the Browse button to the right of the Project field. This opens the Forecasting Project File Selection window, as shown in Figure 54.26.

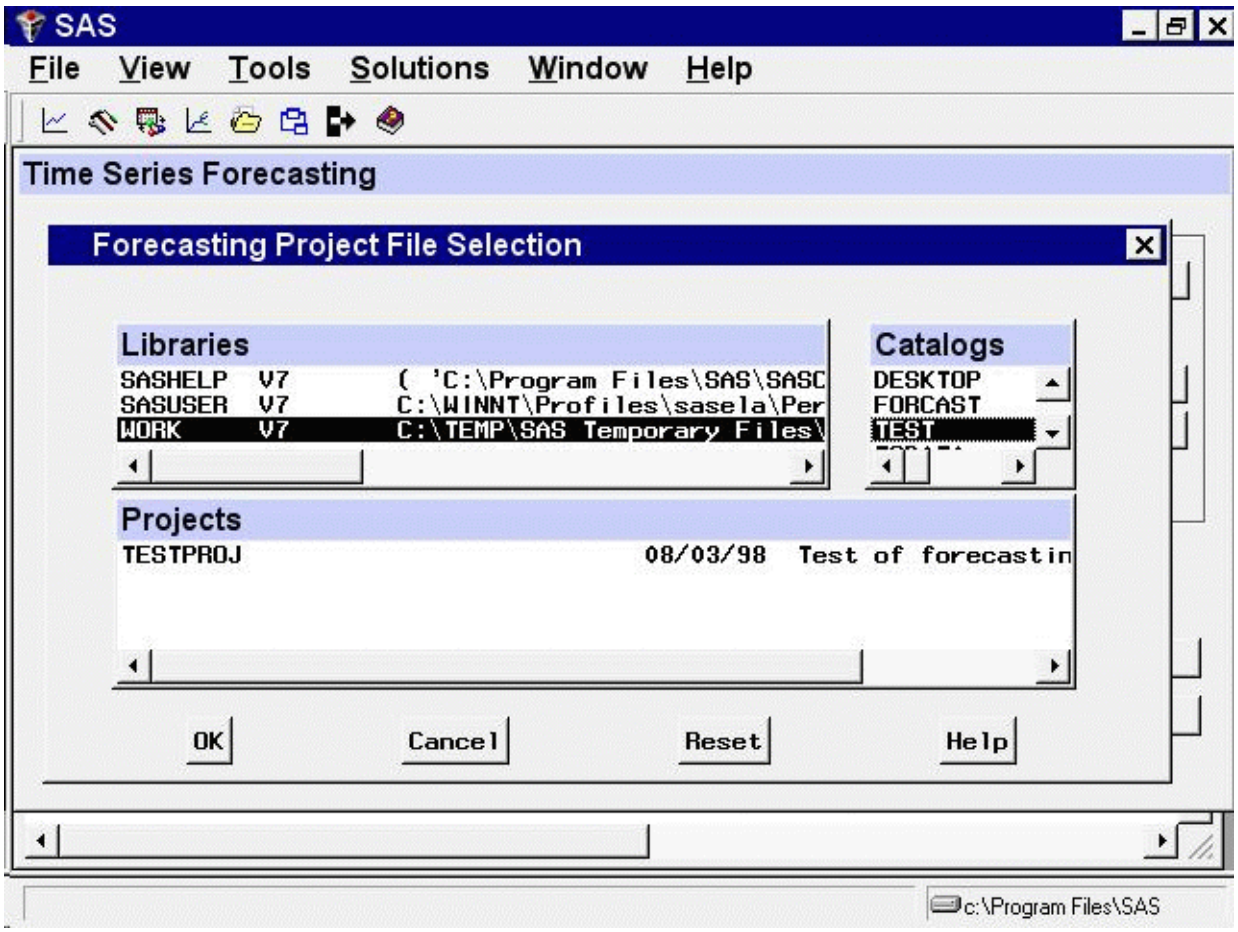
Figure 54.26 Forecasting Project File Selection Window



Select the WORK library from the Libraries list. The Catalogs list now shows all the SAS catalogs in the WORK library.

Select the TEST catalog. The Projects list now shows the list of forecasting projects in the catalog TEST. So far, you have created only one project file, TESTPROJ; so TESTPROJ is the only entry in the Projects list, as shown in Figure 54.27.

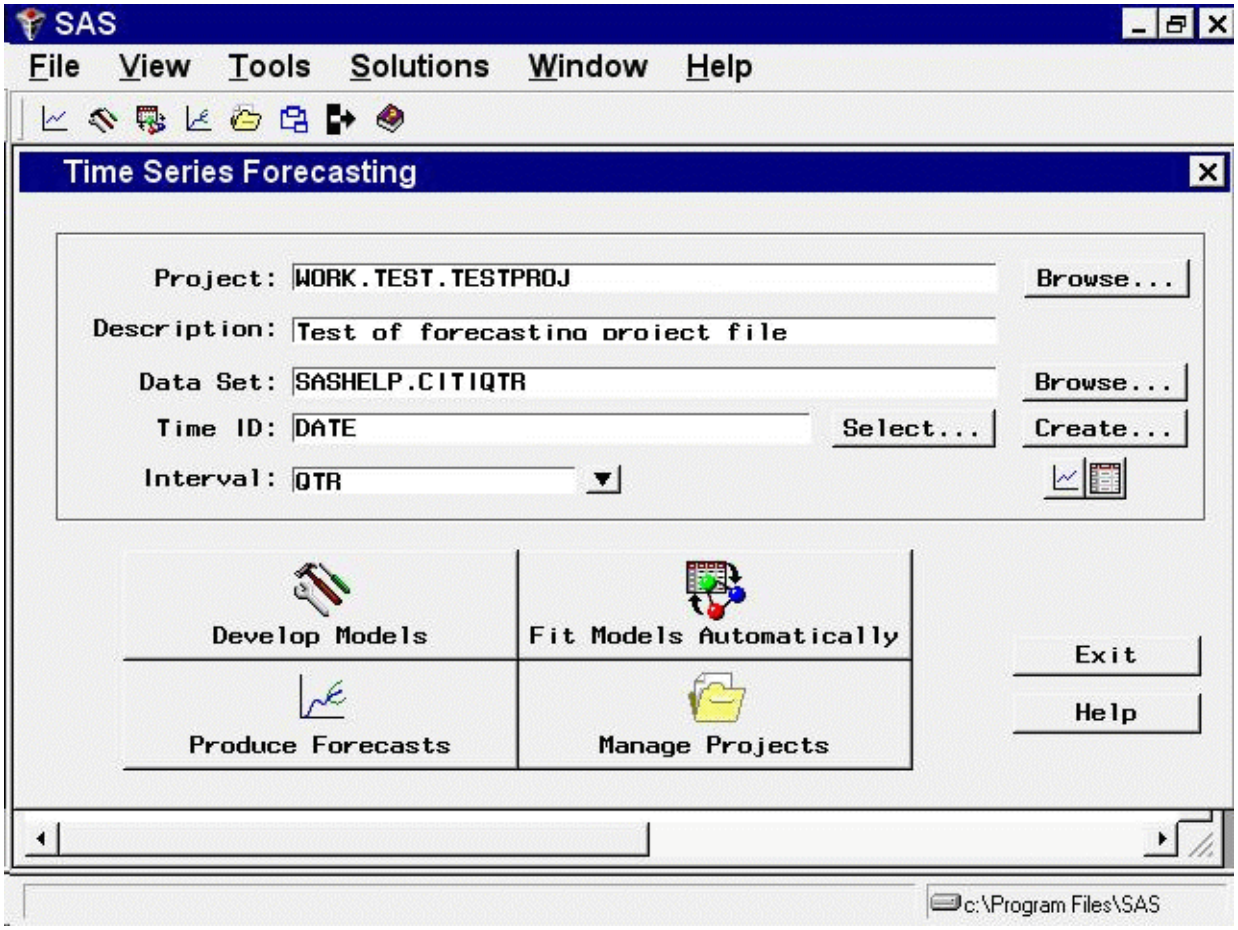
Figure 54.27 Forecasting Projects List



Select TESTPROJ from the Projects list and then select the OK button. This returns you to the Time Series Forecasting window.

The system loads the project information you saved in TESTPROJ and displays a message indicating this. The Project field is now set to WORK.TEST.TESTPROJ, and the description is the description you previously gave to TESTPROJ, as shown in Figure 54.28.

Figure 54.28 Time Series Forecasting Window after Loading Project



If you now select the Manage Projects button, you will see the list of series and forecasting models you created in the previous forecasting session.

Sharing Projects

If you plan to work with others on a forecasting project, you might need to consider how project information can be shared. The series, models, and results of your project are stored in a forecasting project (FMSPROJ) catalog entry in the location you specify, as illustrated in the previous section. You need only read access to the catalog to work with it, but you must have write access to save the project. Multiple users cannot open a project for update at the same time, but they can do so at different times if they all have write access to the catalog where it is stored.

Project options settings such as the *model selection criterion* and *number of models to keep* are stored in an SLIST catalog entry in the SASUSER or TSFSUSER library. Write access to this catalog is required. If you have only read access to the SASUSER library, you can use the -RSASUSER option when starting SAS. You will be prompted for a location for the TSFSUSER library, if it is not already assigned. If you want to use TSFSUSER routinely, assign it before you start the Time Series Forecasting System. Select New from the SAS Explorer file menu. In the New Library window, type TSFSUSER for the name. Click the Browse

button and select the directory or folder you want to use. Turn on the *enable at startup* option so this library will be assigned automatically in subsequent sessions.

The SASUSER library is typically used for private settings saved by individual users. This is the default location for project options. If a work group shares a single options catalog (SASUSER or TSFSUSER points to the same location for all users), then only one user can use the system at a time.

Develop Models Window

In the first forecasting example, you used the Automatic Model Fitting window to fit and select the forecasting model for each series automatically. In addition to this automatic forecasting process, you can also work with time series one at a time to fit forecasting models and apply your own judgment to choose the best forecasting model for each series.

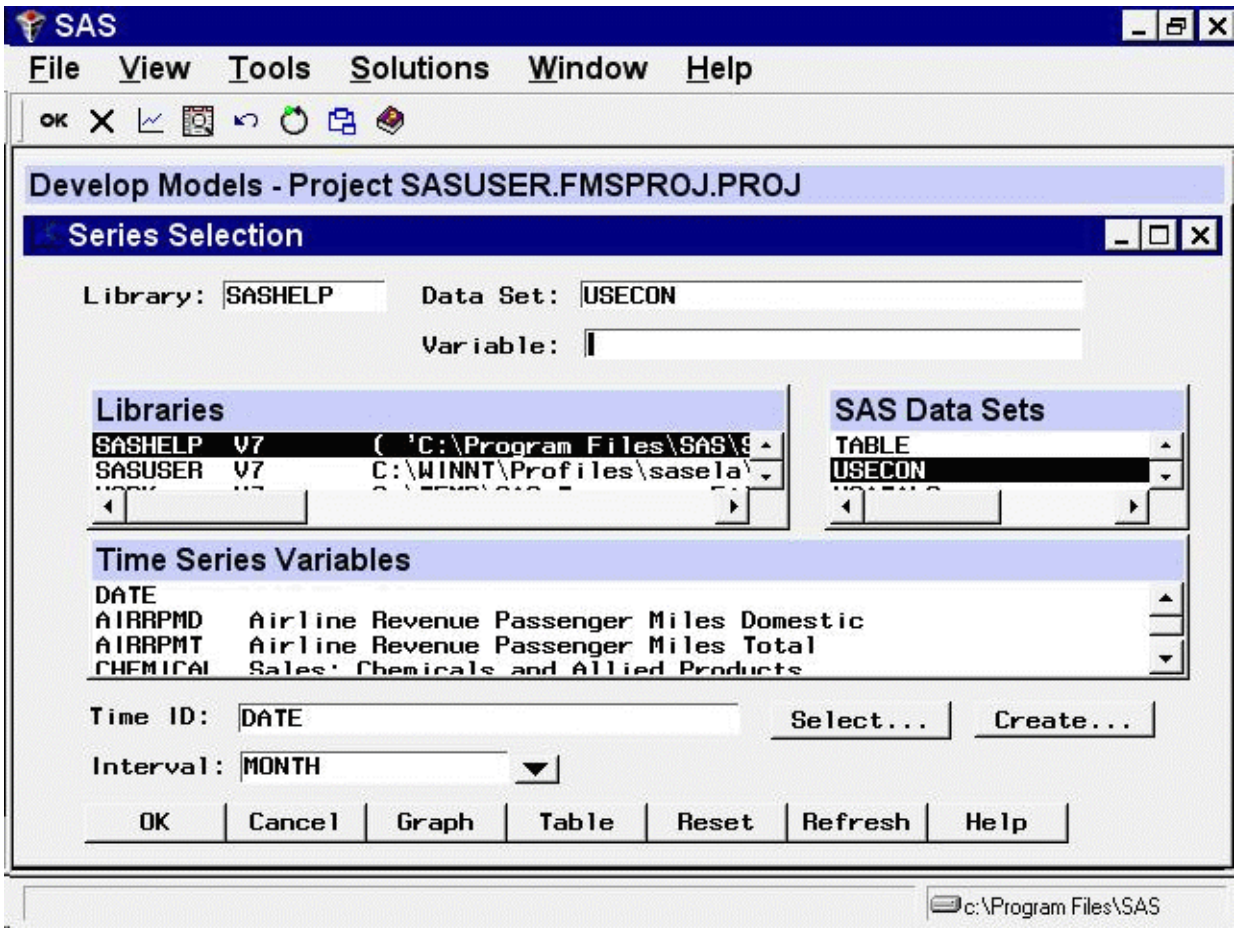
Using the Automatic Model Fitting feature, the system acts like a "black box." This section goes inside the black box to look at the kinds of forecasting methods that the system provides and introduces some of the tools the system offers to help you find the best forecasting model.

Introduction

From the Time Series Forecasting window, select the Browse button to the right of the Data Set field to open the Data Set Selection window. Select the USECON data set from the SASHELP library. This data set contains monthly data on the U.S. economy.

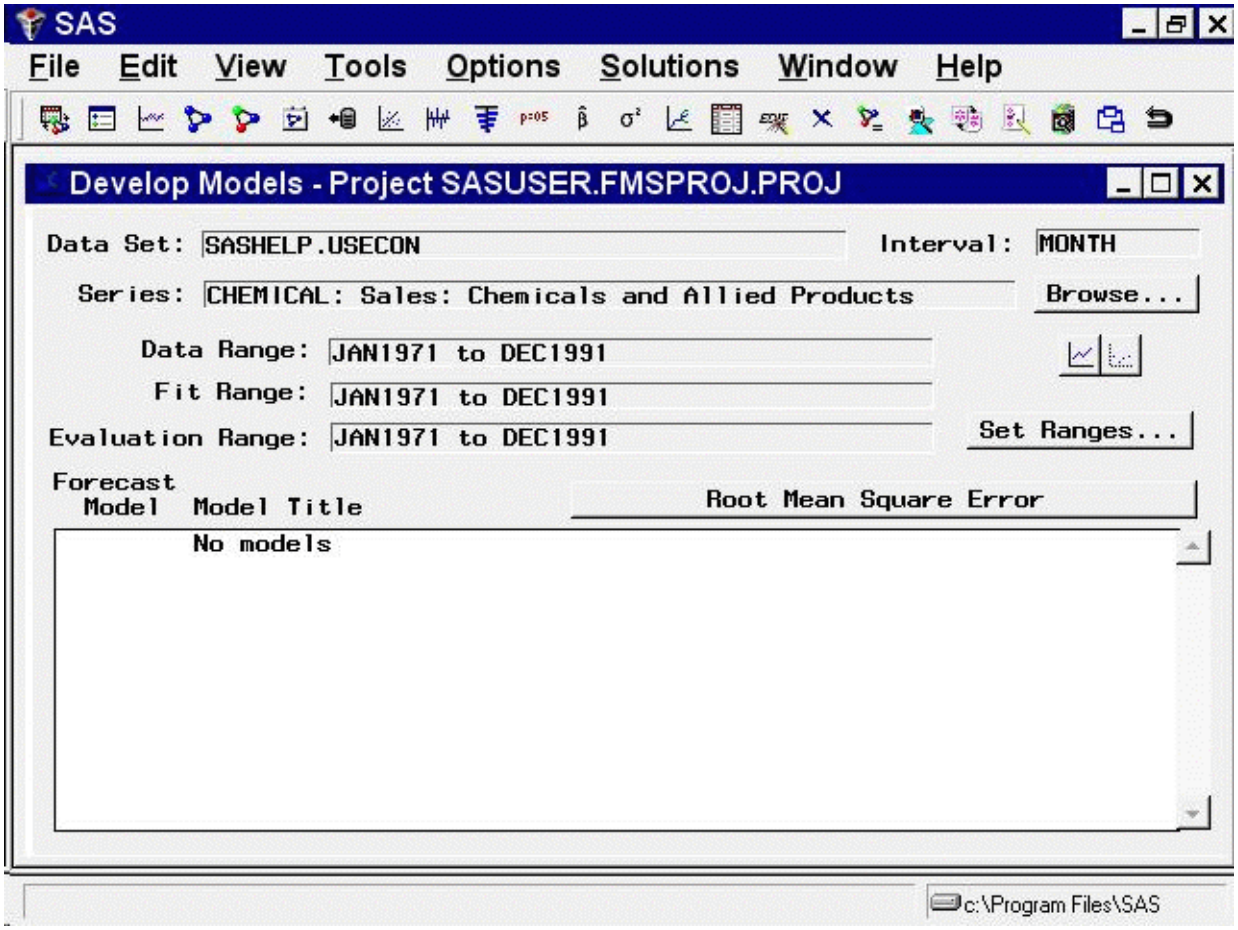
Select OK to close the selection window. Now select the Develop Models button. This opens the Series Selection window, as shown in Figure 54.29. You can enlarge this window for easier viewing of lists of data sets and series.

Figure 54.29 Series Selection Window



Select the series CHEMICAL: Sales of Chemicals and Allied Products, and then select the OK button. This opens the Develop Models window, as shown in Figure 54.30.

Figure 54.30 Develop Models Window



The Data Set, Interval, and Series fields in the upper part of the Develop Models window indicate the series with which you are currently working. You can change the settings of these fields by selecting the Browse button.

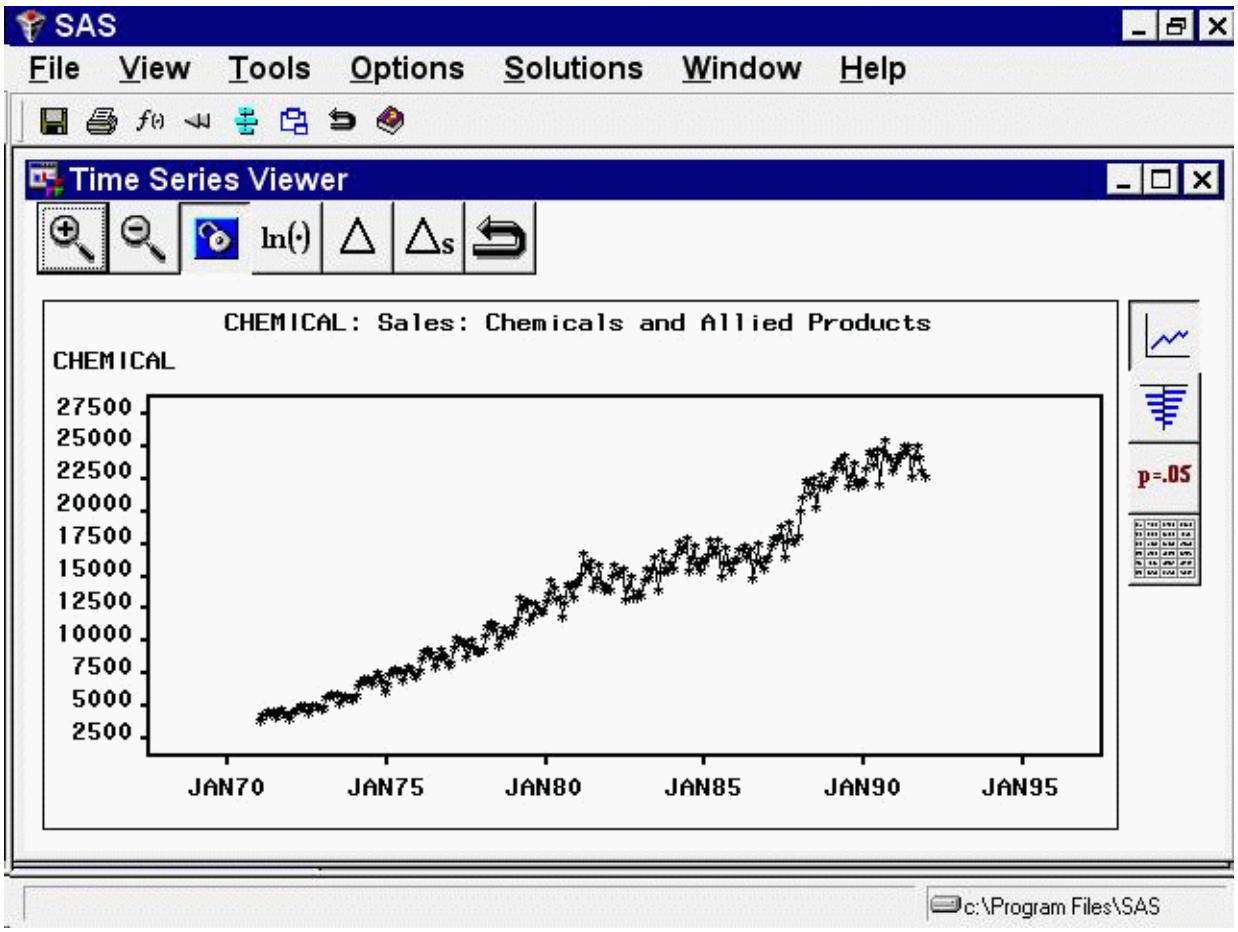
The Data Range, Fit Range, and Evaluation Range fields show the time period over which data are available for the current series, and what parts of that time period are used to fit forecasting models to the series and to evaluate how well the models fit the data. You can change the settings of these fields by selecting the Set Ranges button.

The bottom part of the Develop Models window consists of a table of forecasting models fit to the series. Initially, the list is empty, as indicated by the message "No models." You can fit any number of forecasting models to each series and designate which one you want to use to produce forecasts.

Graphical tools are available for exploring time series and fitted models. The two icons below the Browse button access the Time Series Viewer and the Model Viewer.

Select the left icon. This opens the Time Series Viewer window, as shown in Figure 54.31.

Figure 54.31 Chemical and Allied Product Series



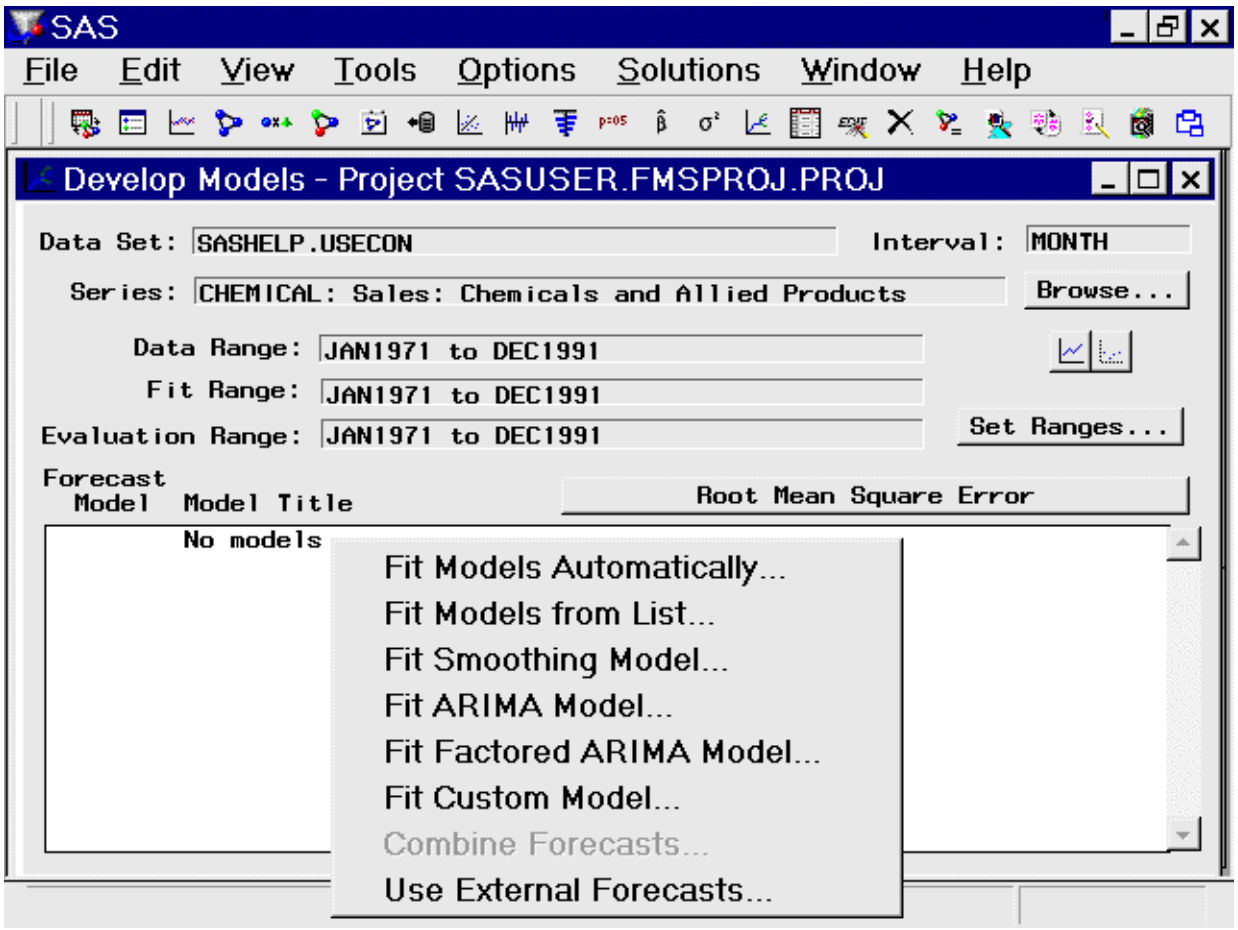
The Time Series Viewer displays a plot of the CHEMICAL series. The Time Series Viewer offers many useful features, which are explored in later sections.

The Time Series Viewer appears in a separate resizable window. You can switch back and forth between the Time Series Viewer window and other windows. For now, return to the Develop Models window. You can close the Time Series Viewer window or leave it open. (To close the Time Series Viewer window, select `Close` from the toolbar or from the File menu.)

Fitting Models

To open a menu of model fitting choices, select `Edit` from the menu bar and then select `Fit Model`, or select `Fit Models from List` in the toolbar, or simply select a blank line in the table as shown in Figure 54.32.

Figure 54.32 Menu of Model Fitting Choices



The Forecasting System provides several ways to specify forecasting models. The eight choices given by the menu shown in Figure 54.32 are as follows:

`Fit Models Automatically`
: performs for the current series the same automatic model selection process that the Automatic Model Fitting window applies to a set of series.

`Fit Models from List`
: presents a list of commonly used forecasting models for convenient point-and-click selection.

`Fit Smoothing Model`
: displays the Smoothing Model Specification window, which enables you to specify several kinds of exponential smoothing and Winters method forecasting models.

`Fit ARIMA Model`
: displays the ARIMA Model Specification window, which enables you to specify many kinds of autoregressive integrated moving average (ARIMA) models, including seasonal ARIMA models and ARIMA models with regressors, transfer functions, and other predictors.

`Fit Factored ARIMA Model`
: displays the Factored ARIMA Model Specification window, which enables you to specify more general ARIMA models, including subset models and models with unusual and/or multiple seasonal cycles. It also supports regressors, transfer functions, and other predictors.

`Fit Custom Model`
: displays the Custom Model Specification window, which enables you to construct a forecasting model by specifying separate options for transforming the data, modeling the trend, modeling seasonality, modeling autocorrelation of the errors, and modeling the effect of regressors and other independent predictors.

`Combine Forecasts`
: displays the Forecast Combination Model Specification window, which enables you to specify models that produce forecasts by combining, or averaging, the forecasts from other models. (This option is not available unless you have fit at least two models.)

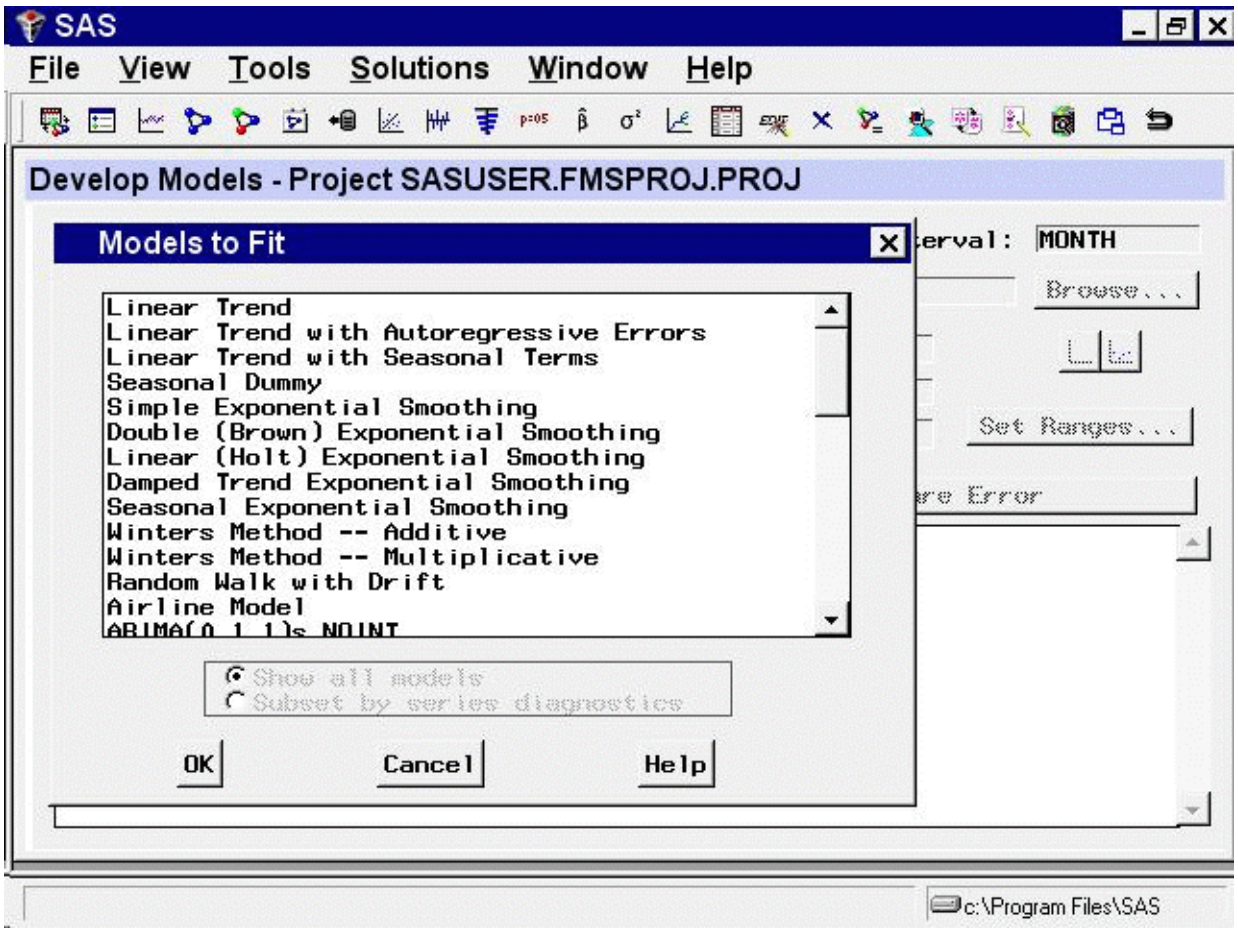
`Use External Forecasts`
: displays the External Forecast Model Specification window, which enables you to use judgmental or externally produced forecasts that have been saved in a separate series in the data set.

All of the forecasting models used by the system are ultimately specified through one of the four windows: Smoothing Method Specification, ARIMA Model Specification, Factored ARIMA Model Specification, or Custom Model Specification. You can specify the same models with either the ARIMA Model Specification window or the Custom Model Specification window, but the Custom Model Specification window can provide a more natural way to specify models for those who are less familiar with the Box-Jenkins style of time series model specification.

The Automatic Model feature, the Models to Fit window, and the Forecast Combination Model Specification window all deal with lists of forecasting models previously defined through the Smoothing Model, ARIMA Model, or Custom Model specification windows. These windows are discussed in detail in later sections.

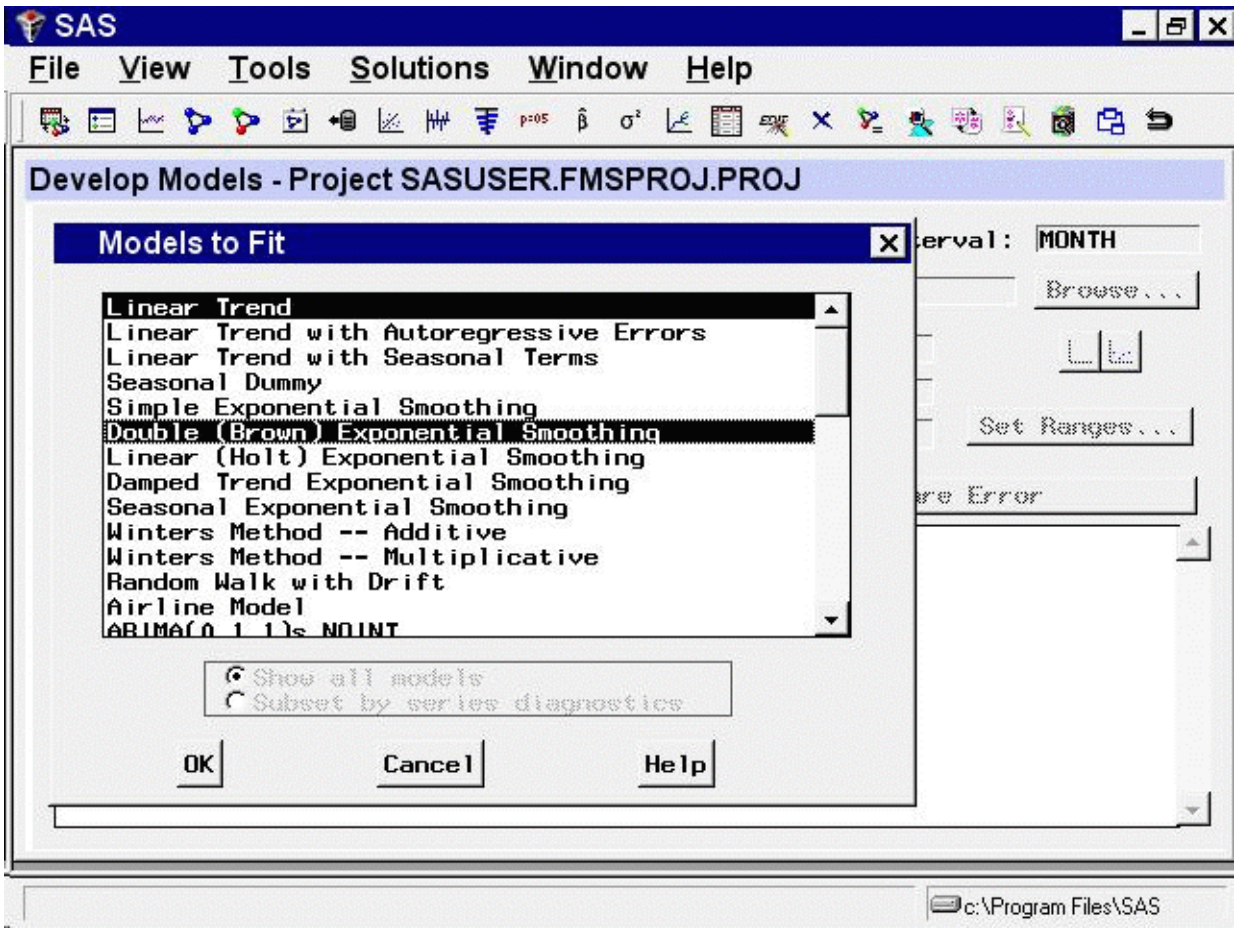
To get started using the Develop Models window, select the Fit Models from List item from the menu shown in Figure 54.32. This opens the Models to Fit window, as shown in Figure 54.33.

Figure 54.33 Models to Fit Window



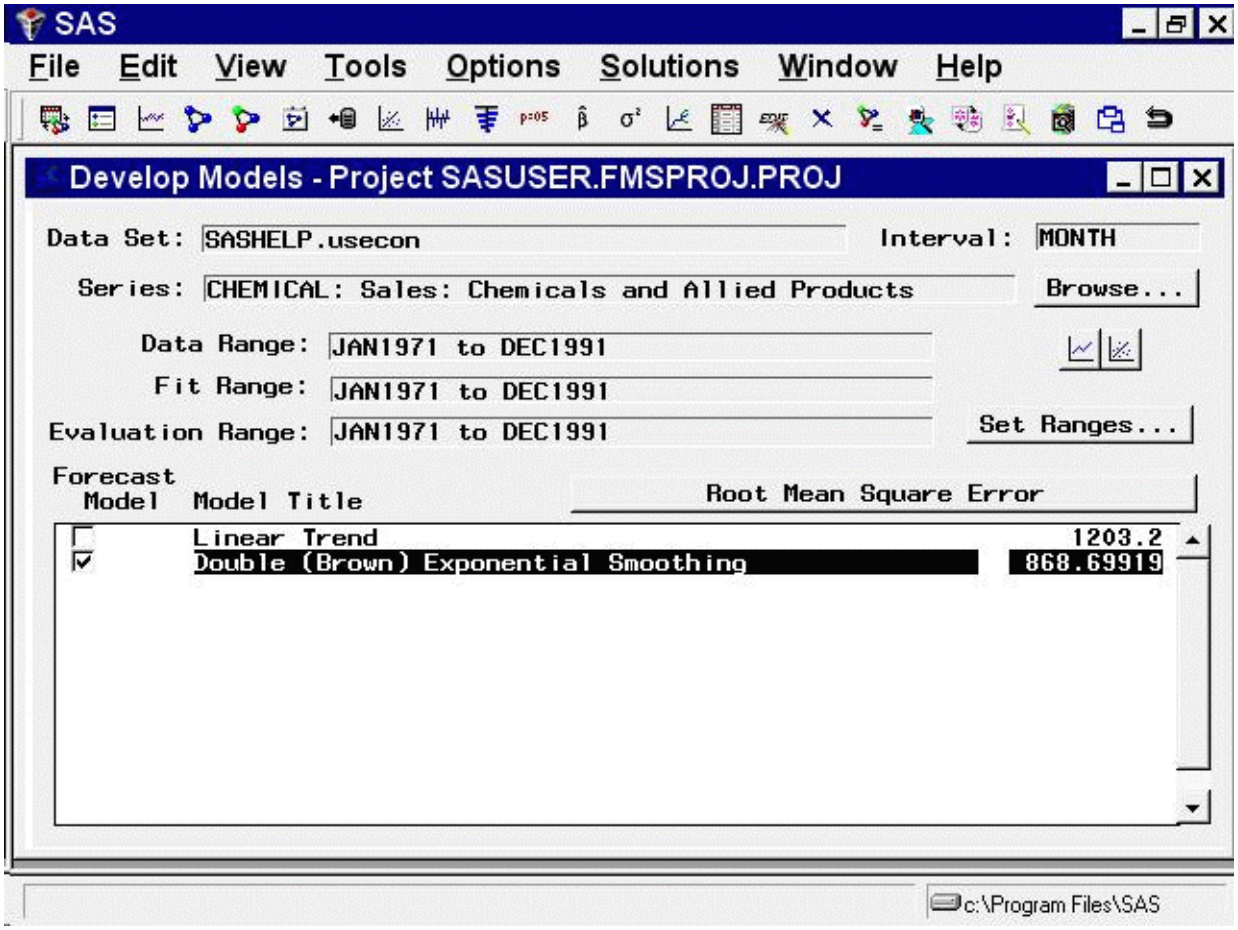
You can select several models to fit at once by holding down the CTRL key as you make the selections. Select `Linear Trend` and `Double (Brown) Exponential Smoothing`, as shown in Figure 54.34, and then select the OK button.

Figure 54.34 Selecting Models to Fit



The system fits the two models you selected. After the models are fit, the labels of the two models and their goodness-of-fit statistic are added to the model table, as shown in Figure 54.35.

Figure 54.35 Fitted Models List



Model List and Statistics of Fit

In the model list, the *Model Title* column shows the descriptive labels for the two fitted models, in this case Linear Trend and Double Exponential Smoothing.

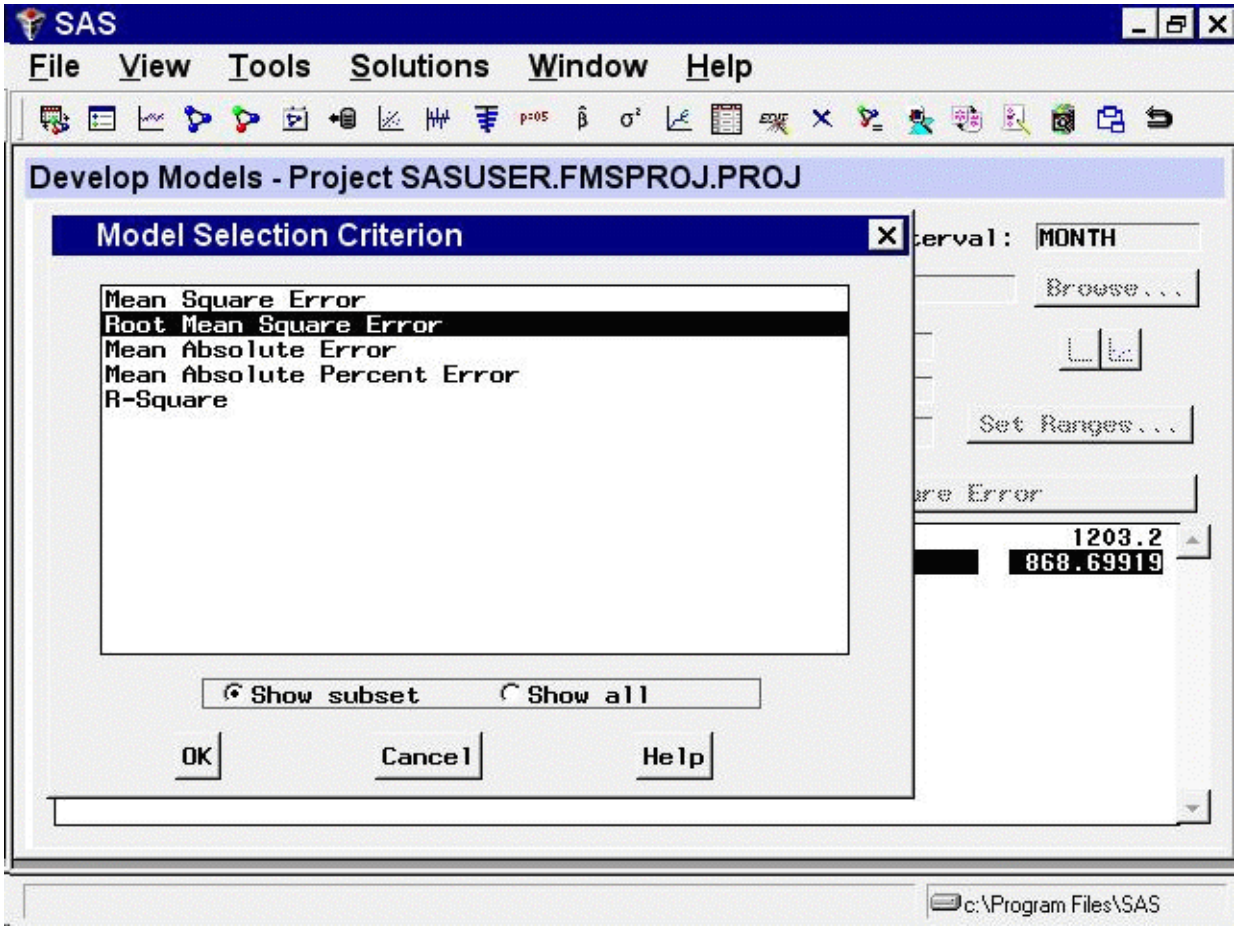
The column labeled *Root Mean Square Error* (or labeled *Mean Absolute Percent Error* if you continued from the example in the previous section) shows the goodness-of-fit criterion used to decide which model fits better. By default, the criterion used is the root mean square error, but you can choose a different measure of fit. The linear trend model has a root mean square error of 1203, while the double exponential smoothing model fits better, with a RMSE of only 869.

The left column labeled *Forecast Model* consists of check boxes that indicate which one of the models in the list has been selected as the model to use to produce the forecasts for the series. When new models are fit and added to the model list, the system sets the Forecast Model flags to designate the one model with the best fit—as measured by the selected goodness-of-fit statistic—as the forecast model. (In the case of ties, the first model with the best fit is selected.)

Because the Double Exponential Smoothing model has the smaller RMSE of the two models in the list, its Forecast Model check box is set. If you would rather produce forecasts by using the Linear Trend model, choose it by selecting the corresponding check box in the Forecast Model column.

To use a different goodness-of-fit criterion, select the button with the current criterion name on it (Root Mean Square Error or Mean Absolute Percent Error). This opens the Model Selection Criterion window, as shown in Figure 54.36.

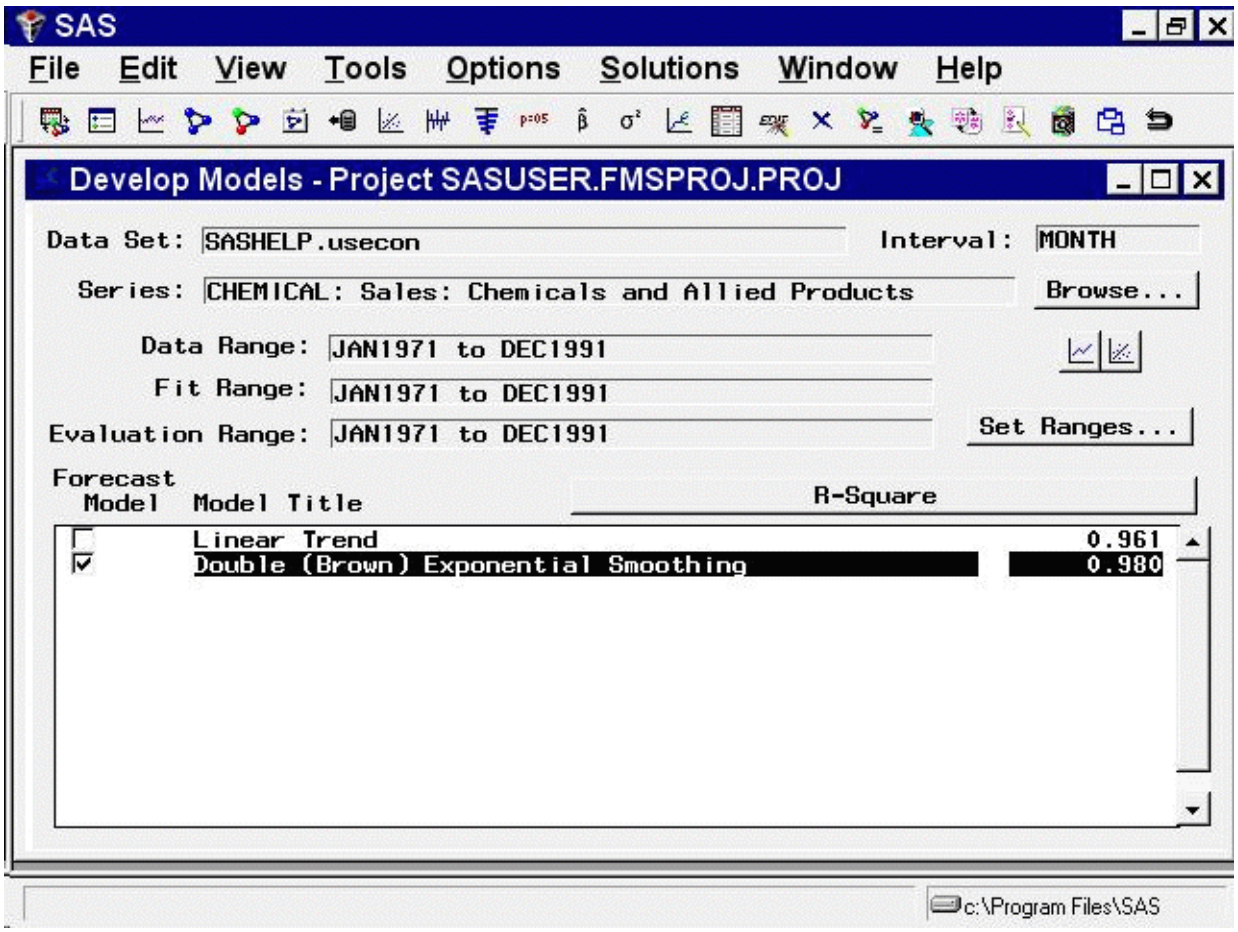
Figure 54.36 Model Selection Criterion Window



The system provides many measures of fit that you can use as the model selection criterion. To avoid confusion, only the most popular of the available fit statistics are shown in this window by default. To display the complete list, you can select the `Show all` option. You can control the subset of statistics listed in this window through the Statistics of Fit item in the Options menu on the Develop Models window.

Initially, Root Mean Square Error is selected. Select R-Square and then select the OK button. This changes the fit statistic displayed in the model list, as shown in Figure 54.37.

Figure 54.37 Model List with R-Square Statistics

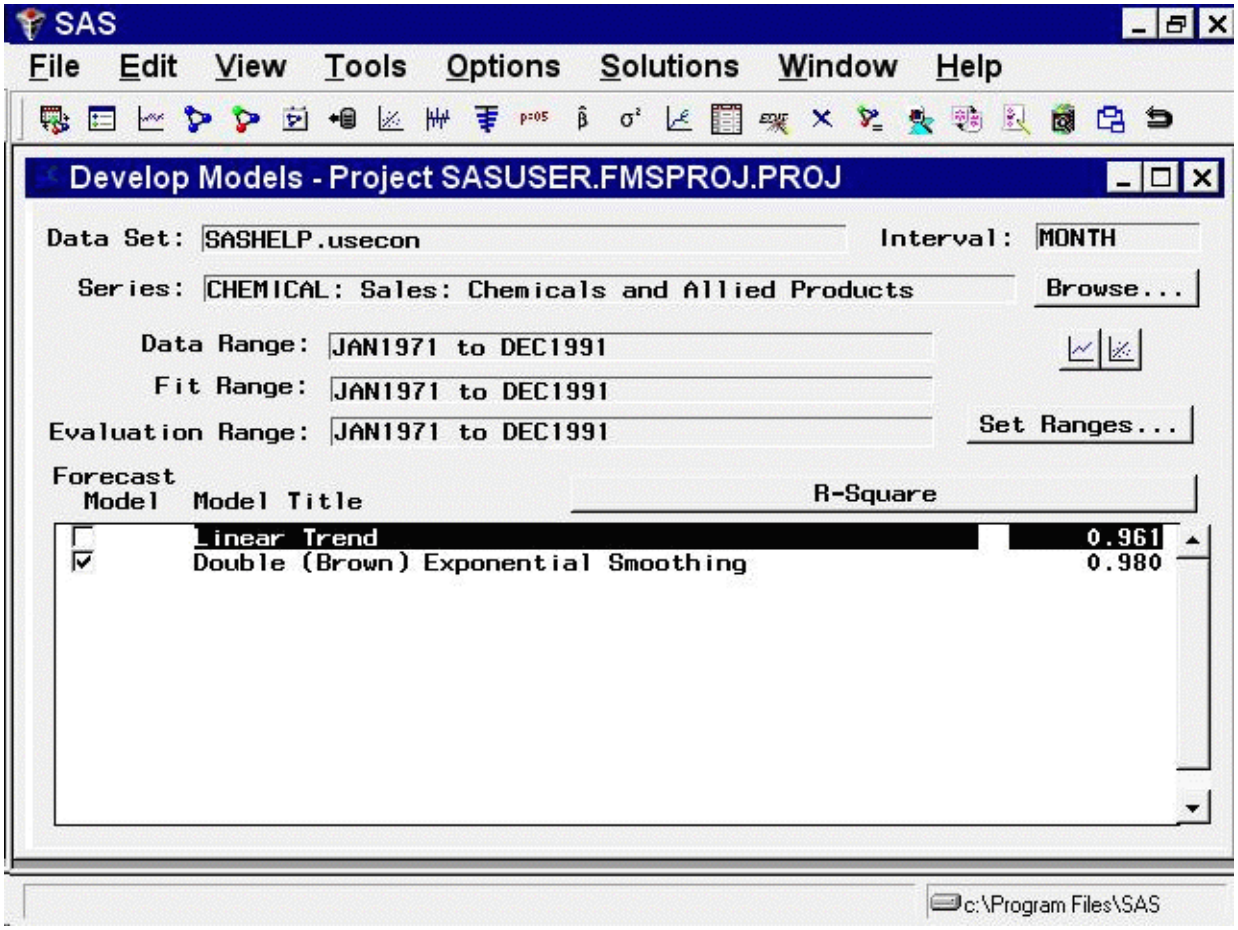


Now that you have fit some models to the series, you can use the Model Viewer button to take a closer look at the predictions of these models.

Model Viewer

In the Develop Models window, select the row in the table containing the Linear Trend model so that this model is highlighted. The model list should now appear as shown in Figure 54.38.

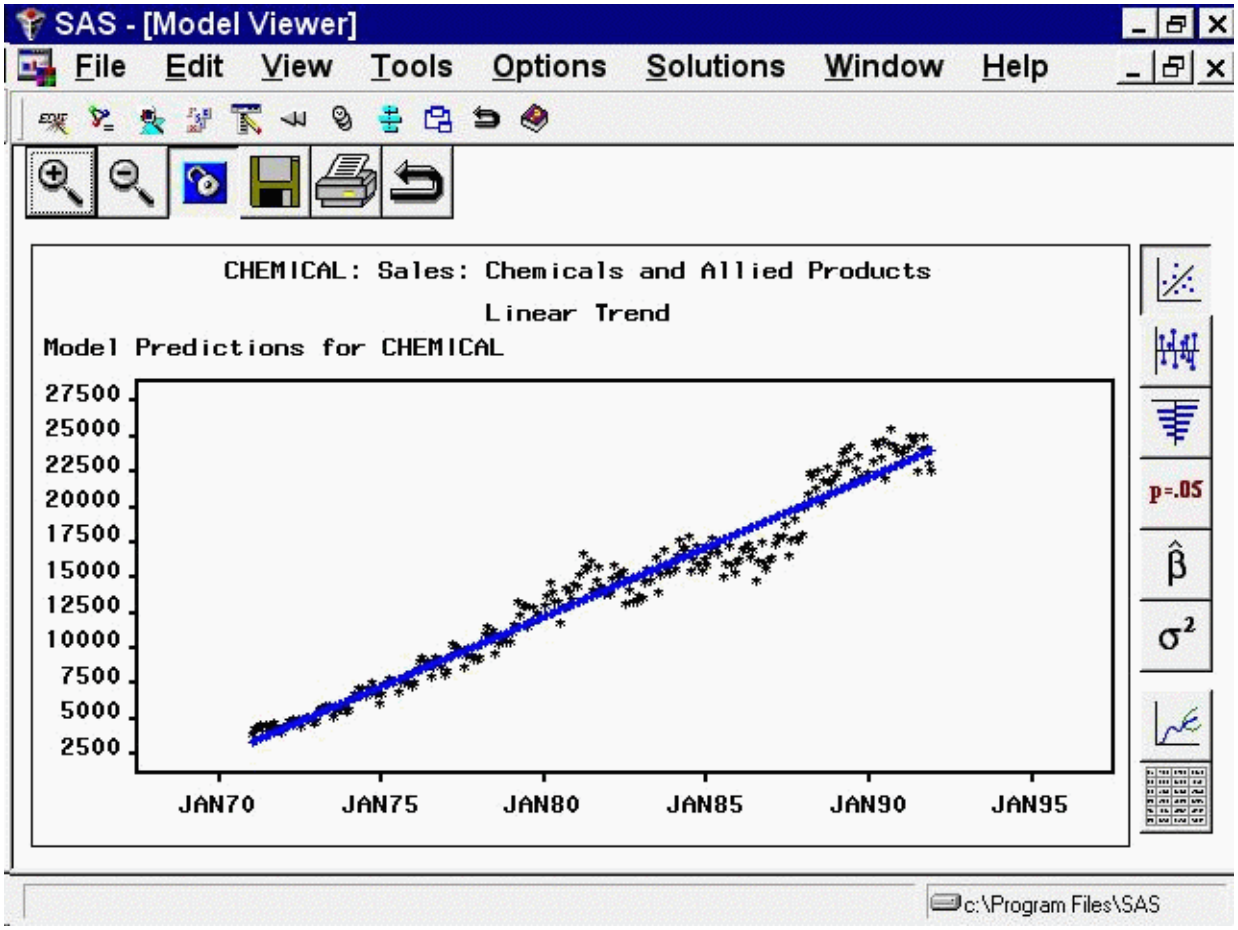
Figure 54.38 Selecting a Model to View



Note that the Linear Trend model is now highlighted, but the Forecast Model column still shows the Double Exponential Smoothing model as the model chosen to produce the final forecasts for the series. Selecting a model in the list means that this is the model that menu items such as `View Model`, `Delete`, `Edit`, and `Refit` will act upon. Choosing a model by selecting its check box in the Forecast Model column means that this model will be used by the Produce Forecasts process to generate forecasts.

Now open the Model Viewer by selecting the right-hand icon under the Browse button, or by selecting `Model Predictions` in the toolbar or from the View menu. The Model Viewer displays the Linear Trend model, as shown in Figure 54.39.

Figure 54.39 Model Viewer: Actual and Predicted Values Plot

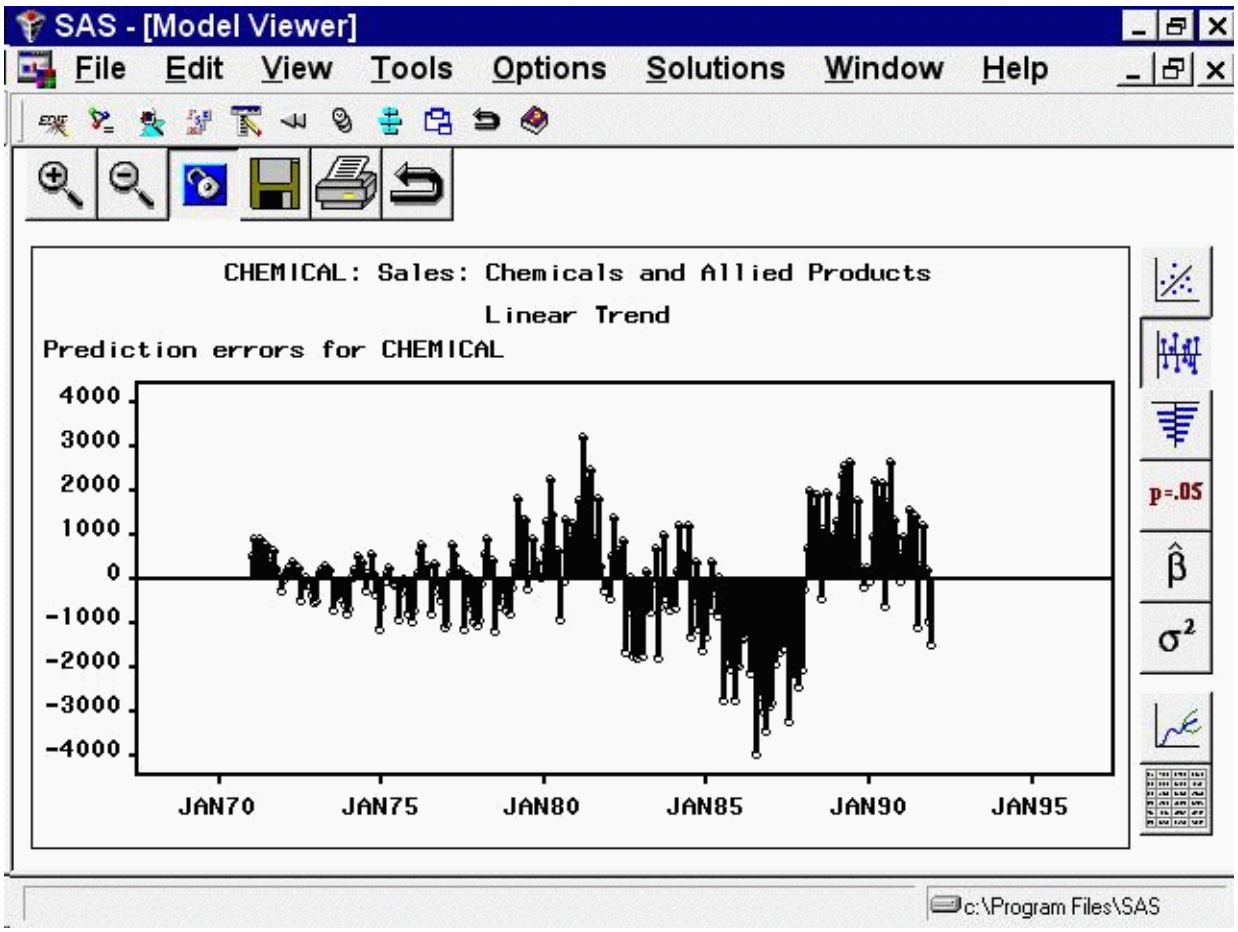


This graph shows the linear trend line representing the model predicted values together with a plot of the actual data values, which fluctuate about the trend line.

Prediction Error Plots

Select the second icon from the top in the vertical toolbar in the Model Viewer window. This switches the Viewer to display a plot of the model prediction errors (actual data values minus the predicted values), as shown in Figure 54.40.

Figure 54.40 Model Viewer: Prediction Errors Plot

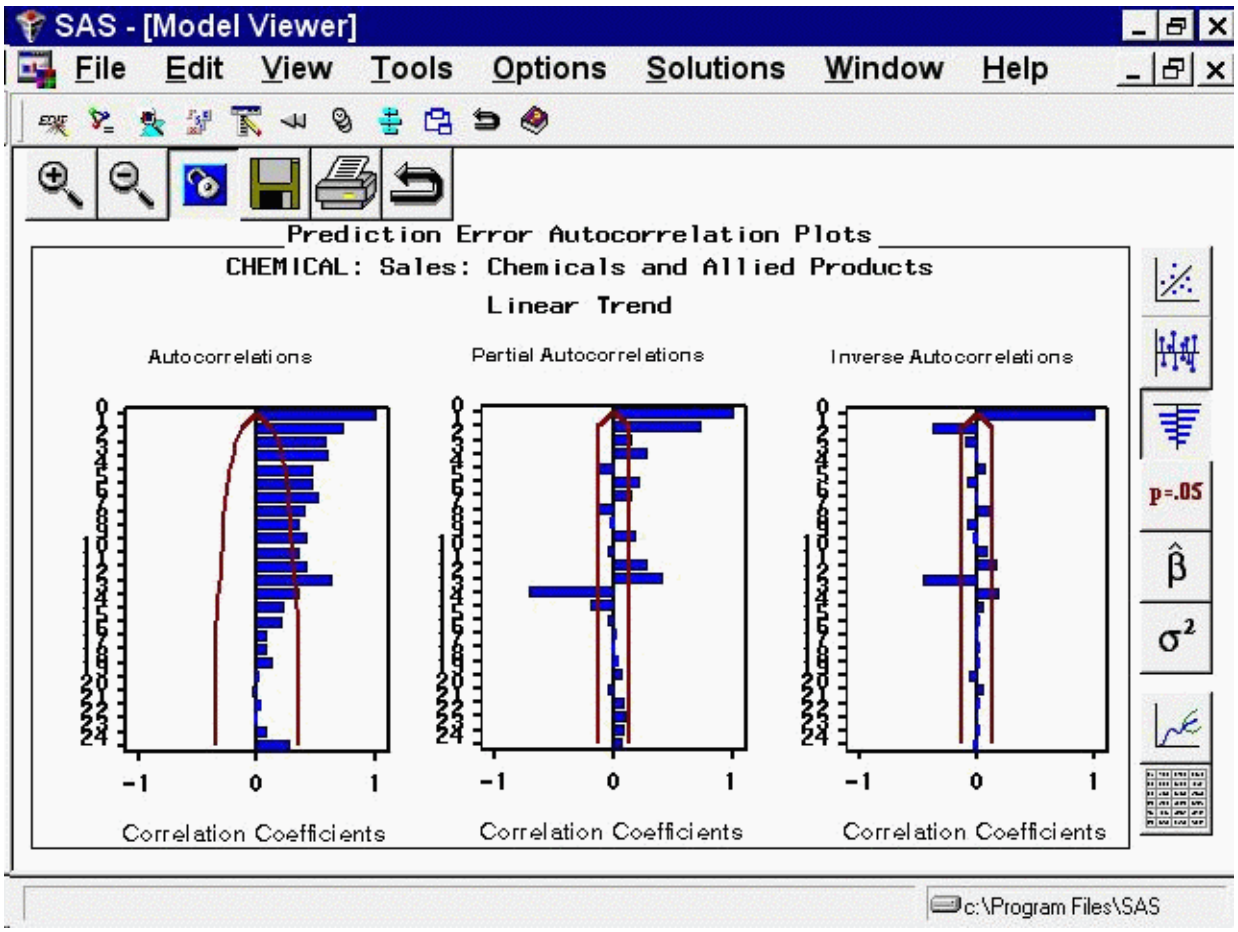


If the model being viewed includes a transformation, prediction errors are defined as the difference between the transformed series actual values and model predictions. You can choose to graph instead the difference between the untransformed series values and untransformed model predictions, which are called *model residuals*. You can also graph normalized prediction errors or normalized model residuals. Use the Residual Plot Options submenu under the Options menu.

Autocorrelation Plots

Select the third icon from the top in the vertical toolbar. This switches the Viewer to display a plot of autocorrelations of the model prediction errors at different lags, as shown in Figure 54.41. Autocorrelations, partial autocorrelations, and inverse autocorrelations are displayed, with lines overlaid at plus and minus two standard errors. You can switch the graphs so that the bars represent significance probabilities by selecting the Correlation Probabilities item on the toolbar or from the View menu. For more information about the meaning and use of autocorrelation plots, see Chapter 8, "The ARIMA Procedure."

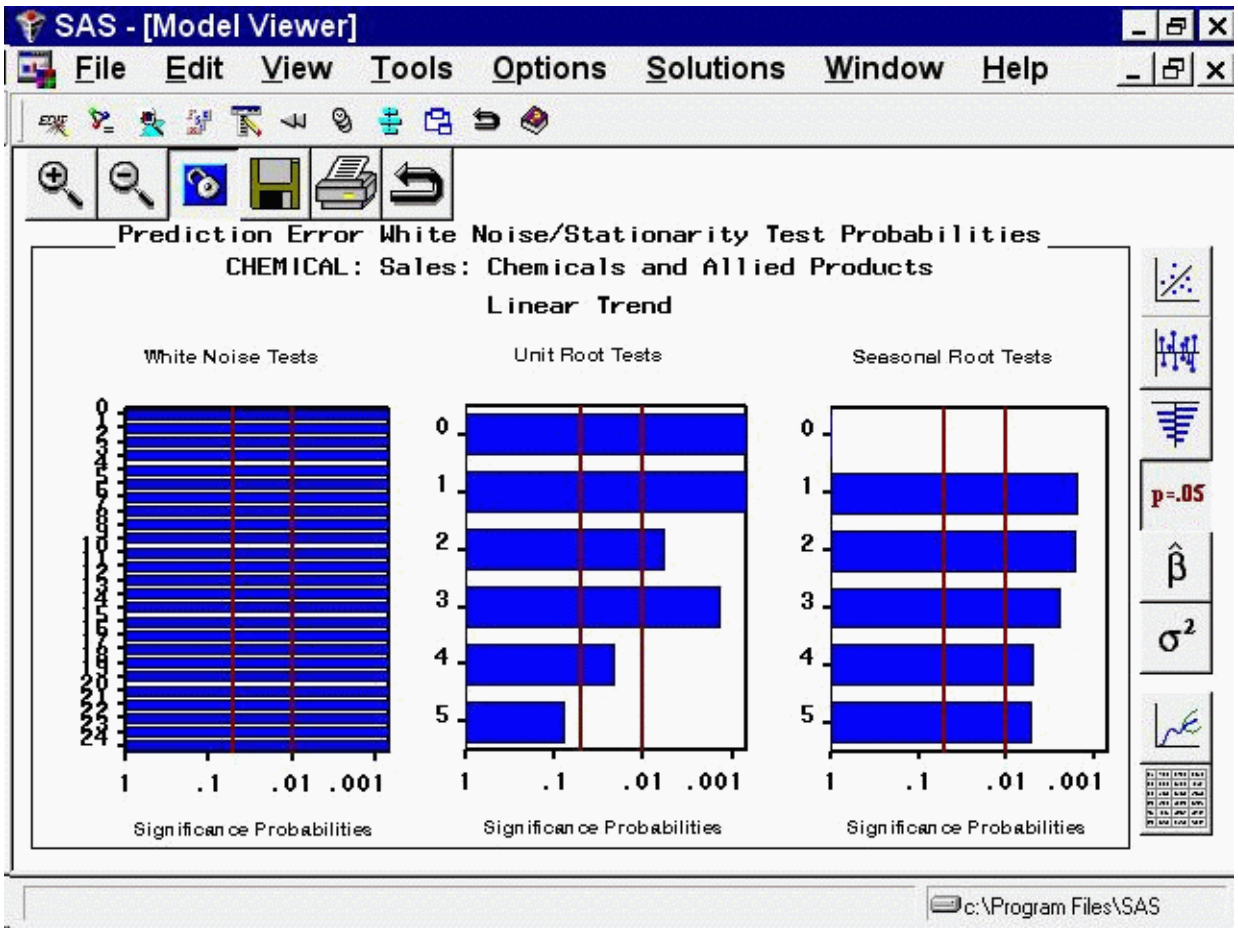
Figure 54.41 Model Viewer: Autocorrelations Plot



White Noise and Stationarity Plots

Select the fourth icon from the top in the vertical toolbar. This switches the Viewer to display a plot of white noise and stationarity tests on the model prediction errors, as shown in Figure 54.42.

Figure 54.42 Model Viewer: White Noise and Stationarity Plot

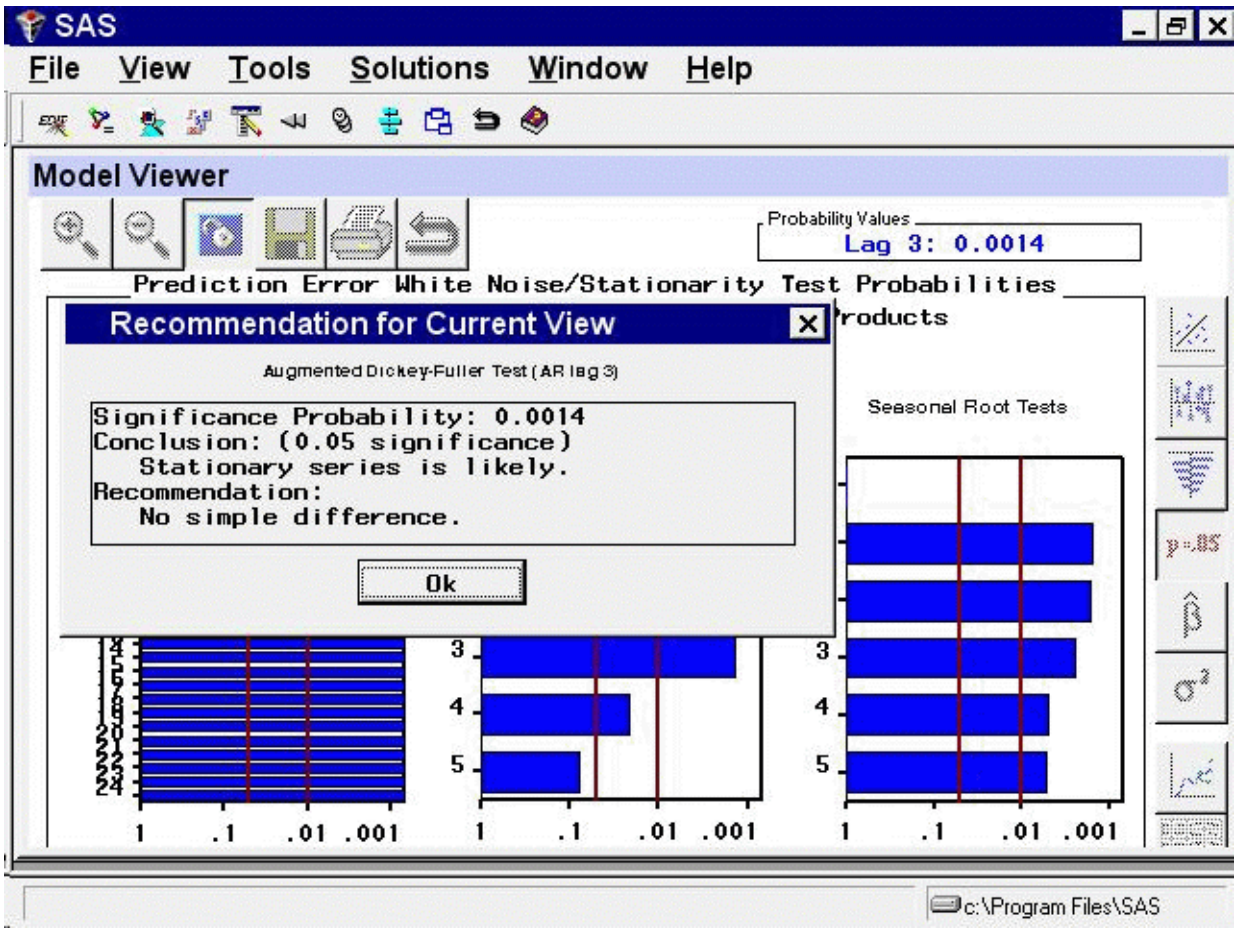


The white noise test bar chart shows significance probabilities of the Ljung-Box chi square statistic. Each bar shows the probability computed on autocorrelations up to the given lag. Longer bars favor rejection of the null hypothesis that the prediction errors represent white noise. In this example, they are all significant beyond the 0.001 probability level, so that you reject the null hypothesis. In other words, the high level of significance at all lags makes it clear that the linear trend model is inadequate for this series.

The second bar chart shows significance probabilities of the augmented Dickey-Fuller test for unit roots. For example, the bar at lag three indicates a probability of 0.0014, so that you reject the null hypothesis that the series is nonstationary. The third bar chart is similar to the second except that it represents the seasonal lags. Since this series has a yearly seasonal cycle, the bars represent yearly intervals.

You can select any of the bars to display an interpretation. Select the fourth bar of the middle chart. This displays the `Recommendation for Current View`, as shown in Figure 54.43. This window gives an interpretation of the test represented by the bar that was selected; it is significant, therefore a stationary series is likely. It also gives a recommendation: You do not need to perform a simple difference to make the series stationary.

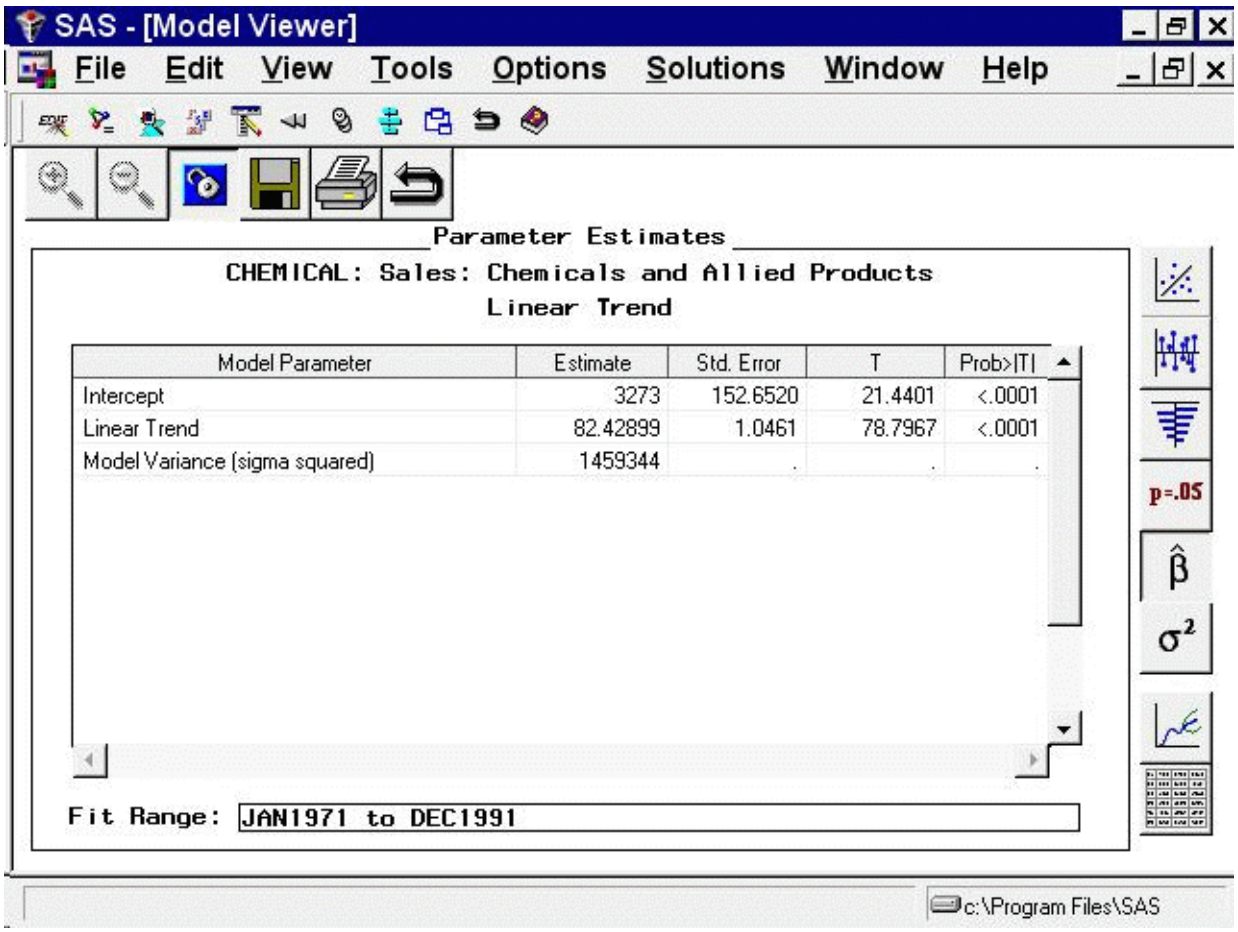
Figure 54.43 Model Viewer: Recommendation for Current View



Parameter Estimates Table

Select the fifth icon from the top in the vertical toolbar to the right of the graph. This switches the Viewer to display a table of parameter estimates for the fitted model, as shown in Figure 54.44.

Figure 54.44 Model Viewer: Parameter Estimates Table

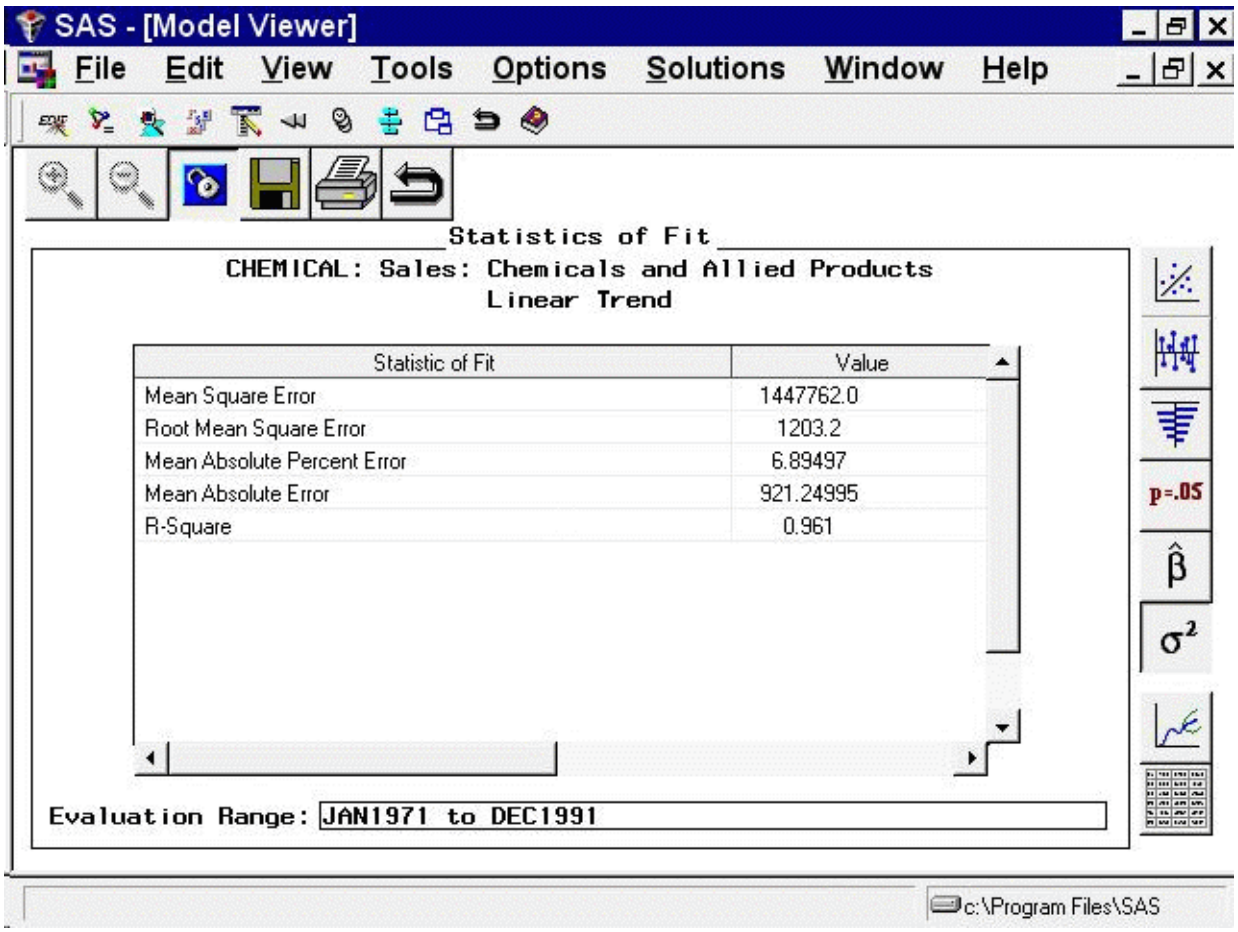


For the linear trend model, the parameters are the intercept and slope coefficients. The table shows the values of the fitted coefficients together with standard errors and *t* tests for the statistical significance of the estimates. The model residual variance is also shown.

Statistics of Fit Table

Select the sixth icon from the top in the vertical toolbar to the right of the table. This switches the Viewer to display a table of statistics of fit computed from the model prediction errors, as shown in Figure 54.45. The list of statistics displayed is controlled by selecting `Statistics of Fit` from the Options menu.

Figure 54.45 Model Viewer: Statistics of Fit Table



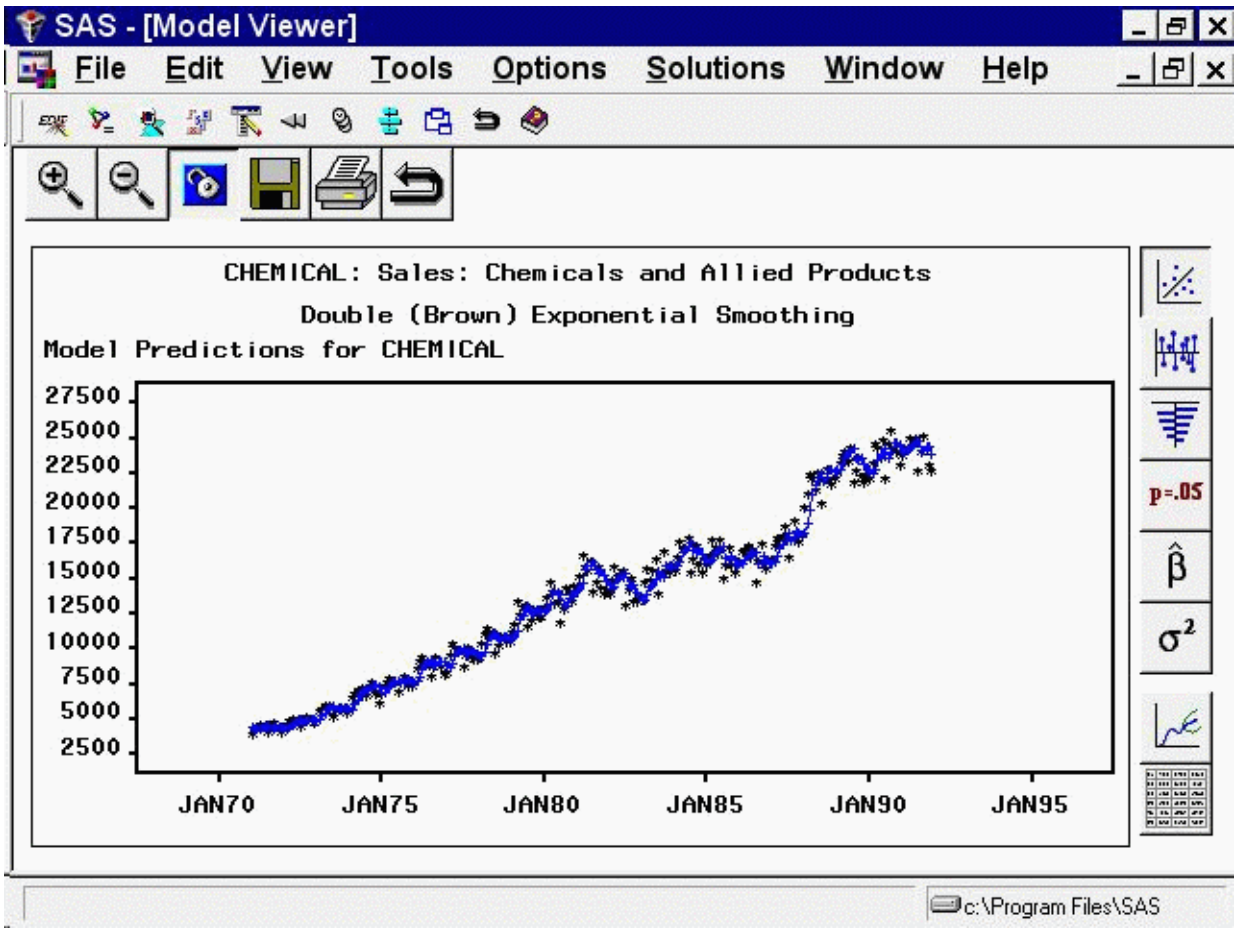
Changing to a Different Model

Select the first icon in the vertical toolbar to the right of the table to return the display to the predicted and actual values plots (Figure 54.39).

Now return to the Develop Models window, but do not close the Model Viewer window. You can use the Next Viewer icon in the toolbar or your system's window manager controls to switch windows. You can resize the windows to make them both visible.

Select the Double Exponential Smoothing model so that this line of the model list is highlighted. The Model Viewer window is now updated to display a plot of the predicted values for the Double Exponential Smoothing model, as shown in Figure 54.46. The Model Viewer is automatically updated to display the currently selected model, unless you specify `Unlink` (the third icon in the window's horizontal toolbar).

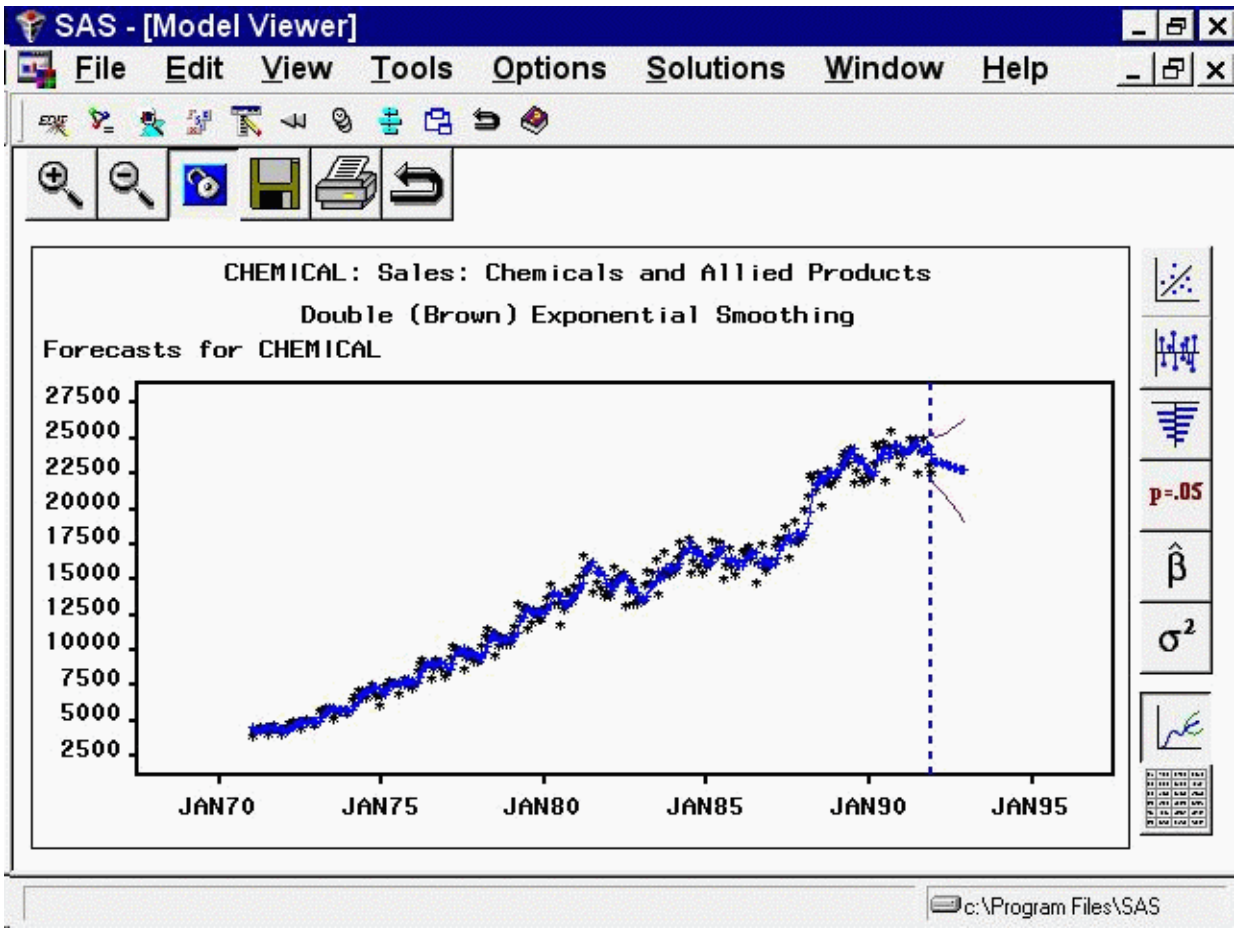
Figure 54.46 Model Viewer Plot for Exponential Smoothing Model



Forecasts and Confidence Limits Plots

Select the seventh icon from the top in the vertical toolbar to the right of the graph. This switches the Viewer to display a plot of forecast values and confidence limits, together with actual values and one-step-ahead within-sample predictions, as shown in Figure 54.47.

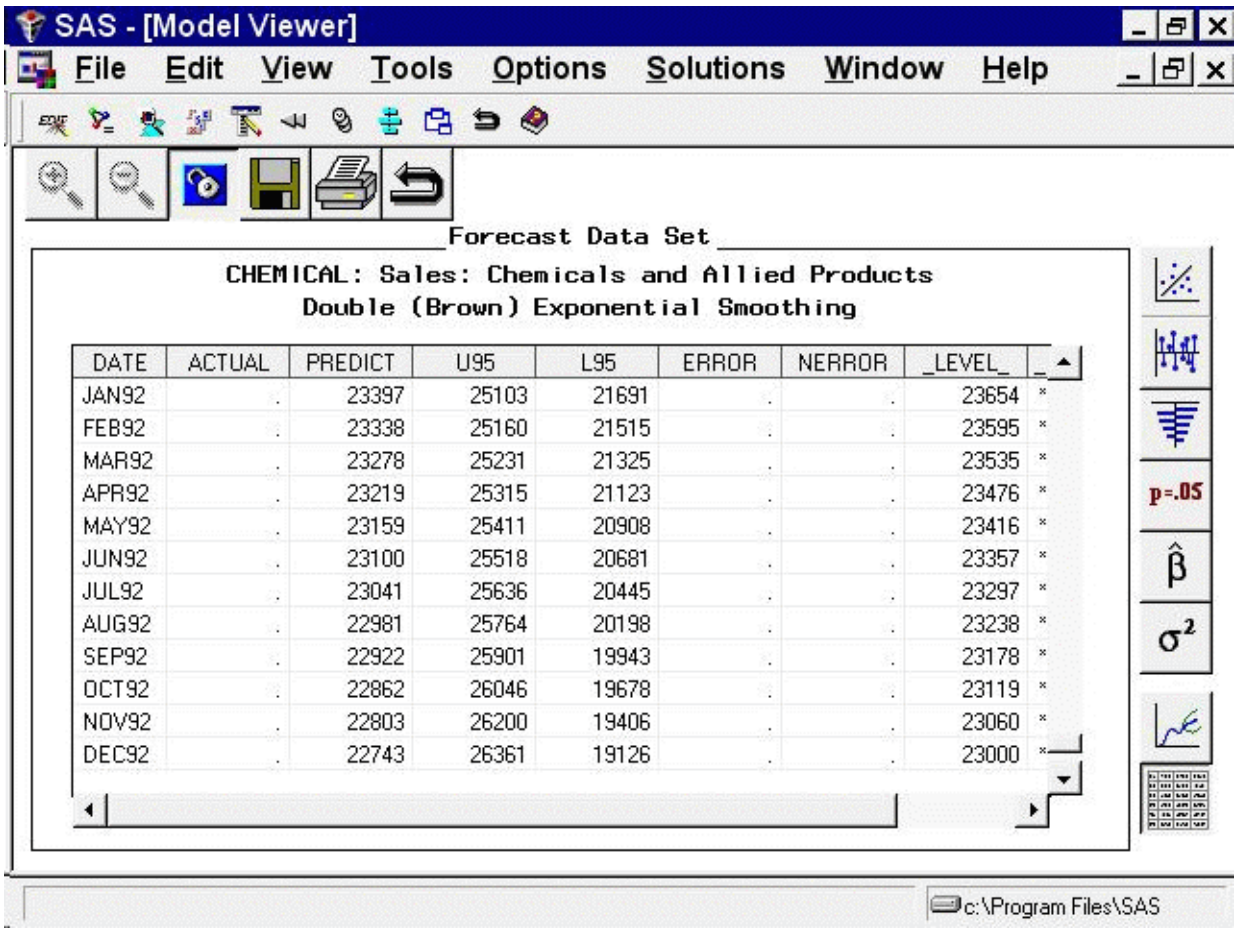
Figure 54.47 Model Viewer: Forecasts and Confidence Limits



Data Table

Select the last icon at the bottom of the vertical toolbar to the right of the graph. This switches the Viewer to display the forecast data set as a table, as shown in Figure 54.48.

Figure 54.48 Model Viewer: Forecast Data Table



To view the full data set, use the vertical and horizontal scroll bars on the data table or enlarge the window.

Closing the Model Viewer

Other features of the Model Viewer and Develop Models window are discussed later in this book. For now, close the Model Viewer window and return to the Time Series Forecasting window.

To close the Model Viewer window, select `Close` from the window's horizontal toolbar or from the File menu.

Chapter 55
Creating Time ID Variables

Contents

Creating a Time ID Value from a Starting Date and Frequency 3667
Using Observation Numbers as the Time ID 3671
Creating a Time ID from Other Dating Variables 3675

The Forecasting System requires that the input data set contain a time ID variable. If the data you want to forecast are not in this form, you can use features of the Forecasting System to help you add time ID variables to your data set. This chapter shows examples of how to use these features.

Creating a Time ID Value from a Starting Date and Frequency

As a first example of adding a time ID variable, use the SAS data set created by the following statements. (Or use your own data set if you prefer.)

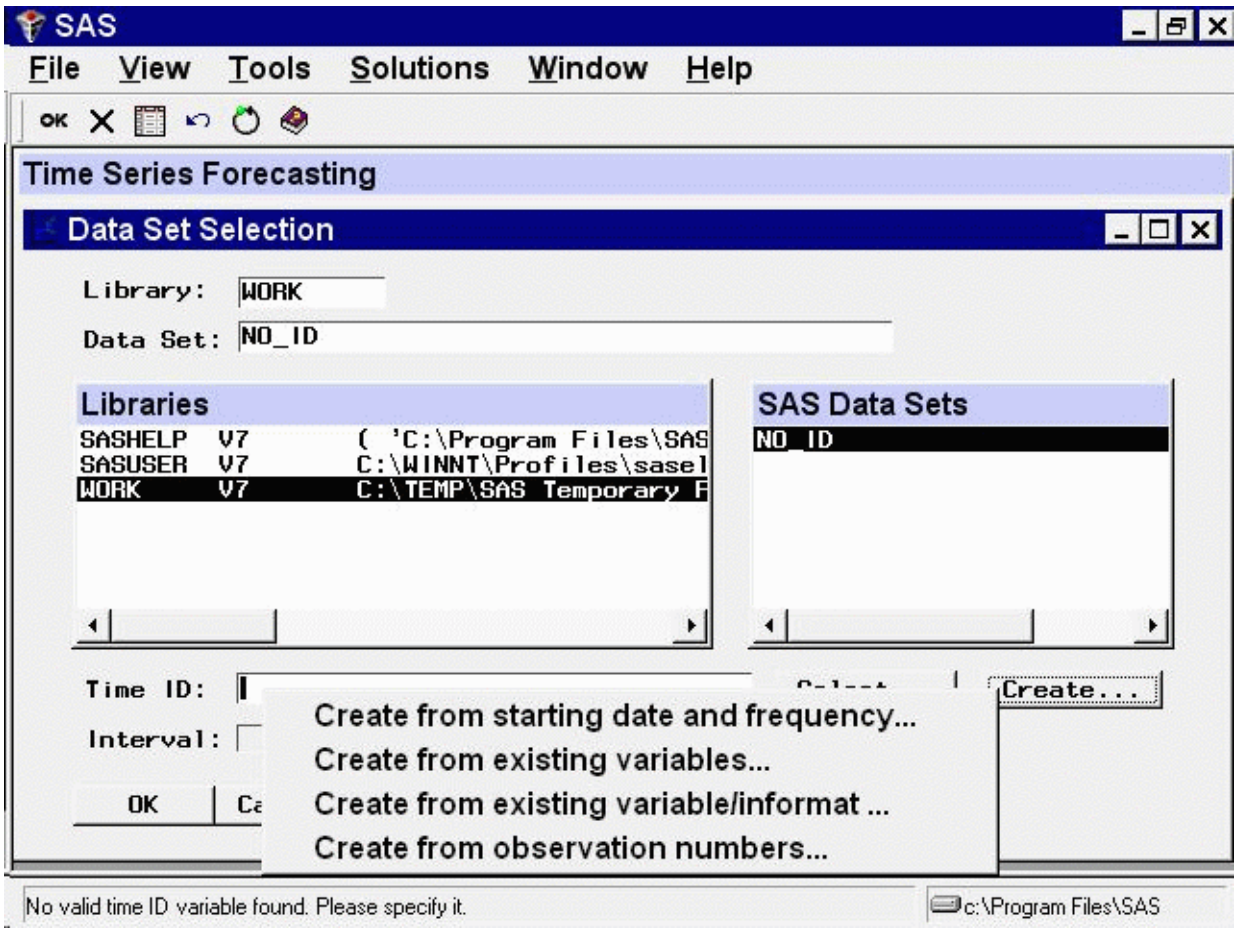
```
data no_id;
   input y @@;
datalines;
  10 15 20 25 30 35 40 45
  50 55 60 65 70 75 80 85
run;
```

Submit these SAS statements to create the data set NO_ID. This data set contains the single variable Y. Assume that Y is a quarterly series and starts in the first quarter of 1991.

In the `Time Series Forecasting` window, use the Browse button to the right of the `Data set` field to bring up the `Data Set Selection` window. Select the WORK library, and then select the NO_ID data set.

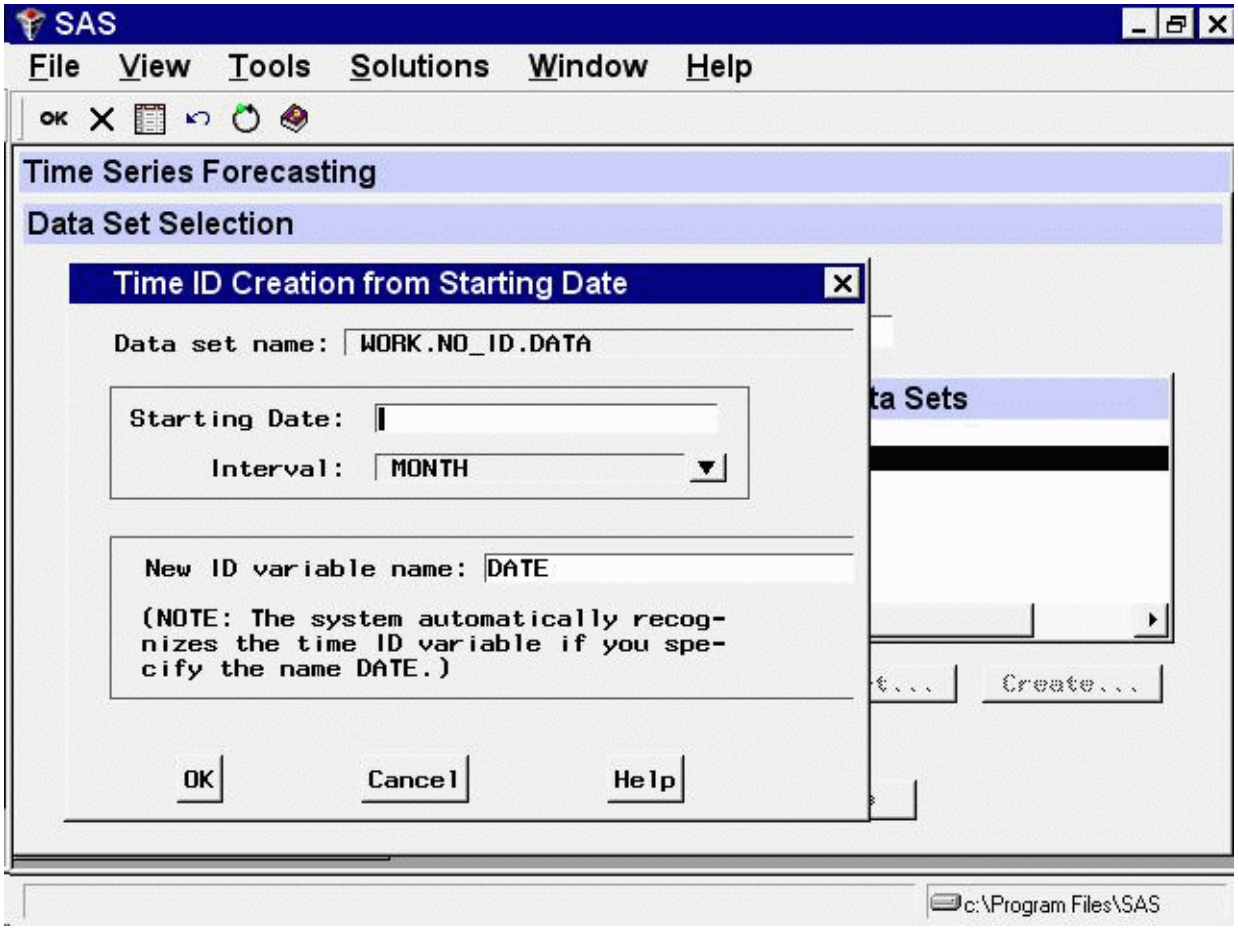
You must create a time ID variable for the data set. Click the Create button to the right of the Time ID field. This opens a menu of choices for creating the Time ID variable, as shown in Figure 55.1.

Figure 55.1 Time ID Creation Popup Menu



Select the first choice, `Create from starting date and frequency`. This opens the `Time ID Creation from Starting Date` window shown in Figure 55.2.

Figure 55.2 Time ID Creation from Starting Date Window



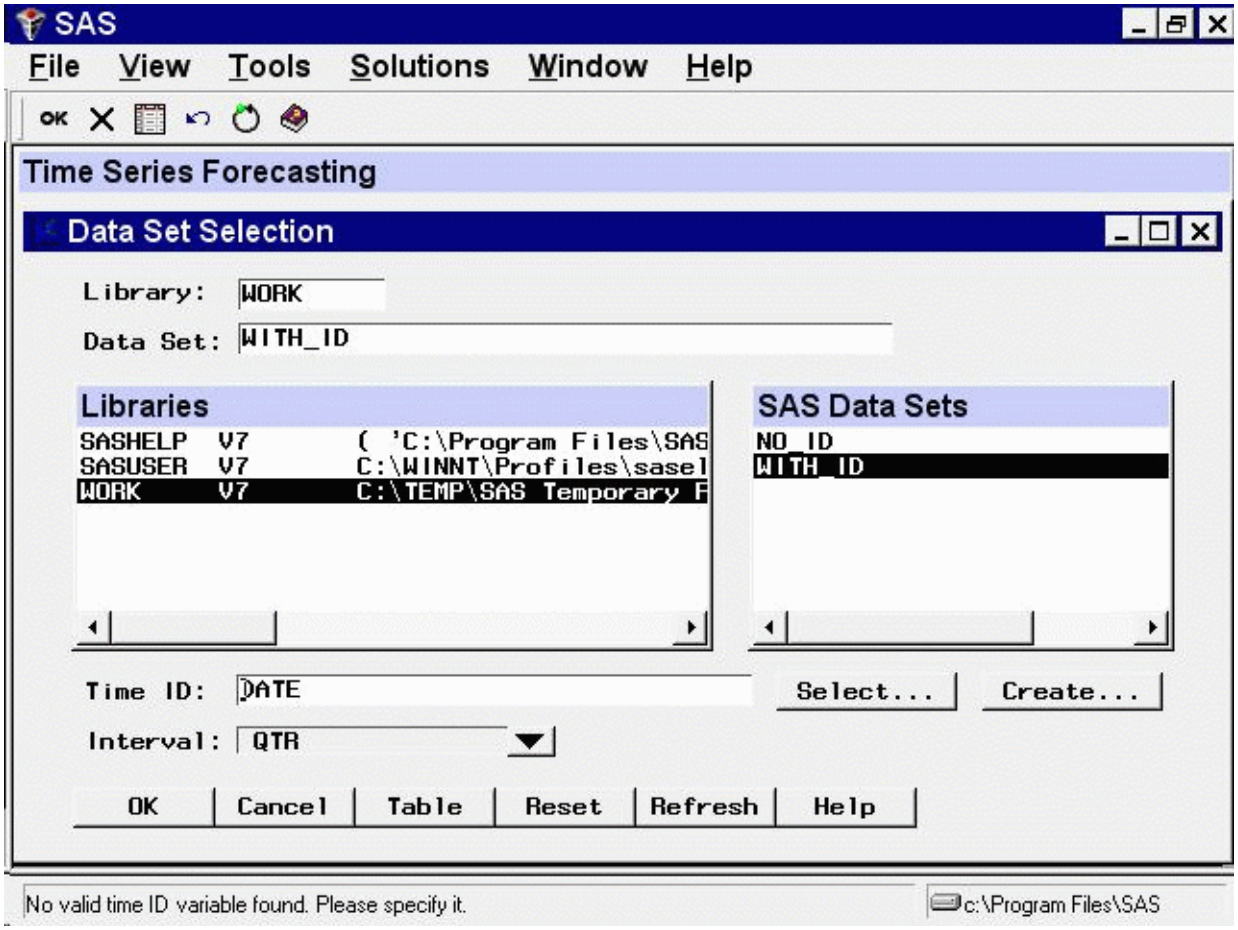
Enter the starting date, 1991:1, in the `Starting Date` field.

Select the `Interval` list arrow and select QTR. The Interval value QTR means that the time interval between successive observations is a quarter of a year; that is, the data frequency is quarterly.

Now select the `OK` button. The system prompts you for the name of the new data set. If you want to create a new copy of the input data set with the DATE variable added, enter a name for the new data set. If you want to replace the NO_ID data set with the new copy containing DATE, just select the `OK` button without changing the name.

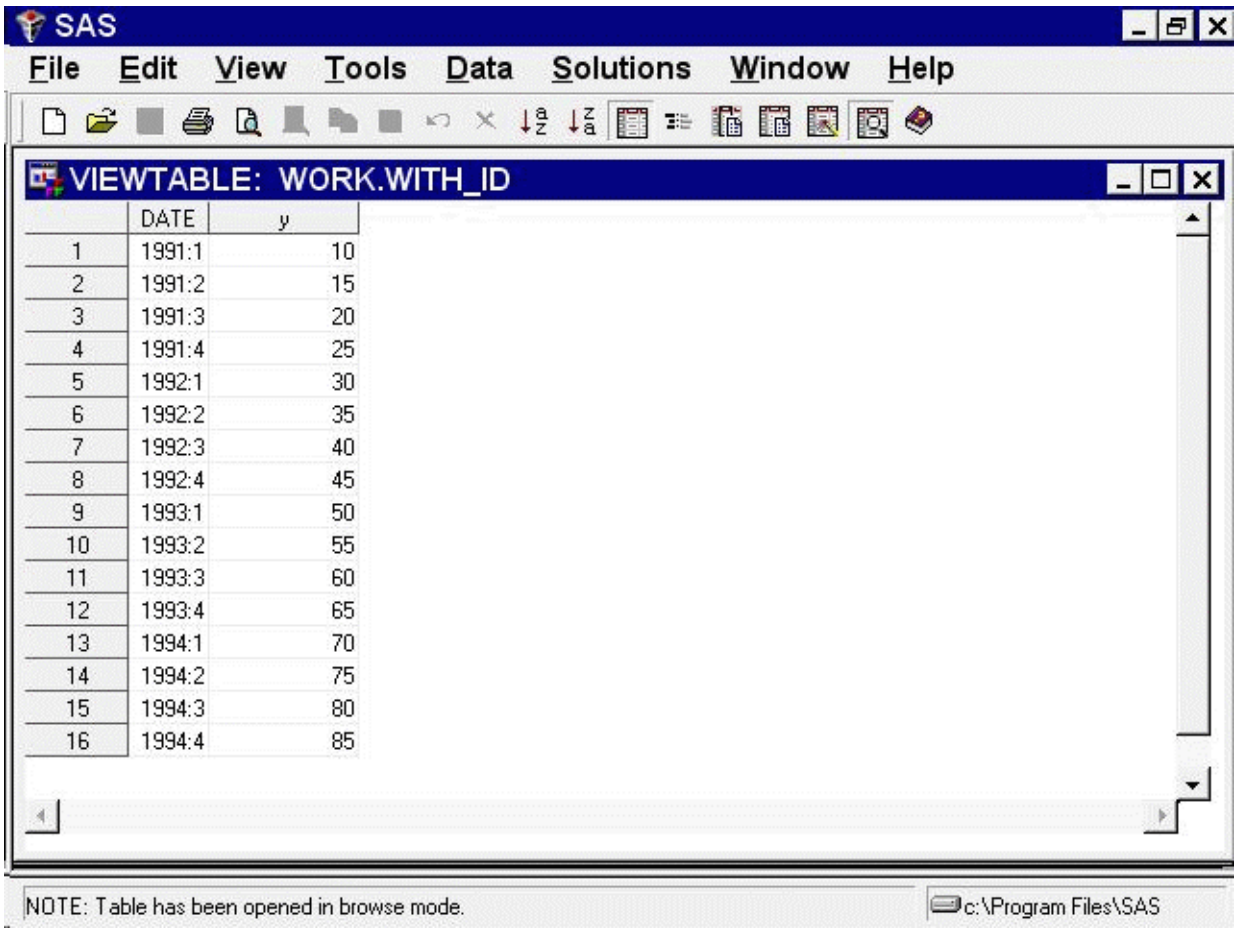
For this example, change the `New name` field to WITH_ID and select the `OK` button. The data set WITH_ID is created containing the series Y from NO_ID and the added ID variable DATE. The system returns to the `Data Set Selection` window, which now appears as shown in Figure 55.3.

Figure 55.3 Data Set Selection Window after Creating Time ID



Select the `Table` button to see the new data set WITH_ID. This opens a VIEWTABLE window for the data set WITH_ID, as shown in Figure 55.4. Select `File` and `Close` to close the VIEWTABLE window.

Figure 55.4 Viewtable Display of Data Set with Time ID Added



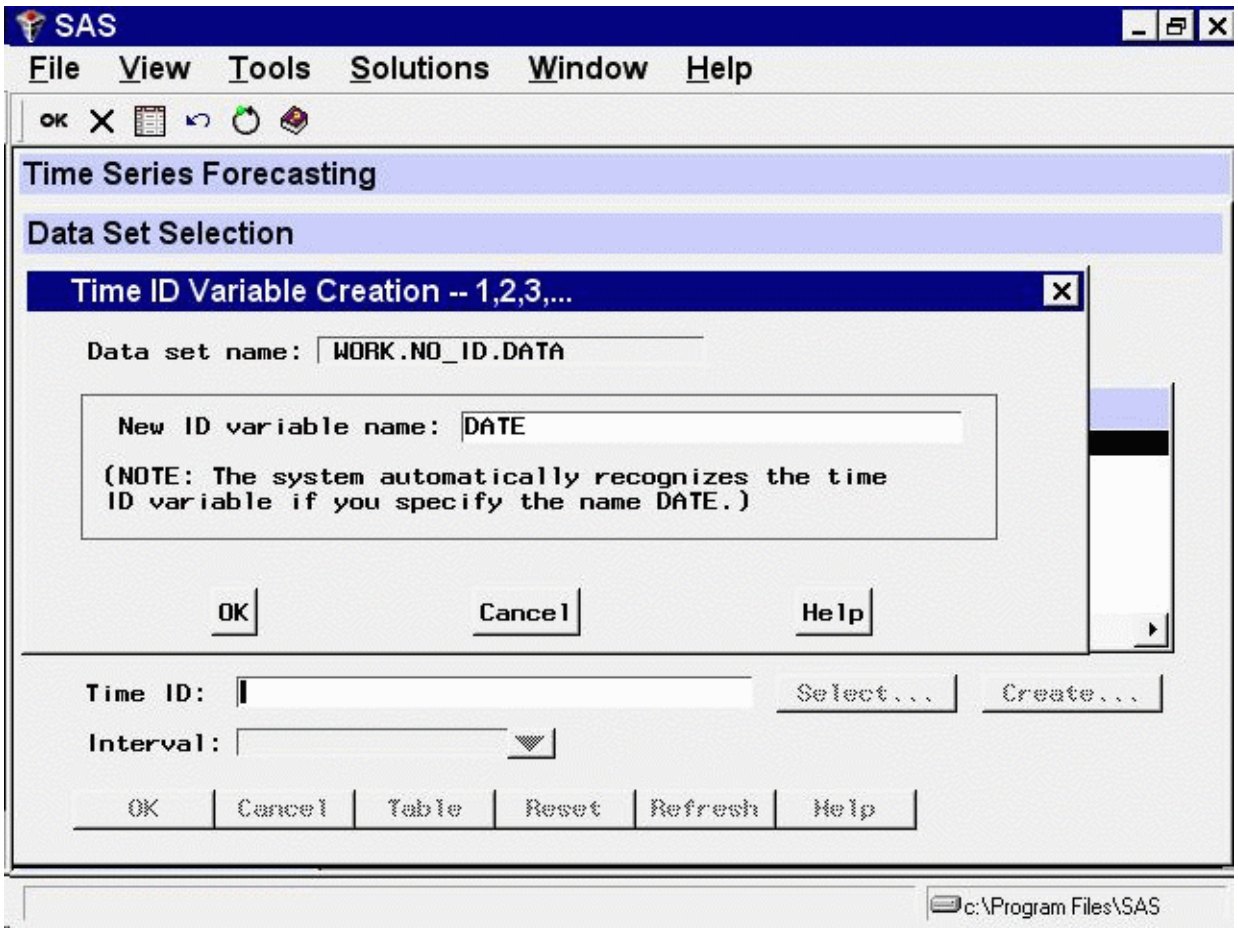
	DATE	y
1	1991:1	10
2	1991:2	15
3	1991:3	20
4	1991:4	25
5	1992:1	30
6	1992:2	35
7	1992:3	40
8	1992:4	45
9	1993:1	50
10	1993:2	55
11	1993:3	60
12	1993:4	65
13	1994:1	70
14	1994:2	75
15	1994:3	80
16	1994:4	85

Using Observation Numbers as the Time ID

Normally, the time ID variable contains date values. If you do not want to have dates associated with your forecasts, you can also use observation numbers as time ID variables. However, you still must have an ID variable. This can be illustrated by adding an observation index time ID variable to the data set NO_ID.

In the Data Set Selection window, select the data set NO_ID again. Select the Create button to the right of the `Time ID` field. Select the fourth choice, `Create from observation numbers`. This opens the `Time ID Variable Creation` window shown in Figure 55.5.

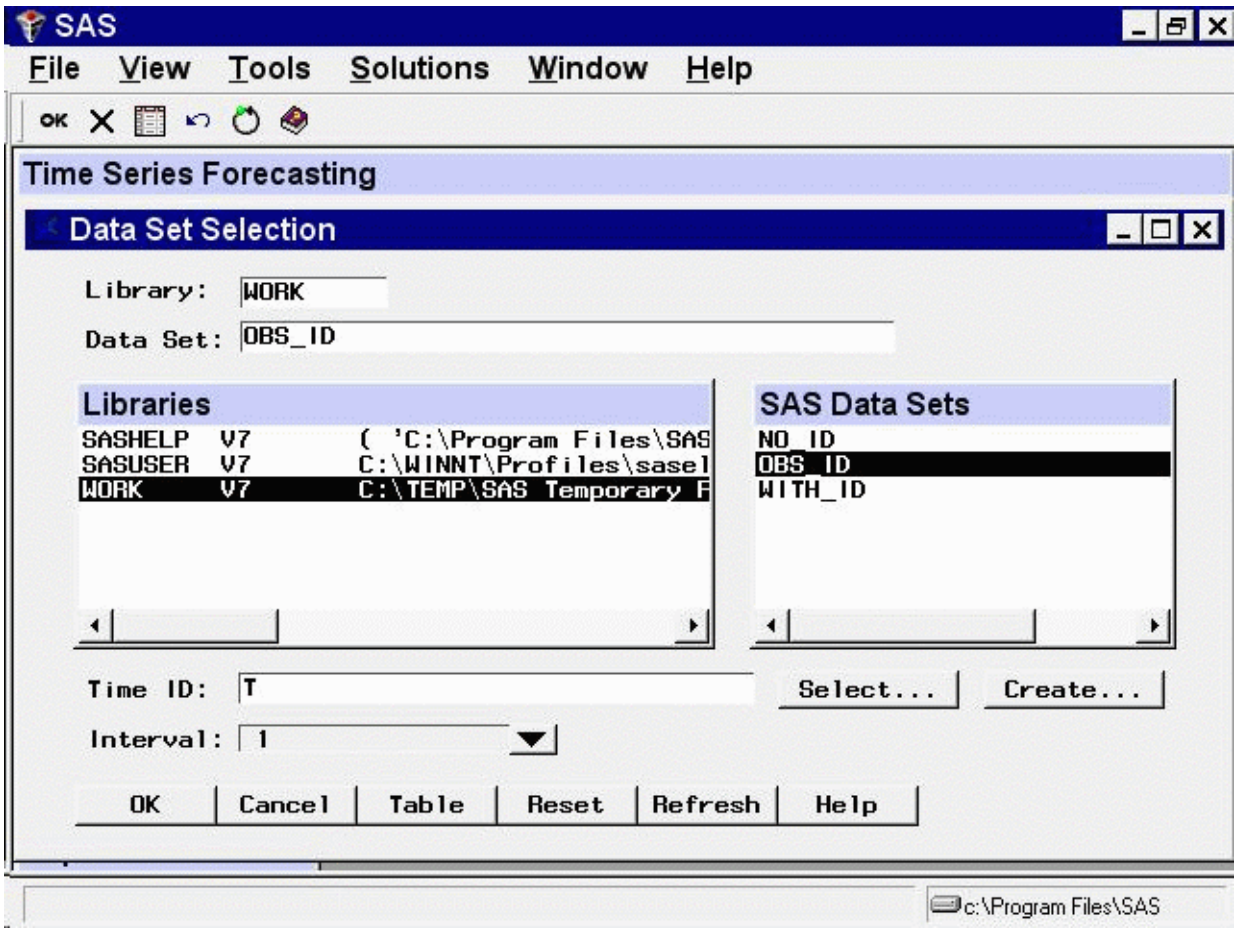
Figure 55.5 Create Time ID Variable Window



Select the `OK` button. This opens the `New Data Set Name` window. Enter "OBS_ID" in the `New data set name` field. Enter "T" in the `New ID variable name` field.

Now select the `OK` button. The new data set OBS_ID is created, and the system returns to the `Data Set Selection` window, which now appears as shown in Figure 55.6.

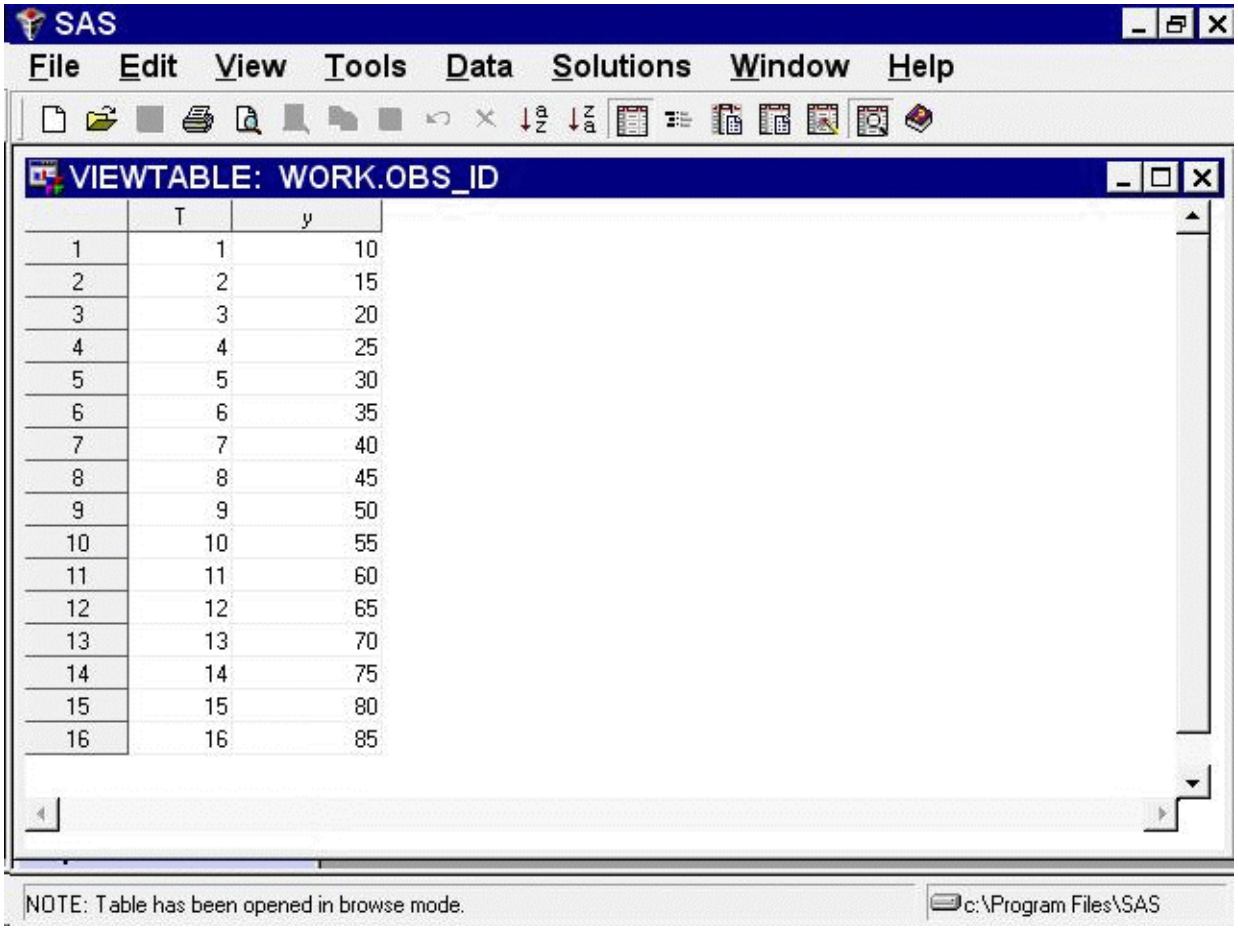
Figure 55.6 Data Set Selection Window after Creating Time ID



The `Interval` field for OBS_ID has the value '1'. This means that the values of the time ID variable T increment by one between successive observations.

Select the `Table` button to look at the OBS_ID data set, as shown in Figure 55.7.

Figure 55.7 VIEWTABLE of Data Set with Observation Index ID



	T	y
1	1	10
2	2	15
3	3	20
4	4	25
5	5	30
6	6	35
7	7	40
8	8	45
9	9	50
10	10	55
11	11	60
12	12	65
13	13	70
14	14	75
15	15	80
16	16	85

Select `File` and `Close` to close the VIEWTABLE window. Select the `OK` button from the `Data Set Selection` window to return to the `Time Series Forecasting` window.

Creating a Time ID from Other Dating Variables

Your data set might contain ID variables that date the observations in a different way than the SAS date valued ID variable expected by the forecasting system. For example, for monthly data, the data set might contain the ID variables YEAR and MONTH, which together date the observations.

In these cases, you can use the Forecasting System's Create Time ID features to compute a time ID variable with SAS date values from the existing dating variables. As an example of this, use the SAS data set read in by the following SAS statements:

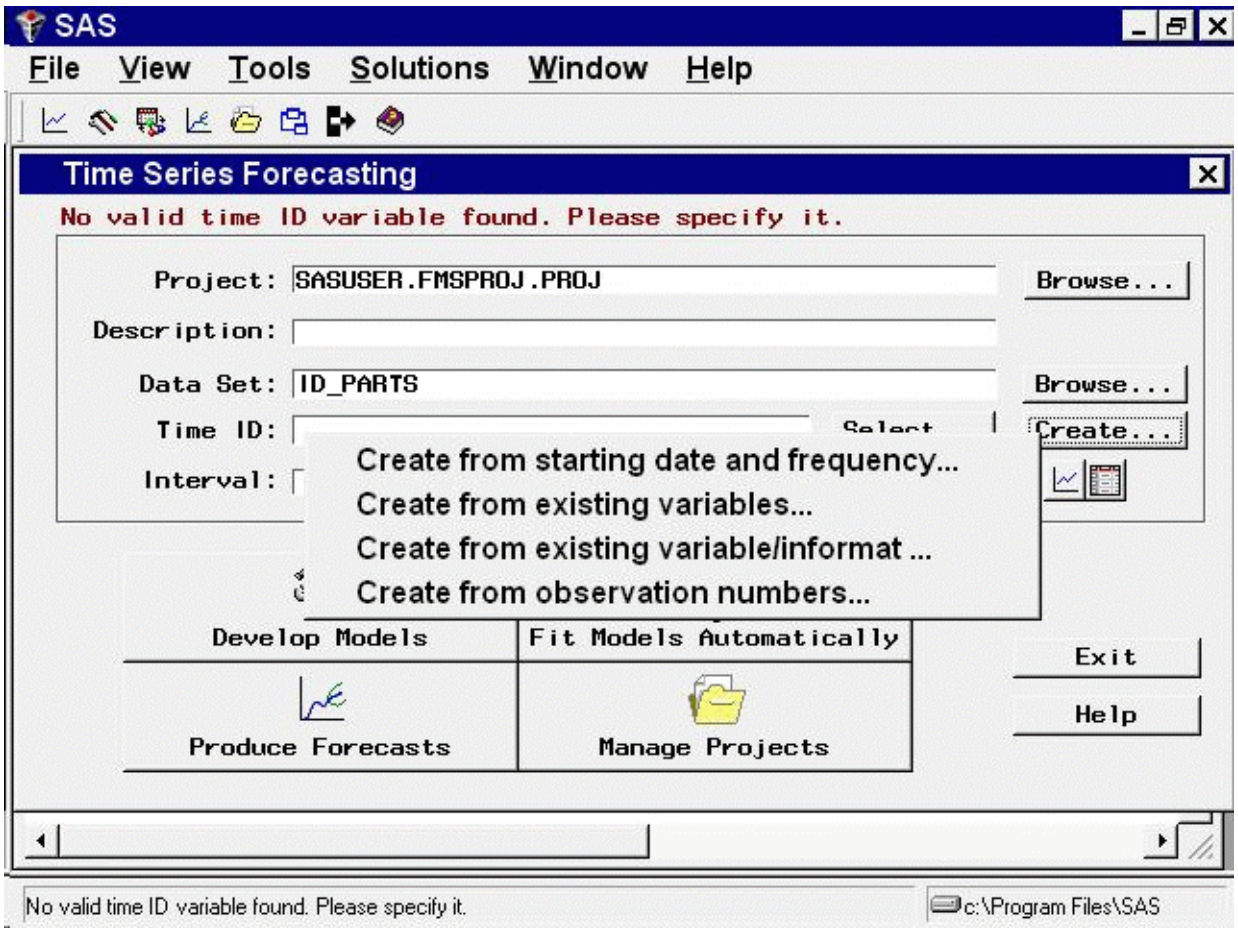
```
data id_parts;
   input yr qtr y;
datalines;
 91 1 10
 91 2 15
 91 3 20
 91 4 25
 92 1 30
 92 2 35
 92 3 40
 92 4 45
 93 1 50
 93 2 55
 93 3 60
 93 4 65
 94 1 70
 94 2 75
 94 3 80
 94 4 85
run;
```

Submit these SAS statements to create the data set ID_PARTS. This data set contains the three variables YR, QTR, and Y. YR and QTR are ID variables that together date the observations, but each variable provides only part of the date information. Because the forecasting system requires a single dating variable containing SAS date values, you need to combine YR and QTR to create a single variable DATE.

Type "ID_PARTS" in the `Data Set` field and press the ENTER key. (You could also use the Browse button to open the Data Set Selection window, as in the previous example, and complete this example from there.)

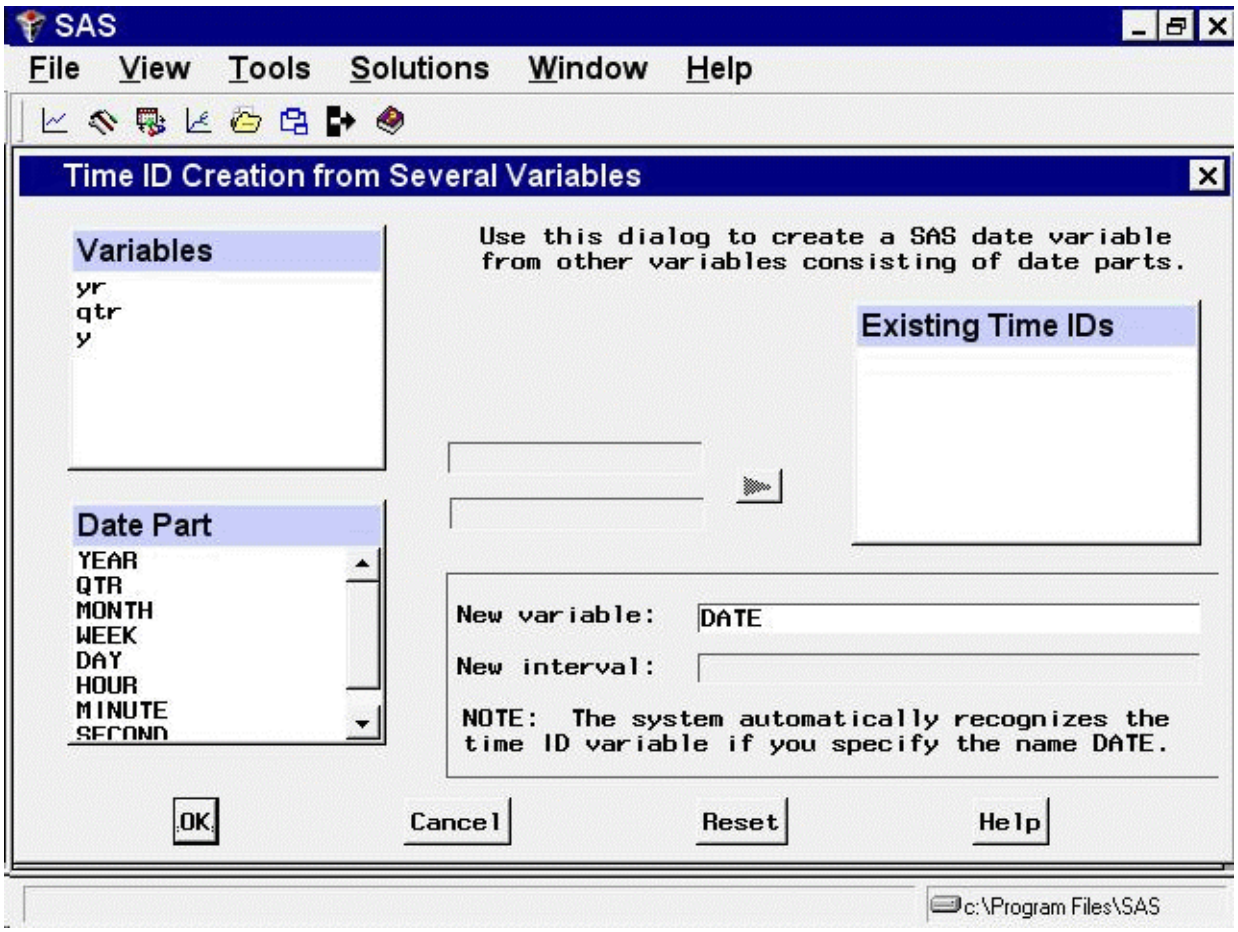
Select the Create button at the right of the `Time ID` field. This opens the menu of Create Time ID choices, as shown in Figure 55.8.

Figure 55.8 Adding a Time ID Variable



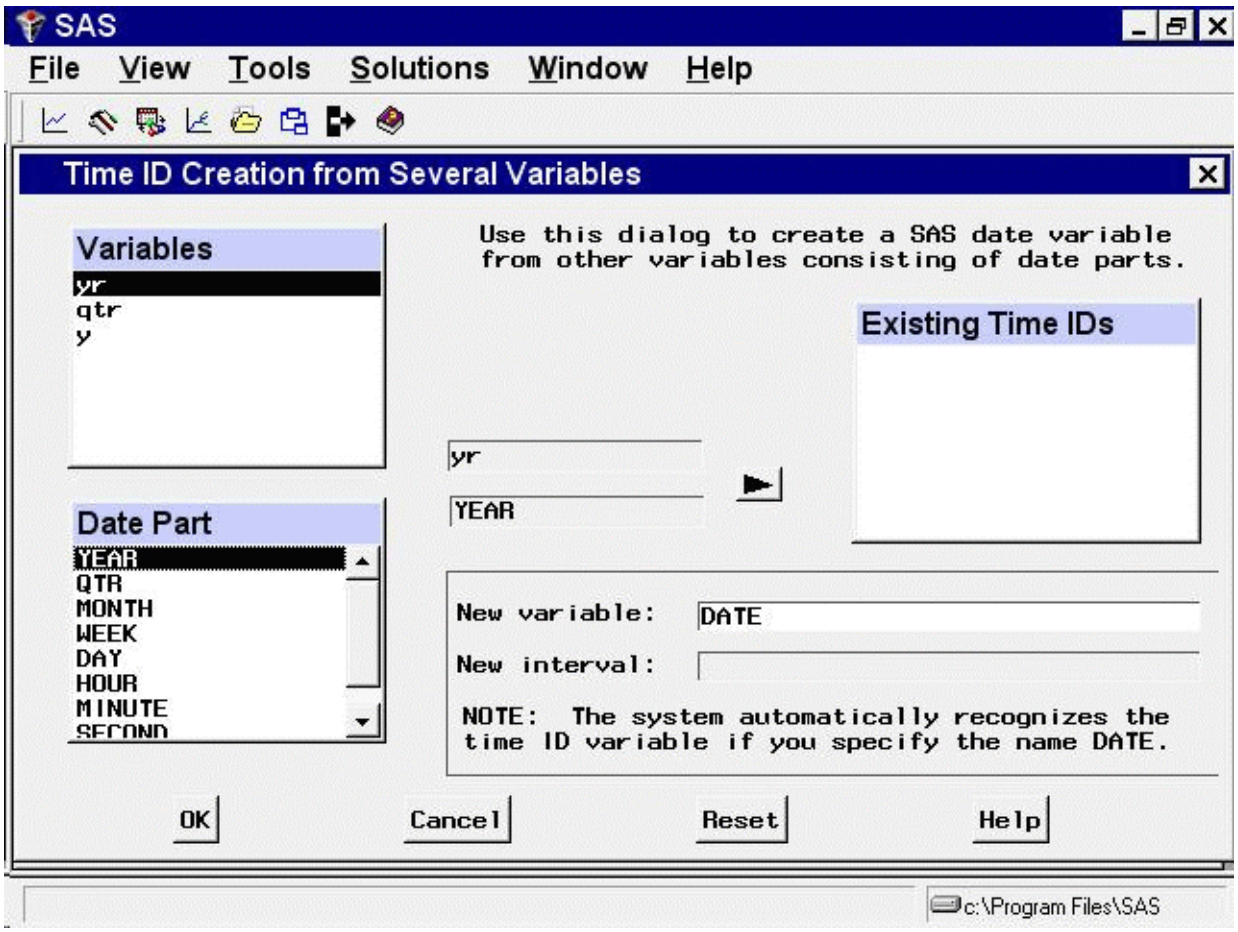
Select the second choice, `Create from existing variables`. This opens the window shown in Figure 55.9.

Figure 55.9 Creating a Time ID Variable from Date Parts



In the `Variables` list, select YR. In the `Date Part` list, select YEAR as shown in Figure 55.10.

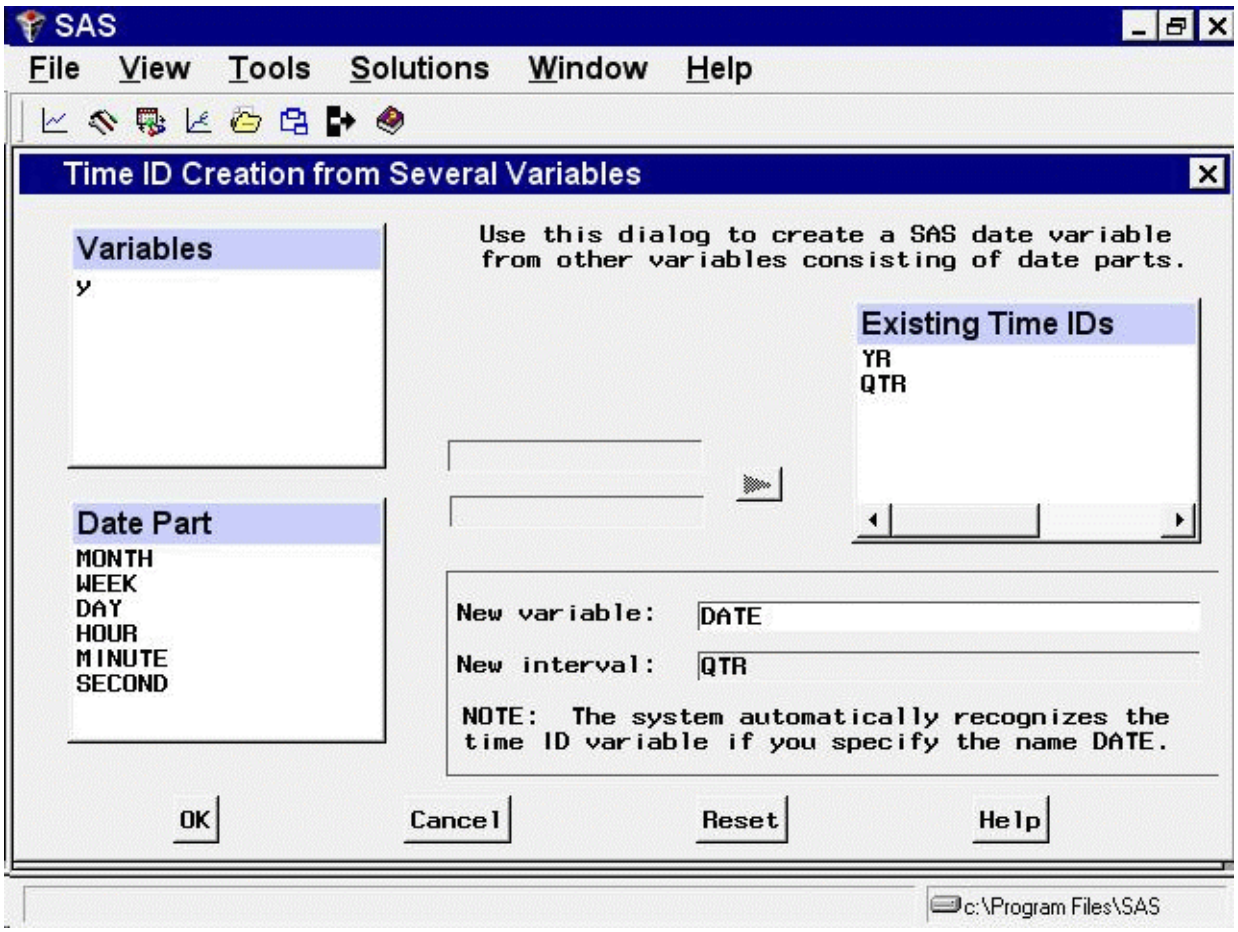
Figure 55.10 Specifying the ID Variable for Years



Now click the right-pointing arrow button. The variable YR and the part code YEAR are added to the `Existing Time IDs` list.

Next select QTR from the `Variables` list, select QTR from the `Date Part` list, and click the arrow button. This adds the variable QTR and the part code QTR to the `Existing Time IDs` list, as shown in Figure 55.11.

Figure 55.11 Creating a Time ID Variable from Date Parts



Now select the `OK` button. This opens the `New Data Set Name` window. Change the `New data set name` field to NEWDATE, and then select the `OK` button.

The data set NEWDATE is created, and the system returns to the `Time Series Forecasting` window with NEWDATE as the selected Data Set. The Time ID field is set to DATE, and the Interval field is set to QTR.

Chapter 56
Specifying Forecasting Models

Contents

Series Diagnostics 3681
Models to Fit Window 3685
Automatic Model Selection 3688
Smoothing Model Specification Window 3690
ARIMA Model Specification Window 3693
Factored ARIMA Model Specification Window 3697
Custom Model Specification Window 3700
Editing the Model Selection List 3707
Forecast Combination Model Specification Window 3710
Incorporating Forecasts from Other Sources 3713

This chapter explores the tools available through the Develop Models window for investigating the properties of time series and for specifying and fitting models. The first section shows you how to diagnose time series properties in order to determine the class of models appropriate for forecasting series with such properties. Later sections show you how to specify and fit different kinds of forecasting models.

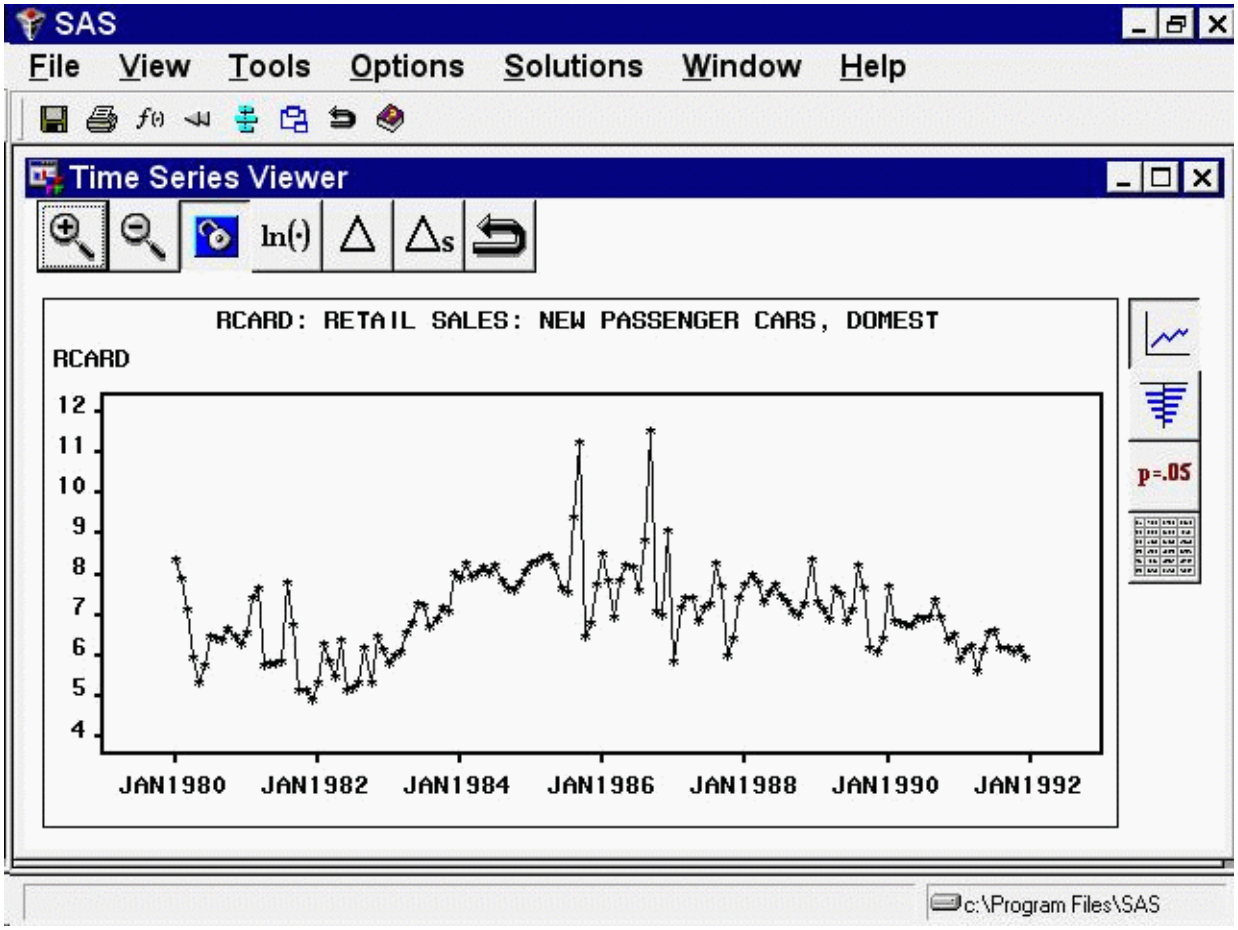
Series Diagnostics

The series diagnostics tool helps you determine the kinds of forecasting models that are appropriate for the data series so that you can limit the search for the best forecasting model. The series diagnostics address these three questions: Is a log transformation needed to stabilize the variance? Is a time trend present in the data? Is there a seasonal pattern to the data?

The automatic model fitting process, which you used in the previous chapter through the Automatic Model Fitting window, performs series diagnostics and selects trial models from a list according to the results. You can also look at the diagnostic information and make your own decisions as to the kinds of models appropriate for the series. The following example illustrates the series diagnostics features.

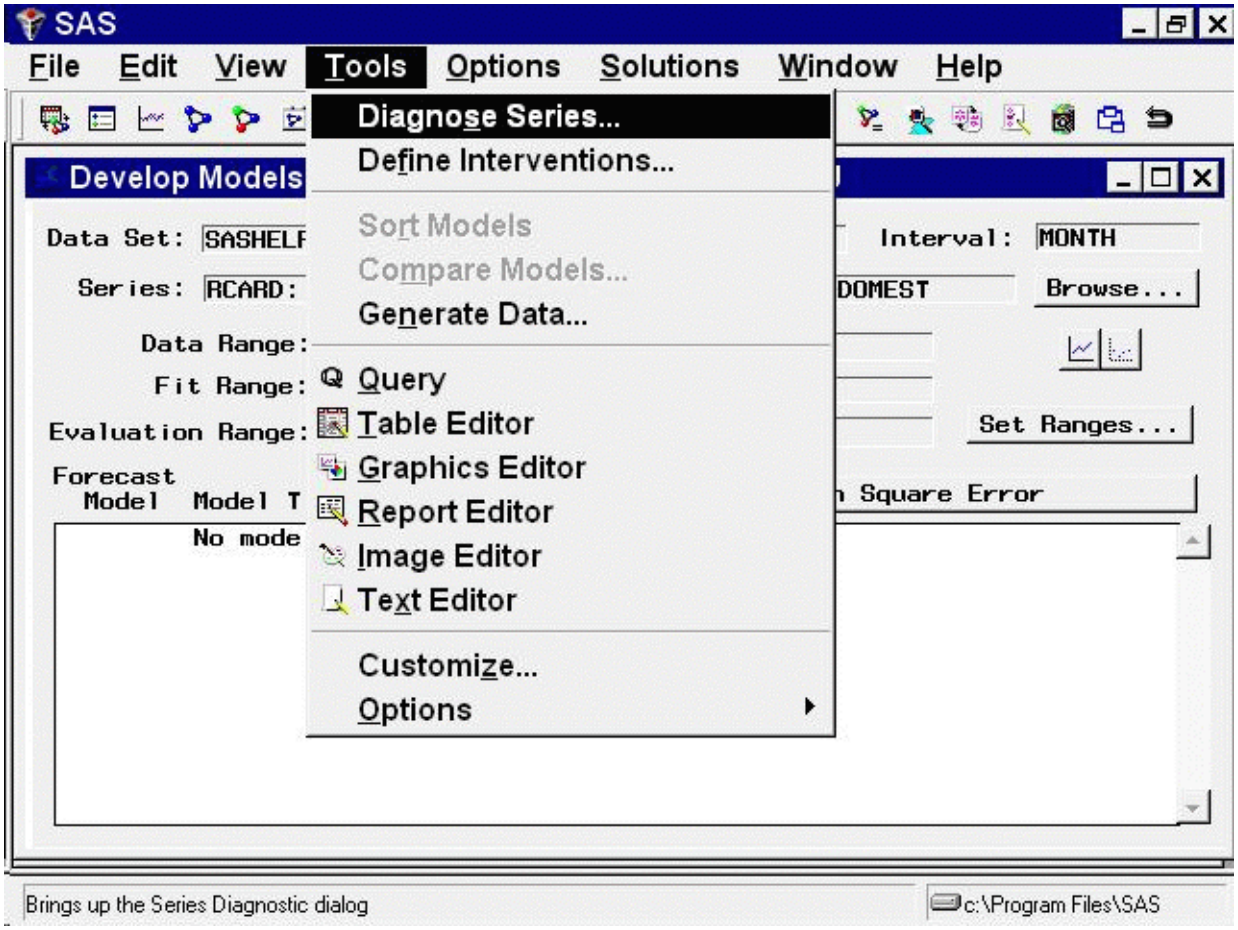
Select “Develop Models” from the Time Series Forecasting window. Select the library SASHELP, the data set CITIMON, and the series RCARD. This series represents domestic retail sales of passenger cars. To look at this series, select “View Series” from the Develop Models window. This opens the Time Series Viewer window, as shown in Figure 56.1.

Figure 56.1 Automobile Sales Series



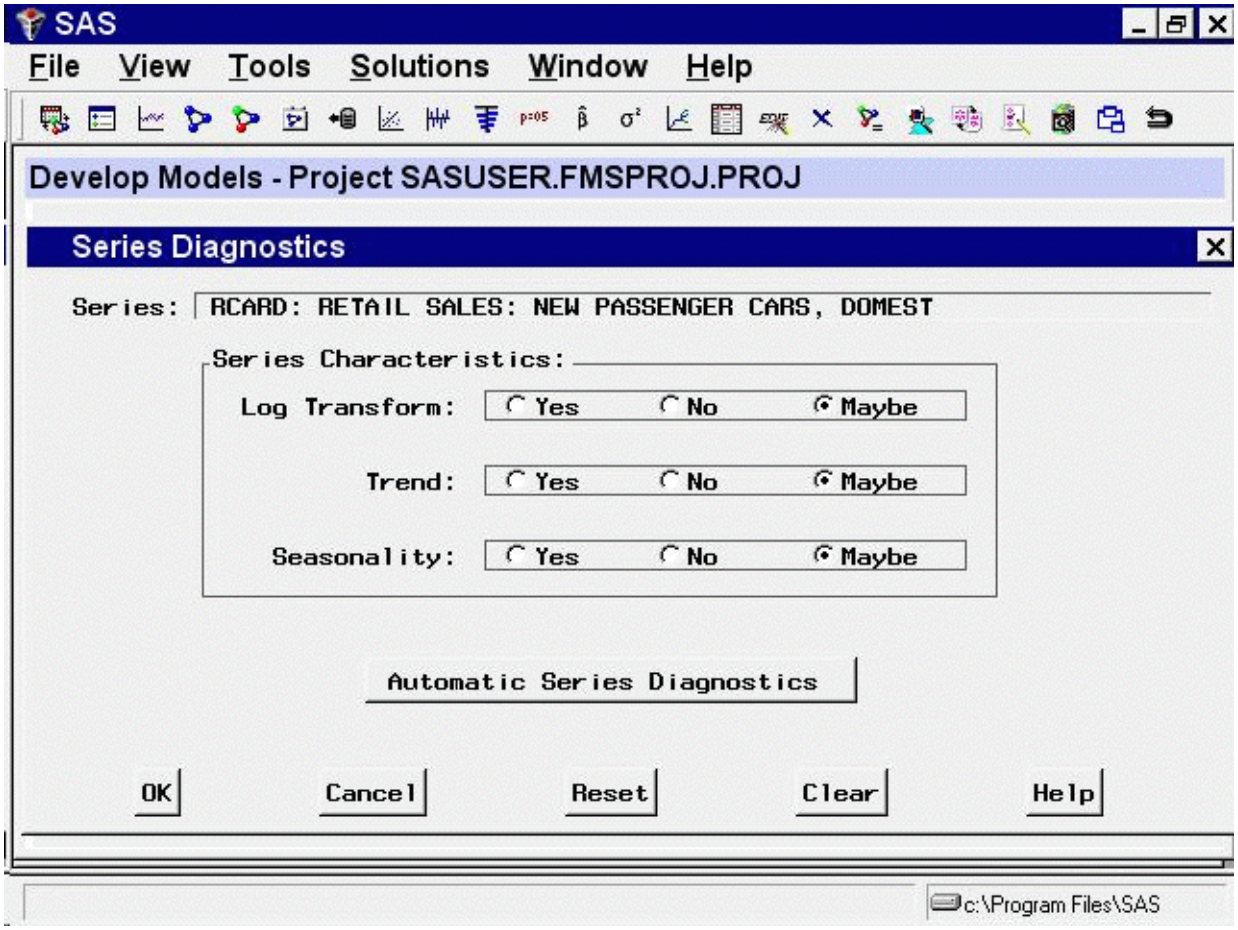
Select “Diagnose Series” from the Tools menu. You can do this from the Develop Models window or from the Time Series Viewer window. Figure 56.2 shows this from the Develop Models window.

Figure 56.2 Selecting Series Diagnostics



This opens the Series Diagnostics window, as shown in Figure 56.3.

Figure 56.3 Series Diagnostics Window



Each of the three series characteristics—need for log transformation, presence of a trend, and seasonality—has a set of options for `Yes`, `No`, and `Maybe`. `Yes` indicates that the series has the characteristic and that forecasting models fit to the series should be able to model and predict this behavior. `No` indicates that you do not need to consider forecasting models designed to predict series with this characteristic. `Maybe` indicates that models with and without the characteristic should be considered. Initially, all these values are set to `Maybe`.

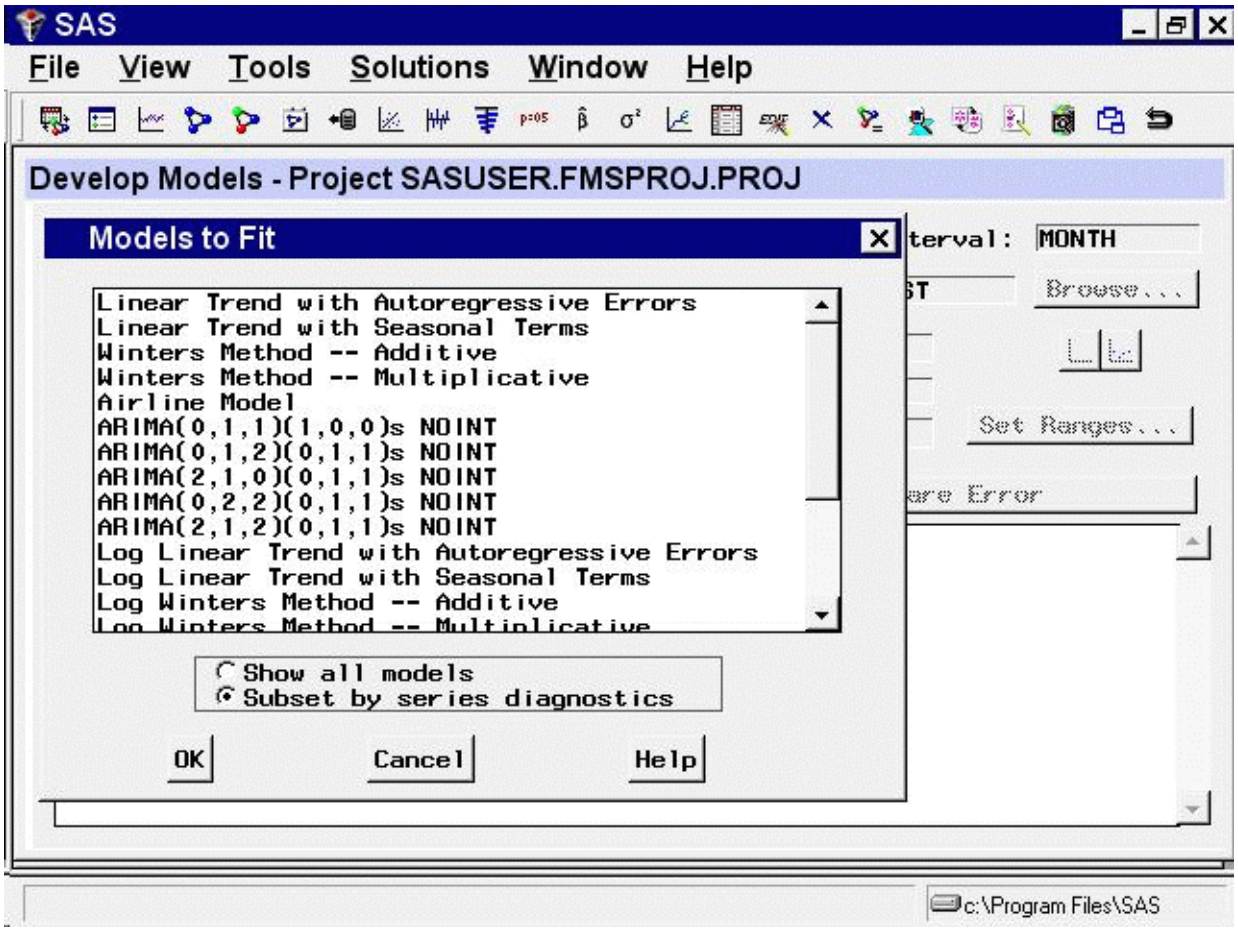
To have the system diagnose the series characteristics, select the Automatic Series Diagnostics button. This runs the diagnostic routines described in Chapter 61, "Forecasting Process Details," and sets the options according to the results. In this example, `Trend` and `Seasonality` are changed from `Maybe` to `Yes`, while `Log Transform` remains set to `Maybe`.

These diagnostic criteria affect the models displayed when you use the Models to Fit window or the Automatic Model Selection model-fitting options described in the following section. You can set the criteria manually, according to your judgment, by selecting any of the options, whether you have used the Automatic Series Diagnostics button or not. For this exercise, leave them as set by the automatic diagnostics. Select the OK button to close the Series Diagnostics window.

Models to Fit Window

As you saw in the previous chapter, you can select models from a list. Invoke the Models to Fit window by clicking the middle of the table and selecting "Fit Models from List" from the menu. This can also be selected from the tool bar or the Fit Model submenu of the Edit menu. The Models to Fit window comes up, as shown in Figure 56.4.

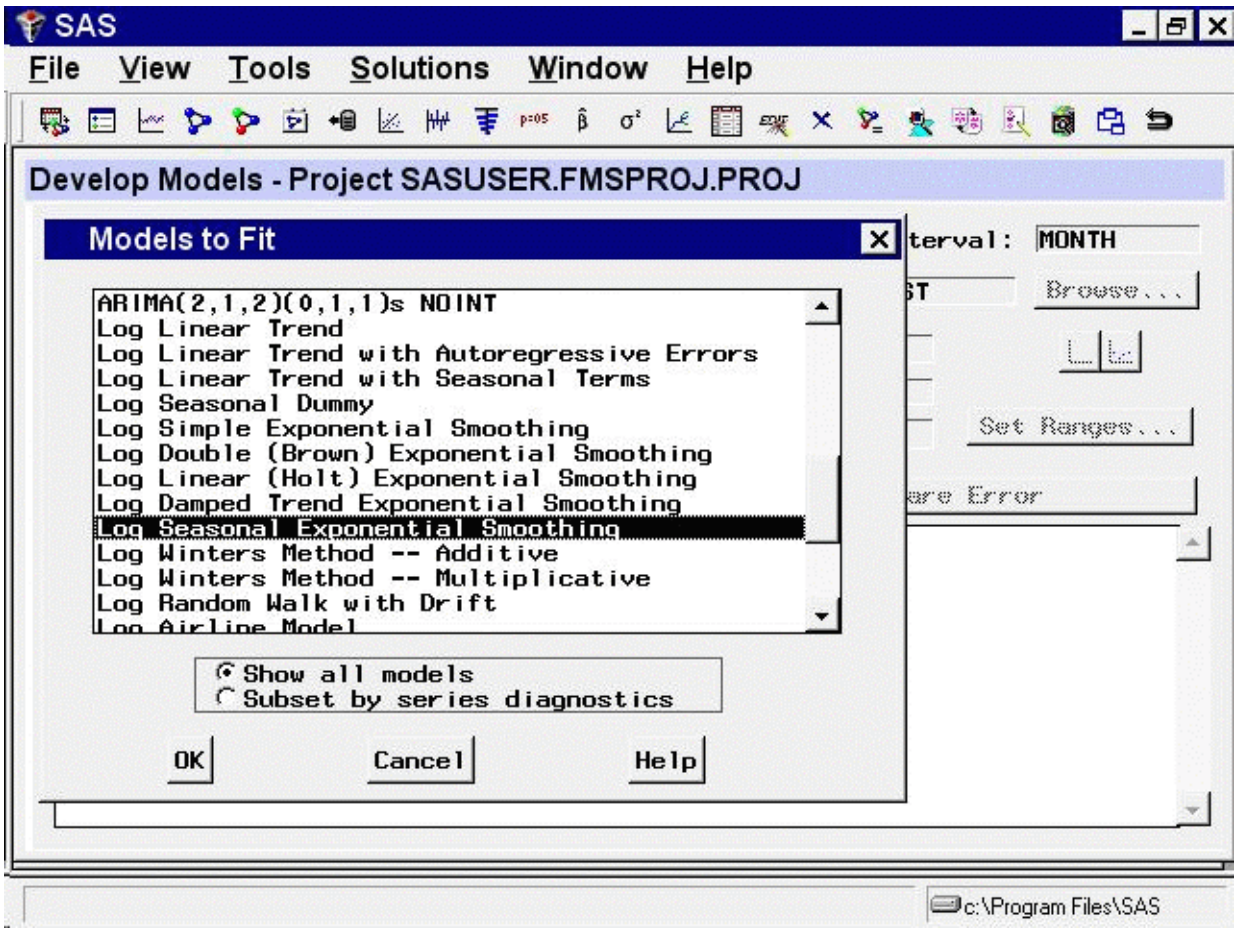
Figure 56.4 Models to Fit Window



Since you have performed series diagnostics, the models shown are the subset that fits the diagnostic criteria.

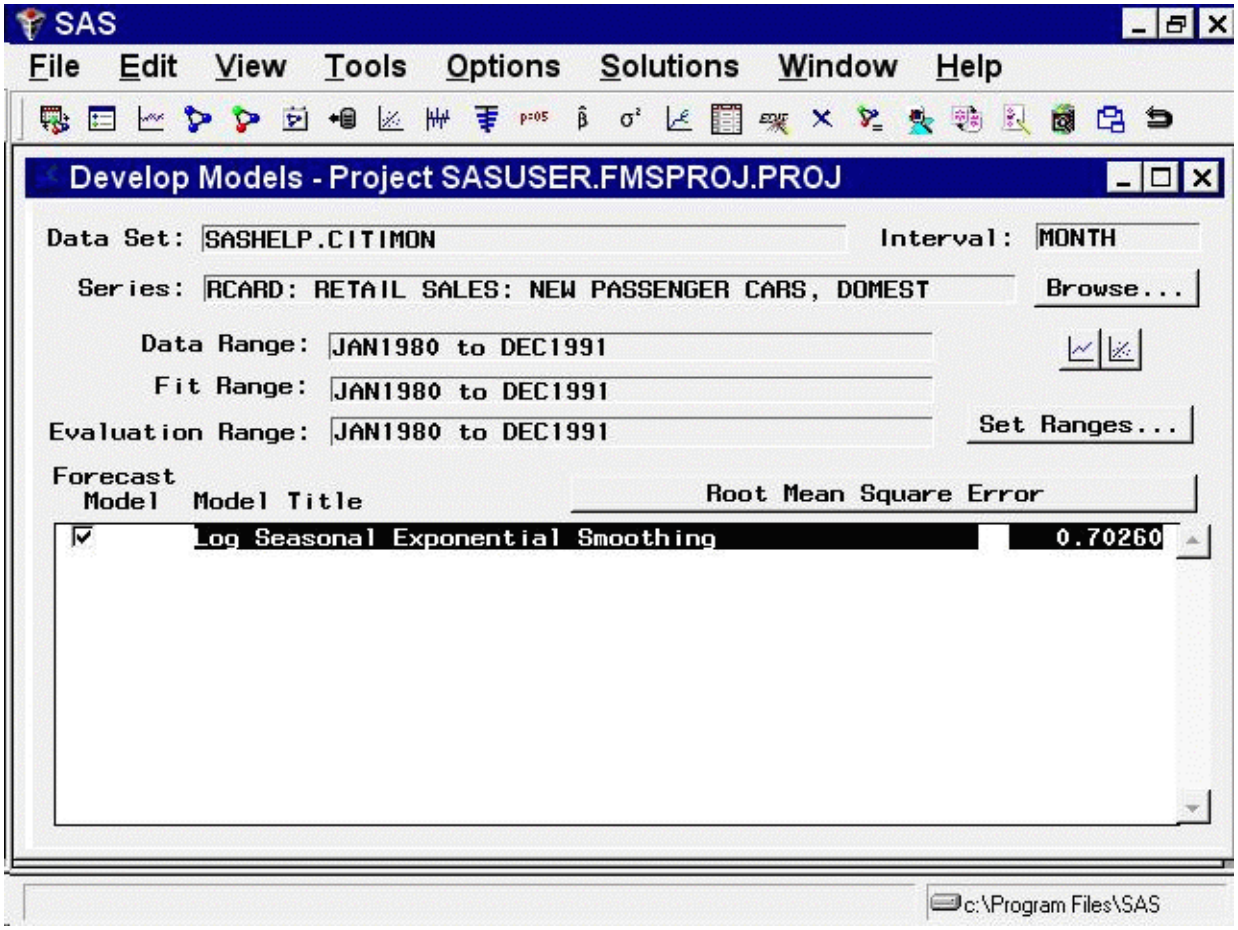
Suppose you want to consider models other than those in this subset because you are undecided about including a trend in the model. Select the Show all models option. Now the entire model selection list is shown. Scroll through the list until you find `Log Seasonal Exponential Smoothing`, as shown in Figure 56.5.

Figure 56.5 Selecting a Model from List



This is a nontrended model, which seems a good candidate. Select this model, and then select the OK button. The model is fit to the series and then appears in the table with the value of the selected fit criterion, as shown in Figure 56.6.

Figure 56.6 Develop Models Window Showing Model Fit



You can edit the model list that appears in the Models to Fit window by selecting “Options” and “Model Selection List” from the menu bar or by selecting the Edit Model List toolbar icon. You can then delete models you are not interested in from the default list and add models using any of the model specification methods described in this chapter. When you save your project, the edited model selection list is saved in the project file. In this way, you can use the Select from List item and the Automatic Model Selection item to select models from a customized search set.

Automatic Model Selection

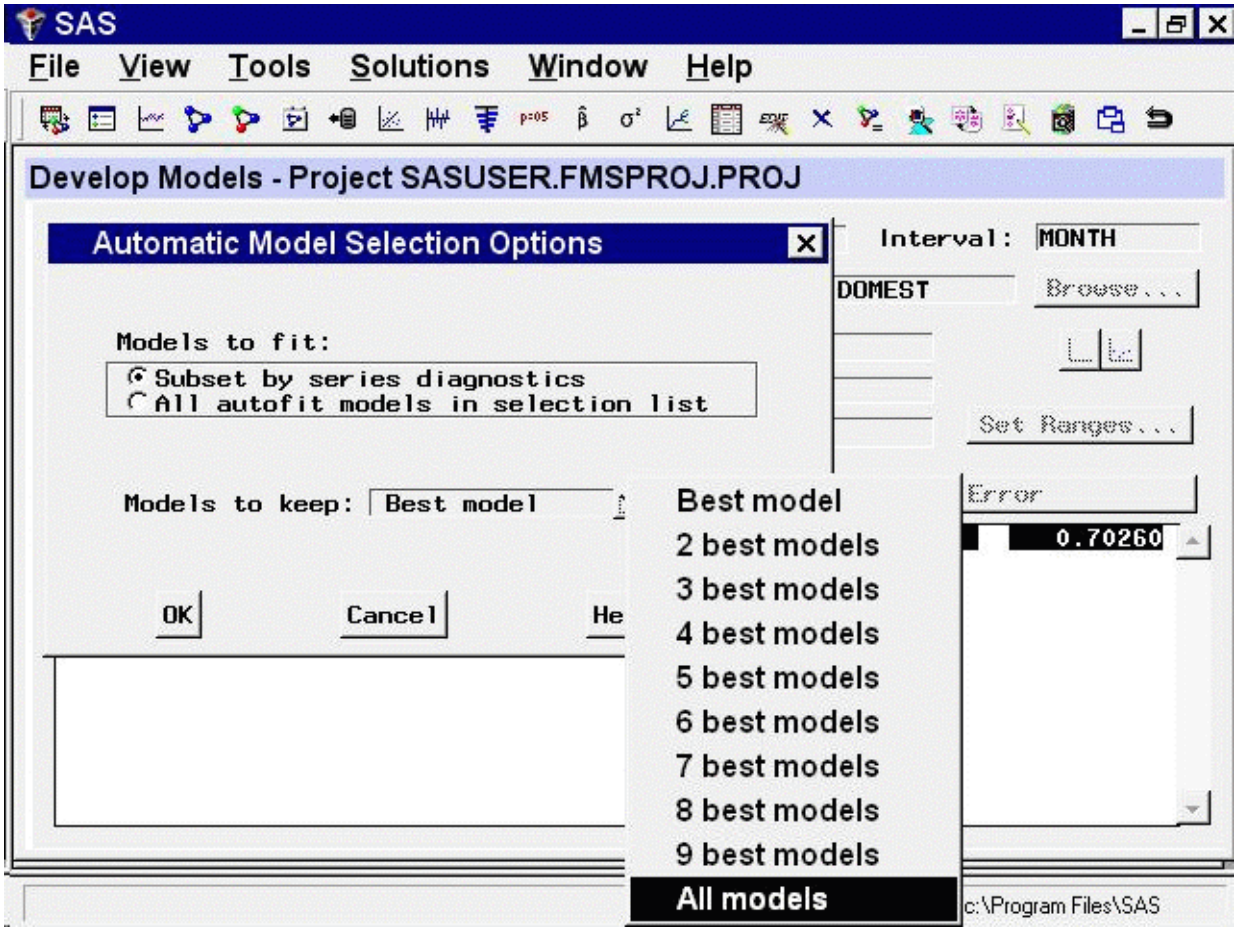
Automatic model selection is equivalent to choosing Select from List, as you did in the preceding section, fitting all the models in the subset list and then deleting all except the best fitting of the models. If series diagnostics have not yet been done, they are performed automatically to determine the model subset to fit. If you set the series diagnostics for log, trend, or seasonal criteria manually using the radio buttons, these choices are honored by the automatic fitting process.

Using automatic selection, the system does not pause to warn you of model fitting errors, such as failure of the estimates to converge (you can track these using the audit trail feature).

By default, only the best fitting model is kept. However, you can control the number of automatically fit models retained in the Develop Models list, and the following example shows how to do this.

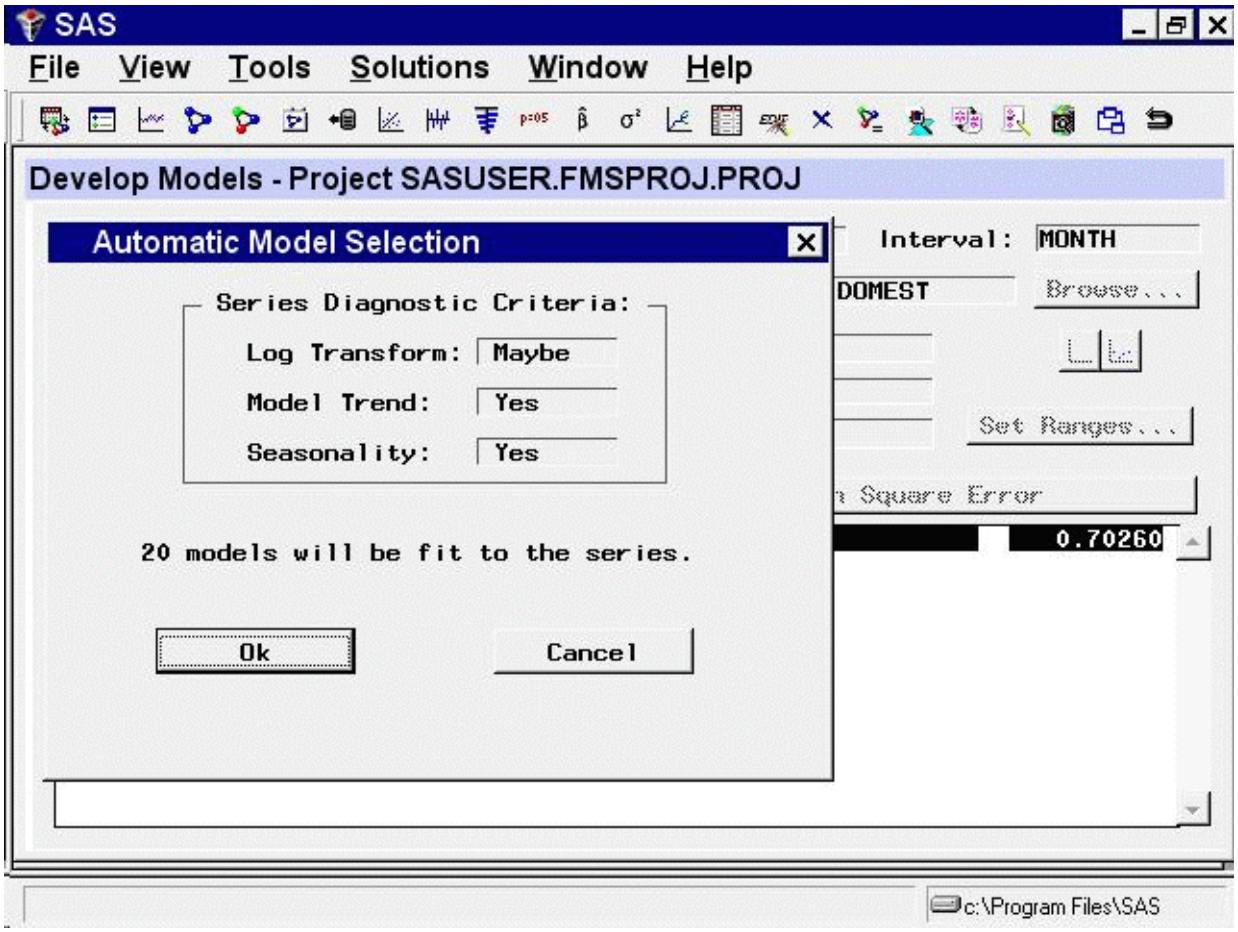
From the menu bar, choose "Options" and "Automatic Fit." This opens the Automatic Model Selection Options window. Click the `Models to Keep` list arrow, and select "All models", as shown in Figure 56.7. Now select OK.

Figure 56.7 Selecting Number of Automatic Fit Models to Keep



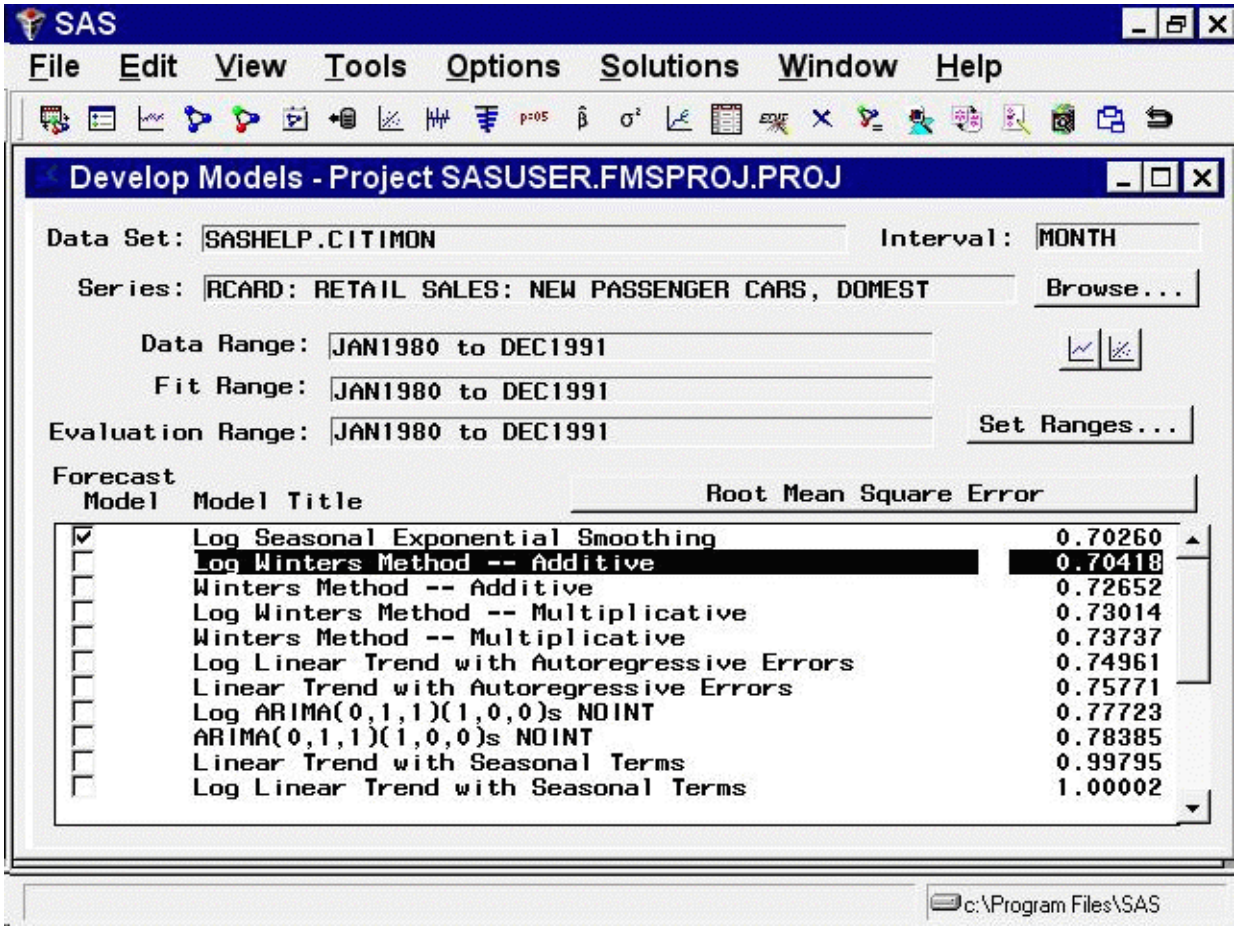
Next, select “Fit Models Automatically” by clicking the middle of the table or using the toolbar or Edit menu. The Automatic Model Selection window appears, showing the diagnostic criteria in effect and the number of models to be fit, as shown in Figure 56.8.

Figure 56.8 Automatic Model Selection Window



Select the OK button. After the models have been fit, all of them appear in the table, in addition to the model which you fit earlier, as shown in Figure 56.9.

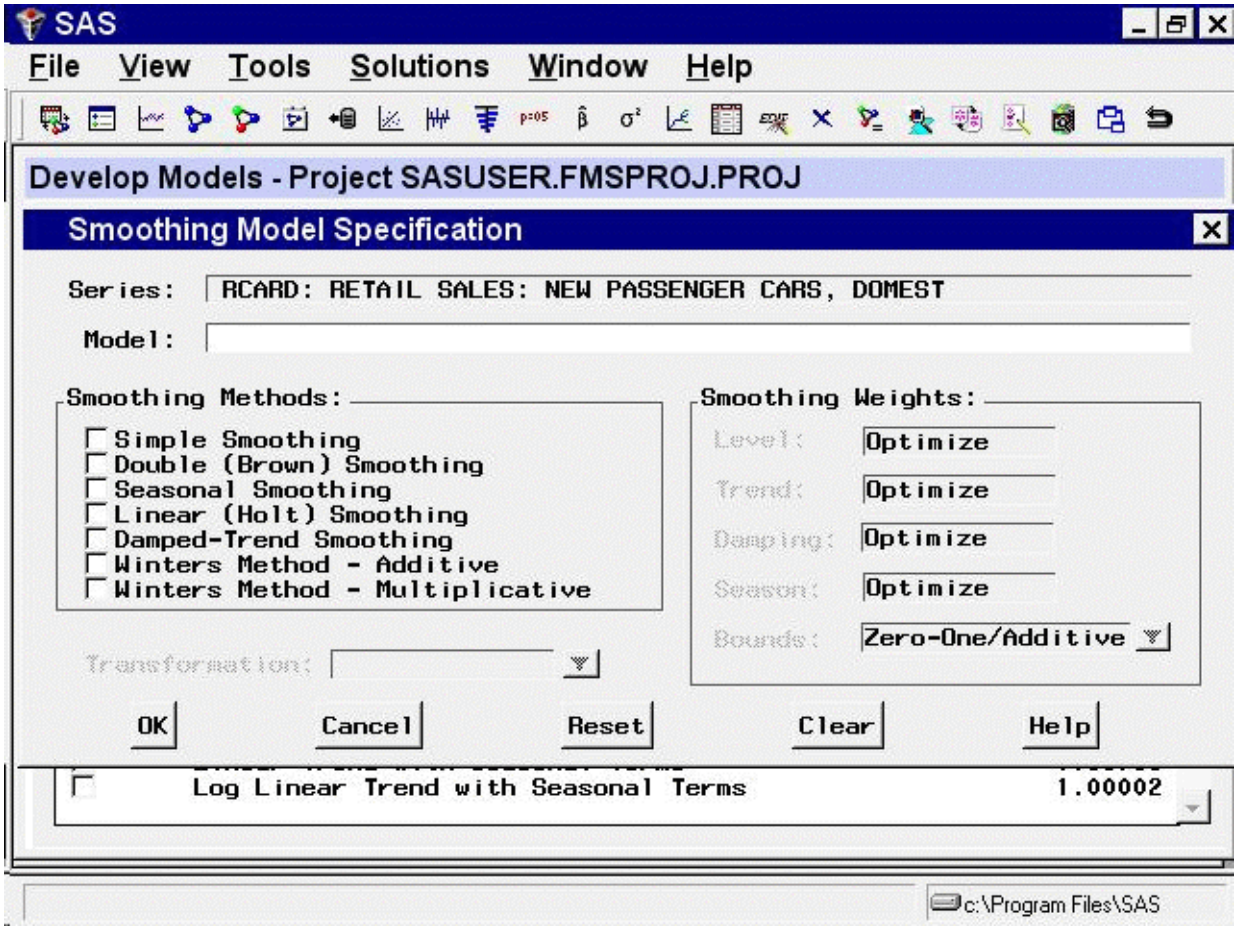
Figure 56.9 Automatically Fit Models



Smoothing Model Specification Window

To fit exponential smoothing and Winters models not already provided in the Models to Fit window, select "Fit Smoothing Model" from the pop-up menu or toolbar or select "Smoothing Model" from the Fit Model submenu of the Edit menu. This opens the Smoothing Model Specification window, as shown in Figure 56.10.

Figure 56.10 Smoothing Model Specification Window



The Smoothing Model Specification window consists of several parts. At the top is the series name and a field for the label of the model you are specifying. The model label is filled in with an automatically generated label as you specify options. You can type over the automatic label with your own label for the model. To restore the automatic label, enter a blank label.

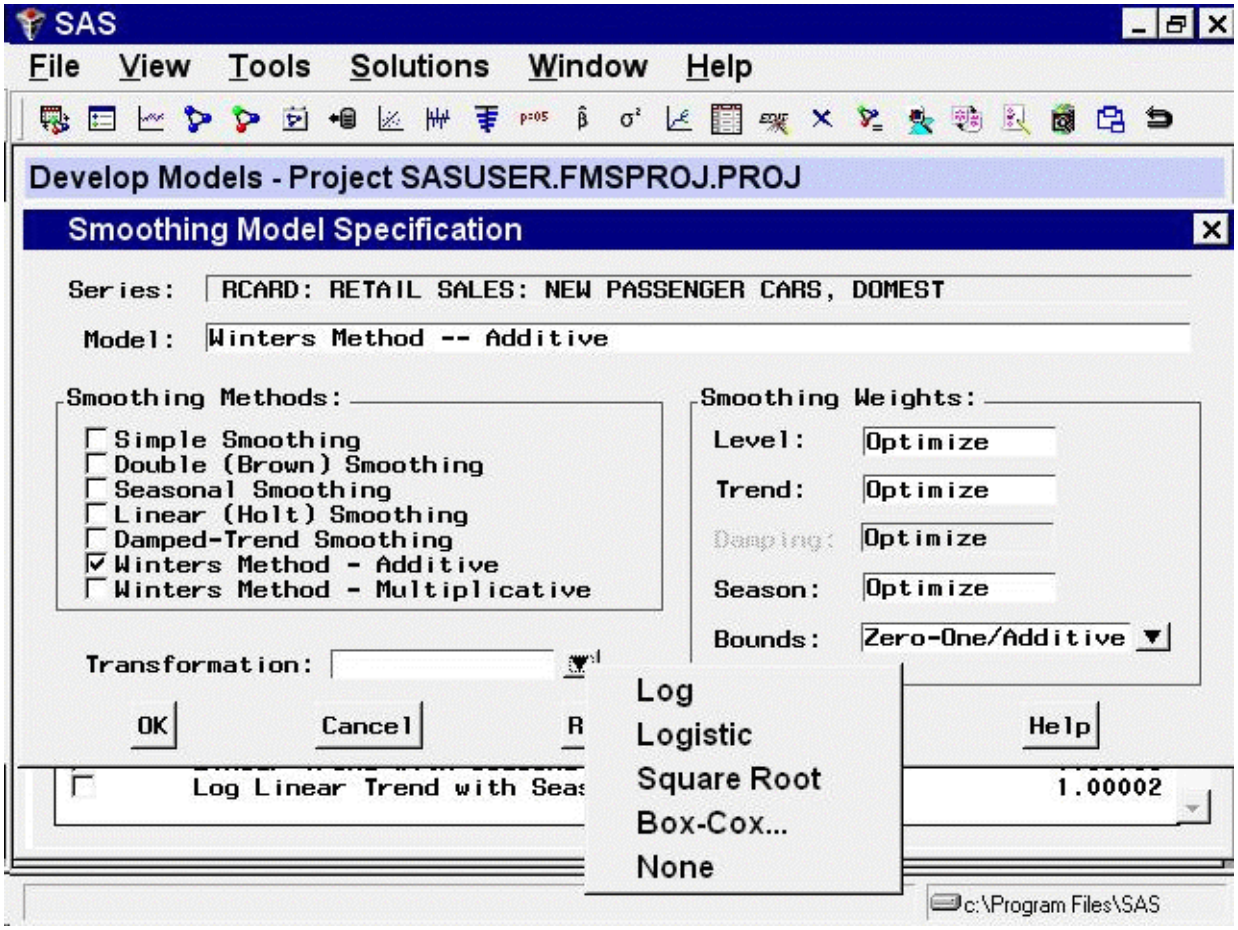
The `Smoothing Methods` box lists the different methods available. Below the `Smoothing Methods` box is the Transformation field, which is used to apply the smoothing method to transformed series values.

The `Smoothing Weights` box specifies how the smoothing weights are determined. By default, the smoothing weights are automatically set to optimize the fit of the model to the data. See Chapter 61, "Forecasting Process Details," for more information about how the smoothing weights are fit.

Under smoothing methods, select "Winters Method – Additive." Notice the smoothing weights box to the right. The third item, `Damping`, is grayed out, while the other items, `Level`, `Trend`, and `Season`, show the word `Optimize`. This tells you that these three smoothing weights are applicable to the smoothing method that you selected and that the system is currently set to optimize these weights for you.

Next, specify a transformation using the `Transformation` list. A menu of transformation choices pops up, as shown in Figure 56.11.

Figure 56.11 Transformation Options

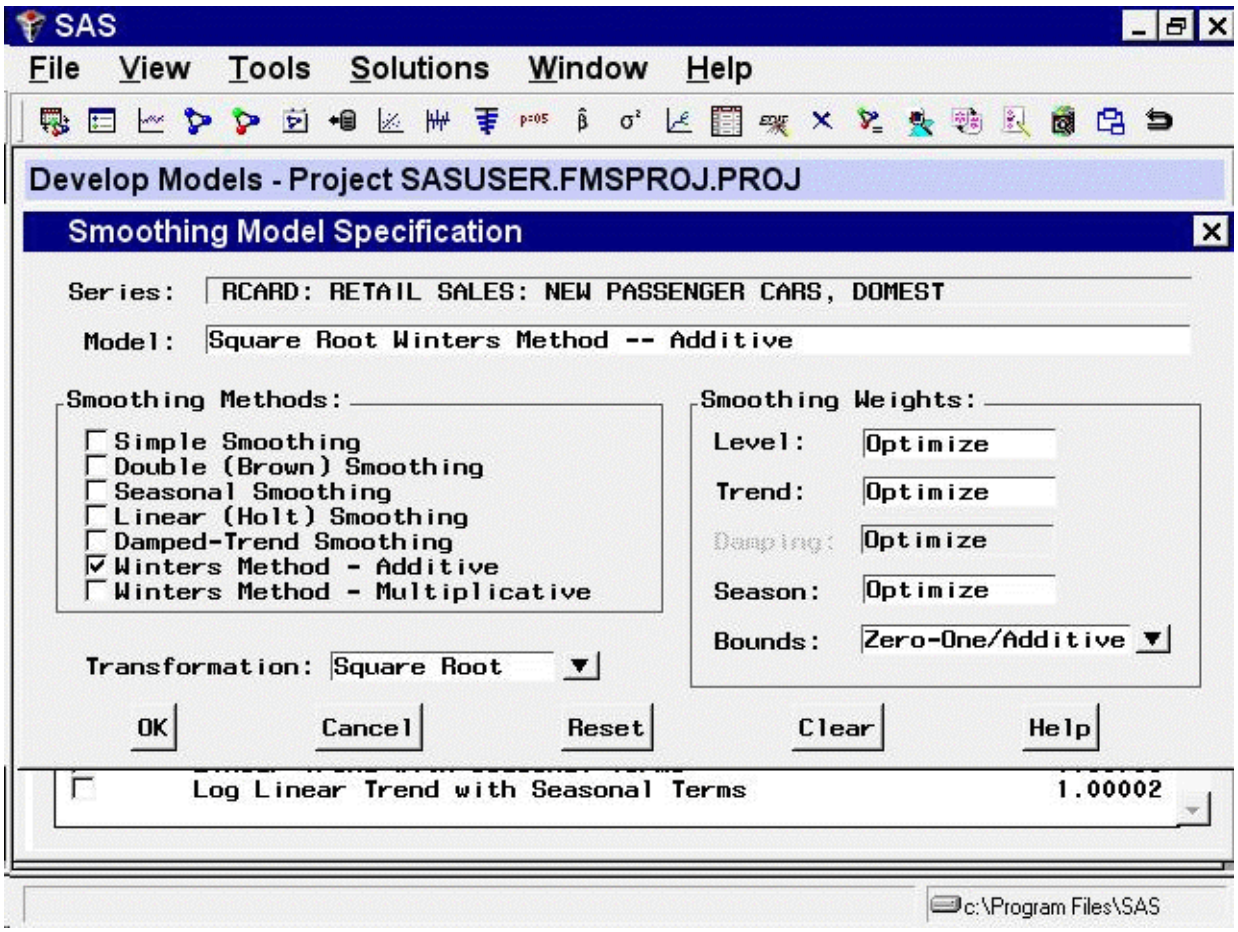


You can specify a logarithmic, logistic, square root, or Box-Cox transformation. For this example, select "Square Root" from the list. The Transformation field is now set to Square Root.

This means that the system will first take the square roots of the series values, apply the additive version of the Winters method to the square root series, and then produce the predictions for the original series by squaring the Winters method predictions (and multiplying by a variance factor if the Mean Prediction option is set in the Forecast Options window). See Chapter 61, "Forecasting Process Details," for more information about predictions from transformed models.

The Smoothing Model Specification window should now appear as shown in Figure 56.12. Select the OK button to fit the model. The model is added to the table of fitted models in the Develop Models window.

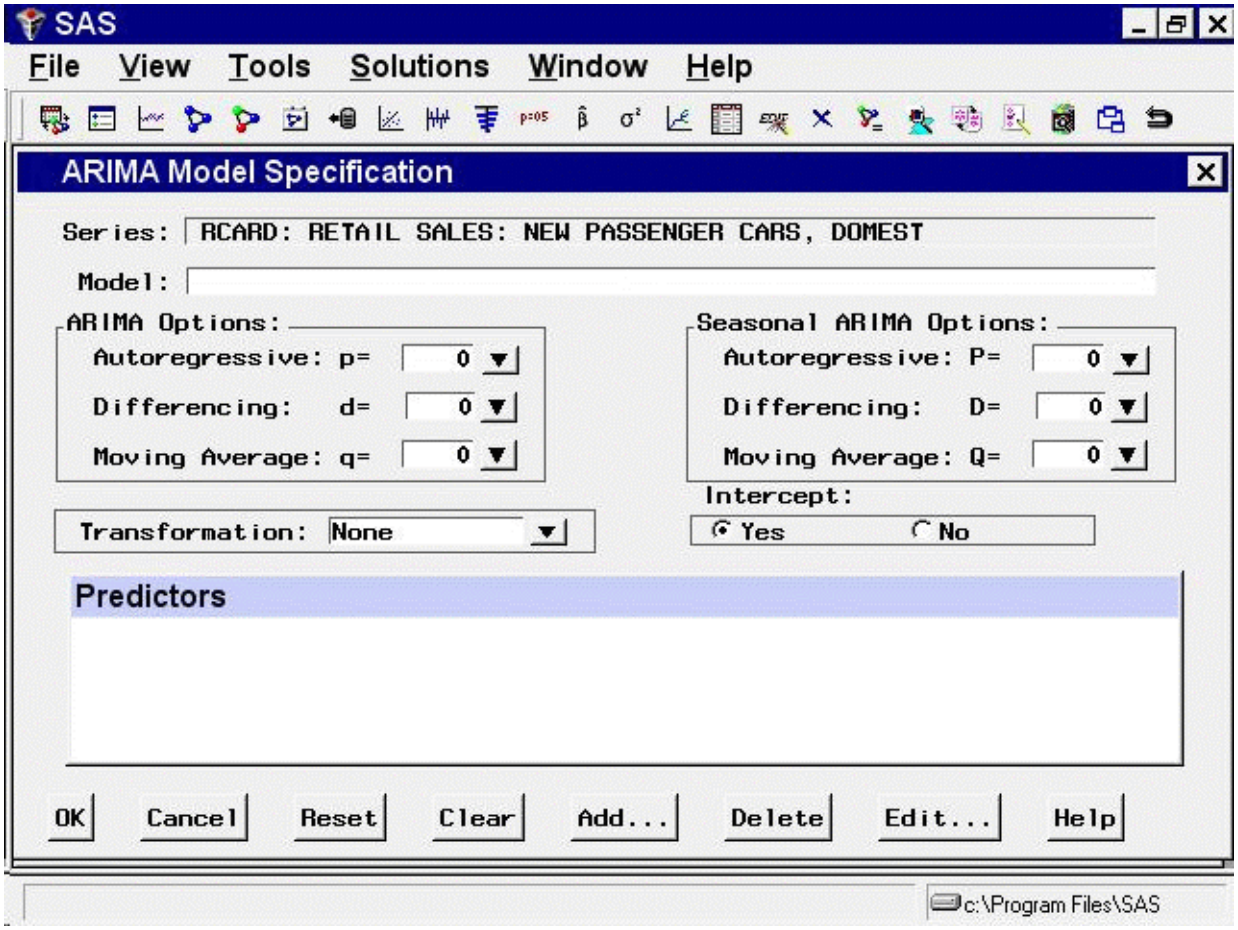
Figure 56.12 Winter's Method Applied to Square Root Series



ARIMA Model Specification Window

To fit ARIMA or Box-Jenkins models not already provided in the Models to Fit window, select the ARIMA model item from the pop-up menu, toolbar, or Edit menu. This opens the ARIMA Model Specification window, as shown in Figure 56.13.

Figure 56.13 ARIMA Model Specification Window



This ARIMA Model Specification window is structured according to the Box and Jenkins approach to time series modeling. You can specify the same time series models with the Custom Model Specification window and the ARIMA Model Specification window, but the windows are structured differently, and you may find one more convenient than the other.

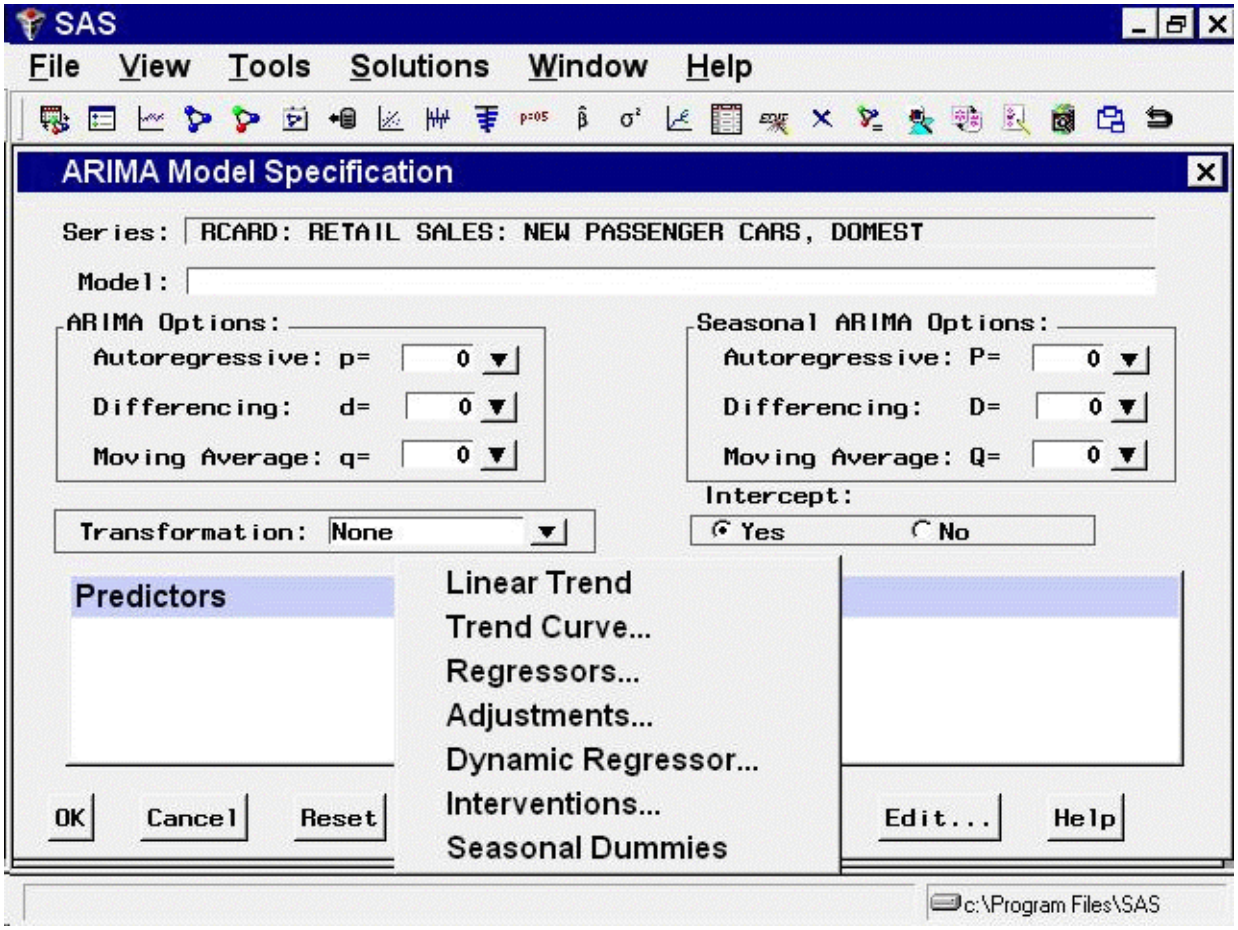
At the top of the ARIMA Model Specification window is the name and label of the series and the label of the model you are specifying. The model label is filled in with an automatically generated label as you specify options. You can type over the automatic label with your own label for the model. To restore the automatic label, enter a blank label.

Using the ARIMA Model Specification window, you can specify autoregressive (p), differencing (d), and moving average (q) orders for both simple and seasonal factors. You can specify transformations with the `Transformation` list. You can also specify whether an intercept is included in the ARIMA model.

In addition to specifying seasonal and nonseasonal ARIMA processes, you can also specify predictor variables and other terms as inputs to the model. ARIMA models with inputs are sometimes called ARIMAX models or Box-Tiao models. Another term for this kind of model is *dynamic regression*.

In the lower part of the ARIMA Model Specification window is the list of *predictors* to the model (initially empty). You can specify predictors by using the Add button. This opens a menu of different kinds of independent effects, as shown in Figure 56.14.

Figure 56.14 Add Predictors Menu



The kinds of predictor effects allowed include time trends, regressors, adjustments, dynamic regression (transfer functions), intervention effects, and seasonal dummy variables. How to use different kinds of predictors is explained in Chapter 58, "Using Predictor Variables."

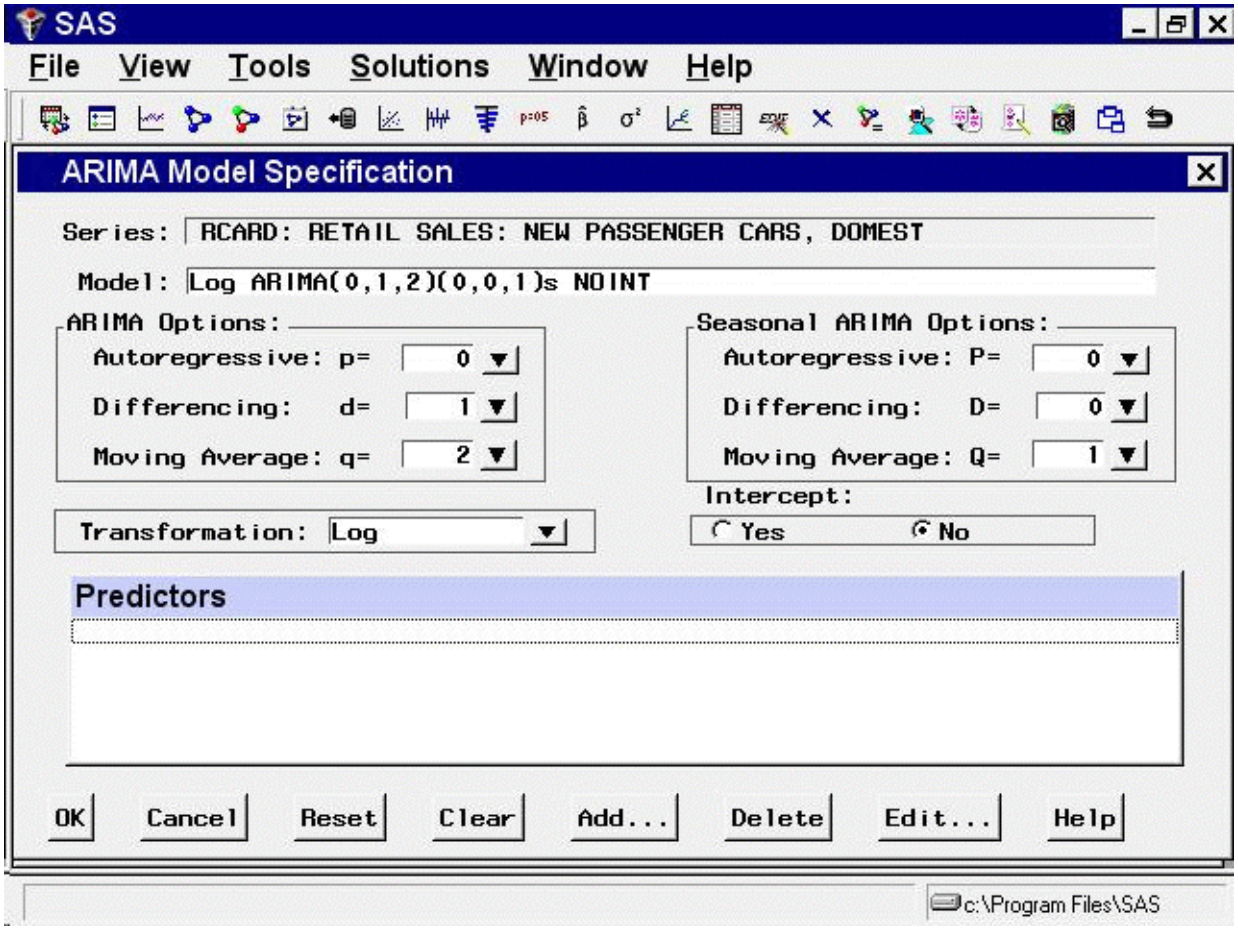
As an example, in the `ARIMA Options` box, set the order of differencing `d` to `1` and the moving average order `q` to `2`. You can either type in these values or click the arrows and select the values from pop-up lists.

These selections specify an ARIMA(0,1,2) or IMA(1,2) model. (See Chapter 8, "The ARIMA Procedure," for more information about the notation used for ARIMA models.) Notice that the model label at the top is now `IMA(1,2) NOINT`, meaning that the data are differenced once and a second-order moving-average term is included with no intercept.

In the `Seasonal ARIMA Options` box, set the seasonal moving-average order `Q` to `1`. This adds a first-order moving-average term at the seasonal (12 month) lag. Finally, select "Log" in the Transformation combo box.

The model label is now `Log ARIMA(0,1,2)(0,0,1)s NOINT`, and the window appears as shown in Figure 56.15.

Figure 56.15 Log ARIMA(0,1,2)(0,0,1)s Specified

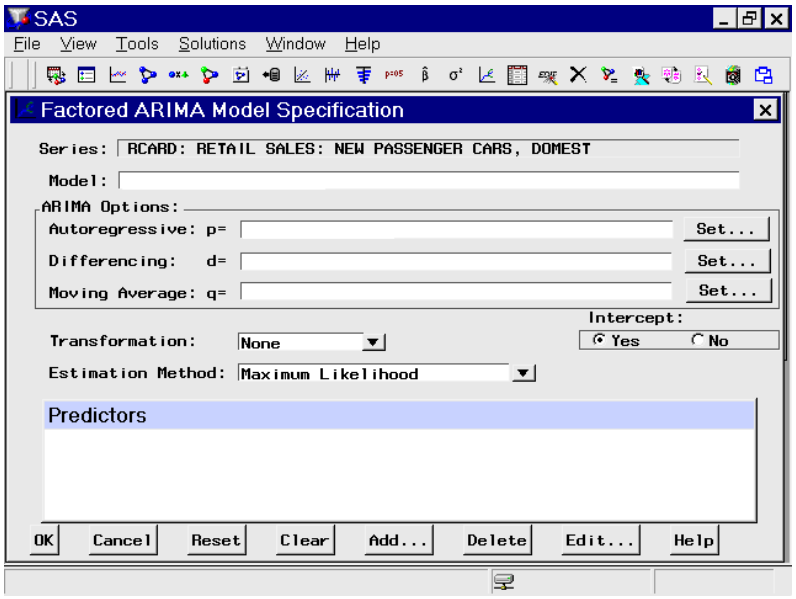


Select the OK button to fit the model. The model is fit and added to the Develop Models table.

Factored ARIMA Model Specification Window

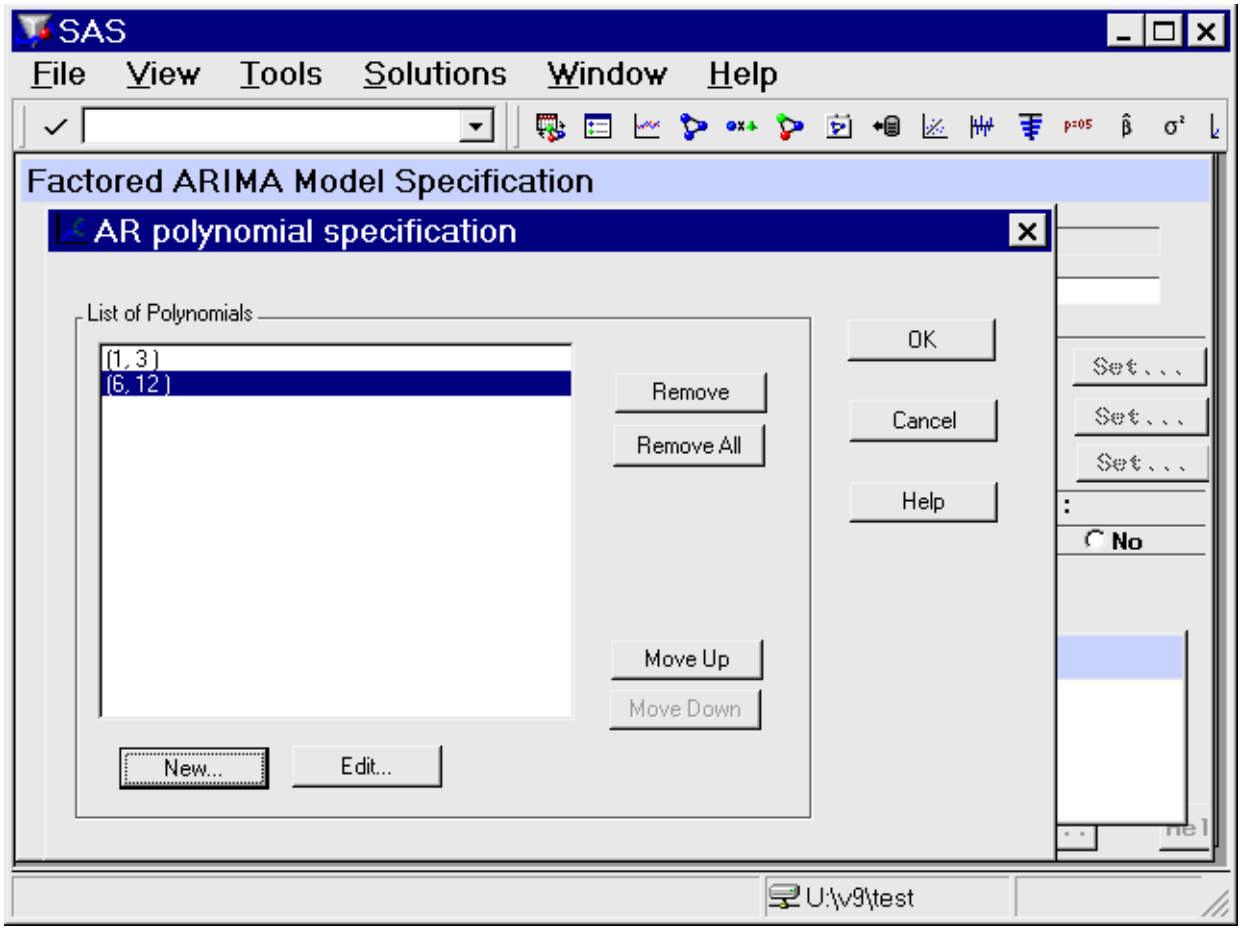
To fit a factored ARIMA model, select the Factored ARIMA model item from the pop-up menu, toolbar, or Edit menu. This brings up the Factored ARIMA Model Specification window, shown in Figure 56.16.

Figure 56.16 Factored ARIMA Model Specification Window



The Factored ARIMA Model Specification window is similar to the ARIMA Model Specification window and has the same features, but it uses a more general specification of the autoregressive (p), differencing (d), and moving-average (q) terms. To specify these terms, select the corresponding Set button, as shown in Figure 56.16. For example, to specify autoregressive terms, select the first Set button. This opens the AR Polynomial Specification Window, shown in Figure 56.17.

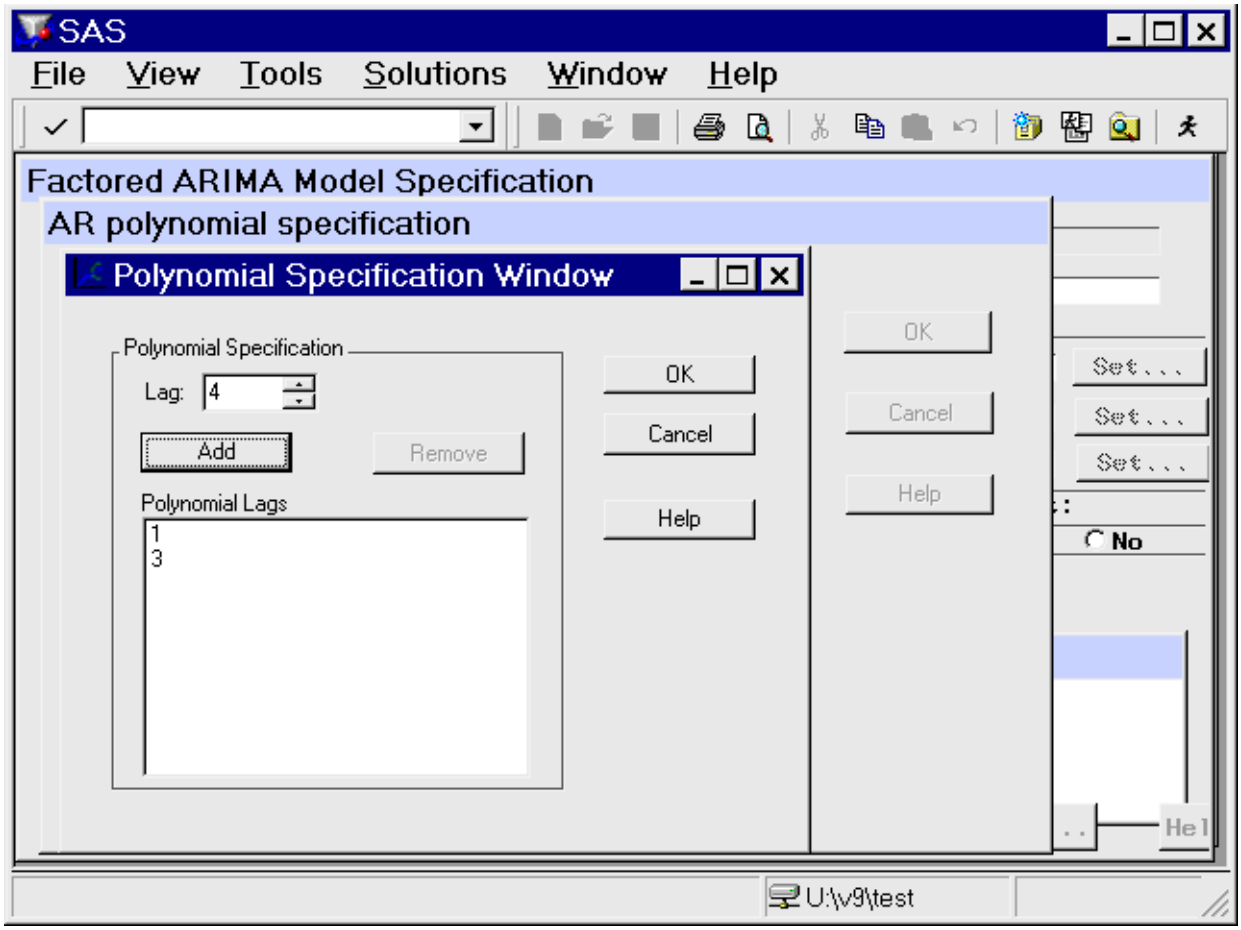
Figure 56.17 AR Polynomial Specification Window



To add AR polynomial terms, select the New button. This opens the Polynomial Specification Window, shown in Figure 56.18. Specify the first lag you want to include by using the `Lag` spin box, then select the Add button. Repeat this process, adding each lag you want to include in the current list. All lags must be specified. For example, if you add only lag 3, the model contains only lag 3, not 1 through 3.

As an example, add lags 1 and 3, then select the OK button. The AR Polynomial Specification Window now shows (1,3) in the list of polynomials. Now select "New" again. Add lags 6 and 12 and select "OK". Now the AR Polynomial Specification Window shows (1,3) and (6,12) as shown in Figure 56.17. Select "OK" to close this window. The Factored ARIMA Model Specification Window now shows the factored model `p=(1,3)(6,12)`. Use the same technique to specify the q terms, or moving-average part of the model. There is no limit to the number of lags or the number of factors you can include in the model.

Figure 56.18 Polynomial Specification Window



To specify differencing lags, select the middle Set button to open the Differencing Specification window. Specify lags using the spin box and add them to the list with the Add button. When you select "OK" to close the window, the differencing lags appear after `d=` in the Factored ARIMA Specification Window, within a single pair of parentheses.

You can use the Factored ARIMA Model Specification Window to specify any model that you can specify with the ARIMA Model and Custom Model windows, but the notation is more similar to that of the ARIMA procedure (see Chapter 8, "The ARIMA Procedure"). Consider as an example the classic Airline model fit to the International Airline Travel series, `SASHELP.AIR`. This is a factored model with one moving-average term at lag one and one moving-average term at the seasonal lag, with first-order differencing at the simple and seasonal lags. Using the ARIMA Model Specification Window, you specify the value 1 for the q and d terms and also for the Q and D terms, which represent the seasonal lags. For monthly data, the seasonal lags represent lag 12, since a yearly seasonal cycle is assumed.

By contrast, the Factored ARIMA Model Specification Window makes no assumptions about seasonal cycles. The Airline model is written as `IMA d=(1,12) q=(1)(12) NOINT`. To specify the differencing terms, add the values 1 and 12 in the Differencing Specification Window and select OK. Then select "New" in the MA Polynomial Specification Window, add the value 1, and select OK. To add the factored term, select "New" again, add the value 12, and select OK. Remember to select "No" in the Intercept radio box, since it is not selected by default. Select OK to close the Factored ARIMA Model Specification Window and fit the model.

You can show that the results are the same as they are when you specify the model by using the ARIMA Model Specification Window and when you select Airline Model from the default model list. If you are familiar with the ARIMA Procedure (Chapter 8, "The ARIMA Procedure"), you might want to turn on the `Show Source Statements` option before fitting the model, then examine the procedure source statements in the log window after fitting the model.

The strength of the Factored ARIMA Specification approach lies in its ability to construct unusual ARIMA models, such as:

Subset models
: These are models of order n, where fewer than n lags are specified. For example, an AR order 3 model might include lags 1 and 3 but not lag 2.

Unusual seasonal cycles
: For example, a monthly series might cycle two or four times per year instead of just once.

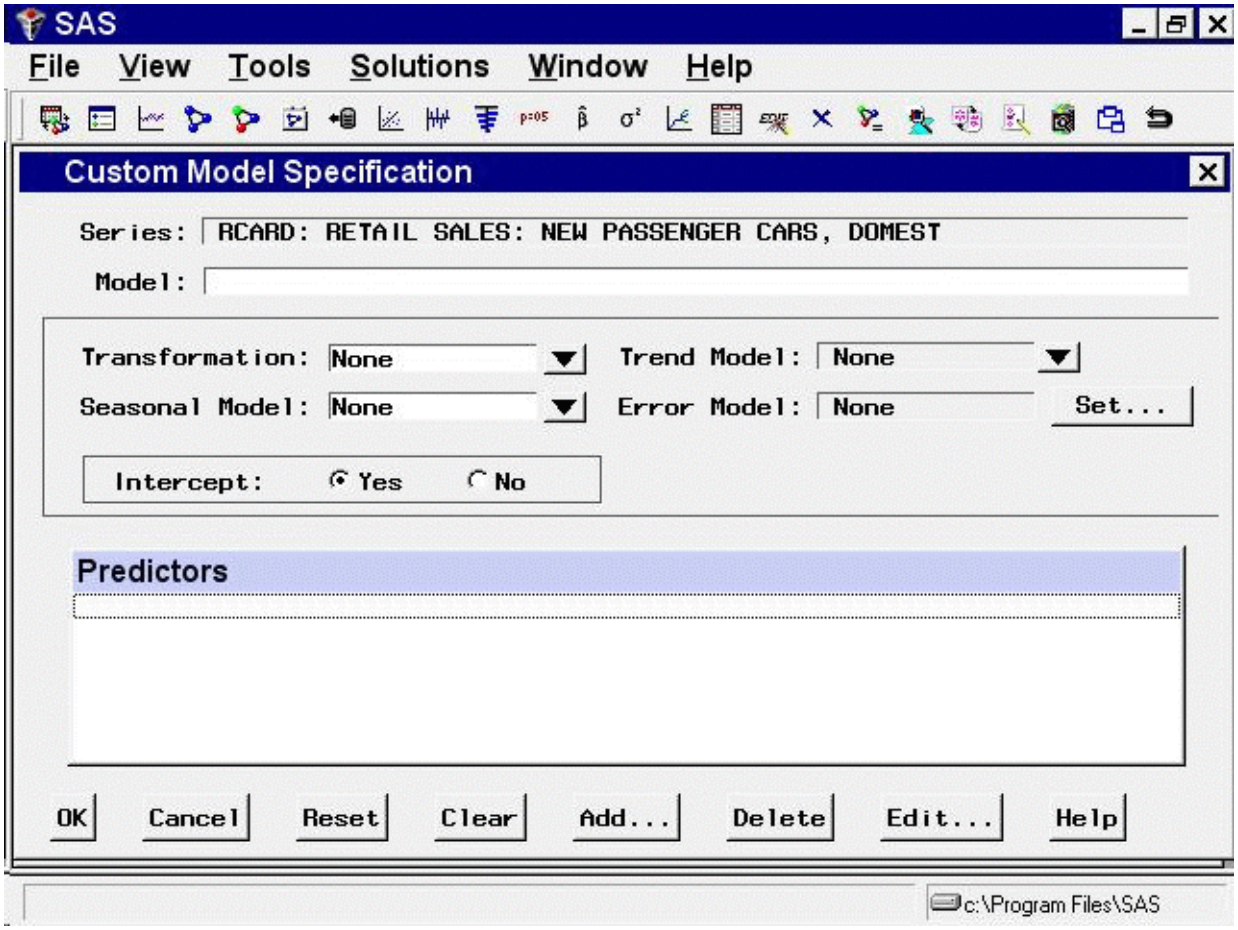
Multiple cycles
: For example, a daily sales series might peak on a certain day each week and also once a year at the Christmas season. Given sufficient data, you can fit a three-factor model, such as `IMA d=(1) q=(1)(7)(365)`.

Models with high order lags take longer to fit and often fail to converge. To save time, select the Conditional Least Squares or Unconditional Least Squares estimation method (see Figure 56.16). Once you have narrowed down the list of candidate models, change to the Maximum Likelihood estimation method.

Custom Model Specification Window

To fit a custom time series model not already provided in the Models to Fit window, select the Custom Model item from the pop-up menu, toolbar, or Edit menu. This opens the Custom Model Specification window, as shown in Figure 56.19.

Figure 56.19 Custom Model Specification Window



You can specify the same time series models with the Custom Model Specification window and the ARIMA Model Specification window, but the windows are structured differently, and you might find one more convenient than the other.

At the top of the Custom Model Specification window is the name and label of the series and the label of the model you are specifying. The model label is filled in with an automatically generated label as you specify options. You can type over the automatic label with your own label for the model. To restore the automatic label, enter a blank label.

The middle part of the Custom Model Specification window consists of four fields: `Transformation`, `Trend Model`, `Seasonal Model`, and `Error Model`. These fields allow you to specify the model in four parts. Each part specifies how a different aspect of the pattern of the time series is modeled and predicted.

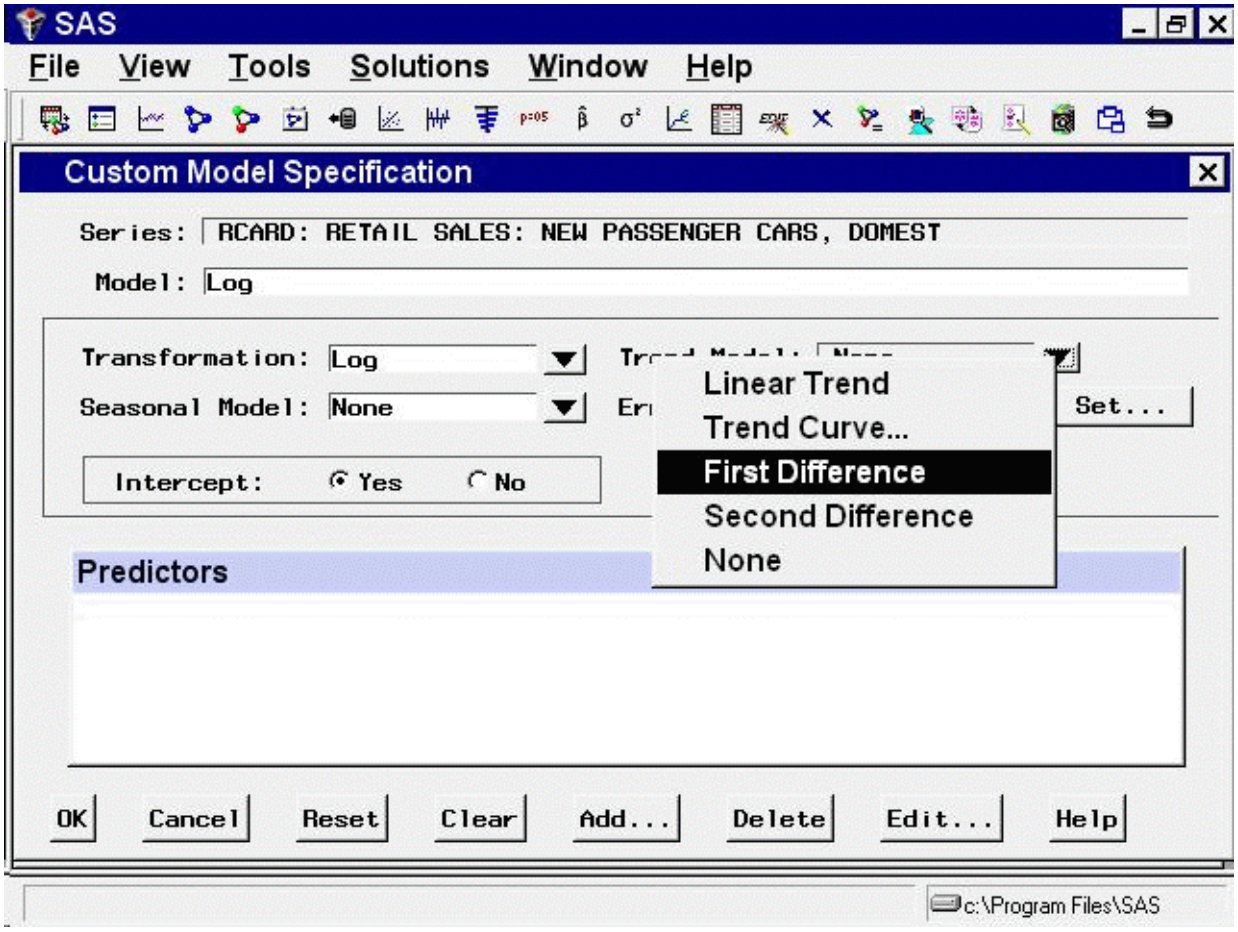
The `Predictors` list at the bottom of the Custom Model Specification window allows you to include different kinds of predictor variables in the forecasting model. The Predictors feature for the Custom Model Specification window is like the Predictors feature for the ARIMA Model Specification window, except that time trend predictors are provided through the Trend Model field and seasonal dummy variable predictors are provided through the Seasonal Model field.

To illustrate how to use the Custom Model Specification window, the following example specifies the same model you fit by using the ARIMA Model Specification window.

First, specify the data transformation to use. Select “Log” using the Transformation combo box.

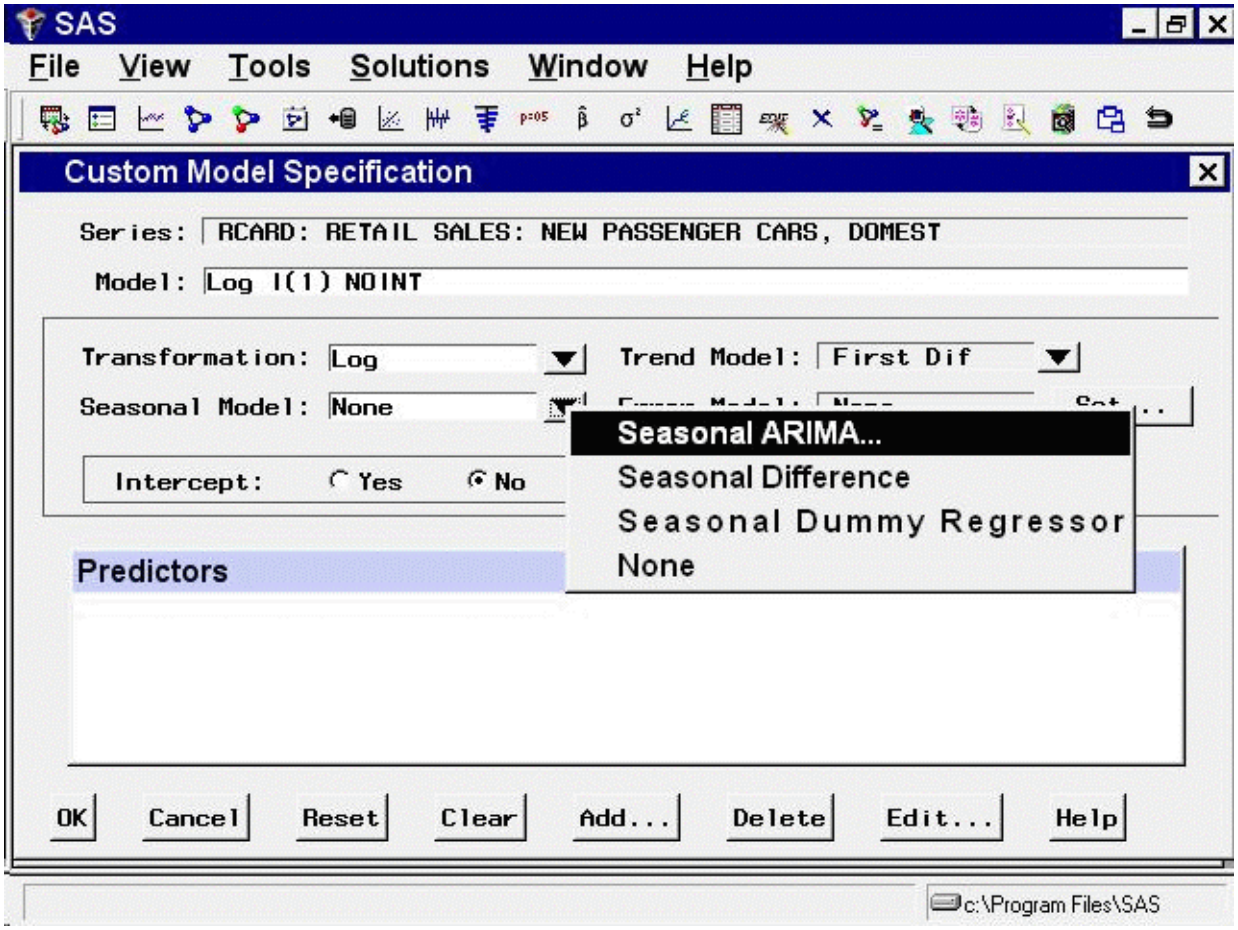
Second, specify how to model the trend in the series. Select `First Difference` in the `Trend Model` combo box, as shown in Figure 56.20.

Figure 56.20 Trend Model Options



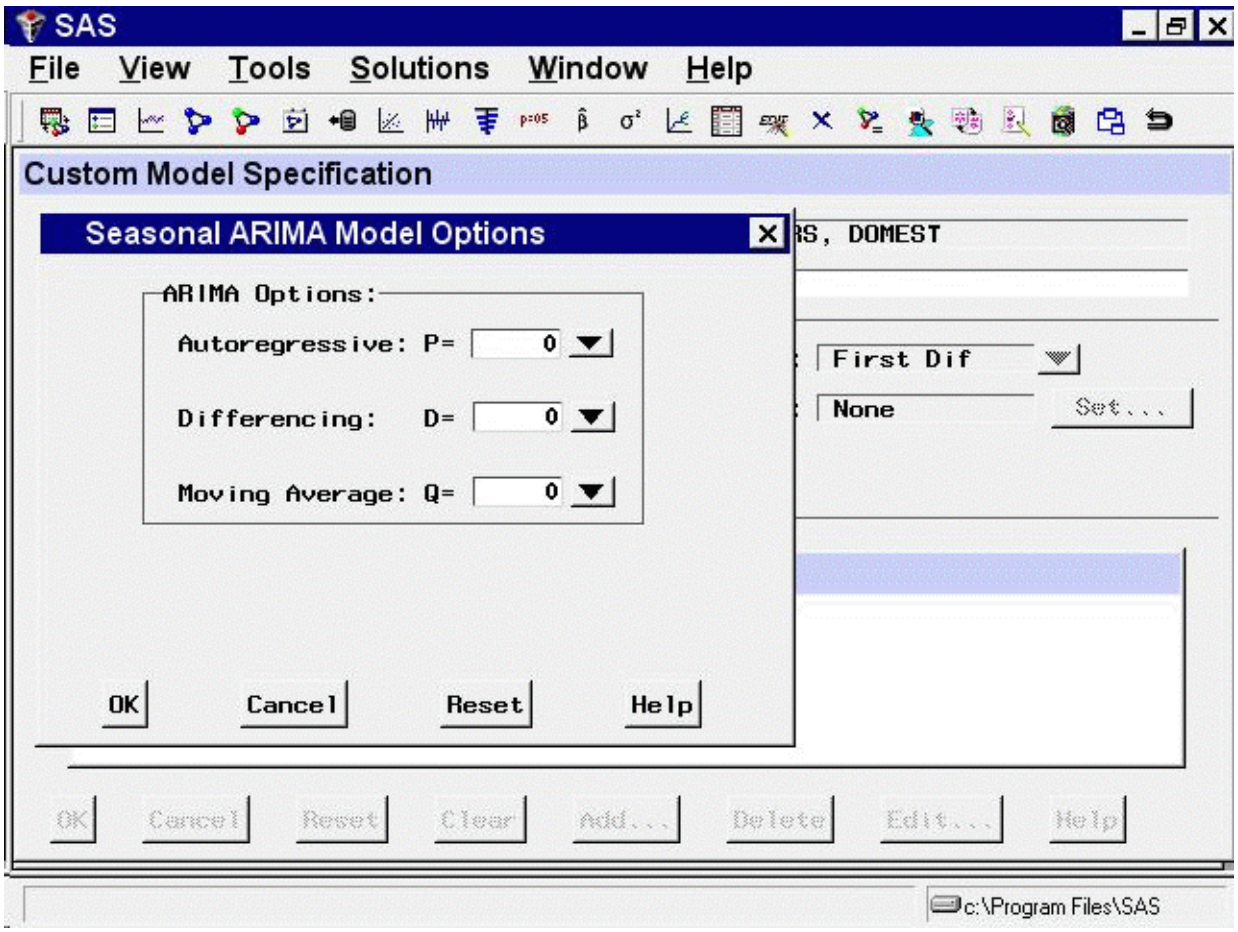
Next, specify how to model the seasonal pattern in the series. Select “Seasonal ARIMA” in the Seasonal Model combo box, as shown in Figure 56.21.

Figure 56.21 Seasonal Model Options



This opens the Seasonal ARIMA Model Options window, as shown in Figure 56.22.

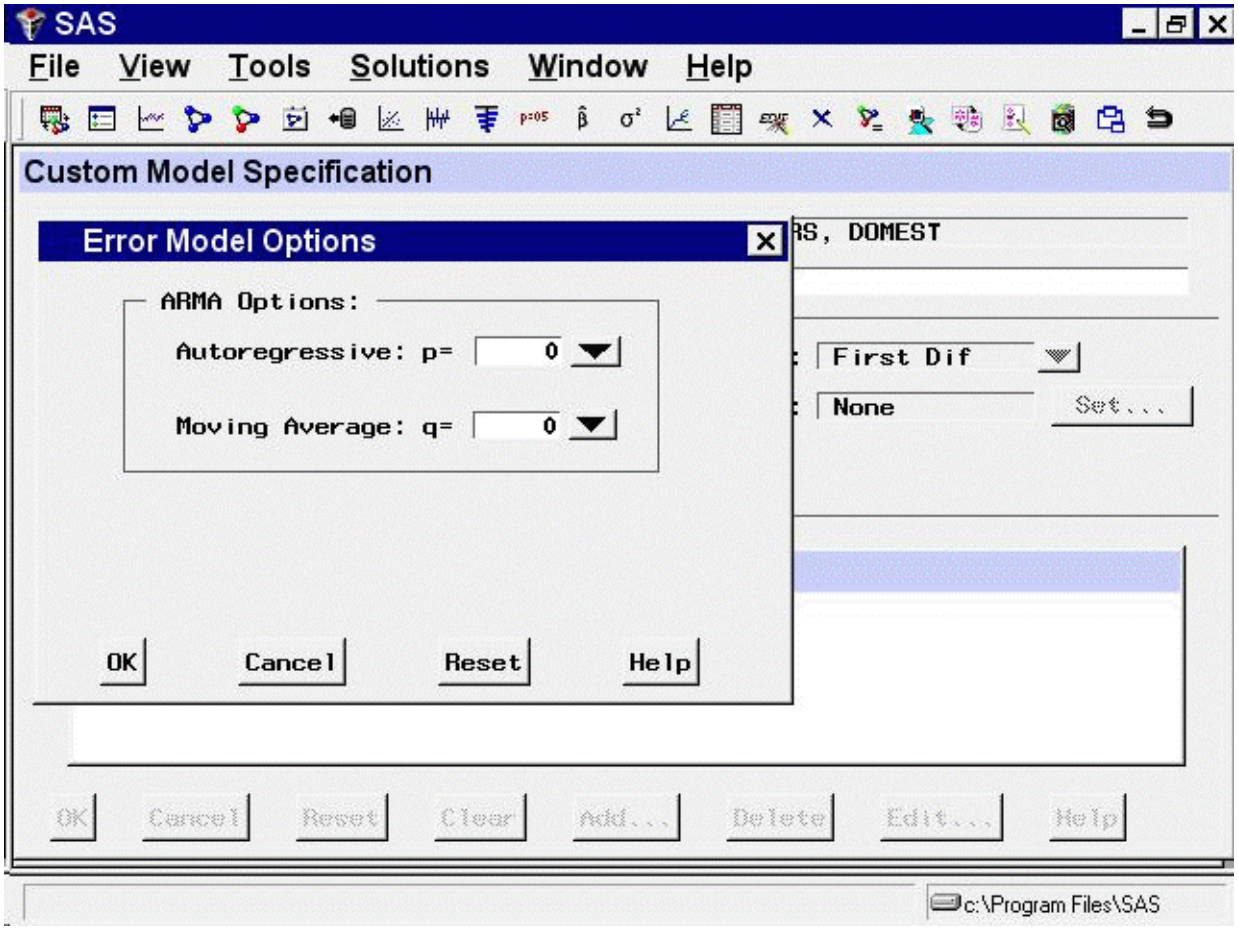
Figure 56.22 Seasonal ARIMA Model Options



Specify a first-order seasonal moving-average term by typing `1` or by selecting "1" from the Moving Average: Q= combo box pop-up menu, and then select the OK button.

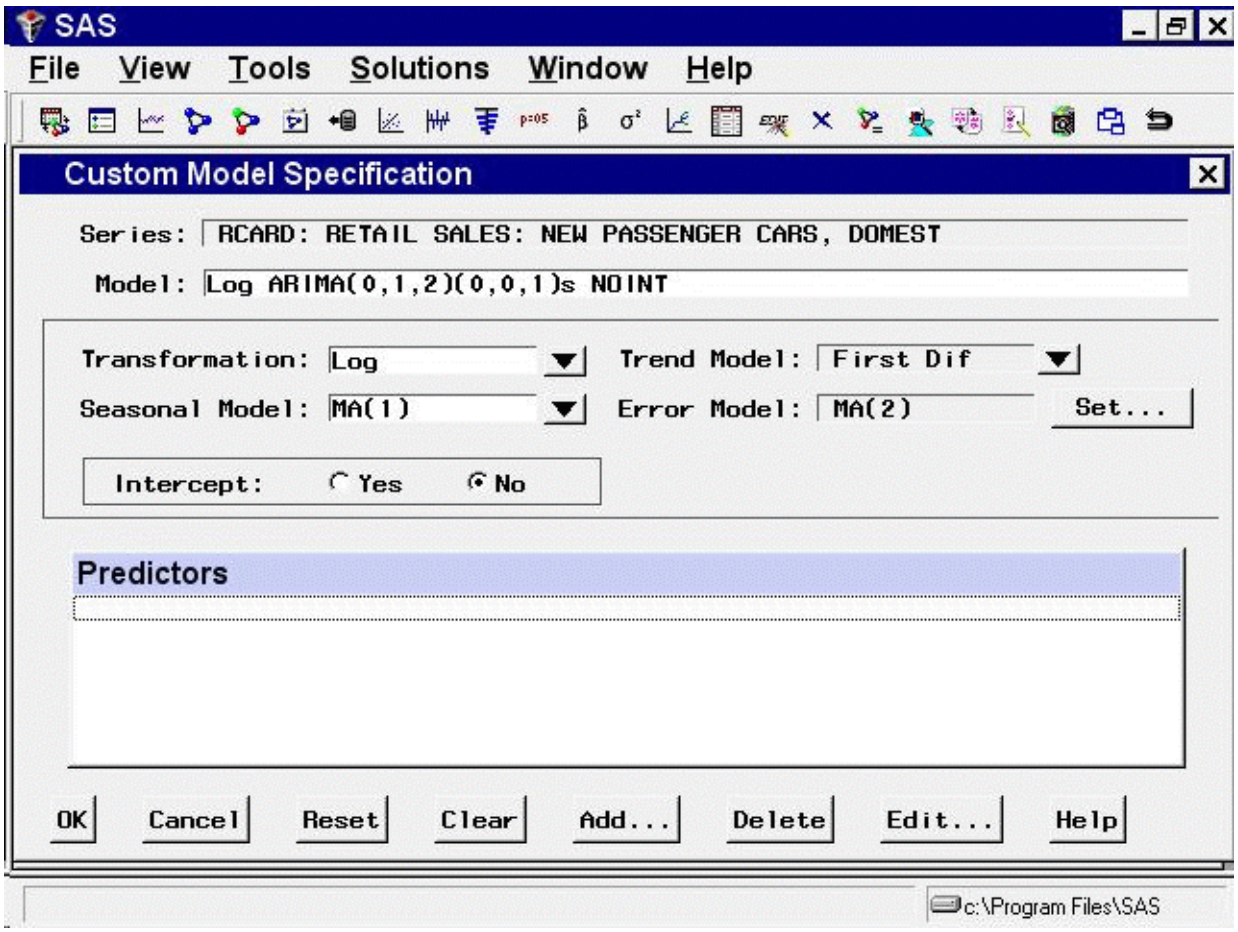
Finally, specify how to model the autocorrelation pattern in the model prediction errors. Select the Set button to the right of the Error Model field. This opens the Error Model Options window, as shown in Figure 56.23. This window allows you to specify an ARMA error process. Set the Moving Average order `q` to `2`, and then select the OK button.

Figure 56.23 Error Model Options



The Custom Model Specification window should now appear as shown in Figure 56.24. The model label at the top of the Custom Model Specification window should now read `Log ARIMA(0,1,2)(0,0,1)s NOINT`, just as it did when you used the ARIMA Model Specification window.

Figure 56.24 Log ARIMA(0,1,2)(0,0,1)s Specified



Now that you have seen how the Custom Model Specification window works, select “Cancel” to exit the window without fitting the model. This should return you to the Develop Models window.

Editing the Model Selection List

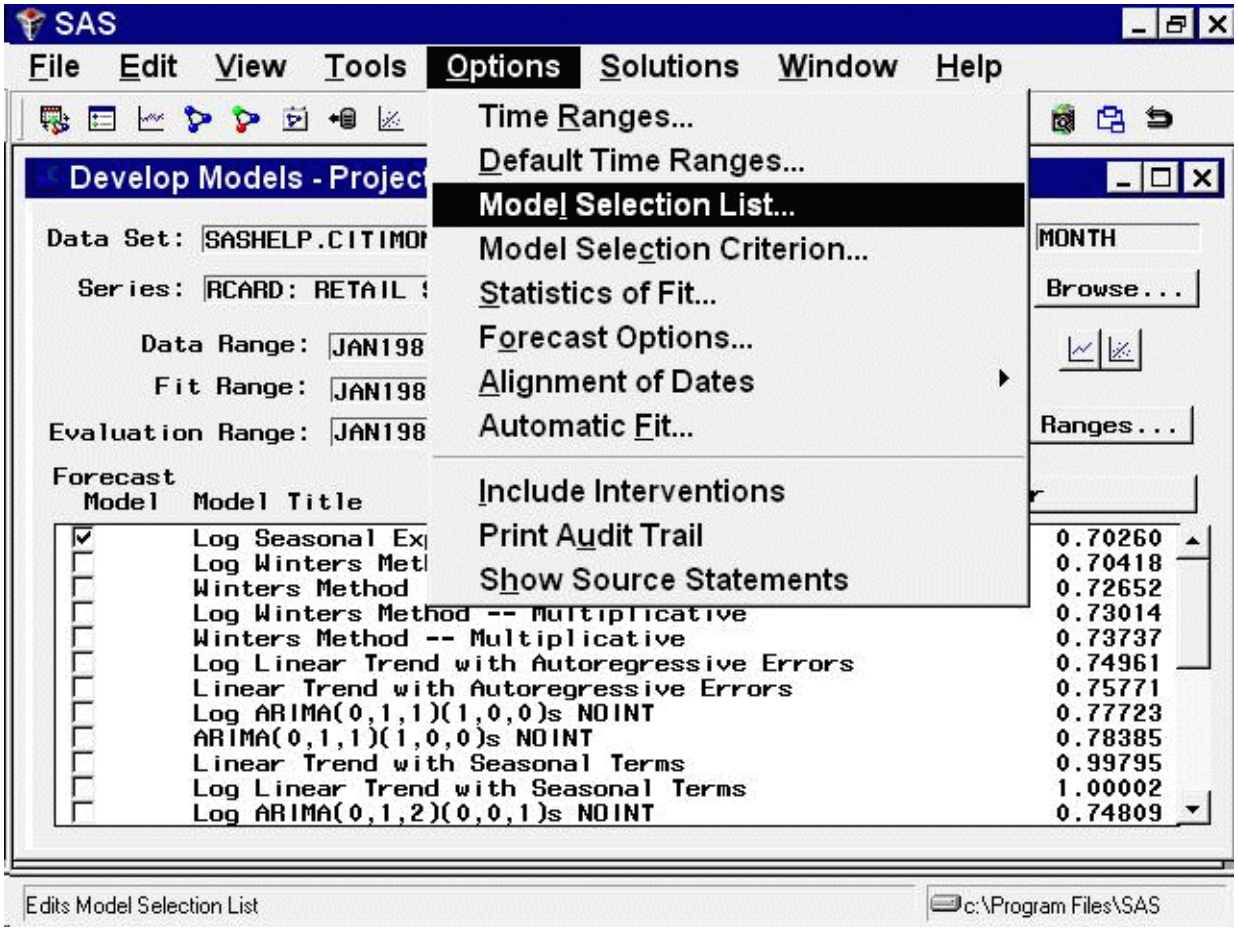
Now that you know how to specify new models that are not included in the system default model selection list, you can edit the model selection list to add models that you expect to use in the future or to delete models that you do not expect to use. When you save the forecasting project to a SAS catalog, the edited model selection list is saved with the project file, and the list is restored when you load the project.

There are two reasons why you would add a model to the model selection list. First, by adding the model to the list, you can fit the model to different time series by selecting it through the `Fit Models from List` action. You do not need to specify the model again every time you use it.

Second, once the model is added to the model selection list, it is available to the automatic model selection process. The model is then considered automatically whenever you use the automatic model selection feature for any series.

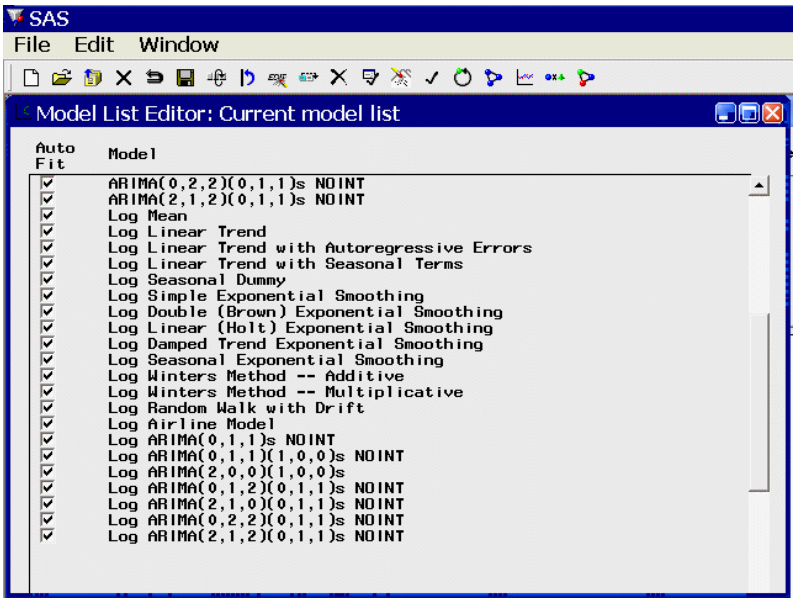
To edit the model selection list, select "Model Selection List" from the Options menu as shown in Figure 56.25, or select the Edit Model List toolbar icon.

Figure 56.25 Model Selection List Option



This selection brings up the Model Selection List editor window, as shown in Figure 56.26. This window consists of the model selection list and an "Auto Fit" column, which controls for each model whether the model is included in the list of models used by the automatic model selection process.

Figure 56.26 Model Selection List Window

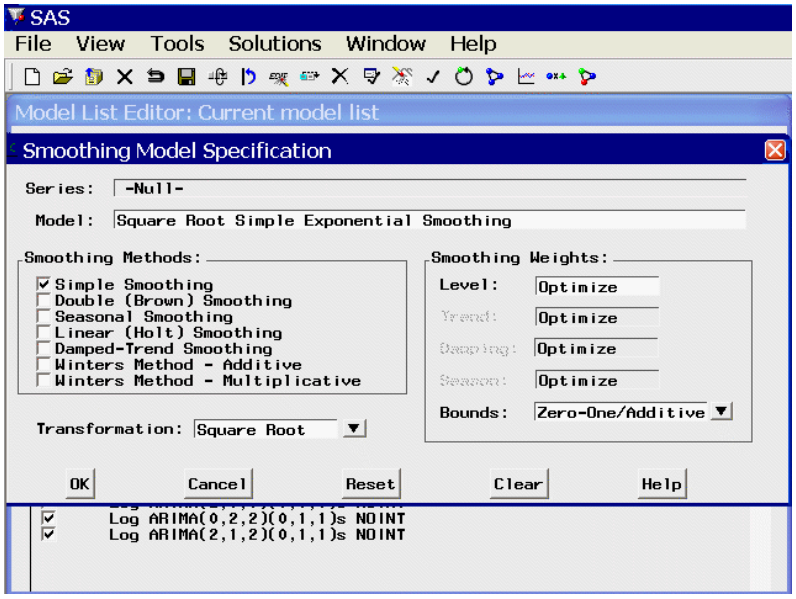


To add a model to the list, select "Add Model" from the Edit menu and then select "Smoothing Model," "ARIMA Model," "Factored ARIMA Model," or "Custom Model" from the submenu. Alternatively, click the corresponding icon on the toolbar.

As an example, select "Smoothing Model." This brings up the Smoothing Model Specification window. Note that the series name is "-Null-." This means that you are not specifying a model to be fit to a particular series, but are specifying a model to be added to the selection list for later reference.

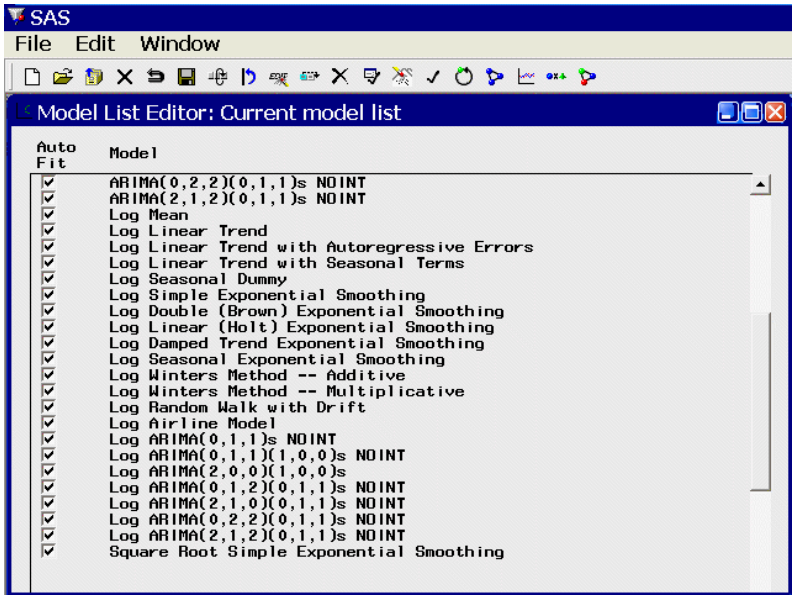
Specify a smoothing model. For example, select "Simple Smoothing" and then select the Square Root transformation. The window appears as shown in Figure 56.27.

Figure 56.27 Adding a Model Specification



Select the OK button to add the model to the end of the model selection list and return you to the Model Selection List window, as shown in Figure 56.28. You can now select the Fit Models from List model-fitting option to use the edited selection list.

Figure 56.28 Model Added to Selection List



If you want to delete one or more models from the list, select the model labels to highlight them in the list. Click a second time to clear a selected model. Then select "Delete" from the Edit pull-down menu, or the corresponding toolbar icon. As an example, delete the `Square Root Simple Exponential Smoothing` model that you just added.

The Model Selection List editor window gives you a lot of flexibility for managing multiple model lists, as explained in the section "Model Selection List Editor Window" on page 3834. For example, you can create your own model lists from scratch or modify or combine previously saved model lists and those provided with the software, and you can save them and designate one as the default for future projects.

Now select "Close" from the File menu (or the Close icon) to close the Model Selection List editor window.

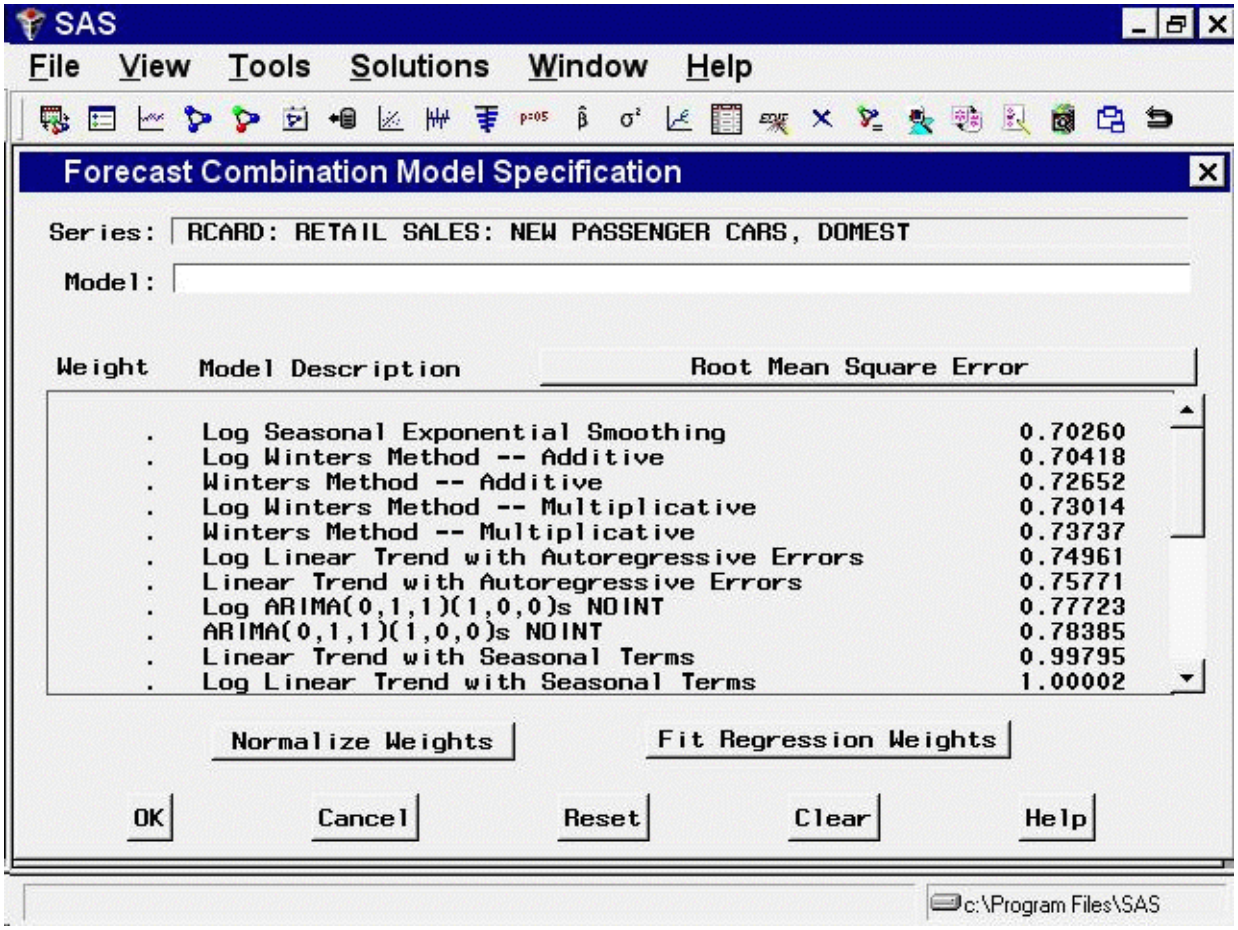
Forecast Combination Model Specification Window

Once you have fit several forecasting models to a series, you face the question of which model to use to produce the final forecasts. One possible answer is to combine or average the forecasts from several models. Combining the predictions from several different forecasting methods is a popular approach to forecasting.

The way that you produce forecast combinations with the Time Series Forecasting System is to use the Forecast Combination Model Specification window to specify a new forecasting model that performs the averaging of forecasts from the models you want to combine. This new model is added to the list of fitted models just like other models. You can then use the Model Viewer window features and Model Fit Comparison window features to examine the fit of the combined model.

To specify a forecast combination model, select "Combine Forecasts" from the pop-up menu or toolbar, or select "Edit" and "Fit Model" from the menu bar. This brings up the Forecast Combination Model Specification window, as shown in Figure 56.29.

Figure 56.29 Forecast Combination Window



At the top of the Forecast Combination window is the name and label of the series and the label of the model you are specifying. The model label is filled in with an automatically generated label as you specify options. You can type over the automatic label with your own label for the model. To restore the automatic label, enter a blank label.

The middle part of the Forecast Combination window consists of the list of models that you have fit to the series. This table shows the label and goodness-of-fit measure for each model and the combining weight assigned to the model.

The `Weight` column controls how much weight is given to each model in the combined forecasts. A missing weight means that the model is not used. Initially, all the models have missing weight values.

You can enter the weight values you want to use in the `Weight` column. Alternatively, you can select models from the Model Description column, and weight values for the models you select are set automatically. To remove a model from the combination, select it again. This resets its weight value to missing.

At the bottom of the Forecast Combination window are two buttons: `Normalize Weights` and `Fit Regression Weights`. The Normalize Weights button adjusts the nonmissing weight values so that they sum to one. The Fit Regression Weights button uses linear regression to compute the weight values that produce the combination of model predictions with the best fit to the series.

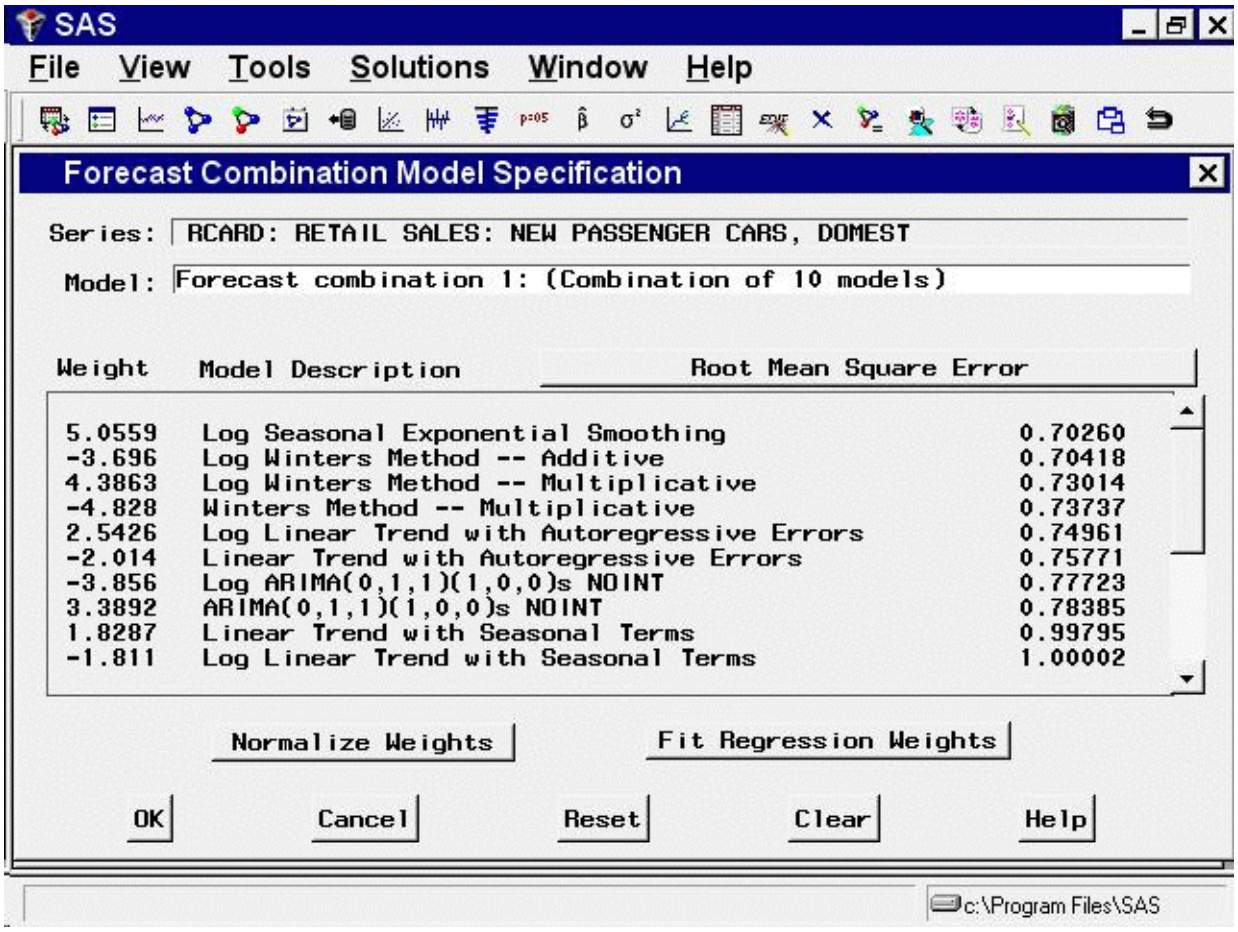
If no models are selected, the Fit Regression Weights button fits weights for all the models in the list. You can compute regression weights for only some of the models by first selecting the models you want to combine and then selecting Fit Regression Weights. In this case, only the nonmissing Weight values are replaced with regression weights.

As an example of how to combine forecasting models, select all the models in the list. After you have finished selecting the models, all the models in the list should now have equal weight values, which implies a simple average of the forecasts.

Now select the Fit Regression Weights button. The system performs a linear regression of the series on the predictions from the models with nonmissing weight values and replaces the weight values with the estimated regression coefficients. These are the combining weights that produce the smallest mean square prediction error within the sample.

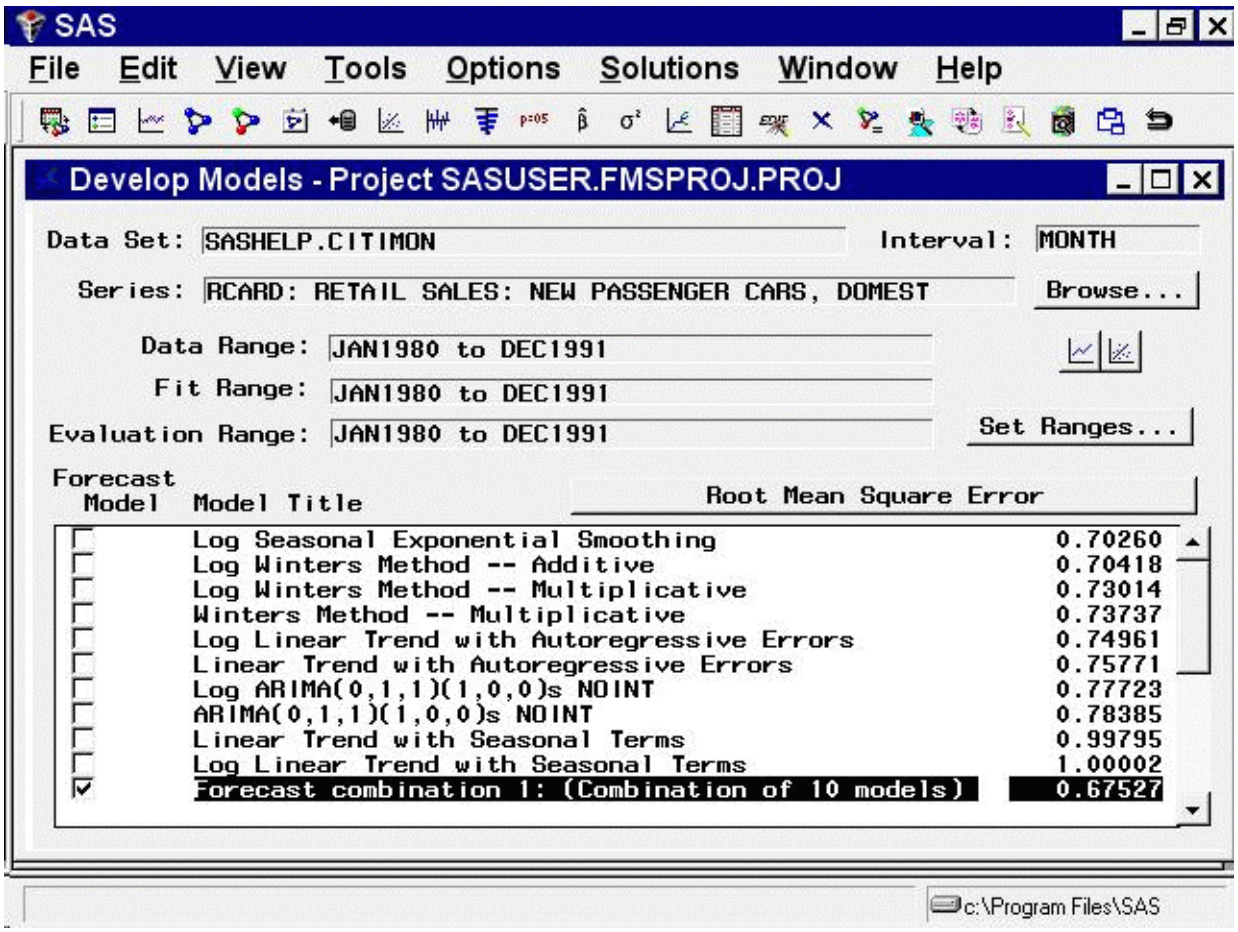
The Forecast Combination window should now appear as shown in Figure 56.30. (Note that some of the regression weight values are negative.)

Figure 56.30 Combining Models



Select the OK button to fit the combined model. Now the Develop Models window shows this model to be the best fitting according to the root mean square error, as shown in Figure 56.31.

Figure 56.31 Develop Models Window Showing All Models Fit



Notice that the combined model has a smaller root mean square error than any one of the models included in the combination. The confidence limits for forecast combinations are produced by taking a weighted average of the mean square prediction errors for the component forecasts, ignoring the covariance between the prediction errors.

Incorporating Forecasts from Other Sources

You might have forecasts from other sources that you want to include in the forecasting process. Examples of other forecasts you might want to use are “best guess” forecasts based on personal judgments, forecasts produced by government agencies or commercial forecasting services, planning scenarios, and reference or “base line” projections. Because such forecasts are produced externally to the Time Series Forecasting System, they are referred to as external forecasts.

You can include external forecasts in combination models to produce compromise forecasts that split the difference between the external forecast and forecasting models that you fit. You can use external forecasts to compare them to the forecasts from models that are fit by the system.

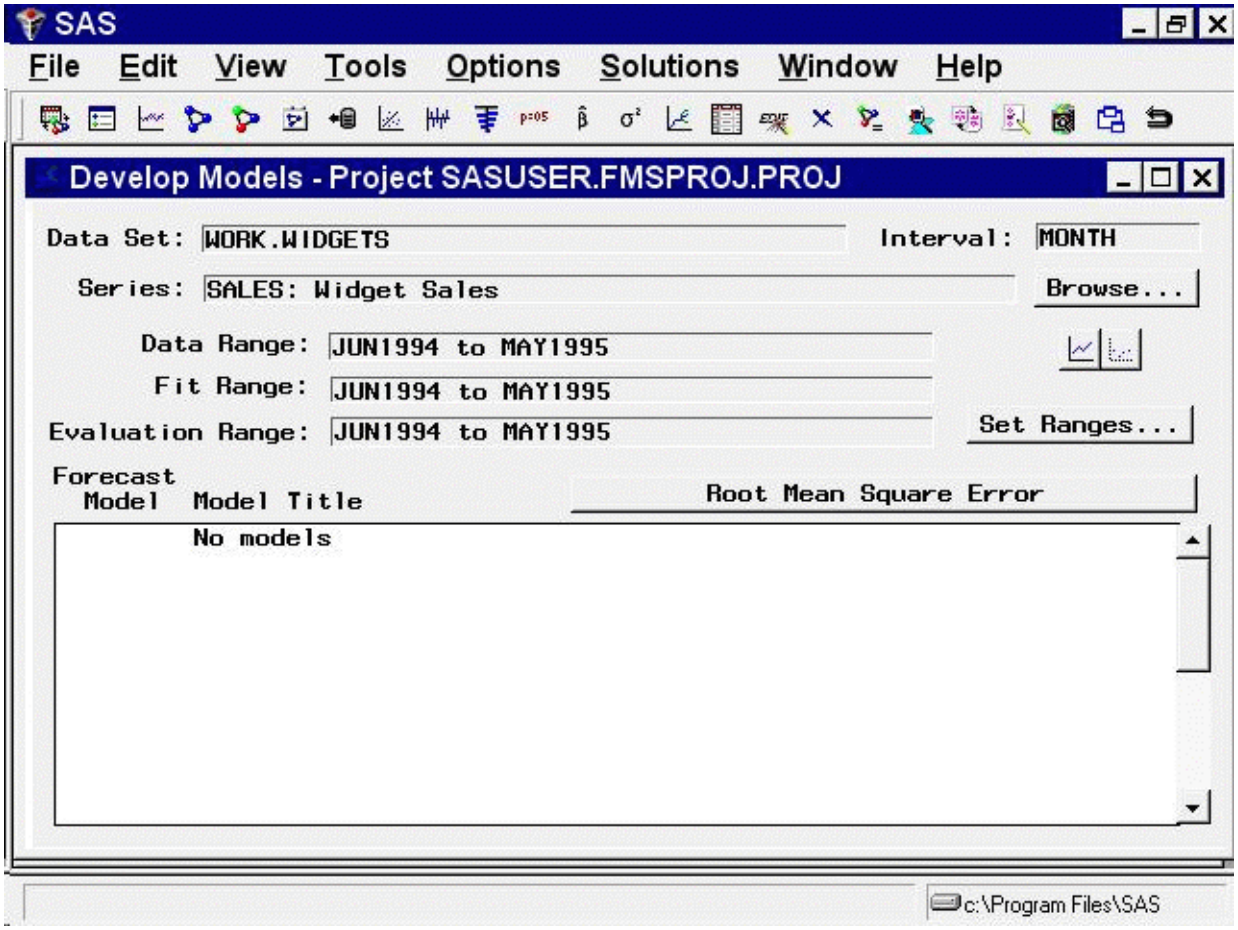
To include external forecasts in the Time Series Forecasting process, you must first supply the external forecast as a variable in the input data set. You then specify a special kind of forecasting "model" whose predictions are identical to the external forecast recorded in the data set.

As an example, suppose you have 12 months of sales data and five months of sales forecasts based on a consensus opinion of the sales staff. The following statements create a SAS data set containing made-up numbers for this situation.

```
data widgets;
   input date monyy5. sales staff;
   format date monyy5.;
   label sales = "Widget Sales"
         staff = "Sales Staff Consensus Forecast";
   datalines;
jun94  142.1    .
jul94  139.6    .
aug94  145.0    .
sep94  150.2    .
oct94  151.1    .
nov94  154.3    .
dec94  158.7    .
jan95  155.9    .
feb95  159.2    .
mar95  160.8    .
apr95  162.0    .
may95  163.3    .
jun95     .  166.
jul95     .  168.
aug95     .  170.
sep95     .  171.
oct95     .  177.
run;
```

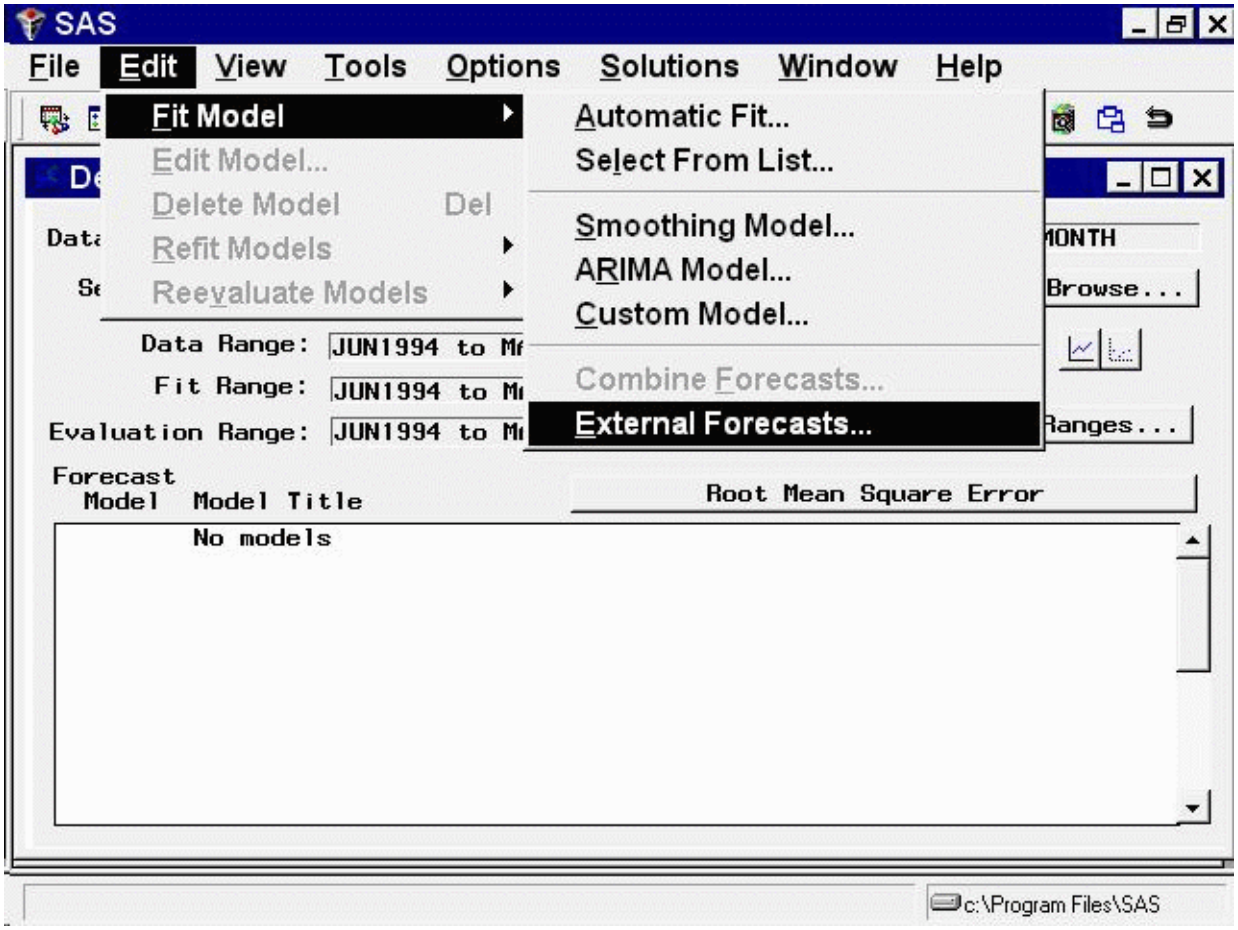
Submit the preceding statements in the SAS Program Editor window. From the Time Series Forecasting window, select "Develop Models." In the Series Selection window, select the data set WORK.WIDGETS and the variable SALES. The Develop Models window should now appear as shown in Figure 56.32.

Figure 56.32 Develop Models Window



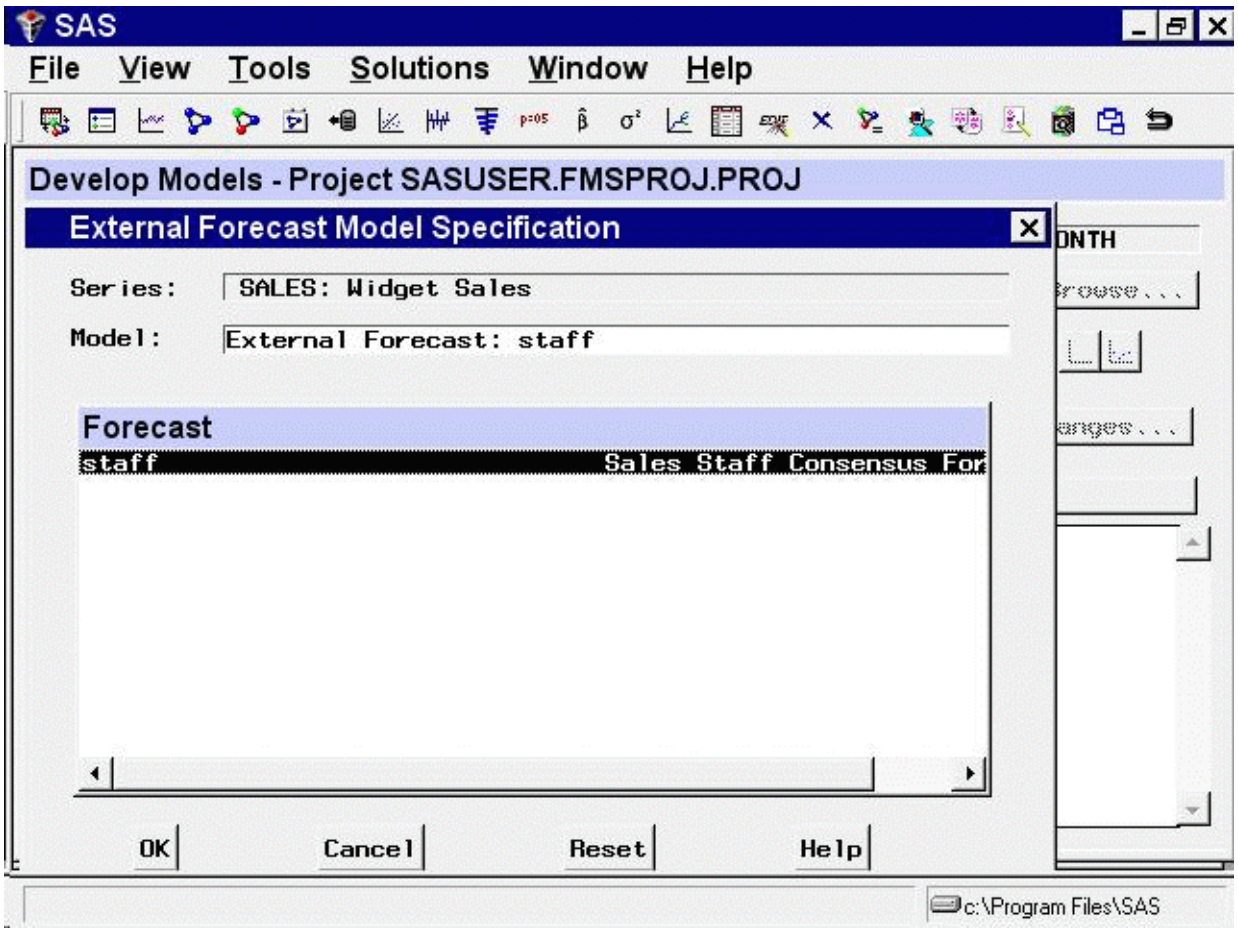
Now select "Edit," "Fit Model," and "External Forecasts" from the menu bar of the Develop Models window, as shown in Figure 56.33, or the `Use External Forecasts` toolbar icon.

Figure 56.33 Adding a Model for an External Forecast Series



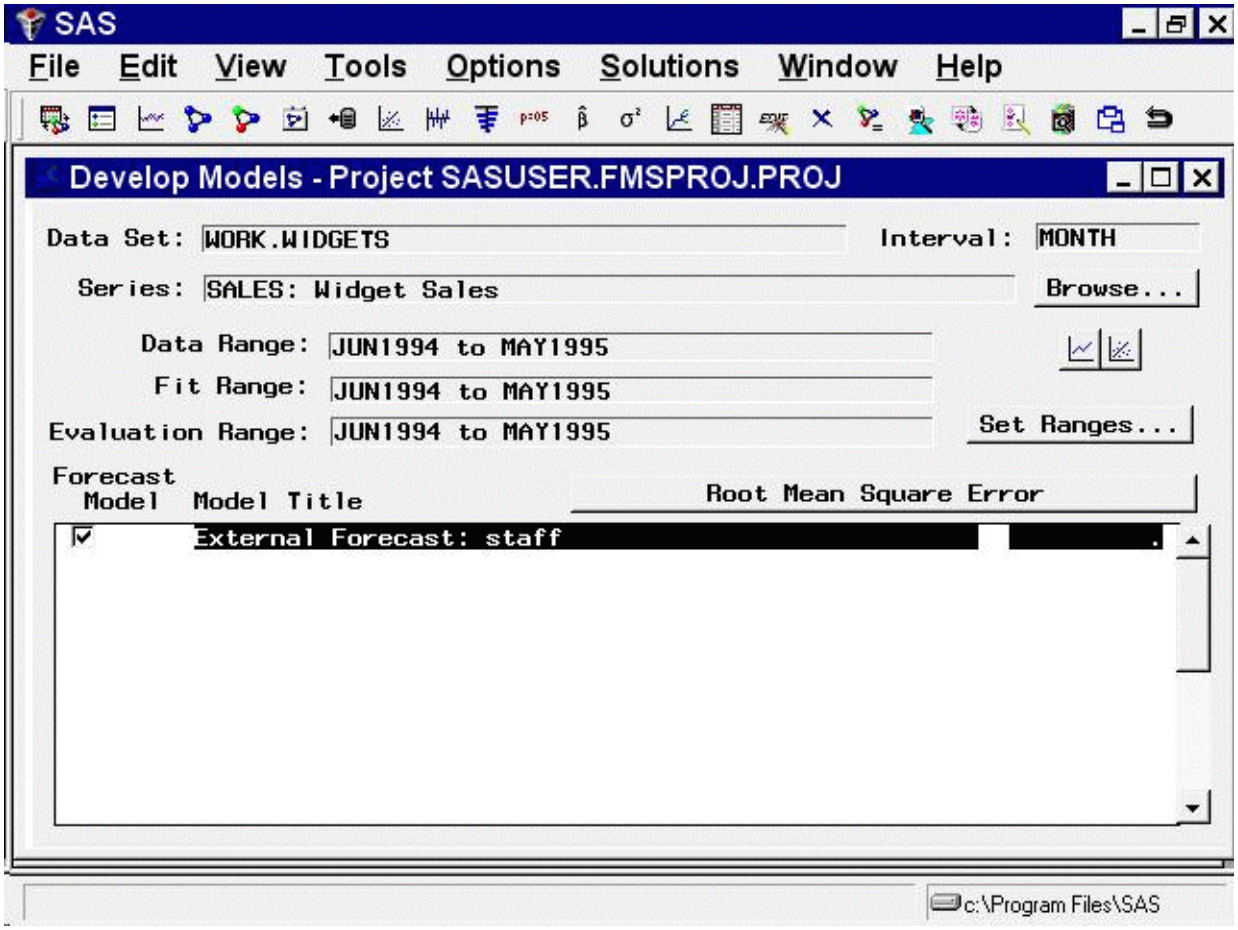
This selection opens the External Forecast Model Specification window. Select the STAFF variable as shown in Figure 56.34.

Figure 56.34 External Forecast Series Selected



Select the OK button. The external forecast model is now "fit" and added to the Develop Models list, as shown in Figure 56.35.

Figure 56.35 Model for External Forecast



You can now use this model for comparison with the predictions from other forecasting models that you fit, or you can include it in a forecast combination model.

Note that no fitting is actually performed for an external forecast model. The predictions of the external forecast model are simply the values of the external forecast series read from the input data set. The goodness-of-fit statistics for such models will depend on the values that the external forecast series contains for observations within the period of fit. In this case, no STAFF values are given for past periods, and therefore the fit statistics for the model are missing.

Chapter 57
Choosing the Best Forecasting Model

Contents

Time Series Viewer Features 3719
Model Viewer Prediction Error Analysis 3726
The Model Selection Criterion 3730
Sorting and Selecting Models 3732
Comparing Models 3733
Controlling the Period of Evaluation and Fit 3734
Refitting and Reevaluating Models 3735
Using Hold-out Samples 3735

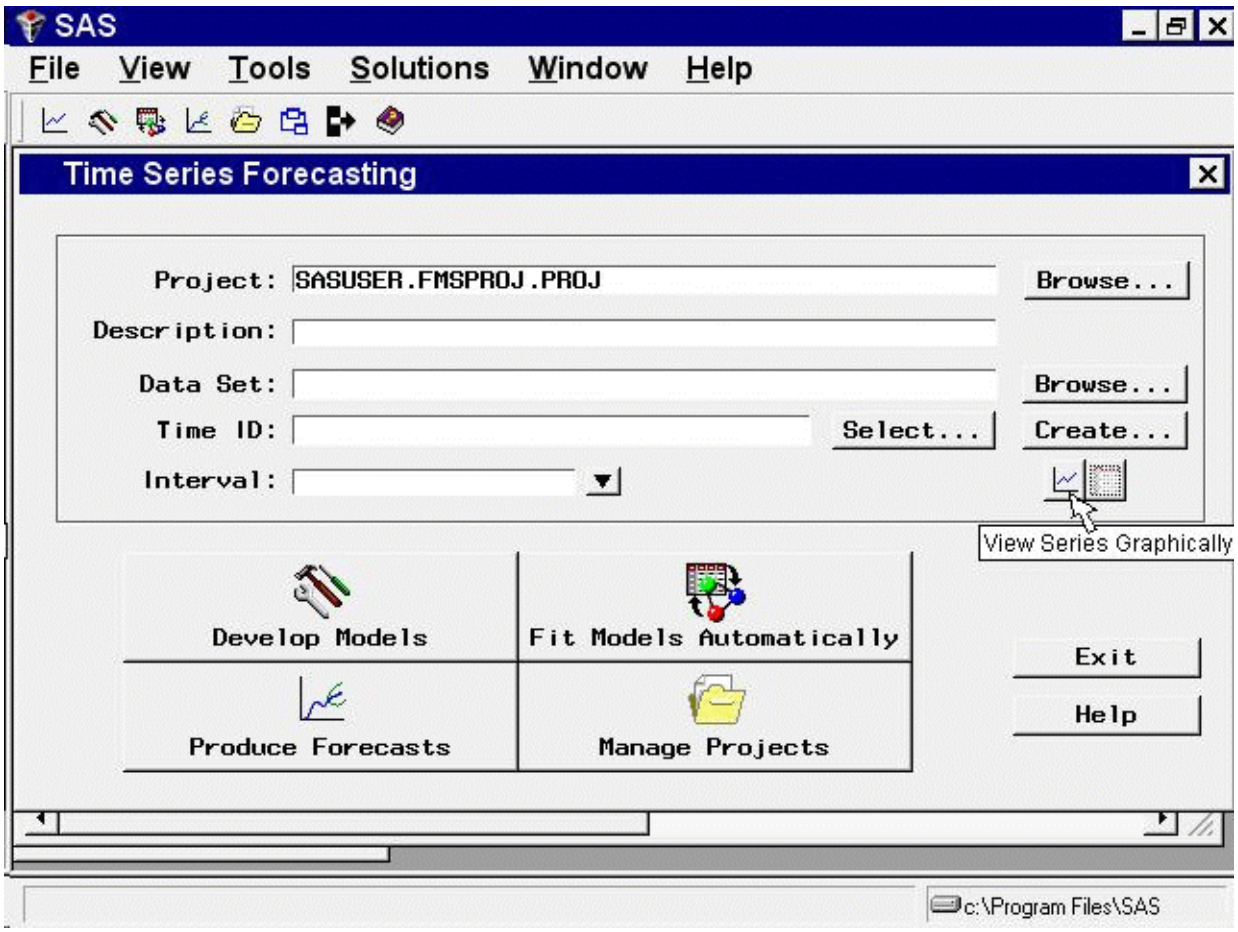
The Time Series Forecasting System provides a variety of tools for identifying potential forecasting models and for choosing the best fitting model. It allows you to decide how much control you want to have over the process, from a hands-on approach to one that is completely automated. This chapter begins with an exploration of the tools available through the Series Viewer and Model Viewer. It presents an example of identifying models graphically and exercising your knowledge of model properties. The remainder of the chapter shows you how to compare models by using a variety of statistics and by controlling the fit and evaluation time ranges. It concludes by showing you how to refit existing models and how to compare models using hold-out samples.

Time Series Viewer Features

The `Time Series Viewer` is a graphical tool for viewing and analyzing time series. It can be used separately from the `Time Series Forecasting System` by using the TSVIEW command or by selecting `Time Series Viewer` from the `Analysis` pull-down menu under `Solutions`.

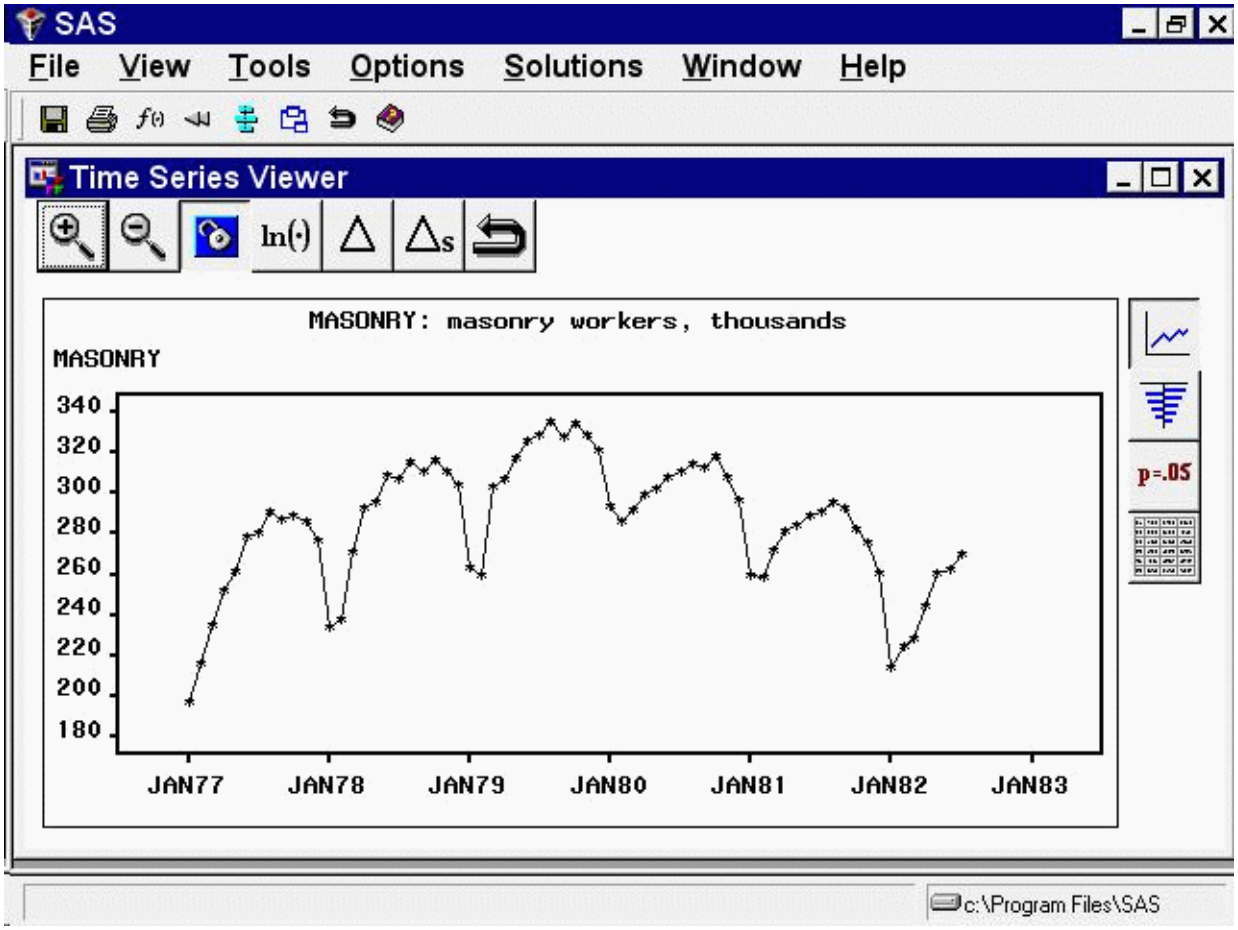
In this chapter you will use the Time Series Viewer to examine plots of your series before fitting models. Begin this example by invoking the Forecasting system and selecting the `View Series Graphically` button, as shown in Figure 57.1, or the `View Series` toolbar icon.

Figure 57.1 Invoking the Time Series Viewer



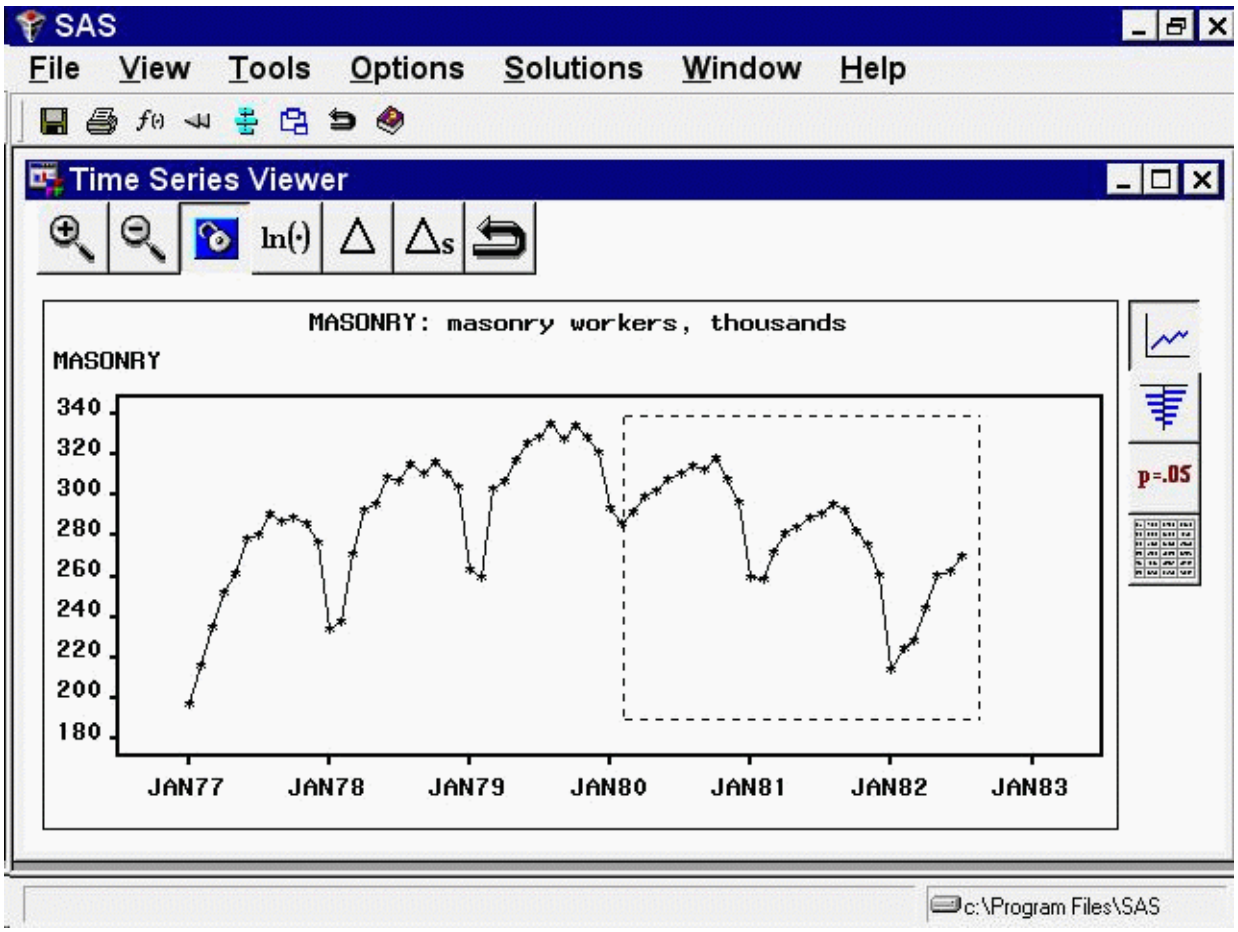
From the Series Selection window, select SASHELP as the library, WORKERS as the data set, and MASONRY as the time series, and then click the `Graph` button. The Time Series Viewer displays a plot of the series, as shown in Figure 57.2.

Figure 57.2 Series Plot



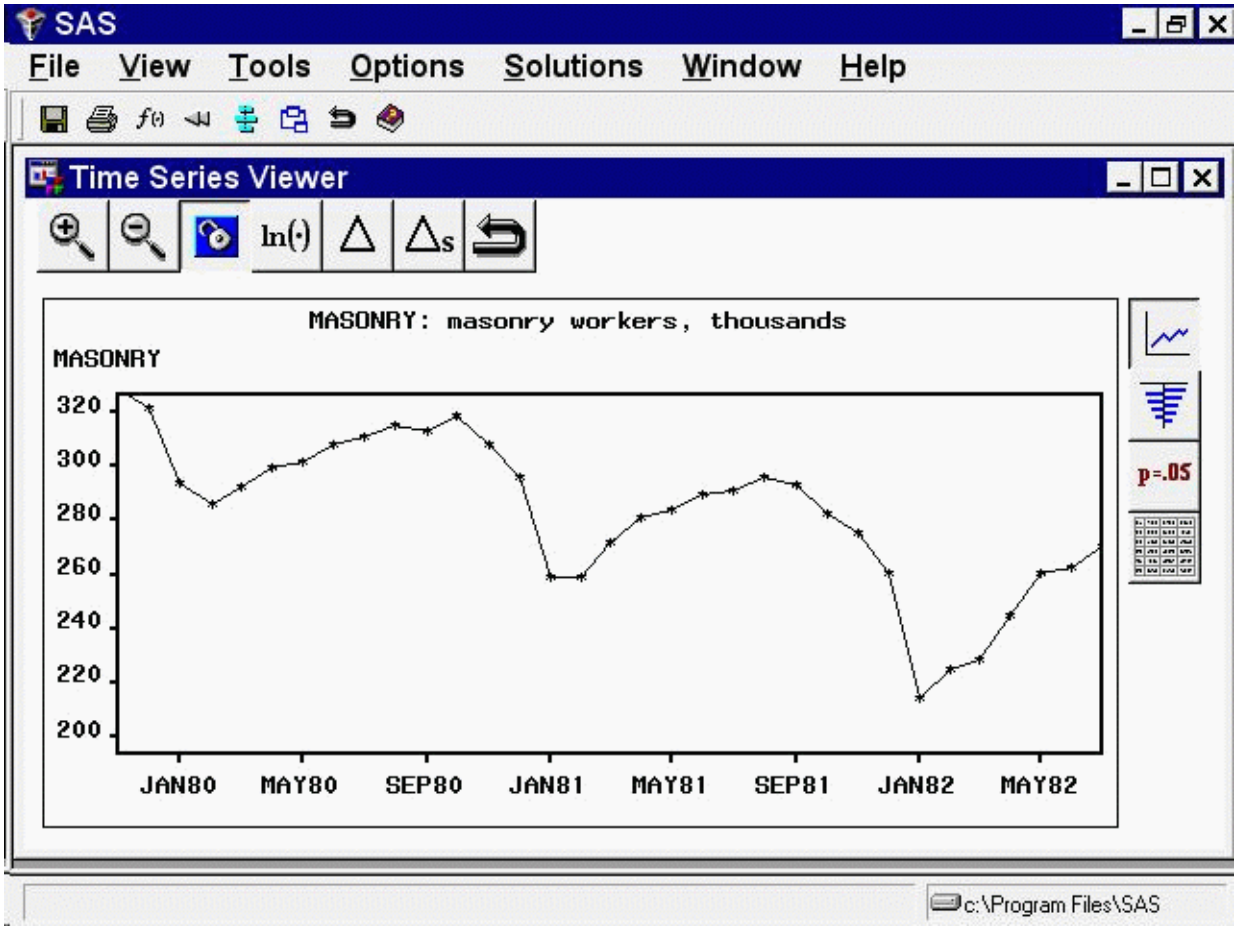
Select the Zoom In icon, the first one on the window's horizontal toolbar. Notice that the mouse cursor changes shape and that "Note: Click on a corner of the region, then drag to the other corner" appears on the message line. Outline an area, as shown in Figure 57.3, by clicking the mouse at the upper-left corner, holding the button down, dragging to the lower-right corner, and releasing the button.

Figure 57.3 Selecting an Area for Zoom



The zoomed plot should appear as shown in Figure 57.4.

Figure 57.4 Zoomed Plot



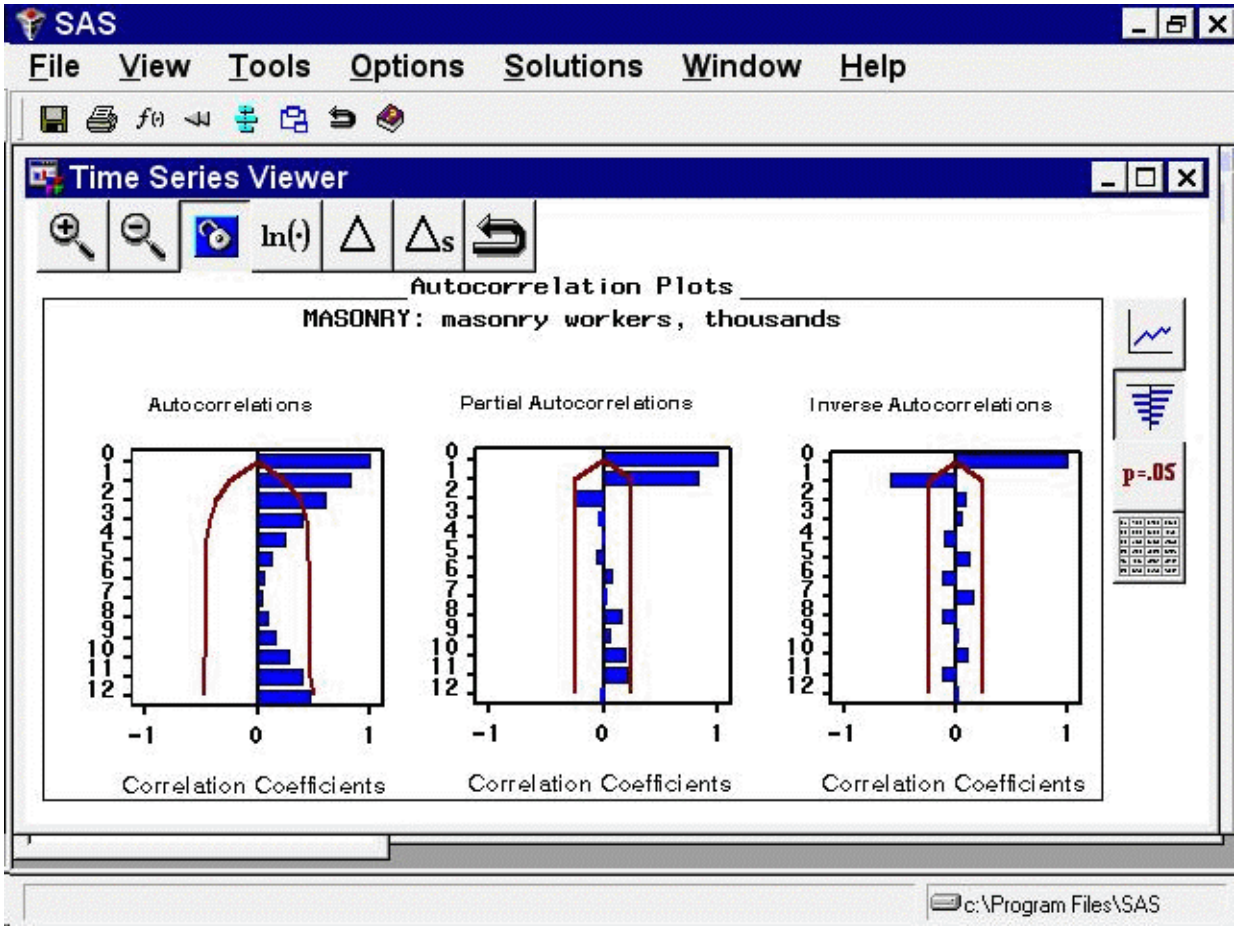
You can repeat the process to zoom in still further. To return to the previous view, select the Zoom Out icon, the second icon on the window's horizontal toolbar.

The third icon on the horizontal toolbar is used to link or unlink the viewer window. By default, the viewer is linked, meaning that it is automatically updated to reflect selection of a different time series. To see this, return to the Series Selection window by clicking on it or using the Window menu or `Next Viewer` toolbar icon. Select the Electric series in the `Time Series Variables` list box. Notice that the Time Series Viewer window is updated to show a plot of the ELECTRIC series. Select the `Link/Unlink` icon if you prefer to unlink the viewer so that it is not automatically updated in this way. Successive selections toggle between the linked and unlinked state. A note on the message line informs you of the state of the Time Series Viewer window.

When a Time Series Viewer window is linked, selecting `View Series` again makes the linked Viewer window active. When no Time Series Viewer window is linked, selecting `View Series` opens an additional Time Series Viewer window. You can bring up as many Time Series Viewer windows as you want.

Having seen the plot in Figure 57.2, you might suspect that the series is nonstationary and seasonal. You can gain further insight into this by examining the sample autocorrelation function (ACF), partial autocorrelation function (PACF), and inverse autocorrelation function (IACF) plots. To switch the display to the autocorrelation plots, select the second icon from the top on the vertical toolbar at the right side of the Time Series Viewer. The plot appears as shown in Figure 57.5.

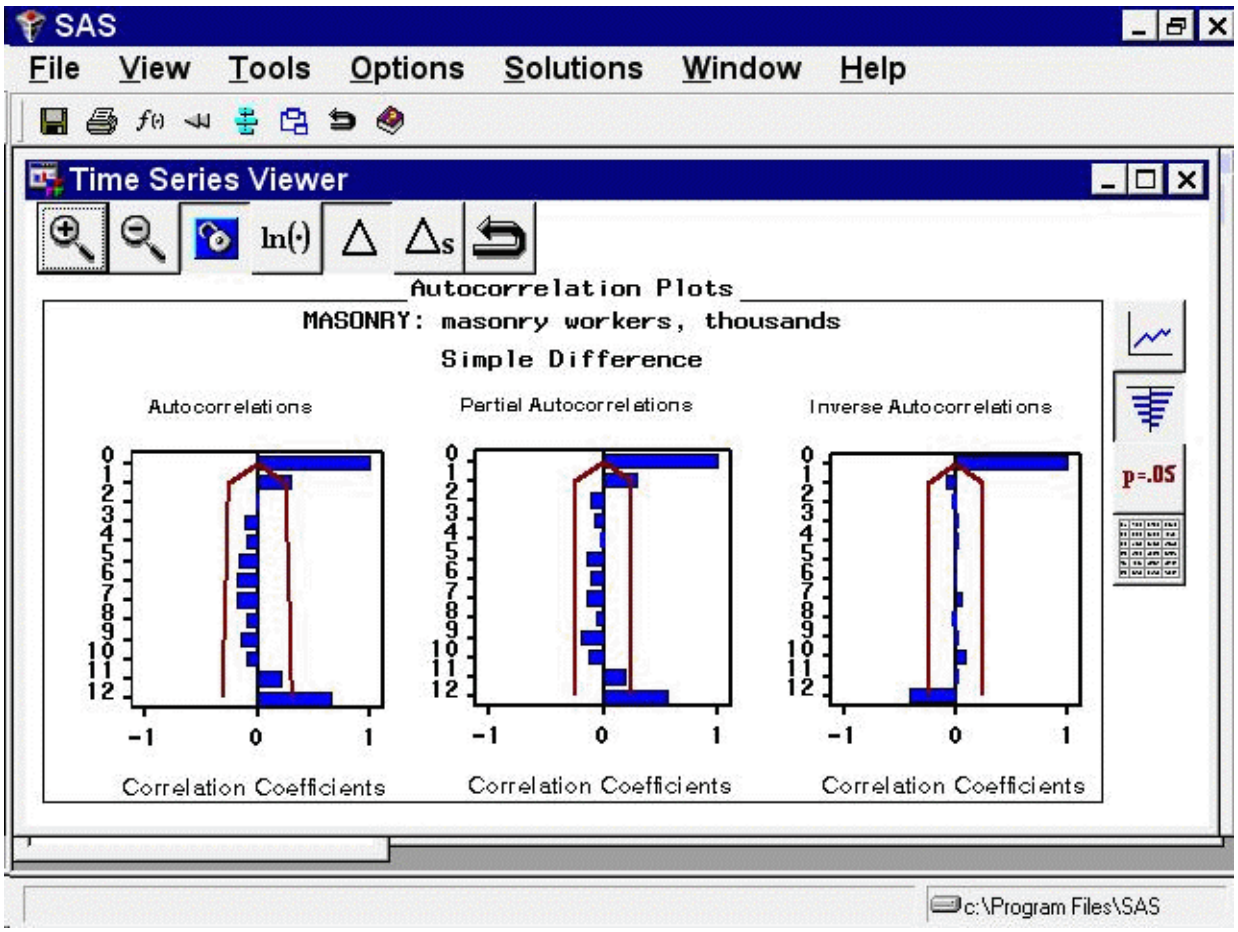
Figure 57.5 Sample Autocorrelation Plots



Each bar represents the value of the correlation coefficient at the given lag. The overlaid lines represent confidence limits computed at plus and minus two standard errors. You can switch the graphs to show significance probabilities by selecting `Correlation Probabilities` under the `Options` pull-down menu, or by selecting the `Toggle ACF Probabilities` toolbar icon.

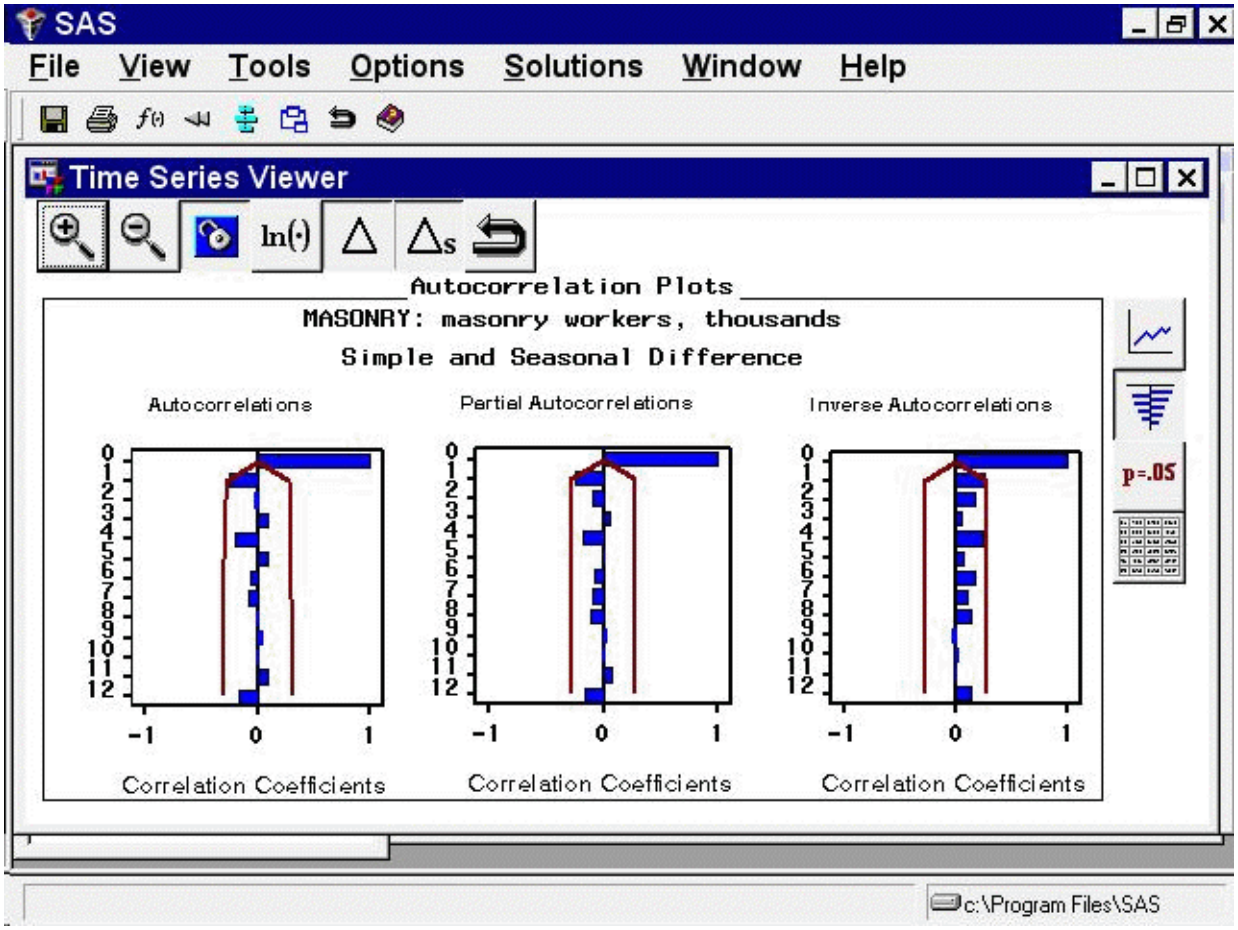
The slow decline of the ACF suggests that first differencing might be warranted. To see the effect of first differencing, select the simple difference icon, the fifth icon from the left on the window's horizontal toolbar. The plot now appears as shown in Figure 57.6.

Figure 57.6 ACF Plots with First Difference Applied



Since the ACF still displays slow decline at seasonal lags, seasonal differencing is appropriate (in addition to the first differencing already applied). Select the `Seasonal Difference` icon, the sixth icon from the left on the horizontal toolbar. The plot now appears as shown in Figure 57.7.

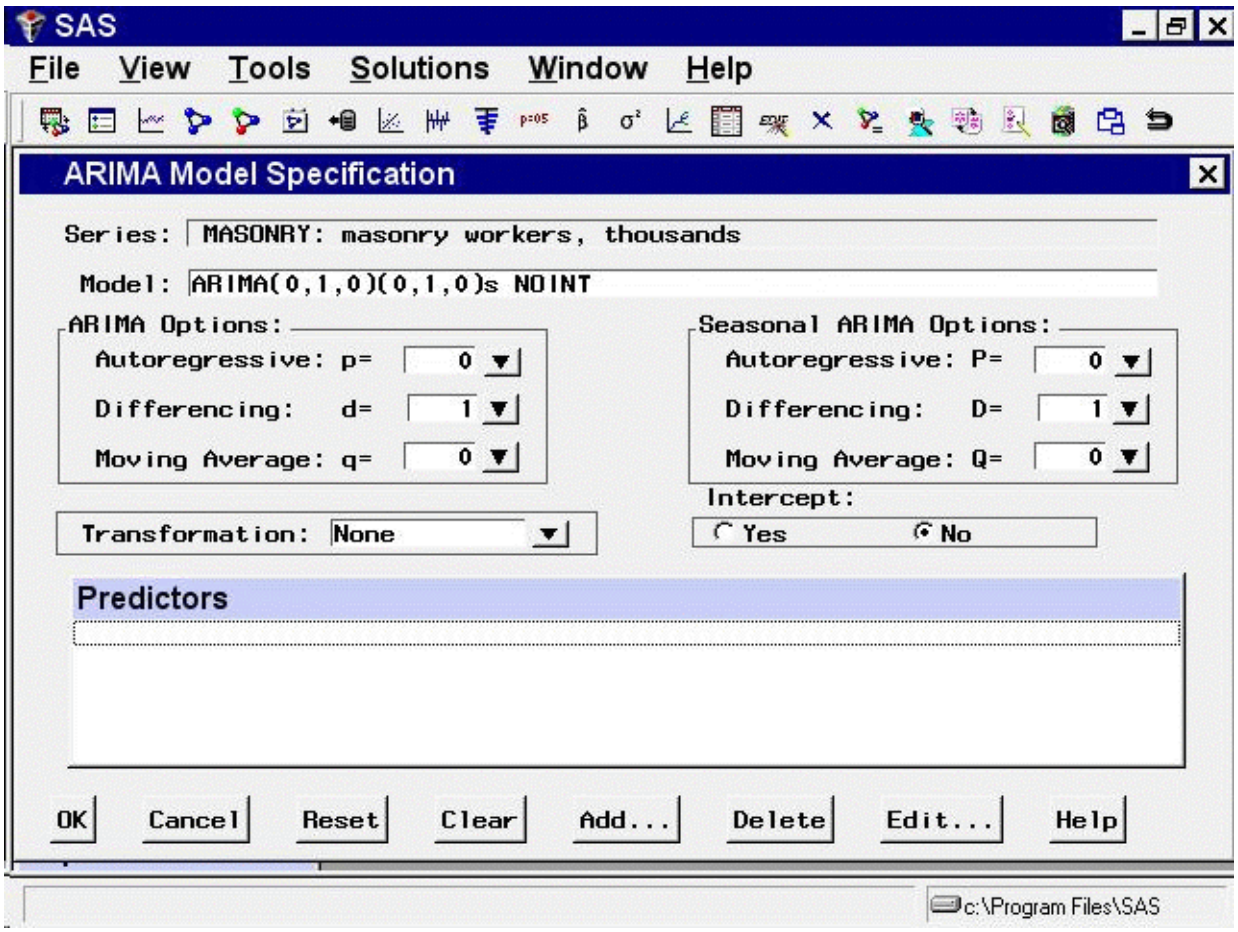
Figure 57.7 ACF Plot with Simple and Seasonal Differencing



Model Viewer Prediction Error Analysis

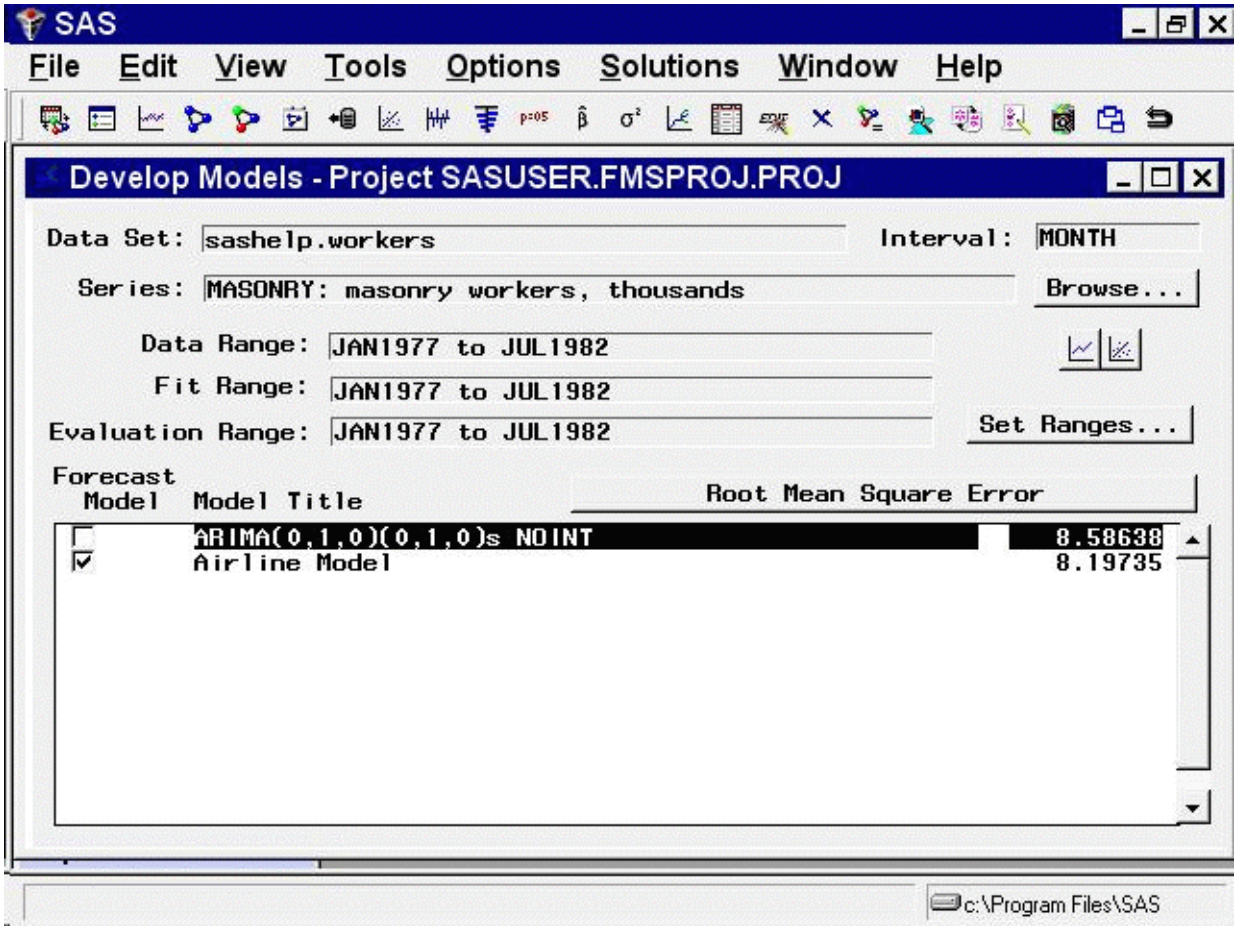
Leave the Time Series Viewer open for the remainder of this exercise. Drag it out of the way or push it to the background so that you can return to the Time Series Forecasting window. Select `Develop Models`, then click an empty part of the table to bring up the pop-up menu, and select `Fit ARIMA Model`. Define the ARIMA(0,1,0)(0,1,0)s model by selecting `1` for Differencing under ARIMA Options, `1` for Differencing under Seasonal ARIMA Options, and `No` for `Intercept`, as shown in Figure 57.8.

Figure 57.8 Specifying the ARIMA(0,1,0)(0,1,0)s Model



When you select the `OK` button, the model is fit and you are returned to the Develop Models window. Click on an empty part of the table and choose `Fit Models from List` from the pop-up menu. Select `Airline Model` from the window. (Airline Model is a common name for the ARIMA(0,1,1)(0,1,1)s model, which is often used for seasonal data with a linear trend.) Select the `OK` button. Once the model has been fit, the table shows the two models and their root mean square errors. Notice that the Airline Model provides only a slight improvement over the differencing model, ARIMA(0,1,0)(0,1,0)s. Select the first row to highlight the differencing model, as shown in Figure 57.9.

Figure 57.9 Selecting a Model

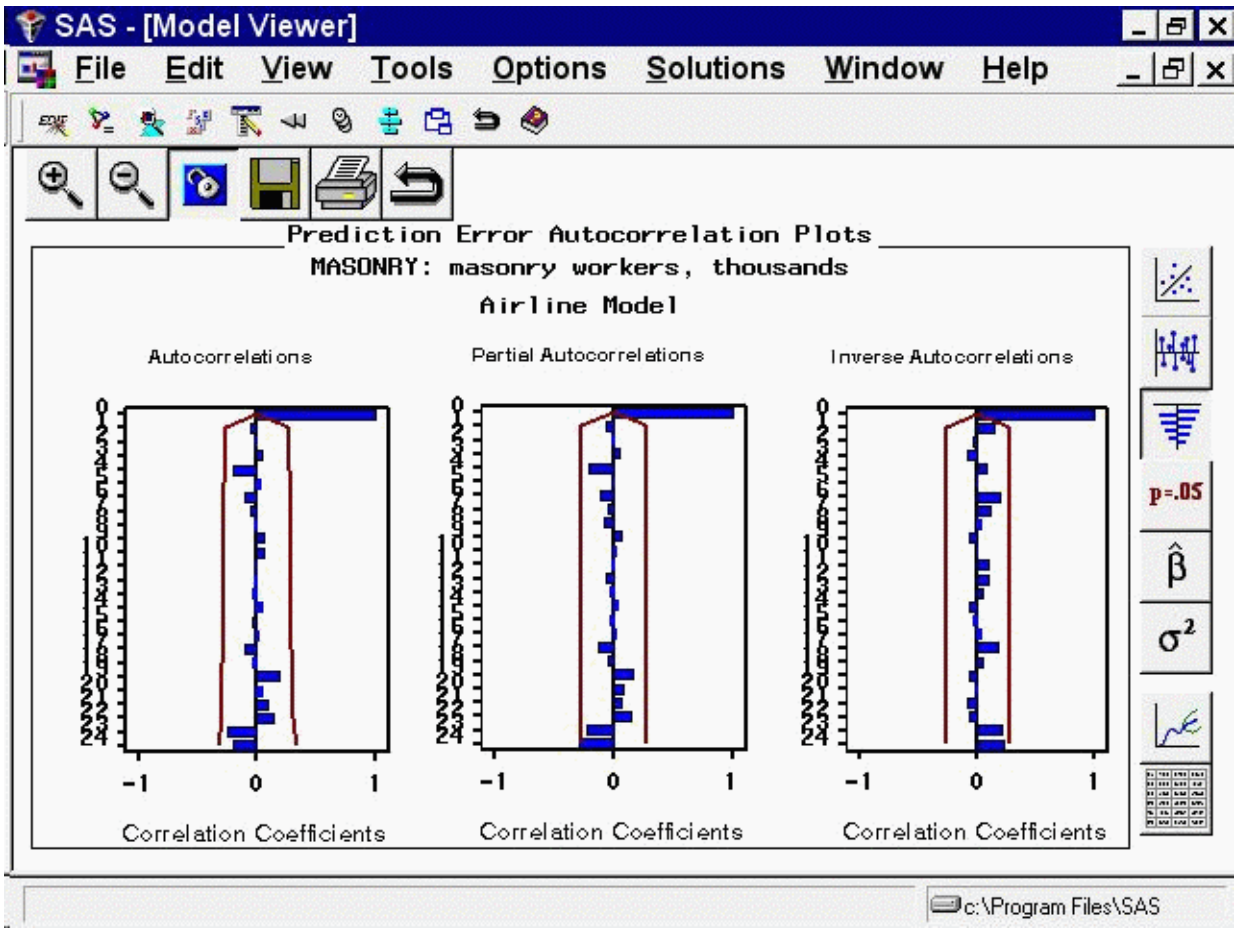


Now select the `View Selected Model Graphically` button, below the `Browse` button at the right side of the Develop Models window. The `Model Viewer` window appears, showing the actual data and model predictions for the MASONRY series. (Note that predicted values are missing for the first 13 observations due to simple and seasonal differencing.)

To examine the ACF plot for the model prediction errors, select the third icon from the top on the vertical toolbar. For this model, the prediction error ACF is the same as the ACF of the original data with first differencing and seasonal differencing applied. This differencing is apparent if you bring the Time Series Viewer back into view for comparison.

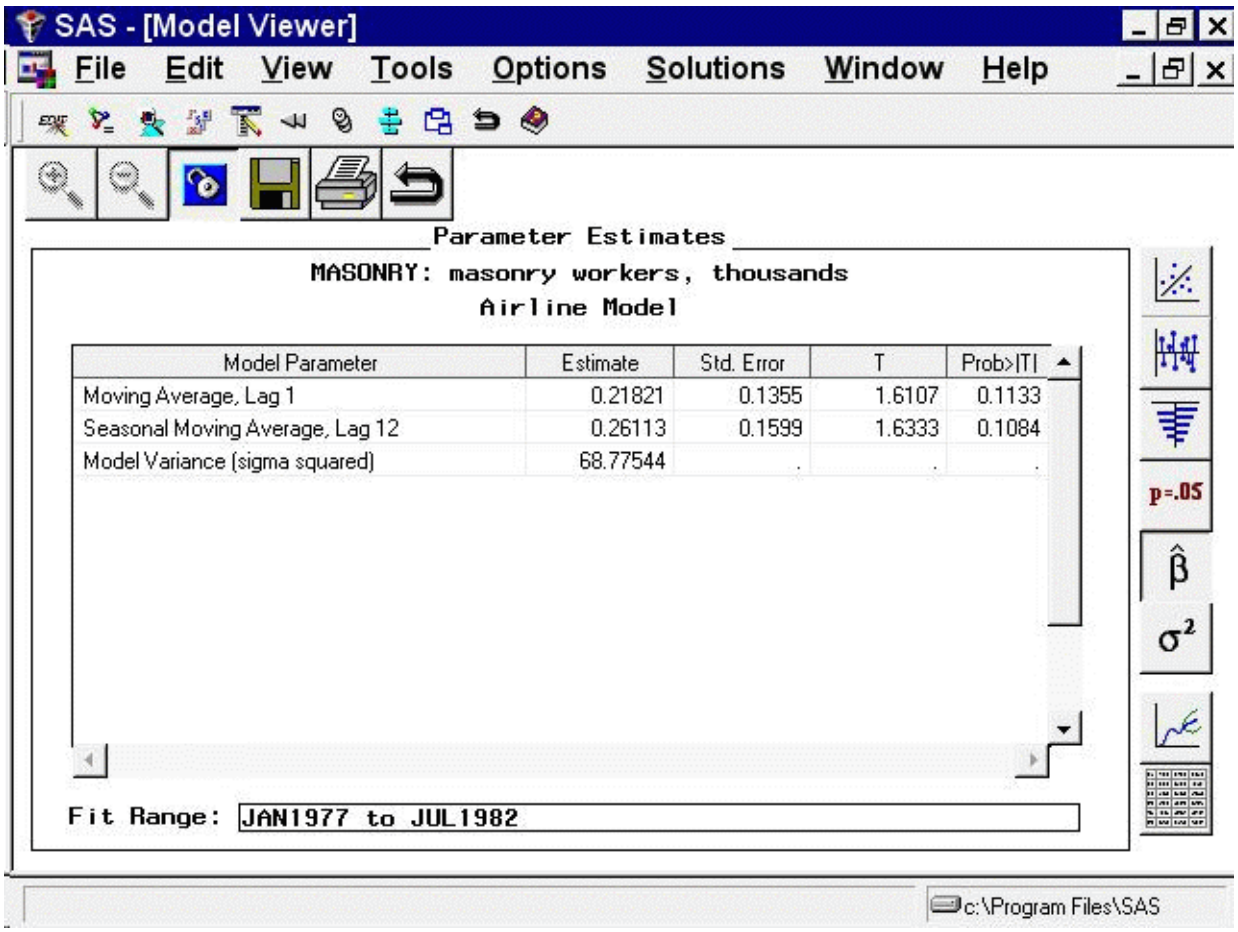
Return to the Develop Models Window by clicking on it or using the window pull-down menu or the Next Viewer toolbar icon. Select the second row of the table in the Develop Models window to highlight the Airline Model. The Model Viewer is automatically updated to show the prediction error ACF of the newly selected model, as shown in Figure 57.10.

Figure 57.10 Prediction Error ACF Plot for the Airline Model



Another helpful tool available within the Model Viewer is the parameter estimates table. Select the fifth icon from the top of the vertical toolbar. The table gives the parameter estimates for the two moving-average terms in the Airline Model, as well as the model residual variance, as shown in Figure 57.11.

Figure 57.11 Parameter Estimates for the Airline Model



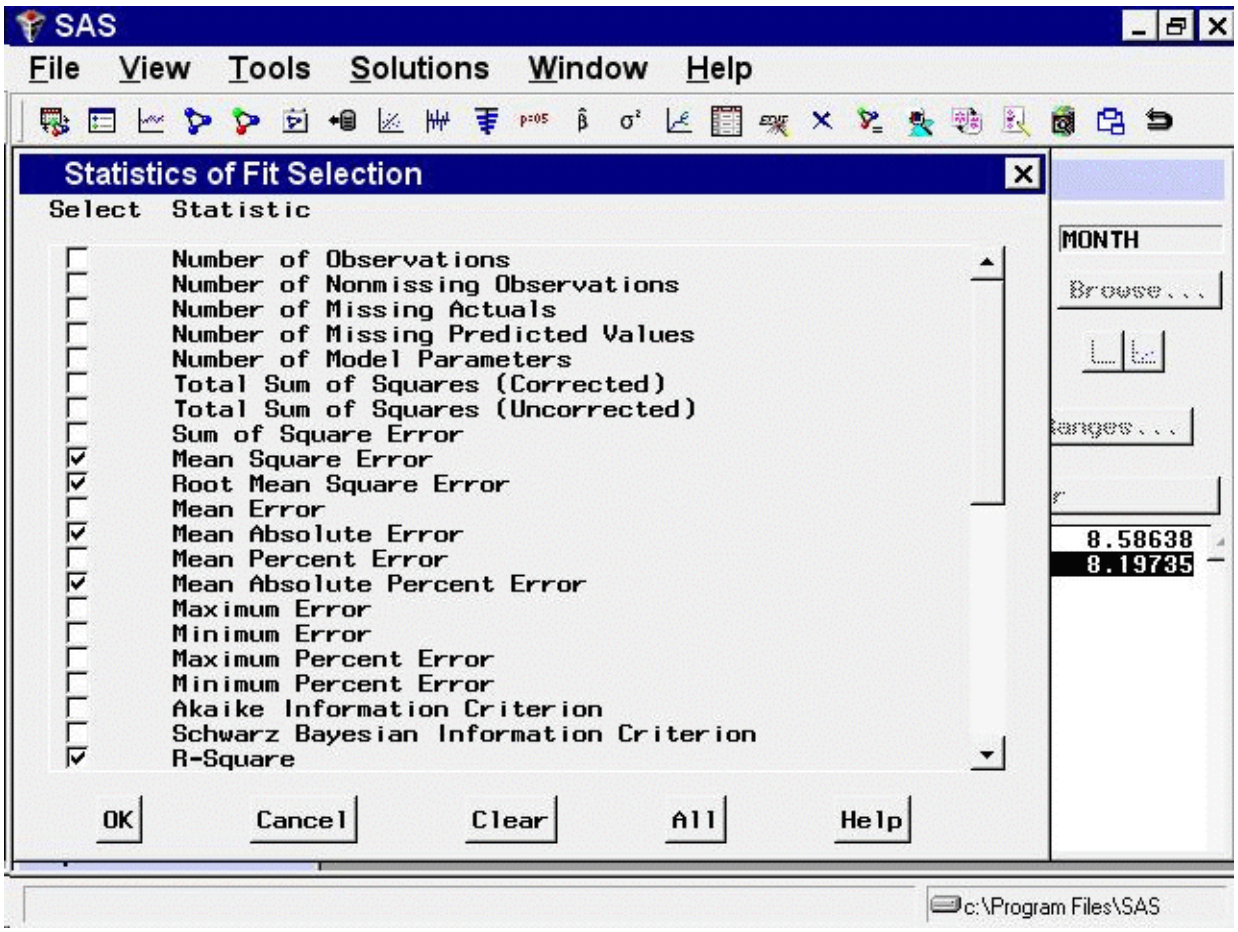
You can adjust the column widths in the table by dragging the vertical borders of the column titles with the mouse. Notice that neither of the parameter estimates is significantly different from zero at the 0.05 level of significance, since `Prob>|t|` is greater than 0.05. This suggests that the Airline Model should be discarded in favor of the more parsimonious differencing model, which has no parameters to estimate.

The Model Selection Criterion

Return to the Develop Models window (Figure 57.9) and notice the Root Mean Square Error button at the right side of the table banner. This is the model selection criterion—the statistic used by the system to select the best fitting model. So far in this example you have fit two models and have left the default criterion, root mean square error (RMSE), in effect. Because the Airline Model has the smaller value of this criterion and because smaller values of the RMSE indicate better fit, the system has chosen this model as the forecasting model, indicated by the check box in the `Forecast Model` column.

The statistics available as model selection criteria are a subset of the statistics available for informational purposes. To access the entire set, select `Options` from the menu bar, and then select `Statistics of Fit`. The `Statistics of Fit Selection` window appears, as shown in Figure 57.12.

Figure 57.12 Statistics of Fit



By default, five of the more well known statistics are selected. You can select and deselect statistics by clicking the check boxes in the left column. For this exercise, select `All`, and notice that all the check boxes become checked. Select the `OK` button to close the window. Now if you choose `Statistics of Fit` in the `Model Viewer` window, all of the statistics will be shown for the selected model.

To change the model selection criterion, click the `Root Mean Square Error` button or select `Options` from the menu bar and then select `Model Selection Criterion`. Notice that most of the statistics of fit are shown, but those which are not relevant to model selection, such as number of observations, are not shown. Select `Schwarz Bayesian Information Criterion` and click `OK`. Since this statistic puts a high penalty on models with larger numbers of parameters, the ARIMA(0,1,0)(0,1,0)s model comes out with the better fit.

Notice that changing the selection criterion does not automatically select the model that is best according to that criterion. You can always choose the model you want to use for forecasts by selecting its check box in the `Forecast Model` column.

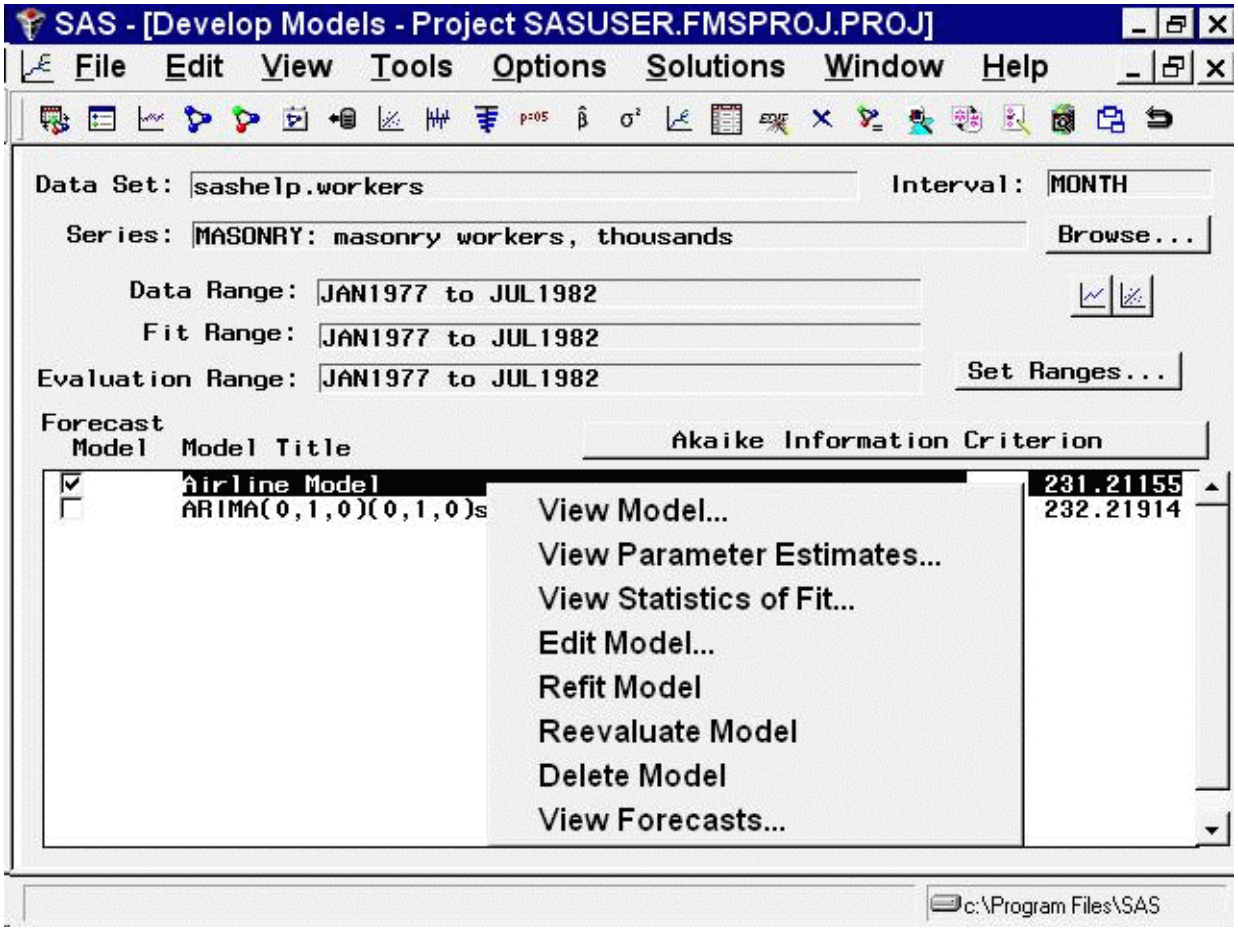
Now bring up the Model Selection Criterion window again and select `Akaike Information Criterion`. This statistic puts a lesser penalty on number of parameters, and the Airline Model comes out as the better fitting model.

Sorting and Selecting Models

Select `Sort Models` on the `Tools` menu or from the toolbar. This sorts the current list of fitted models by the current selection criterion. Although some selection criteria assign larger values to better fitting models (for example, R-square) while others assign smaller values to better fitting models, `Sort Models` always orders models with the best fitting model—in this case, the Airline Model—at the top of the list.

When you select a model in the table, its name and criterion value become highlighted, and actions that apply to that model become available. If your system supports a right mouse button, you can click it to invoke a pop-up menu, as shown in Figure 57.13.

Figure 57.13 Right Mouse Button Pop-up Menu



Whether or not you have a right mouse button, the same choices are available under `Edit` and `View` from the menu bar. If the model viewer has been invoked, it is automatically updated to show the selected model, unless you have unlinked the viewer by using the `Link/Unlink` toolbar button.

Select the highlighted model in the table again. Notice that it is no longer highlighted. When no models are highlighted, the right mouse button pop-up menu changes, and items on the menu bar that apply to a selected model become unavailable. For example, you can choose `Edit` from the menu bar, but you can't choose the `Edit Model` or `Delete Model` selections unless you have highlighted a model in the table.

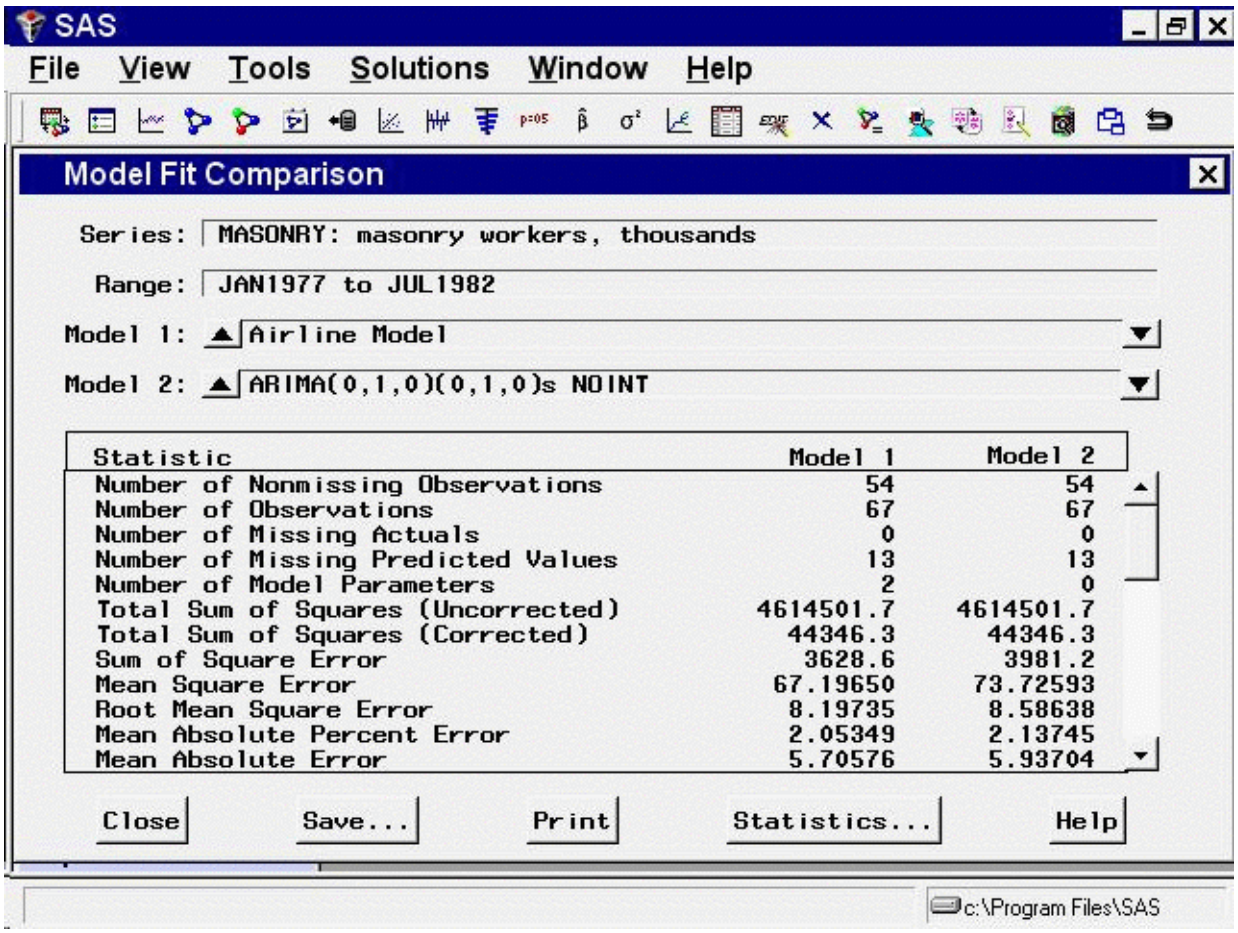
When you select the check box in the `Forecast Model` column of the table, the model in that row becomes

the forecasting model. This is the model that will be used the next time forecasts are generated by choosing `View Forecasts` or by using the `Produce Forecasts` window. Note that this forecasting model flag is automatically set when you use `Fit Automatic Model` or when you fit an individual model that fits better, using the current selection criterion, than the current forecasting model.

Comparing Models

Select `Tools` and `Compare Models` from the menu bar. This displays the `Model Fit Comparison` table, as shown in Figure 57.14.

Figure 57.14 Model Comparison Window



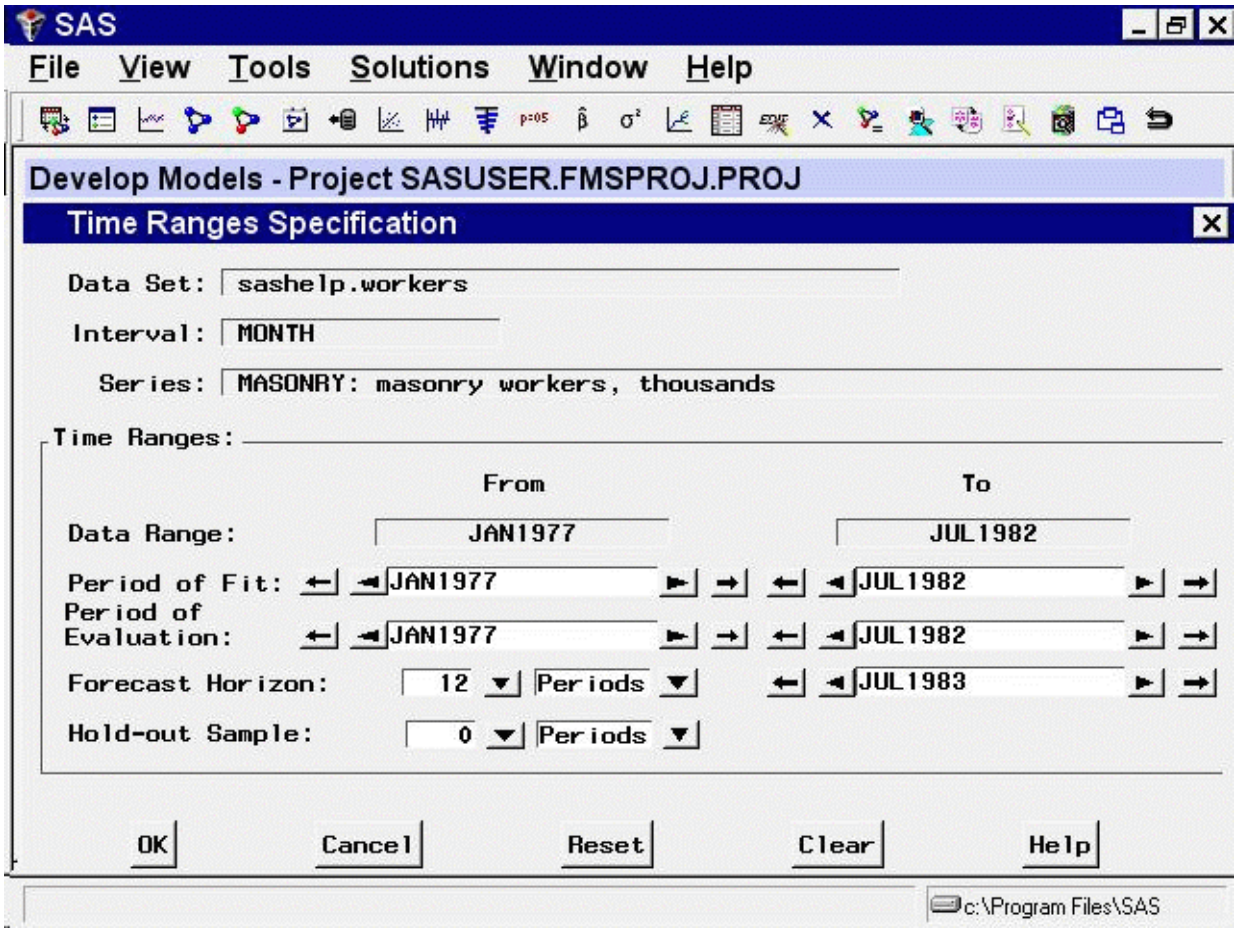
The two models you have fit are shown as `Model 1` and `Model 2`. When there are more than two models, you can bring any two of them into the table by selecting the up and down arrows. In this way, it is easy to do pairwise comparisons on any number of models, looking at as many statistics of fit as you like. Since you previously chose to display all statistics of fit, all of them are shown in the comparison table. Use the vertical scroll bar to move through the list.

After you have examined the model comparison table, select the `Close` button to return to the Develop Models window.

Controlling the Period of Evaluation and Fit

Notice the three time ranges shown on the Develop Models window (Figure 57.9). The data range shows the beginning and ending dates of the MASONRY time series. The period of fit shows the beginning and ending dates of data used to fit the models. The period of evaluation shows the beginning and ending dates of data used to compute statistics of fit. By default, the fit and evaluate ranges are the same as the data range. To change these ranges, select the `Set Ranges` button, or select `Options` and `Time Ranges` from the menu bar. This brings up the `Time Ranges Specification` window, as shown in Figure 57.15.

Figure 57.15 Time Ranges Specification Window



For this example, suppose the early data in the series is unreliable, and you want to use the range June 1978 to the latest available for both model fitting and model evaluation. You can either type JUN1978 in the `From` column for `Period of Fit` and `Period of Evaluation`, or you can advance these dates by clicking the right pointing arrows. The outer arrow advances the date by a large amount (in this case, by a year), and the inner arrow advances it by a single period (in this case, by a month). Once you have changed the `Period of Fit` and the `Period of Evaluation` to JUN1978 in the `From` column, select the `OK` button to return to the `Develop Models` window. Notice that these time ranges are updated at the top of the window, but the models already fit have not been affected. Your changes to the time ranges affect *subsequently fit* models.

Refitting and Reevaluating Models

If you fit the ARIMA(0,1,0)(0,1,0)s and Airline models again in the same way as before, they will be added to the model list, with the same names but with different values of the model selection criterion. Parameter estimates will be different, due to the new fit range, and statistics of fit will be different, due to the new evaluation range.

For this exercise, instead of specifying the models again, refit the existing models by selecting `Edit` from the menu bar and then selecting `Refit Models` and `All Models`. After the models have been refit, you should see the same two models listed in the table but with slightly different values for the selection criterion. The ARIMA (0,1,0)(0,1,0)s and Airline models have now been fit to the MASONRY series by using data from June 1978 to July 1982, since this is the period of fit you specified. The statistics of fit have been computed for the period of evaluation, which was the same as the period of fit. If you had specified a period of evaluation different from the period of fit, the statistics would have been computed accordingly.

In practice, another common reason for refitting models is the availability of new data. For example, when data for a new month become available for a monthly series, you might add them to the input data set, then invoke the forecasting system, open the project containing models fit previously, and refit the models prior to generating new forecasts. Unless you specify the period of fit and period of evaluation in the `Time Ranges Specification` window, they default to the full data range of the series found in the input data set at the time of refitting.

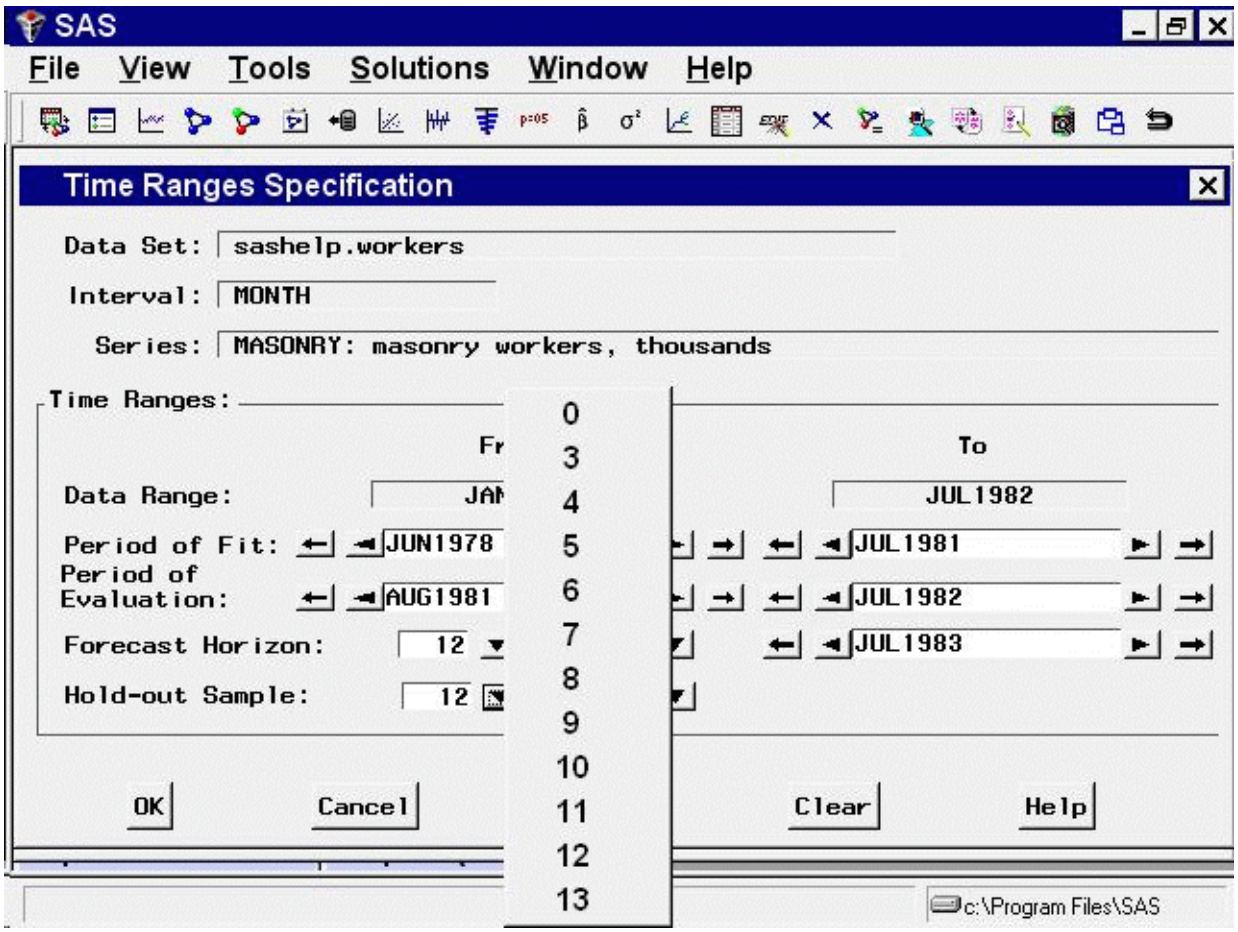
If you prefer to apply previously fit models to revised data without refitting, use `Reevaluate Models` instead of `Refit Models`. This recomputes the statistics of fit by using the current evaluation range, but does not re-estimate the model parameters.

Using Hold-out Samples

One important application of model fitting where the period of fit is different from the period of evaluation is the use of hold-out samples. With this technique of model evaluation, the period of fit ends at a time point before the end of the data series, and the remainder of the data are held out as a nonoverlapping period of evaluation. With respect to the period of fit, the hold-out sample is a period in the future, used to compare the forecasting accuracy of models fit to past data.

For this exercise, use a hold-out sample of 12 months. Bring up the `Time Ranges Specification` window again by selecting the `Set Ranges` button. Set `Hold-out Sample` to 12 using the combo box, as shown in Figure 57.16. You can also type in a value. To specify a hold-out sample period in different units, you can use the `Periods` combo box. In this case, it allows you to select years as the unit, instead of periods.

Figure 57.16 Specifying the Hold-out Sample Size

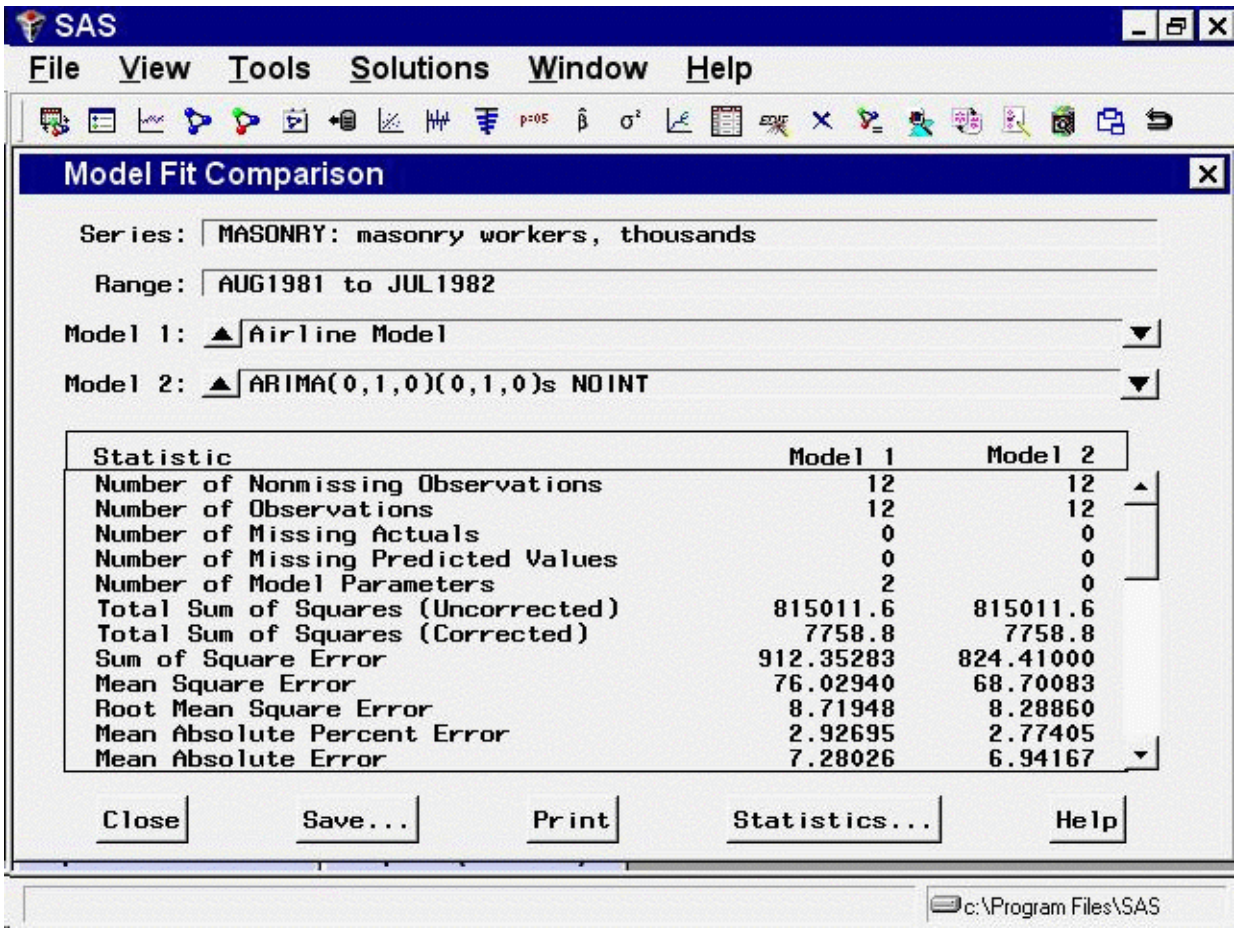


Notice that setting the hold-out sample to 12 automatically sets the fit range to JUN1978–JUL1981 and the evaluation range to AUG1981–JUL1982. If you had set the period of fit and period of evaluation to these ranges, the hold-out sample would have been automatically set to 12 periods.

Select the `OK` button to return to the `Develop Models` window. Now refit the models again. Select `Tools` and `Compare Models` to compare the models now that they have been fit to the period June 1978 through July 1981 and evaluated for the hold-out sample period August 1981 through July 1982. Note that the fit statistics for the hold-out sample are based on one-step-ahead forecasts. (See *Statistics of Fit* in Chapter 61, "Forecasting Process Details.")

As shown in Figure 57.17, the ARIMA (0,1,0)(0,1,0)s model now seems to provide a better fit to the data than does the Airline model. It should be noted that the results can be quite different if you choose a different size hold-out sample.

Figure 57.17 Using 12 Month Hold-out Sample



Chapter 58
Using Predictor Variables

Contents

Linear Trend **3741**
Time Trend Curves **3743**
Regressors **3747**
Adjustments **3749**
Dynamic Regressor **3750**
Interventions **3755**
The Intervention Specification Window 3755
Specifying a Trend Change Intervention 3757
Specifying a Level Change Intervention 3760
Modeling Complex Intervention Effects 3761
Fitting the Intervention Model 3762
Limitations of Intervention Predictors 3766
Seasonal Dummies **3766**
References **3770**

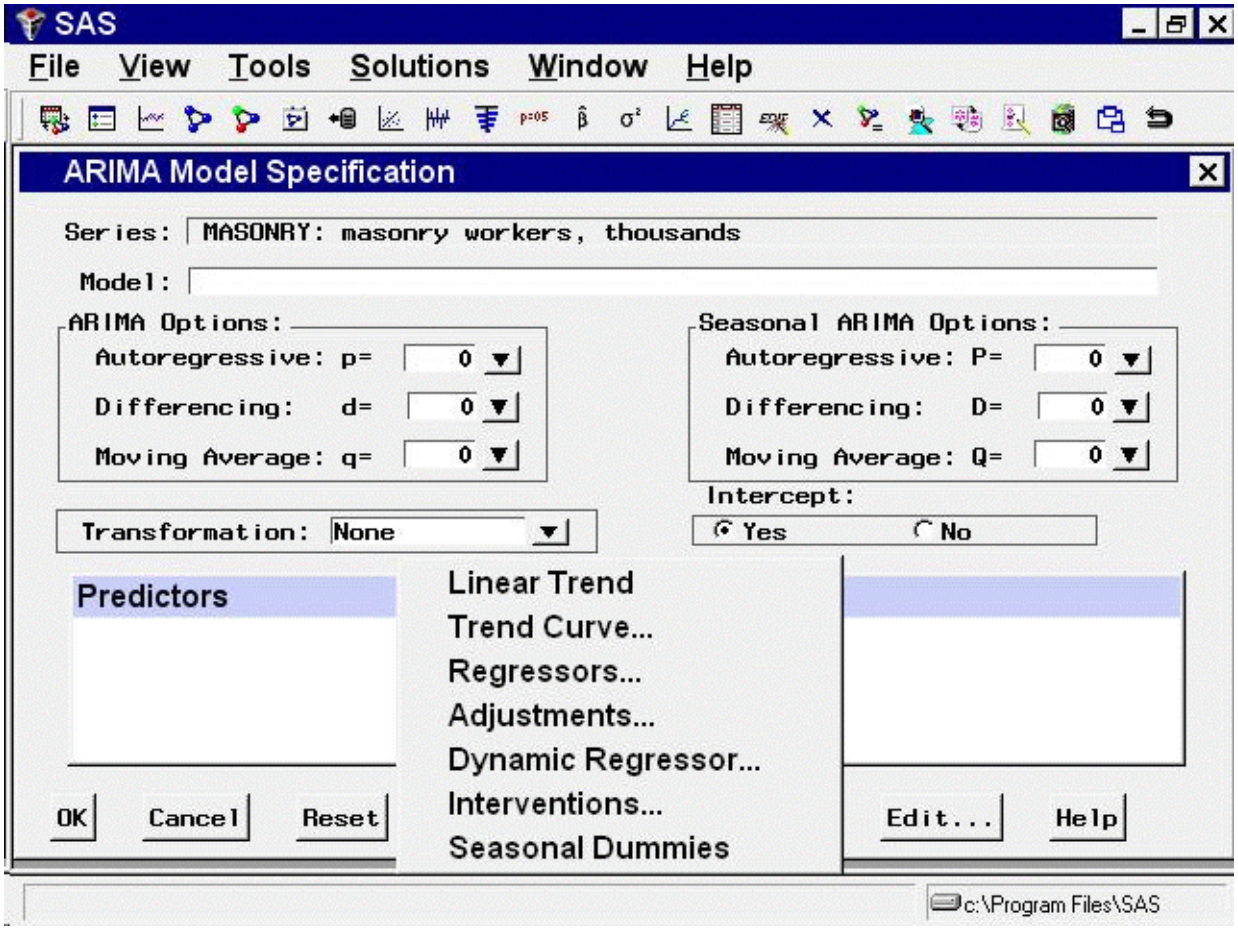
Forecasting models predict the future values of a series by using two sources of information: the past values of the series and the values of other time series variables. Other variables used to predict a series are called *predictor variables*.

Predictor variables that are used to predict the dependent series can be variables in the input data set, such as regressors and adjustment variables, or they can be special variables computed by the system as functions of time, such as trend curves, intervention variables, and seasonal dummies.

You can specify seven different types of predictors in forecasting models by using the ARIMA Model or Custom Model Specification windows. You cannot specify predictor variables with the Smoothing Model Specification window.

Figure 58.1 shows the menu of options for adding predictors to an ARIMA model that is opened by clicking the `Add` button. The Add menu for the Custom Model Specification menu is similar.

Figure 58.1 Add Predictors Menu



These types of predictors are as follows.

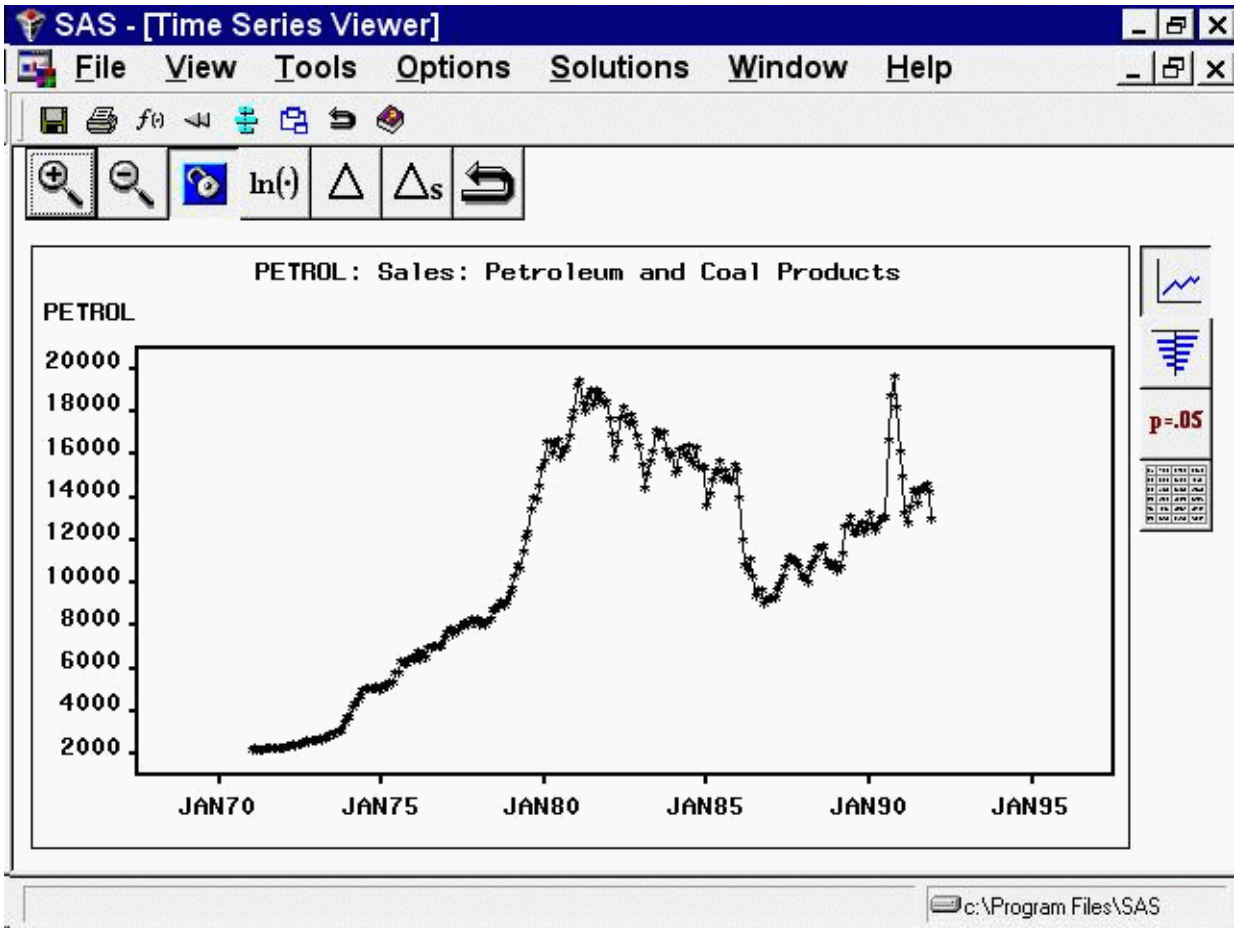
Linear Trend	adds a variable that indexes time as a predictor series. A straight line time trend is fit to the series by regression when you specify a linear trend.
Trend Curve	provides a menu of various functions of time that you can add to the model to fit nonlinear time trends. The Linear Trend option is a special case of the Trend Curve option for which the trend curve is a straight line.
Regressors	allows you to predict the series by regressing it on other variables in the data set.
Adjustments	allows you to specify other variables in the data set that supply adjustments to the forecast.
Dynamic Regressor	allows you to select a predictor variable from the input data set and specify a complex model for the way that the predictor variable affects the dependent series.
Interventions	allows you to model the effect of special events that "intervene" to change the pattern of the dependent series. Examples of intervention effects are strikes, tax increases, and special sales promotions.
Seasonal Dummies	adds seasonal indicator or "dummy" variables as regressors to model seasonal effects.

You can add any number of predictors to a forecasting model, and you can combine predictor variables with other model options.

The following sections explain these seven kinds of predictors in greater detail and provide examples of their use. The examples illustrate these different kinds of predictors by using series in the SASHELP.USECON data set.

Select the `Develop Models` button from the main window. Select the data set SASHELP.USECON and select the series PETROL. Then select the `View Series Graphically` button from the Develop Models window. The plot of the example series PETROL appears as shown in Figure 58.2.

Figure 58.2 Sales of Petroleum and Coal

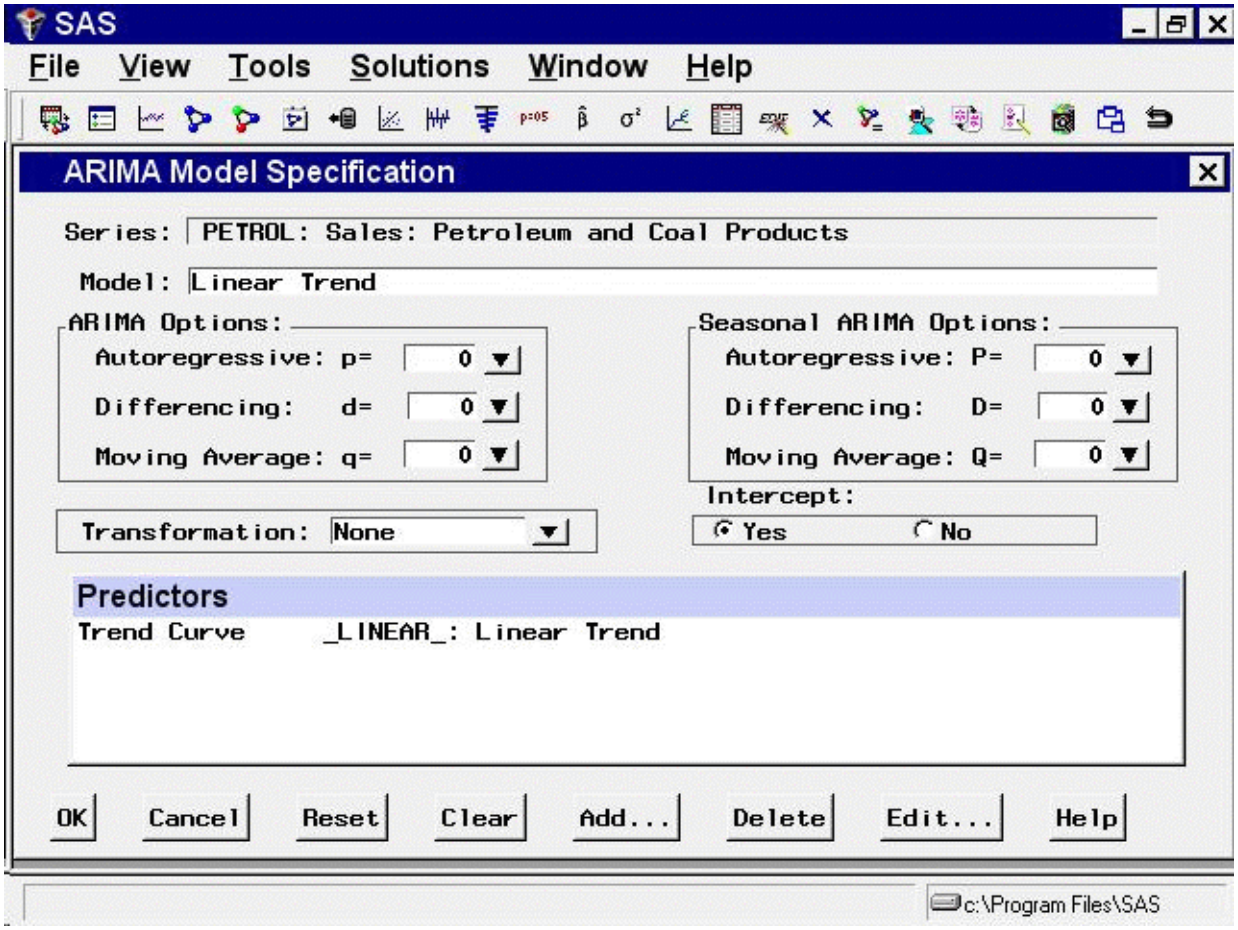


Linear Trend

From the Develop Models window, select `Fit ARIMA Model`. From the ARIMA Model Specification window, select `Add` and then select `Linear Trend` from the menu (shown in Figure 58.1).

A linear trend is added to the Predictors list, as shown in Figure 58.3.

Figure 58.3 Linear Trend Predictor Specified



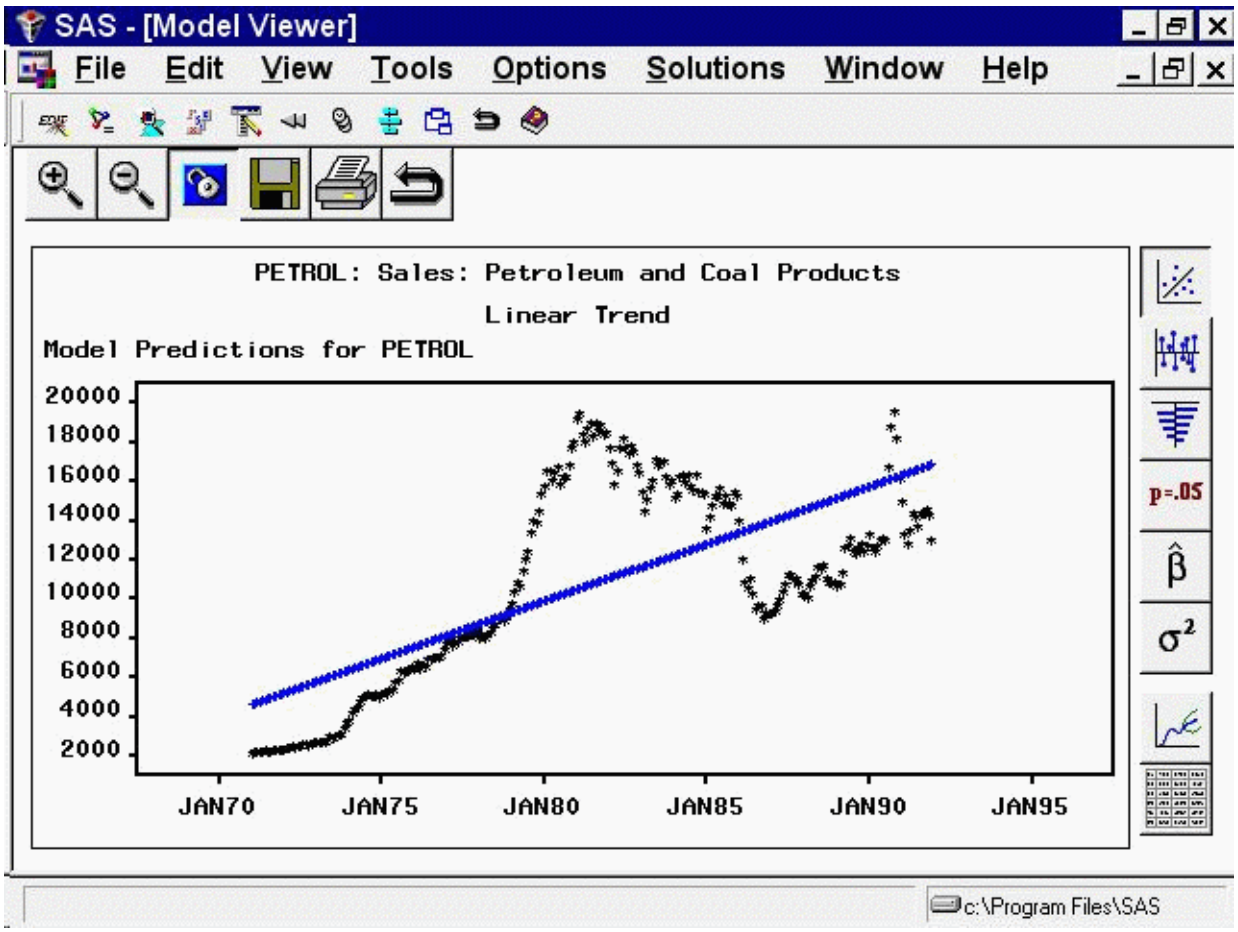
The description for the linear trend item shown in the Predictors list has the following meaning. The first part of the description, Trend Curve, describes the type of predictor. The second part, _LINEAR_, gives the variable name of the predictor series. In this case, the variable is a time index that the system computes. This variable is included in the output forecast data set. The final part, Linear Trend, describes the predictor.

Notice that the model you have specified consists only of the time index regressor _LINEAR_ and an intercept. Although this window is normally used to specify ARIMA models, in this case no ARIMA model options are specified, and the model is a simple regression on time.

Select the `OK` button. The Linear Trend model is fit and added to the model list in the Develop Models window.

Now open the Model Viewer by using the `View Model Graphically` icon or the `Model Predictions` item under the `View` pull-down menu or toolbar. This displays a plot of the model predictions and actual series values, as shown in Figure 58.4. The predicted values lie along the least squares trend line.

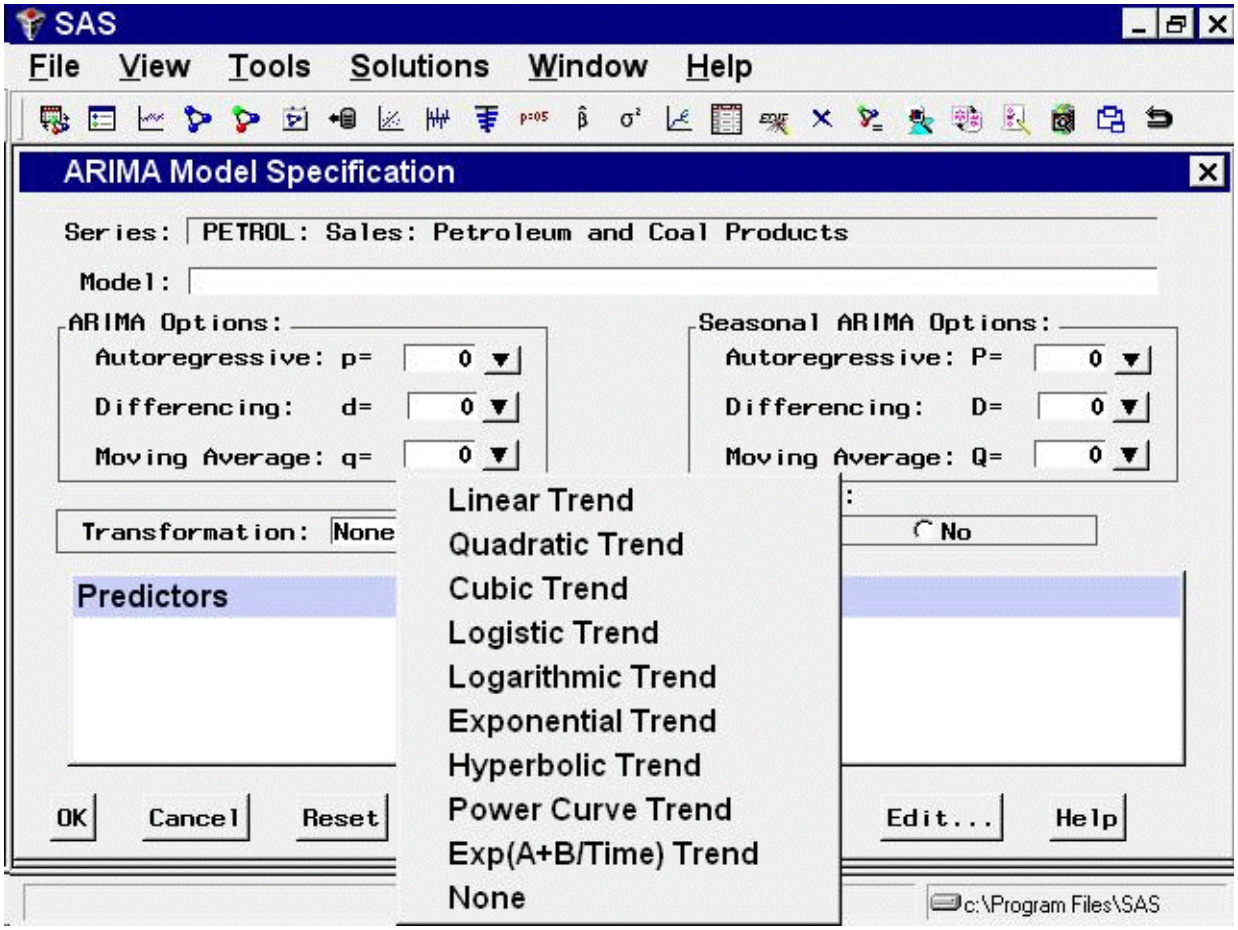
Figure 58.4 Linear Trend Model



Time Trend Curves

From the Develop Models window, select `Fit ARIMA Model`. From the ARIMA Model Specification window, select `Add` and then select `Trend Curve` from the menu (shown in Figure 58.1). A menu of different kinds of trend curves is displayed, as shown in Figure 58.5.

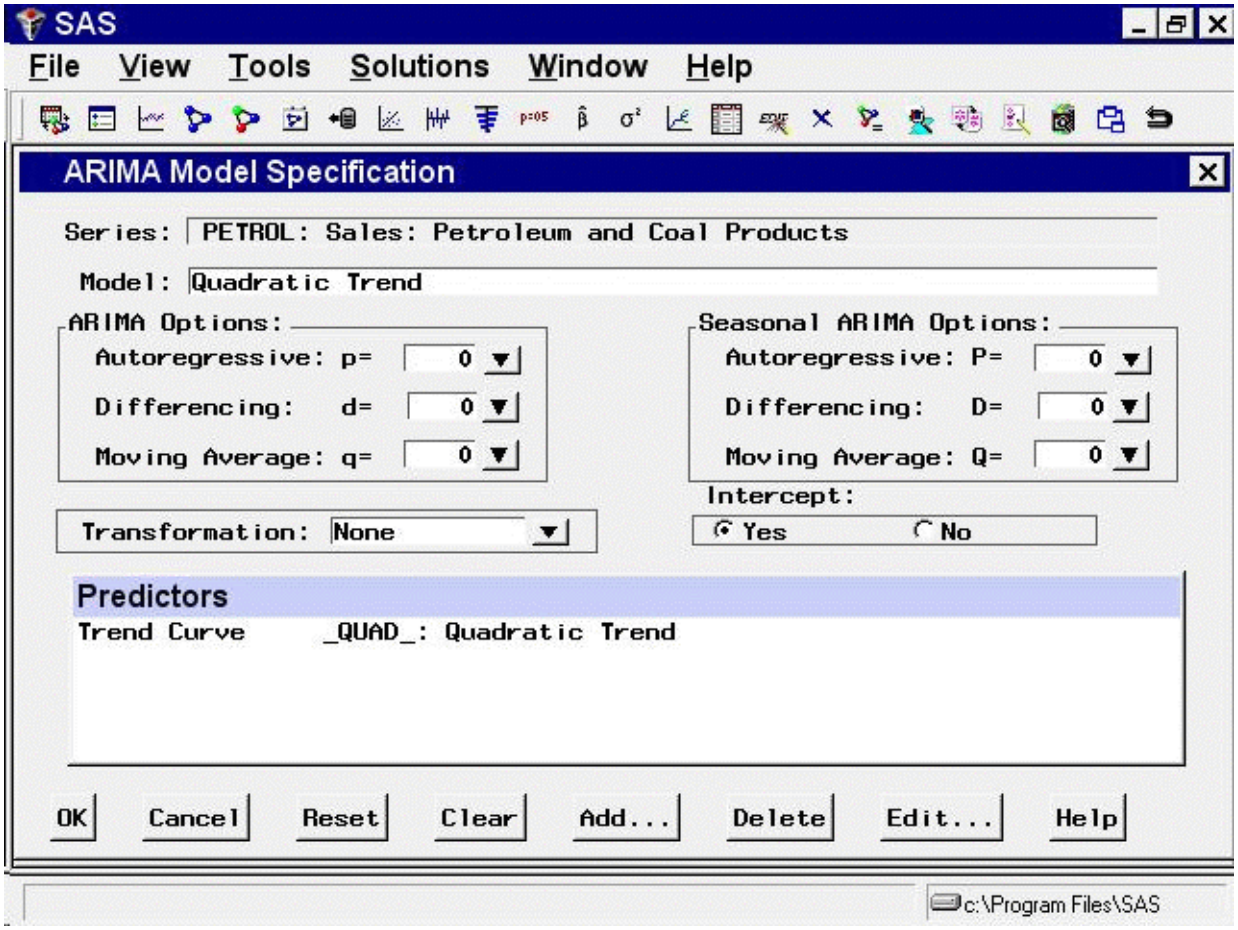
Figure 58.5 Time Trend Curves Menu



These trend curves work in a similar way as the Linear Trend option (which is a special case of a trend curve and one of the choices on the menu), but with the Trend Curve menu you have a choice of various nonlinear time trends.

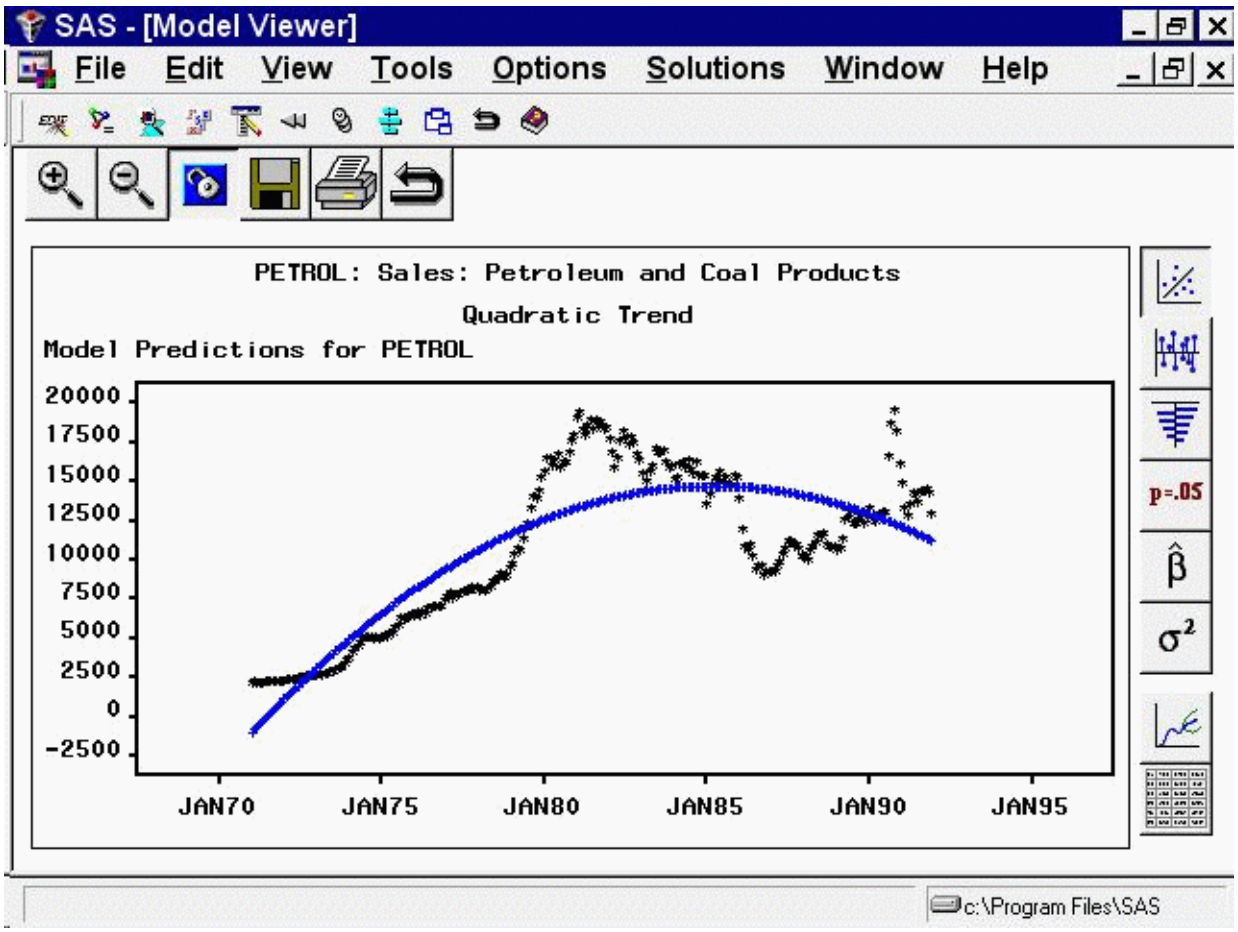
Select `Quadratic Trend`. This adds a quadratic time trend to the Predictors list, as shown in Figure 58.6.

Figure 58.6 Quadratic Trend Specified



Now select the OK button. The quadratic trend model is fit and added to the list of models in the Develop Models window. The Model Viewer displays a plot of the quadratic trend model, as shown in Figure 58.7.

Figure 58.7 Quadratic Trend Model



This curve does not fit the PETROL series very well, but the View Model plot illustrates how time trend models work. You might want to experiment with different trend models to see what the different trend curves look like.

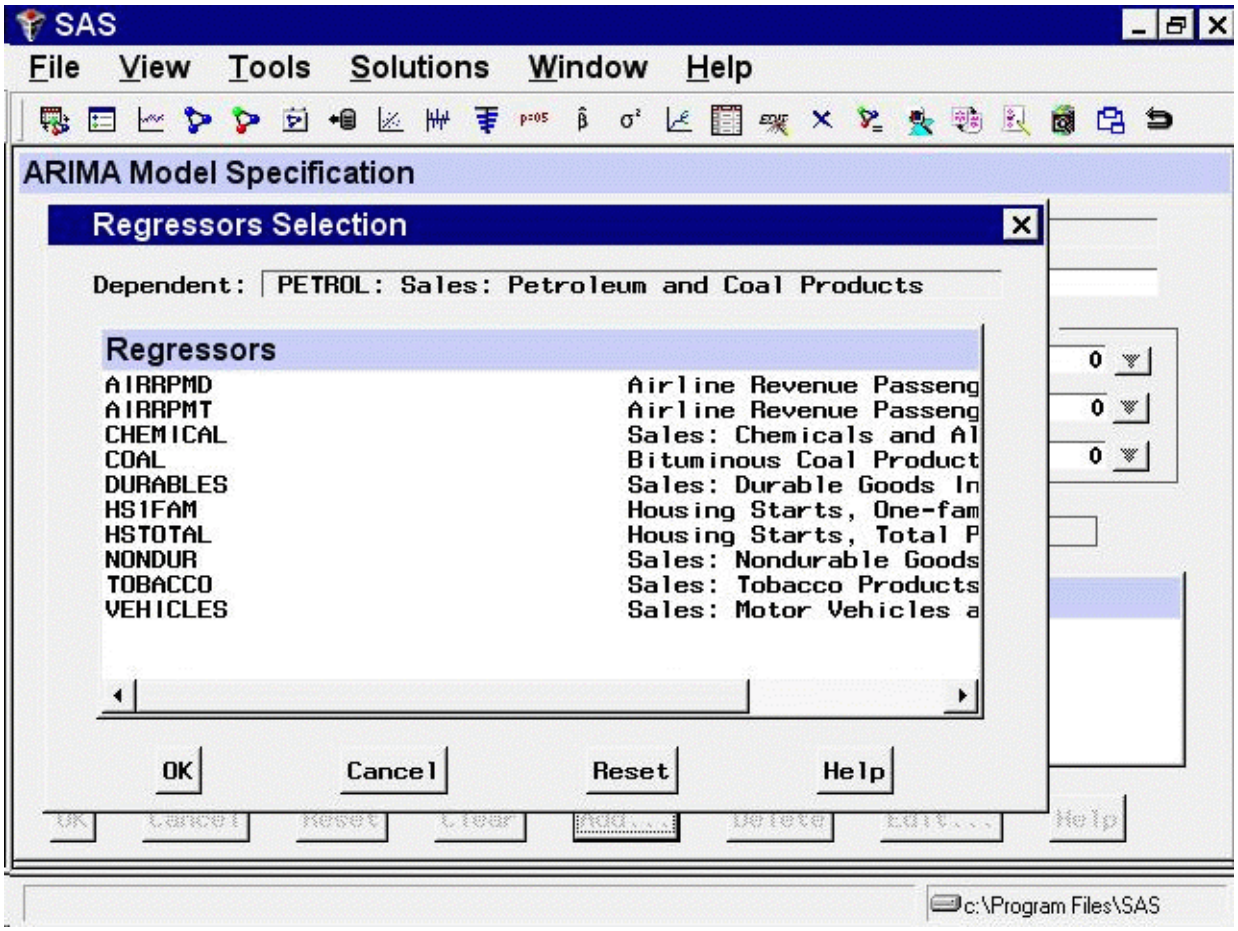
Some of the trend curves require transforming the dependent series. When you specify one of these curves, a notice is displayed reminding you that a transformation is needed, and the Transformation field is automatically filled in. Therefore, you cannot control the Transformation specification when some kinds of trend curves are specified.

See the section "Time Trend Curves" on page 3743 in Chapter 61, "Forecasting Process Details," for more information about the different trend curves.

Regressors

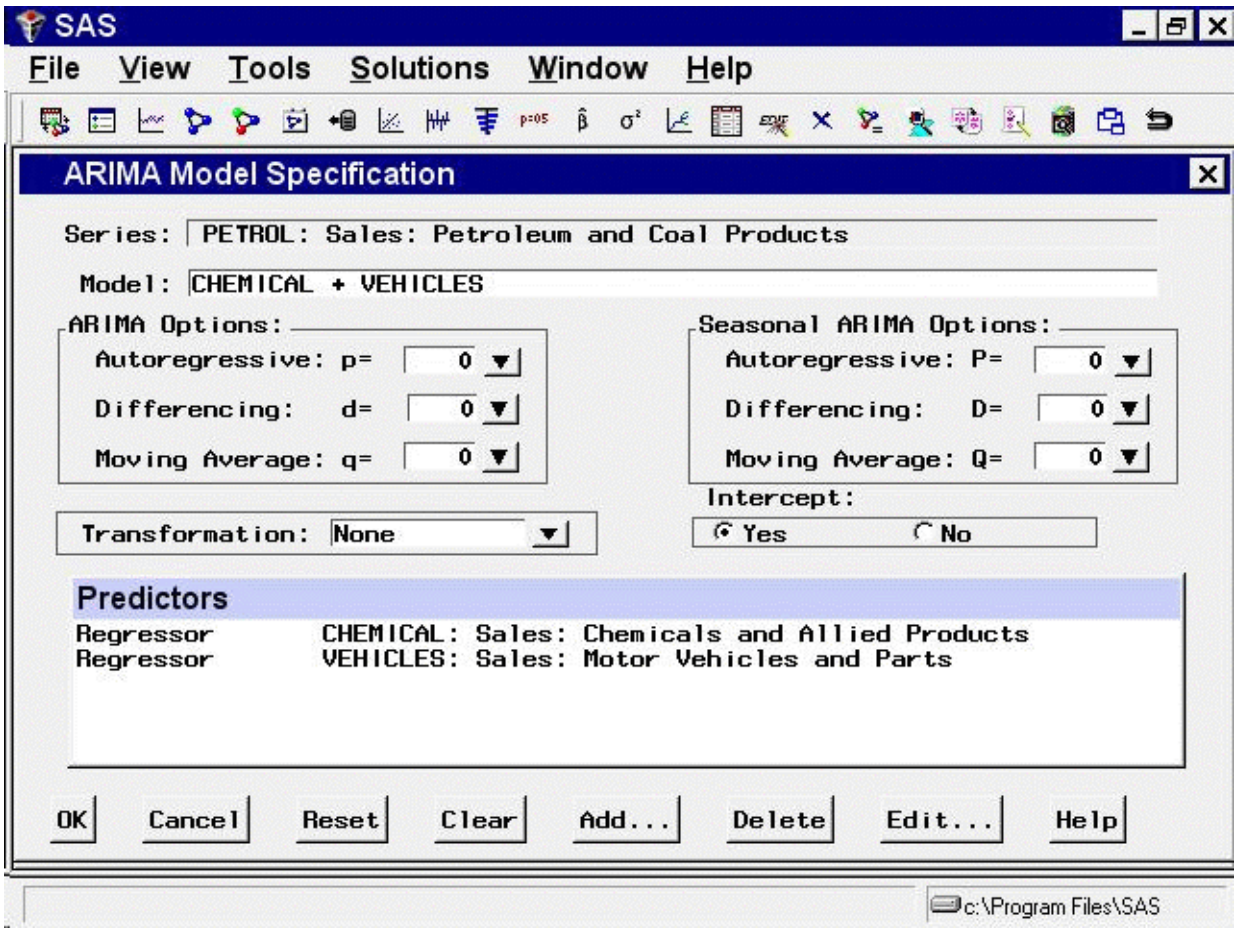
From the Develop Models window, select `Fit ARIMA Model`. From the ARIMA Model Specification window, select `Add` and then select `Regressors` from the menu (shown in Figure 58.1). This displays the `Regressors Selection` window, as shown in Figure 58.8. This window allows you to select any number of other series in the input data set as regressors to predict the dependent series.

Figure 58.8 Regressors Selection Window



For this example, select `CHEMICAL, Sales: Chemicals and Allied Products`, and `VEHICLES, Sales: Motor Vehicles and Parts`. (Note: You do not need to use the CTRL key when selecting more than one regressor.) Then select the `OK` button. The two variables you selected are added to the Predictors list as regressor type predictors, as shown in Figure 58.9.

Figure 58.9 Regressors Selected

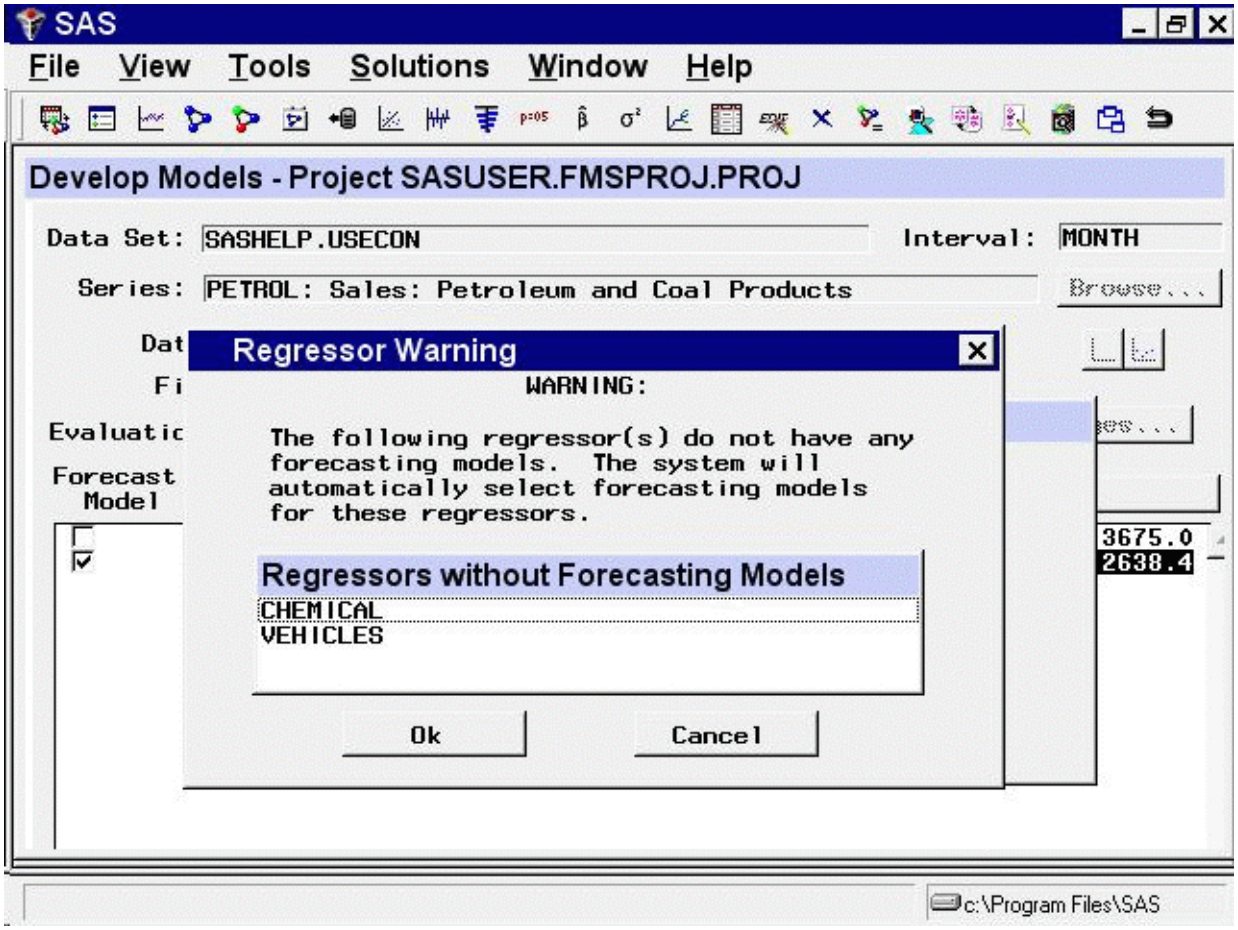


You must have forecasts of the future values of the regressor variables in order to use them as predictors. To do this, you can specify a forecasting model for each regressor, have the system automatically select forecasting models for the regressors, or supply predicted future values for the regressors in the input data set.

Even if you have supplied future values for a regressor variable, the system requires a forecasting model for the regressor. Future values that you supply in the input data set take precedence over predicted values from the regressor's forecasting model when the system computes the forecasts for the dependent series.

Select the OK button. The system starts to fit the regression model but then stops and displays a warning that the regressors that you selected do not have forecasting models, as shown in Figure 58.10.

Figure 58.10 Regressors Needing Models Warning



If you want the system to create forecasting models automatically for the regressor variables by using the automatic model selection process, select the `OK` button. If not, you can select the `Cancel` button to abort fitting the regression model.

For this example, select the `OK` button. The system now performs the automatic model selection process for CHEMICAL and VEHICLES. The selected forecasting models for CHEMICAL and VEHICLES are added to the model lists for those series. If you switch the current time series in the Develop Models window to CHEMICAL or VEHICLES, you will see the model that the system selected for that series.

Once forecasting models have been fit for all regressors, the system proceeds to fit the regression model for PETROL. The fitted regression model is added to the model list displayed in the Develop Models window.

Adjustments

An *adjustment* predictor is a variable in the input data set that is used to adjust the forecast values produced by the forecasting model. Unlike a regressor, an adjustment variable does not have a regression coefficient. No model fitting is performed for adjustments. Nonmissing values of the adjustment series are simply added to the model prediction for the corresponding period. Missing adjustment values are ignored. If you supply

adjustment values for observations within the period of fit, the adjustment values are subtracted from the actual values, and the model is fit to these adjusted values.

To add adjustments, select `Add` and then select `Adjustments` from the pop-up menu (shown in Figure 58.1). This displays the `Adjustments Selection` window. The Adjustments Selection window functions the same as the Regressor Selection window (which is shown in Figure 58.8). You can select any number of adjustment variables as predictors.

Unlike regressors, adjustments do not require forecasting models for the adjustment variables. If a variable that is used as an adjustment does have a forecasting model fit to it, the adjustment variable's forecasting model is ignored when the variable is used as an adjustment.

You can use forecast adjustments to account for expected future events that have no precedent in the past and so cannot be modeled by regression. For example, suppose you are trying to forecast the sales of a product, and you know that a special promotional campaign for the product is planned during part of the period you want to forecast. If such sales promotion programs have been frequent in the past, then you can record the past and expected future level of promotional efforts in a variable in the data set and use that variable as a regressor in the forecasting model.

However, if this is the first sales promotion of its kind for this product, you have no way to estimate the effect of the promotion from past data. In this case, the best you can do is to make an educated guess at the effect the promotion will have and add that guess to what your forecasting model would predict in the absence of the special sales campaign.

Adjustments are also useful for making judgmental alterations to forecasts. For example, suppose you have produced forecast sales data for the next 12 months. Your supervisor believes that the forecasts are too optimistic near the end and asks you to prepare a forecast graph in which the numbers that you have forecast are reduced by 1000 in the last three months. You can accomplish this task by editing the input data set so that it contains observations for the actual data range of sales plus 12 additional observations for the forecast period, and a new variable called, for example, ADJUSTMENT. The variable ADJUSTMENT contains the value 1000 for the last three observations and is missing for all other observations. You fit the same model previously selected for forecasting by using the `ARIMA Model Specification` or `Custom Model Specification` window, but with an adjustment added that uses the variable ADJUSTMENT. Now when you graph the forecasts by using the Model Viewer, the last three periods of the forecast are reduced by 1000. The confidence limits are unchanged, which helps draw attention to the fact that the adjustments to the forecast deviate from what would be expected statistically.

Dynamic Regressor

Selecting `Dynamic Regressor` from the `Add Predictors` menu (shown in Figure 58.1) allows you to specify a complex time series model of the way that a predictor variable influences the series that you are forecasting.

When you specify a predictor variable as a simple regressor, only the current period value of the predictor effects the forecast for the period. By specifying the predictor with the Dynamic Regression option, you can use past values of the predictor series, and you can model effects that take place gradually.

Dynamic regression models are an advanced feature that you are unlikely to find useful unless you have studied the theory of statistical time series analysis. You might want to skip this section if you are not trained in time series modeling.

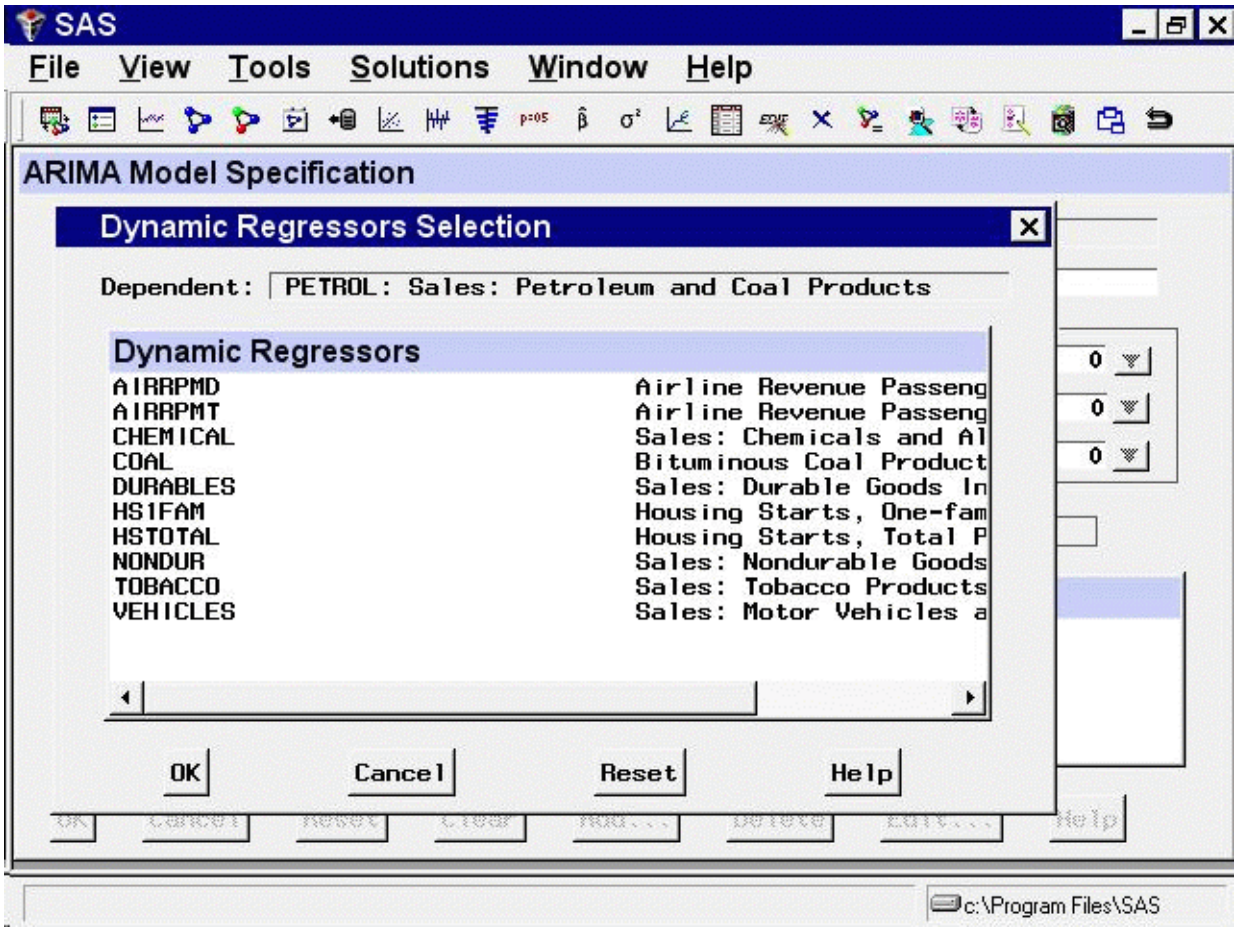
The term *dynamic regression* was introduced by Pankratz (1991) and refers to what Box and Jenkins (1976) named *transfer function models*. In dynamic regression, you have a time series model, similar to an ARIMA model, that predicts how changes in the predictor series affect the dependent series over time.

The dynamic regression model relates the predictor variable to the expected value of the dependent series in the same way that an ARIMA model relates the fluctuations of the dependent series about its conditional mean to the random error term (which is also called the innovation series). For more information about dynamic regression or transfer function see Pankratz (1991); Box and Jenkins (1976). See also Chapter 8, "The ARIMA Procedure."

From the Develop Models window, select `Fit ARIMA Model`. From the ARIMA Model Specification window, select `Add` and then select `Linear Trend` from the menu (shown in Figure 58.1).

Now select `Add` and select `Dynamic Regressor`. This displays the `Dynamic Regressors Selection` window, as shown in Figure 58.11.

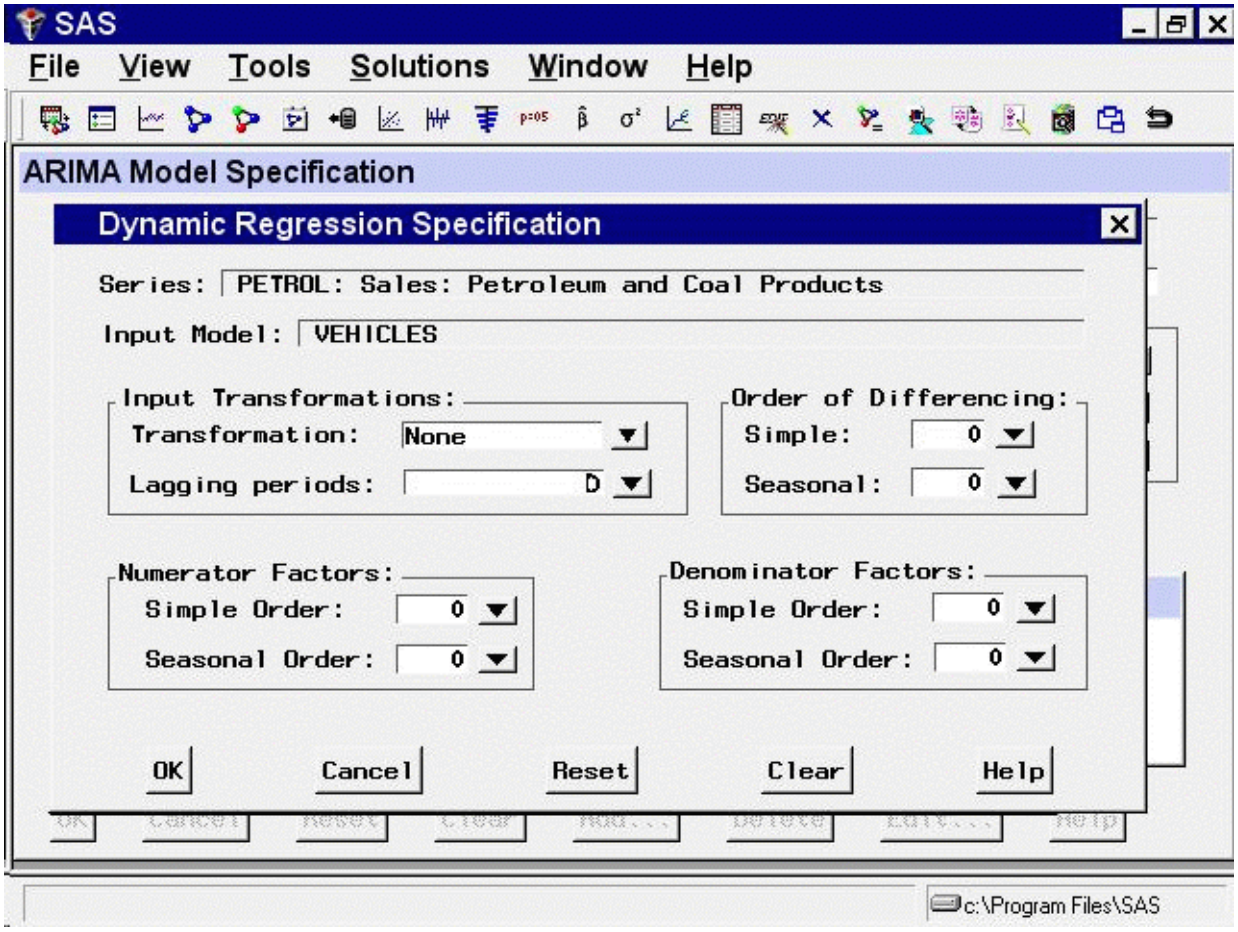
Figure 58.11 Dynamic Regressors Selection Window



You can select only one predictor series when specifying a dynamic regression model. For this example, select `VEHICLES, Sales: Motor Vehicles and Parts`. Then select the `OK` button.

This displays the `Dynamic Regression Specification` window, as shown in Figure 58.12.

Figure 58.12 Dynamic Regression Specification Window



This window consists of four parts. The `Input Transformations` fields enable you to transform or lag the predictor variable. For example, you might use the lagged logarithm of the variable as the predictor series.

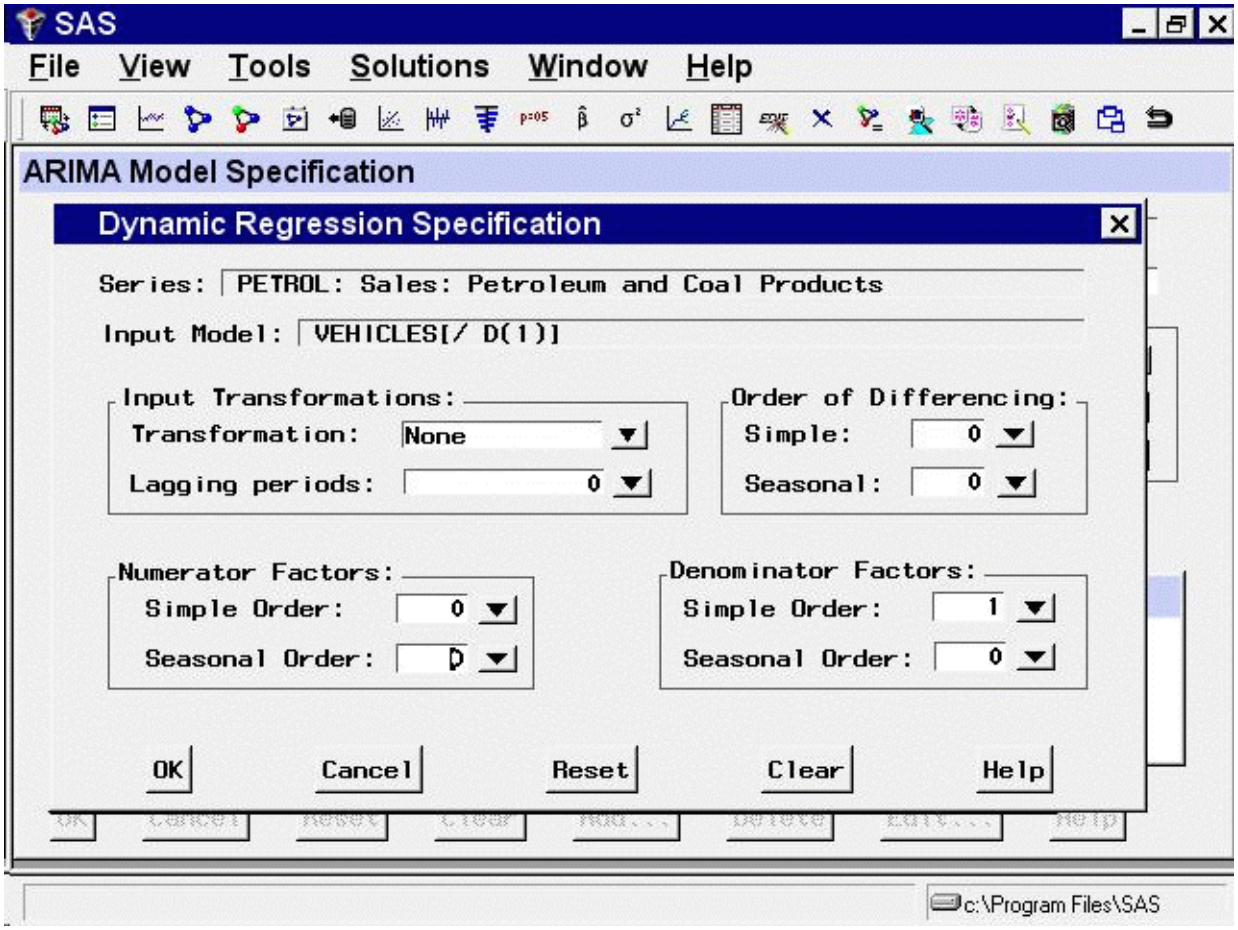
The `Order of Differencing` fields enable you to specify simple and seasonal differencing of the predictor series. For example, you might use changes in the predictor variable instead of the variable itself as the predictor series.

The `Numerator Factors` and `Denominator Factors` fields enable you to specify the orders of simple and seasonal numerator and denominator factors of the transfer function.

Simple regression is a special case of dynamic regression in which the dynamic regression model consists of only a single regression coefficient for the current value of the predictor series. If you select the `OK` button without specifying any options in the Dynamic Regression Specification window, a simple regressor will be added to the model.

For this example, use the `Simple Order` combo box for `Denominator Factors` and set its value to 1. The window now appears as shown in Figure 58.13.

Figure 58.13 Distributed Lag Regression Specified

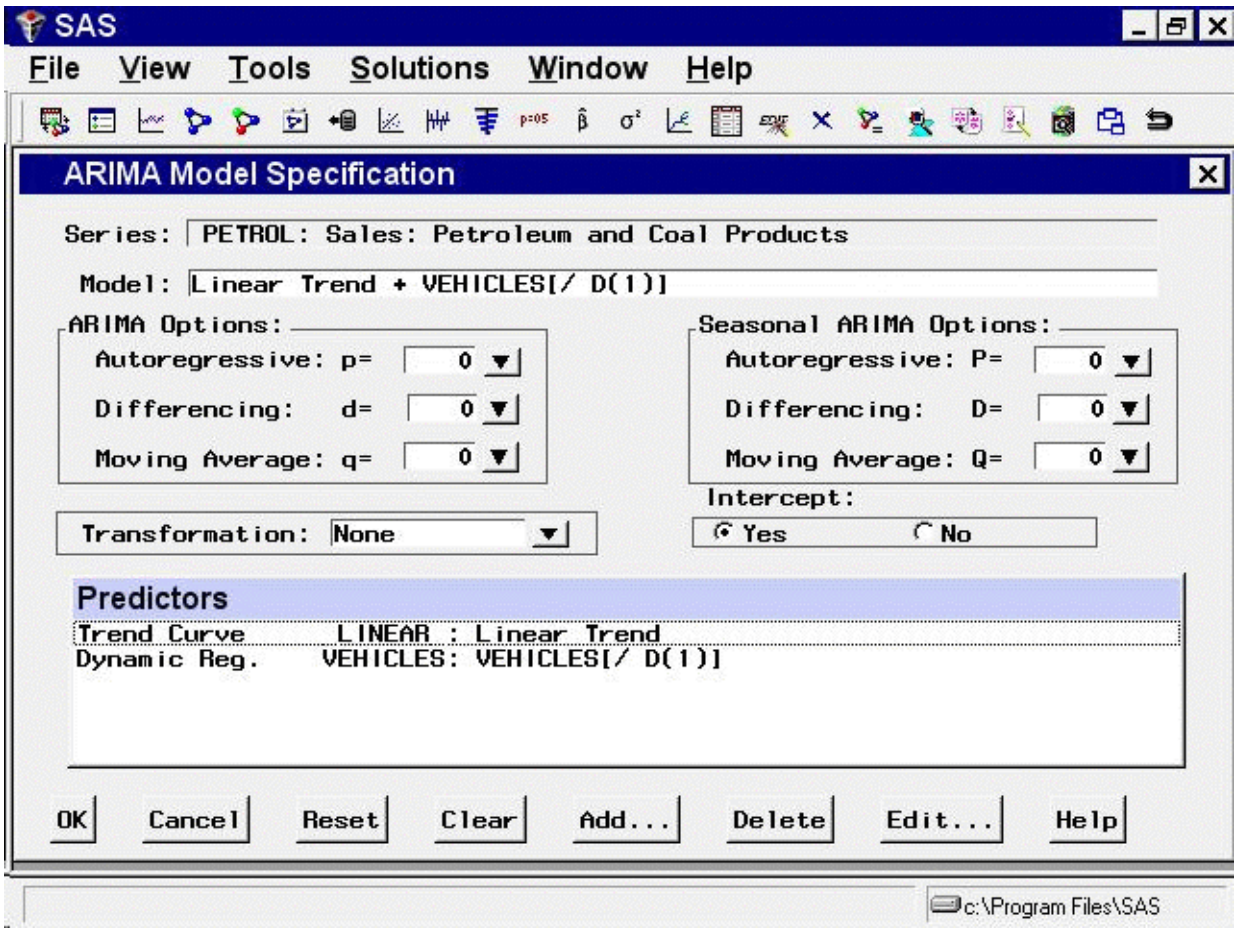


This model is equivalent to regression on an exponentially weighted infinite distributed lag of VEHICLES (in the same way an MA(1) model is equivalent to single exponential smoothing).

Select the `OK` button to add the dynamic regressor to the model predictors list.

In the ARIMA Model Specification window, the Predictors list should now contain two items, a linear trend and a dynamic regressor for VEHICLES, as shown in Figure 58.14.

Figure 58.14 Dynamic Regression Model



This model is a multiple regression of PETROL on a time trend variable and an infinite distributed lag of VEHICLES. Select the OK button to fit the model.

As with simple regressors, if VEHICLES does not already have a forecasting model, an automatic model selection process is performed to find a forecasting model for VEHICLES before the dynamic regression model for PETROL is fit.

Interventions

An *intervention* is a special indicator variable, computed automatically by the system, that identifies time periods affected by unusual events that influence or intervene in the normal path of the time series you are forecasting. When you add an intervention predictor, the indicator variable of the intervention is used as a regressor, and the impact of the intervention event is estimated by regression analysis.

To add an intervention to the Predictors list, you must use the Intervention Specification window to specify the time or times that the intervening event took place and to specify the type of intervention. You can add interventions either through the `Interventions` item of the `Add` action or by selecting `Tools` from the menu bar and then selecting `Define Interventions`.

Intervention specifications are associated with the series. You can specify any number of interventions for each series, and once you define interventions you can select them for inclusion in forecasting models. If you select the `Include Interventions` option in the `Options` menu, any interventions that you have previously specified for a series are automatically added as predictors to forecasting models for the series.

From the Develop Models window, invoke the series viewer by selecting the `View Series Graphically` icon or `Series` under the `View` menu. This displays the Time Series Viewer, as was shown in Figure 58.2.

Note that the trend in the PETROL series shows several clear changes in direction. The upward trend in the first part of the series reverses in 1981. There is a sharp drop in the series towards the end of 1985, after which the trend is again upwardly sloped. Finally, in 1991 the series takes a sharp upward excursion but quickly returns to the trend line.

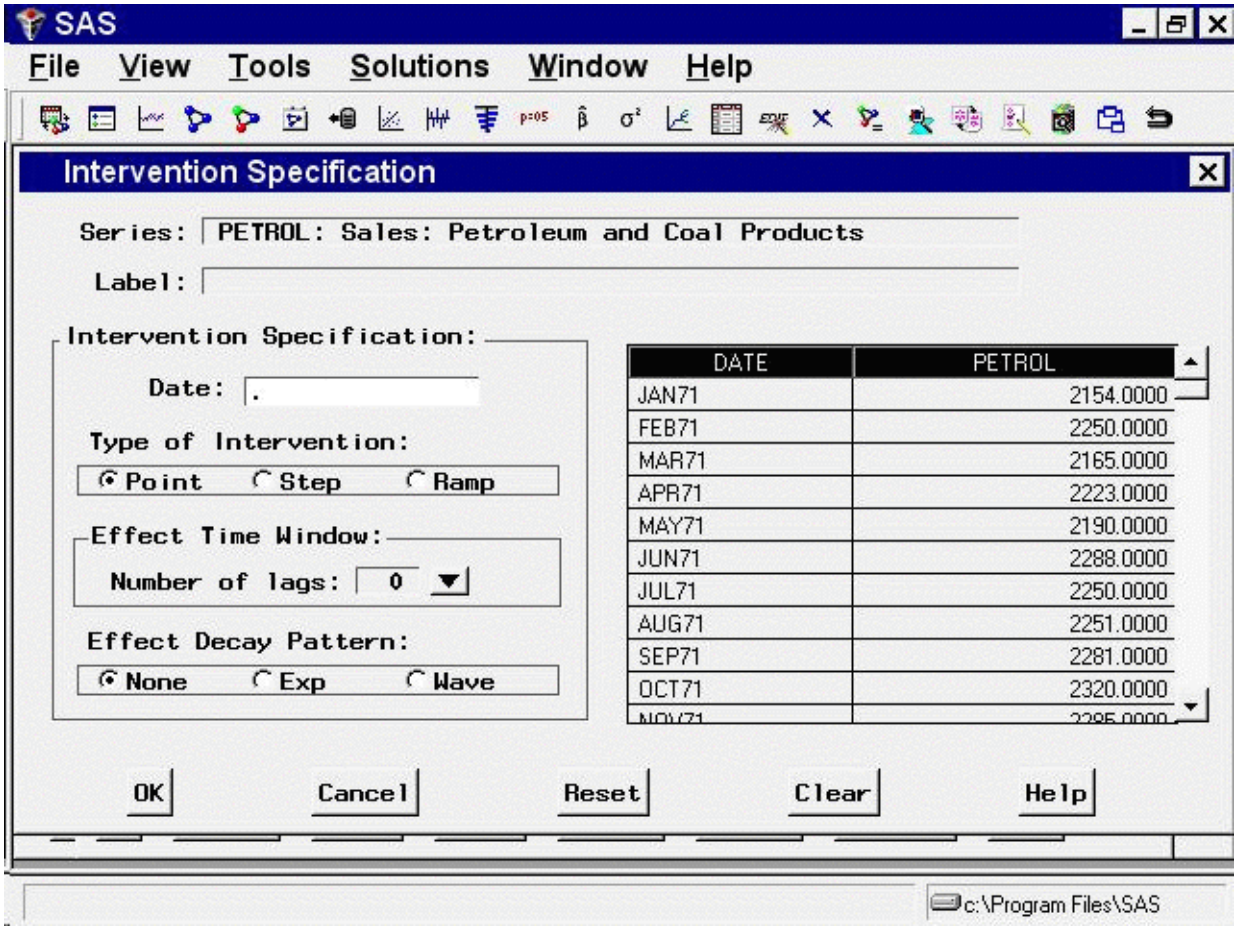
You might have no idea what events caused these changes in the trend of the series, but you can use these patterns to illustrate the use of intervention predictors. To do this, you fit a linear trend model to the series, but modify that trend line by adding intervention effects to model the changes in trend you observe in the series plot.

The Intervention Specification Window

From the Develop Models window, select `Fit ARIMA` model. From the ARIMA Model Specification window, select `Add` and then select `Linear Trend` from the menu (shown in Figure 58.1).

Select `Add` again and then select `Interventions`. If you have any interventions already defined for the series, this selection displays the `Interventions for Series` window. However, since you have not previously defined any interventions, this list is empty. Therefore, the system assumes that you want to add an intervention and displays the `Intervention Specification` window instead, as shown in Figure 58.15.

Figure 58.15 Interventions Specification Window



The top of the Intervention Specification window shows the current series and the label for the new intervention (initially blank). At the right side of the window is a scrollable table showing the values of the series. This table helps you locate the dates of the events you want to model.

At the left of the window is an area titled `Intervention Specification` that contains the options for defining the intervention predictor. The `Date` field specifies the time that the intervention occurs. You can type a date value in the `Date` field, or you can set the Date value by selecting a row from the table of series values at the right side of the window.

The area titled `Type of Intervention` controls the kind of indicator variable constructed to model the intervention effect. You can specify the following kinds of interventions:

Point — is used to indicate an event that occurs in a single time period. An example of a point event is a strike that shuts down production for part of a time period. The value of the intervention's indicator variable is zero except for the date specified.

Step — is used to indicate a continuing event that changes the level of the series. An example of a step event is a change in the law, such as a tax rate increase. The value of the intervention's indicator variable is zero before the date specified and 1 thereafter.

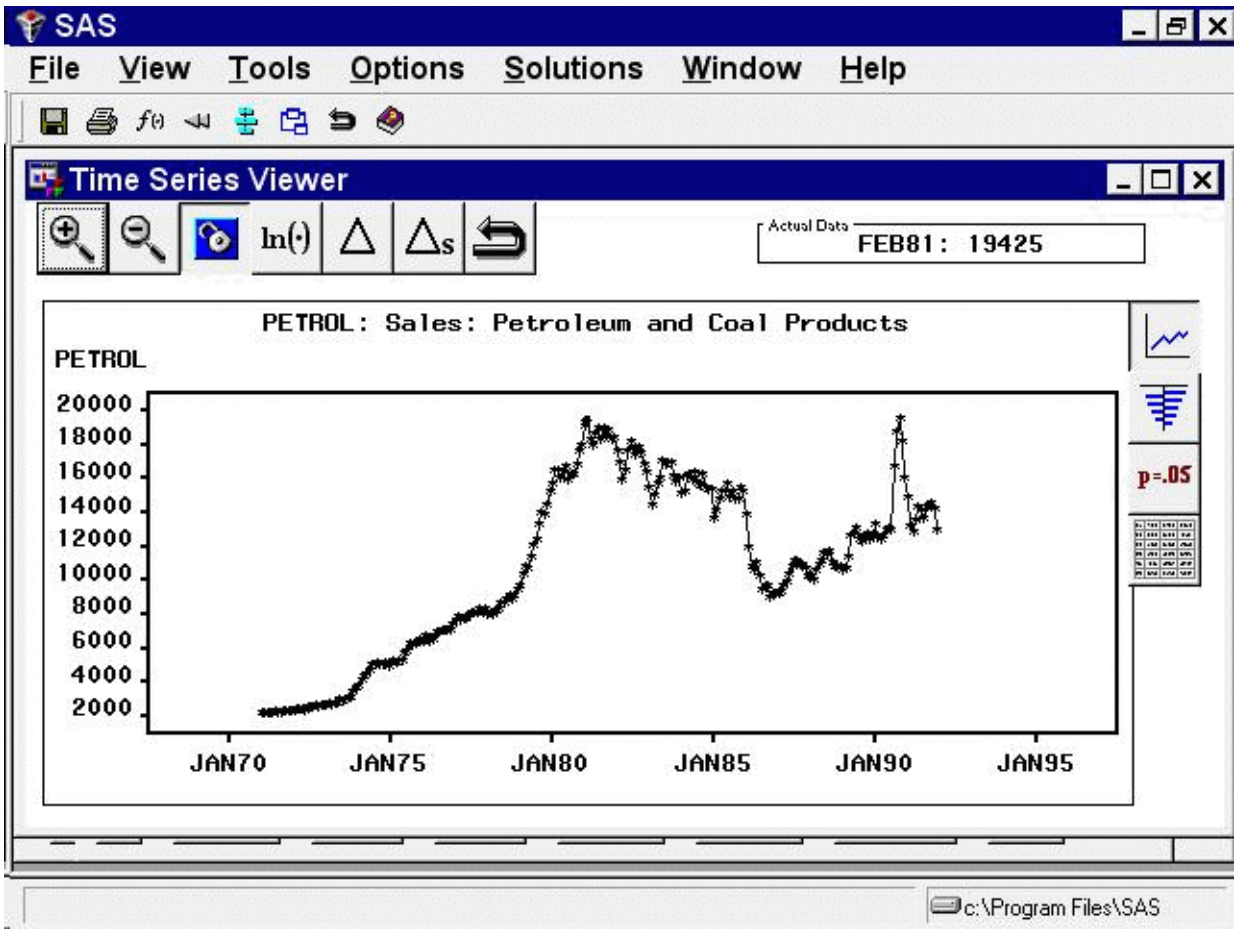
Ramp — is used to indicate a continuing event that changes the trend of the series. The value of the intervention's indicator variable is zero before the date specified, and it increases linearly with time thereafter.

The areas titled `Effect Time Window` and `Effect Decay Pattern` specify how to model the effect that the intervention has on the dependent series. These options are not used for simple interventions, they will be discussed later in this chapter.

Specifying a Trend Change Intervention

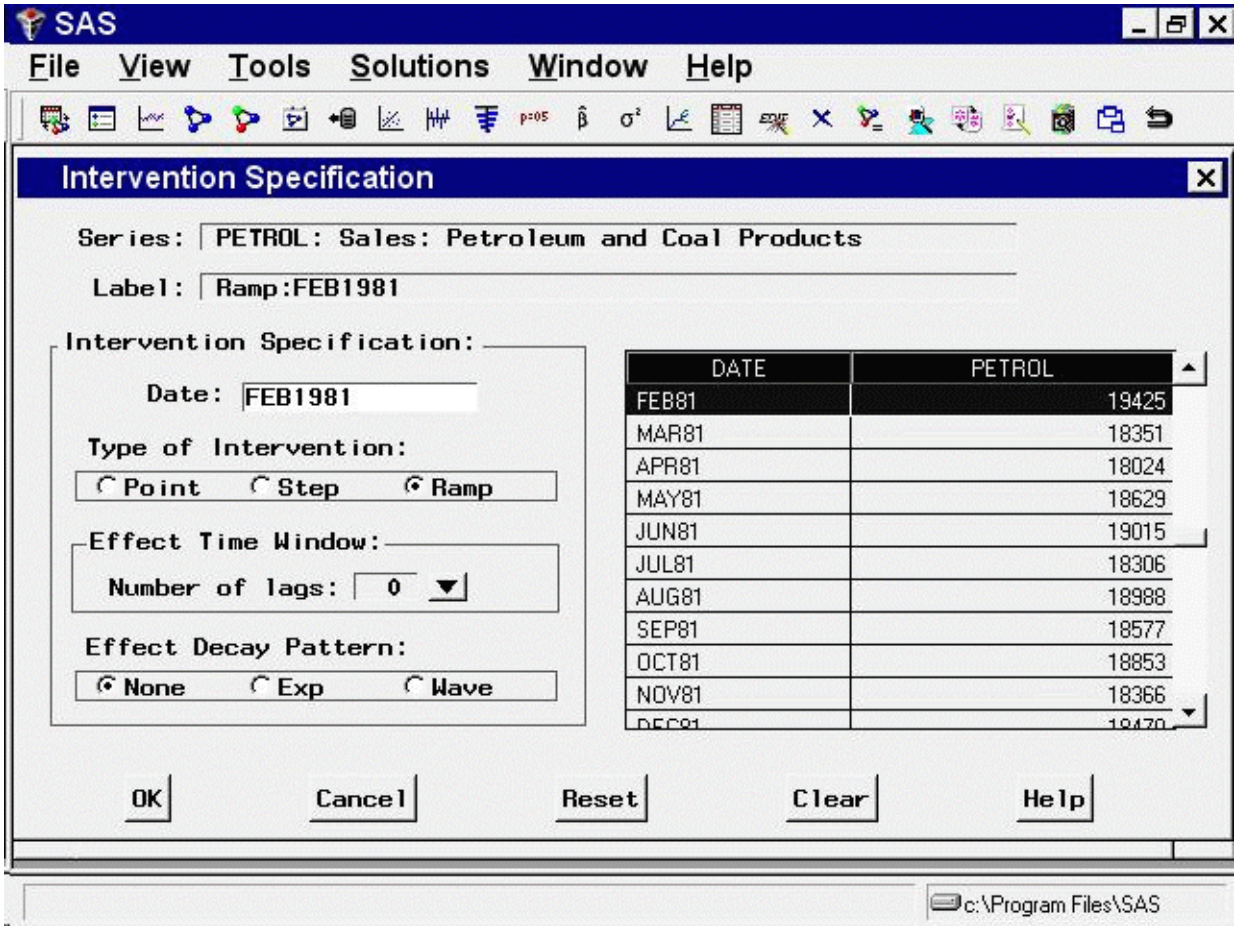
In the Time Series Viewer window position the mouse over the highest point in 1981 and select the point. This displays the data value, 19425, and date, February 1981, of that point in the upper-right corner of the Time Series Viewer, as shown in Figure 58.16.

Figure 58.16 Identifying the Turning Point



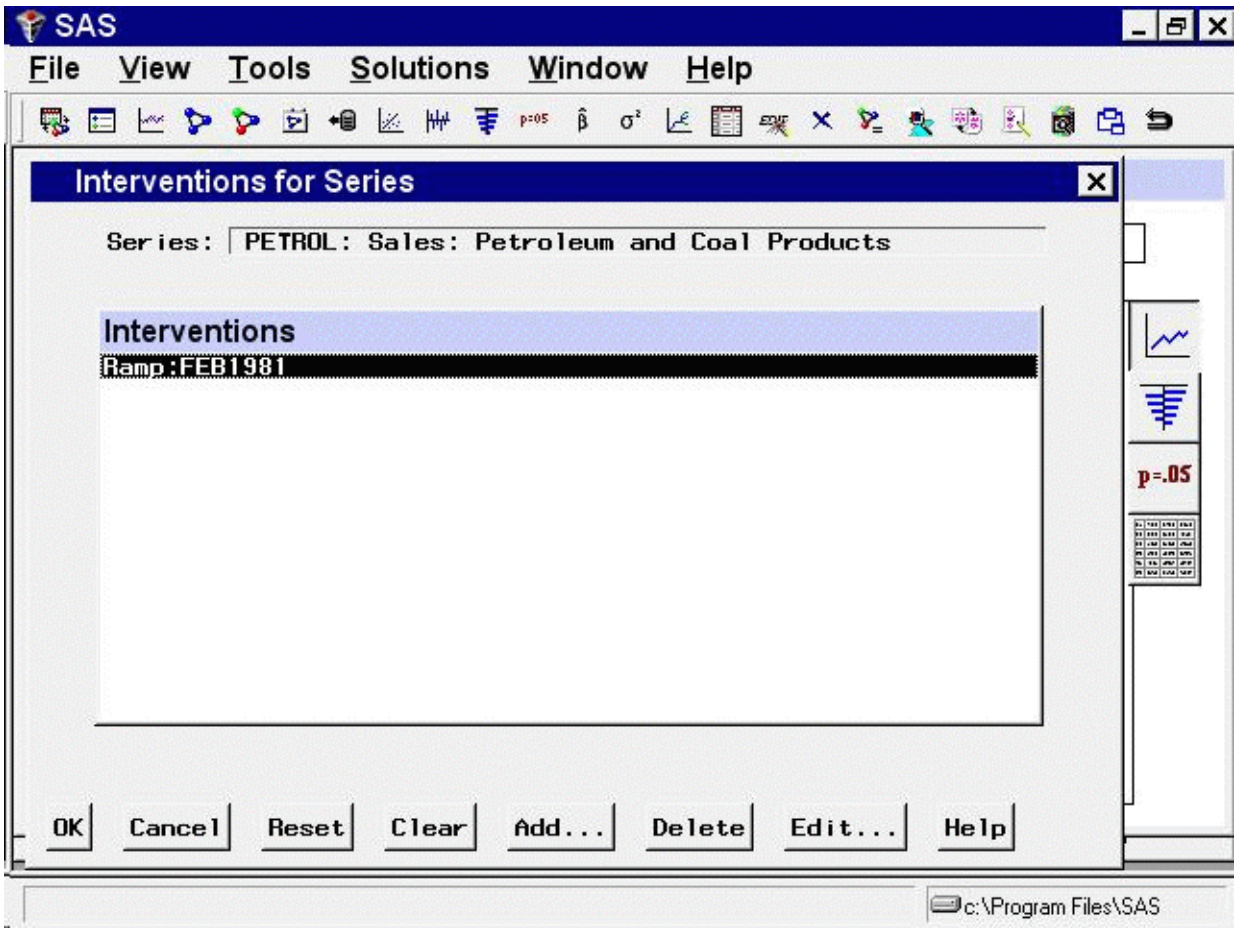
Now that you know the date that the trend reversal occurred, enter that date in the `Date` field of the Intervention Specification window. Select `Ramp` as the type of intervention. The window should now appear as shown in Figure 58.17.

Figure 58.17 Ramp Intervention Specified



Select the `OK` button. This adds the intervention to the list of interventions for the PETROL series, and returns you to the `Interventions for Series` window, as shown in Figure 58.18.

Figure 58.18 Interventions for Series Window



This window allows you to select interventions for inclusion in the forecasting model. Since you need to define other interventions, select the `Add` button. This returns you to the Intervention Specification window (shown in Figure 58.15).

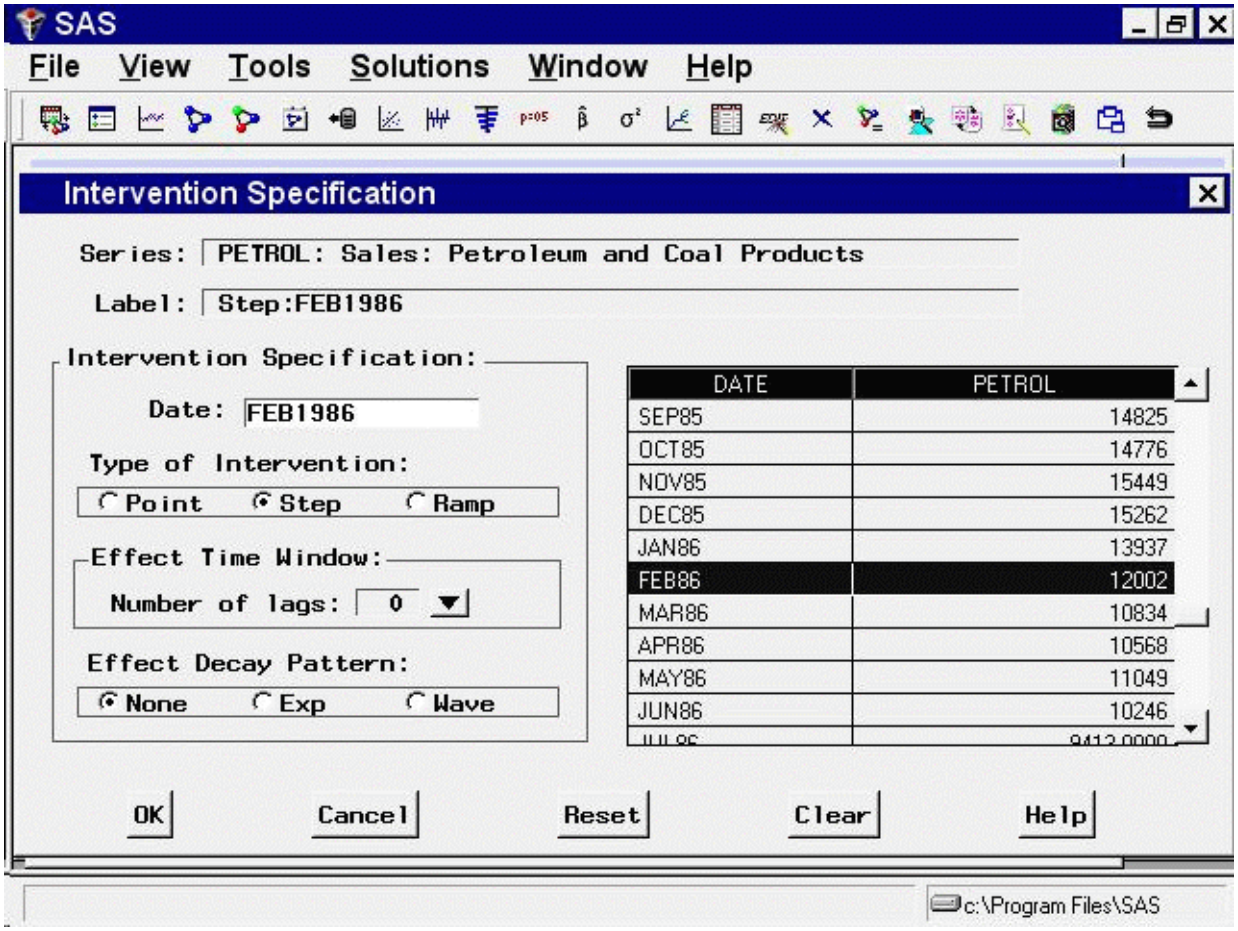
Specifying a Level Change Intervention

Now add an intervention to account for the drop in the series in late 1985. You can locate the date of this event by selecting points in the Time Series Viewer plot or by scrolling through the data values table in the Interventions Specification window. Use the latter method so that you can see how this works.

Scrolling through the table, you see that the drop was from 15262 in December 1985, to 13937 in January 1986, to 12002 in February, to 10834 in March. Since the drop took place over several periods, you could use another ramp type intervention. However, this example represents the drop as a sudden event by using a step intervention and uses February 1986 as the approximate time of the drop.

Select the table row for February 1986 to set the `Date` field. Select `Step` as the intervention type. The window should now appear as shown in Figure 58.19.

Figure 58.19 Step Intervention Specified



Select the `OK` button to add this intervention to the list for the series.

Since the trend reverses again after the drop, add a ramp intervention for the same date as the step intervention. Select `Add` from the Interventions for Series window. Enter FEB86 in the `Date` field, select `Ramp`, and then select the `OK` button.

Modeling Complex Intervention Effects

You have now defined three interventions to model the changes in trend and level. The excursion near the end of the series remains to be dealt with.

Select `Add` from the Interventions for Series window. Scroll through the data values and select the date on which the excursion began, August 1990. Leave the intervention type as Point.

The pattern of the series from August 1990 through January 1991 is more complex than a simple shift in level or trend. For this pattern, you need a complex intervention model for an event that causes a sharp rise followed by a rapid return to the previous trend line. To specify this model, use the `Effect Time Window` and `Effect Decay Rate` options.

The `Effect Time Window` option controls the number of lags of the intervention's indicator variable used to model the effect of the intervention on the dependent series. For a simple intervention, the number of lags is zero, which means that the effect of the intervention is modeled by fitting a single regression coefficient to the intervention's indicator variable.

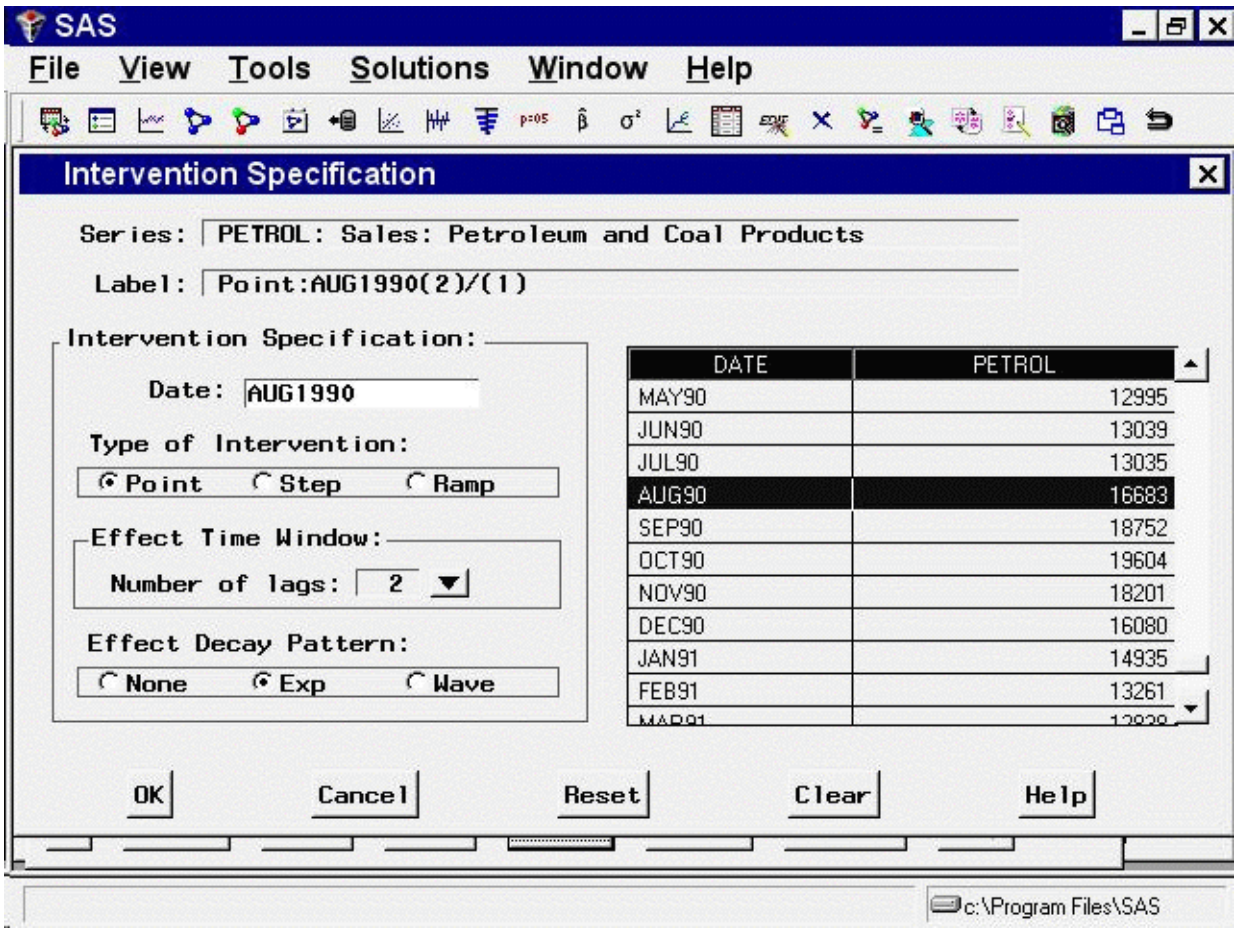
When you set the number of lags greater than zero, regression coefficients are fit to lags of the indicator variable. This allows you to model interventions whose effects take place gradually, or to model rebound effects. For example, severe weather might reduce production during one period but cause an increase in production in the following period as producers struggle to catch up. You could model this by using a point intervention with an effect time window of 1 lag. This would fit two coefficients for the intervention, one for the immediate effect and one for the delayed effect.

The `Effect Decay Pattern` option controls how the effect of the intervention dissipates over time. `None` specifies that there is no gradual decay: for point interventions, the effect ends immediately; for step and ramp interventions, the effect continues indefinitely. `Exp` specifies that the effect declines at an exponential rate. `Wave` specifies that the effect declines like an exponentially damped sine wave (or as the sum of two exponentials, depending on the fit to the data).

If you are familiar with time series analysis, these options might be clearer if you note that together the Effect Time Window and Effect Decay Pattern options define the numerator and denominator orders of a transfer function or dynamic regression model for the indicator variable of the intervention. See the section "Dynamic Regressor" on page 3750 for more information.

For this example, select 2 lags as the value of the Event Time Window option, and select `Exp` as the Effect Decay Pattern option. The window should now appear as shown in Figure 58.20.

Figure 58.20 Complex Intervention Model

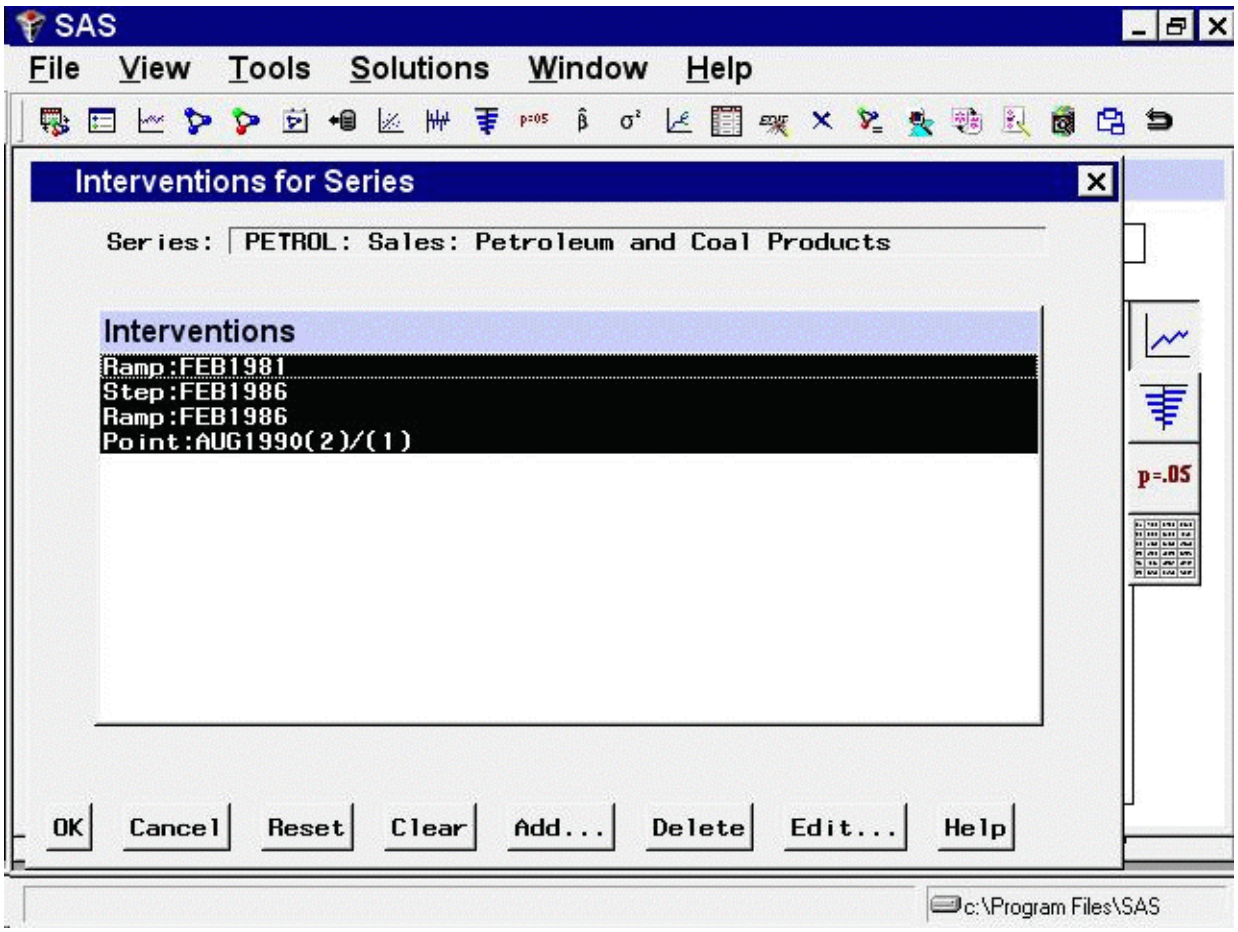


Select the OK button to add the intervention to the list.

Fitting the Intervention Model

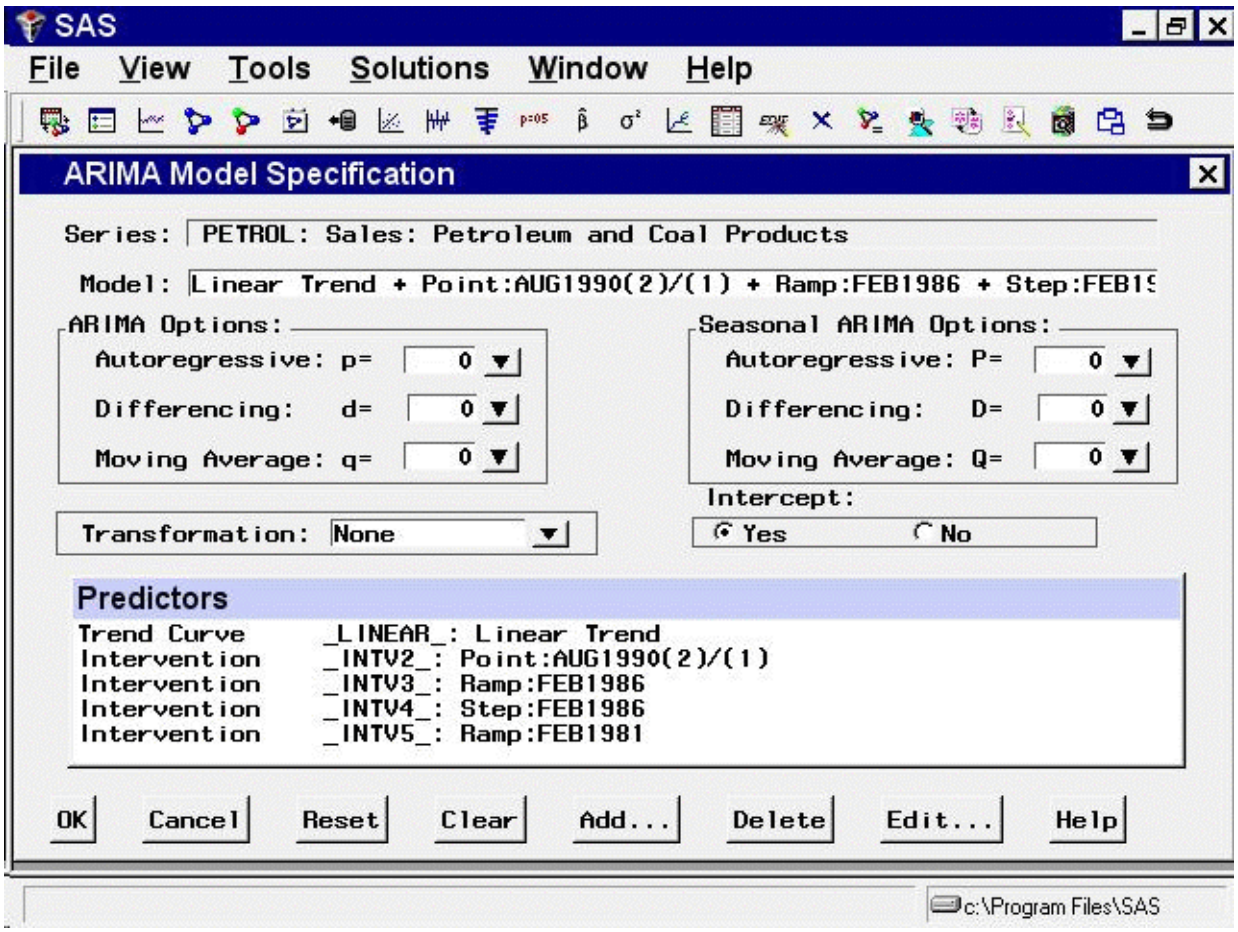
The Interventions for Series window now contains definitions for four intervention predictors. Select all four interventions, as shown in Figure 58.21.

Figure 58.21 Interventions for Series Window



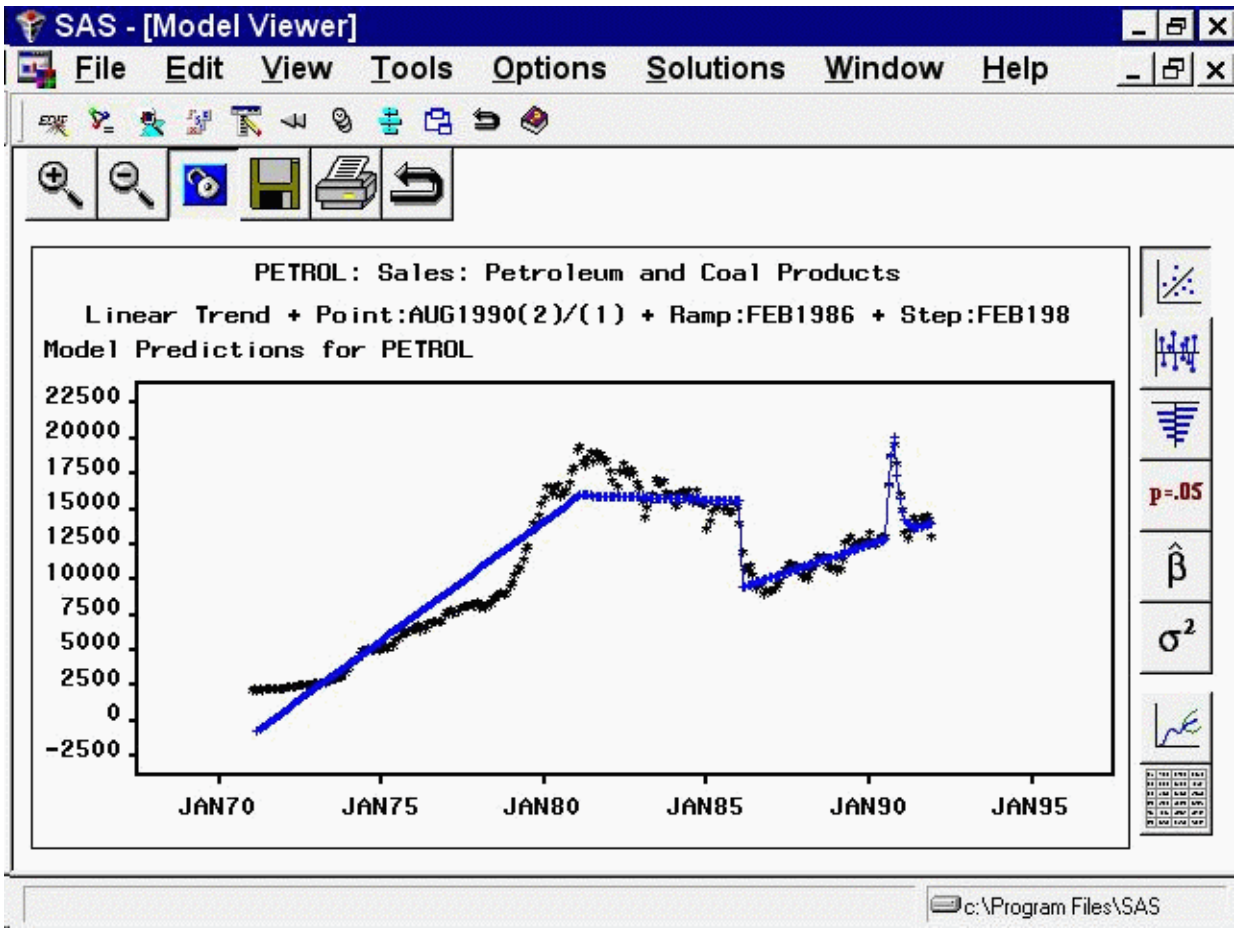
Select the OK button. This returns you to the ARIMA Model Specification window, which now lists items in the Predictors list, as shown in Figure 58.22.

Figure 58.22 Linear Trend with Interventions Specified



Select the OK button to fit this model. After the model is fit, bring up the Model Viewer. You will see a plot of the model predictions, as shown in Figure 58.23.

Figure 58.23 Linear Trend with Interventions Model



You can use the Zoom In feature to take a closer look at how the complex intervention effect fits the excursion in the series starting in August 1990.

Limitations of Intervention Predictors

Note that the model you have just fit is intended only to illustrate the specification of interventions. It is not intended as an example of good forecasting practice.

The use of continuing (step and ramp type) interventions as predictors has some limitations that you should consider. If you model a change in trend with a simple ramp intervention, then the trend in the data before the date of the intervention has no influence on the forecasts. Likewise, when you use a step intervention, the average level of the series before the intervention has no influence on the forecasts.

Only the final trend and level at the end of the series are extrapolated into the forecast period. If a linear trend is the only pattern of interest, then instead of specifying step or ramp interventions, it would be simpler to adjust the period of fit so that the model ignores the data before the final trend or level change.

Step and ramp interventions are valuable when there are other patterns in the data—such as seasonality, autocorrelated errors, and error variance—that are stable across the changes in level or trend. Step and ramp interventions enable you to fit seasonal and error autocorrelation patterns to the whole series while fitting the trend only to the latter part of the series.

Point interventions are a useful tool for dealing with outliers in the data. A point intervention will fit the series value at the specified date exactly, and it has the effect of removing that point from the analysis. When you specify an effect time window, a point intervention will exactly fit as many additional points as the number of lags specified.

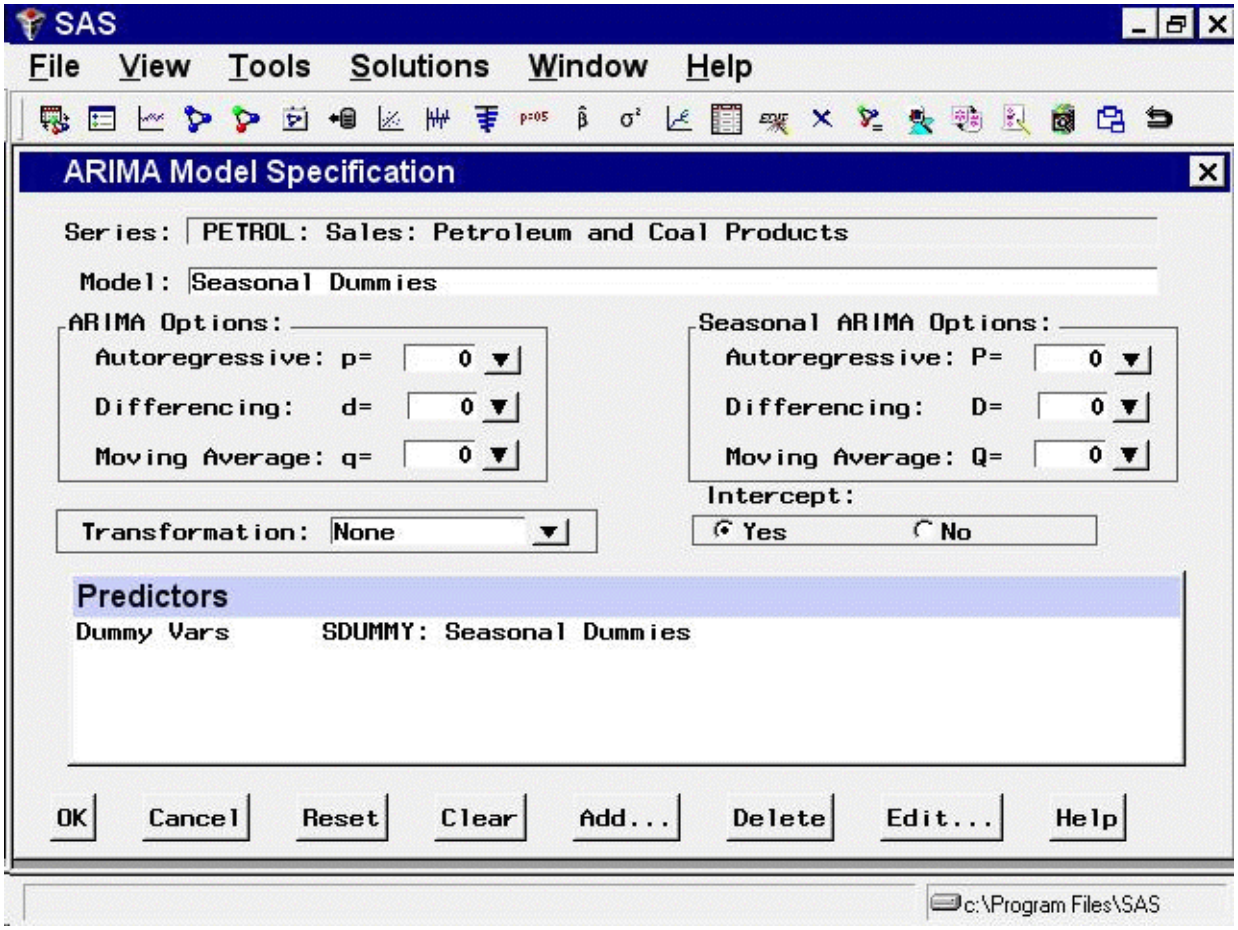
Seasonal Dummies

A *Seasonal Dummies* predictor is a special feature that adds to the model seasonal indicator or "dummy" variables to serve as regressors for seasonal effects.

From the Develop Models window, select `Fit ARIMA Model`. From the ARIMA Model Specification window, select `Add` and then select `Seasonal Dummies` from the menu (shown in Figure 58.1).

A Seasonal Dummies input is added to the Predictors list, as shown in Figure 58.24.

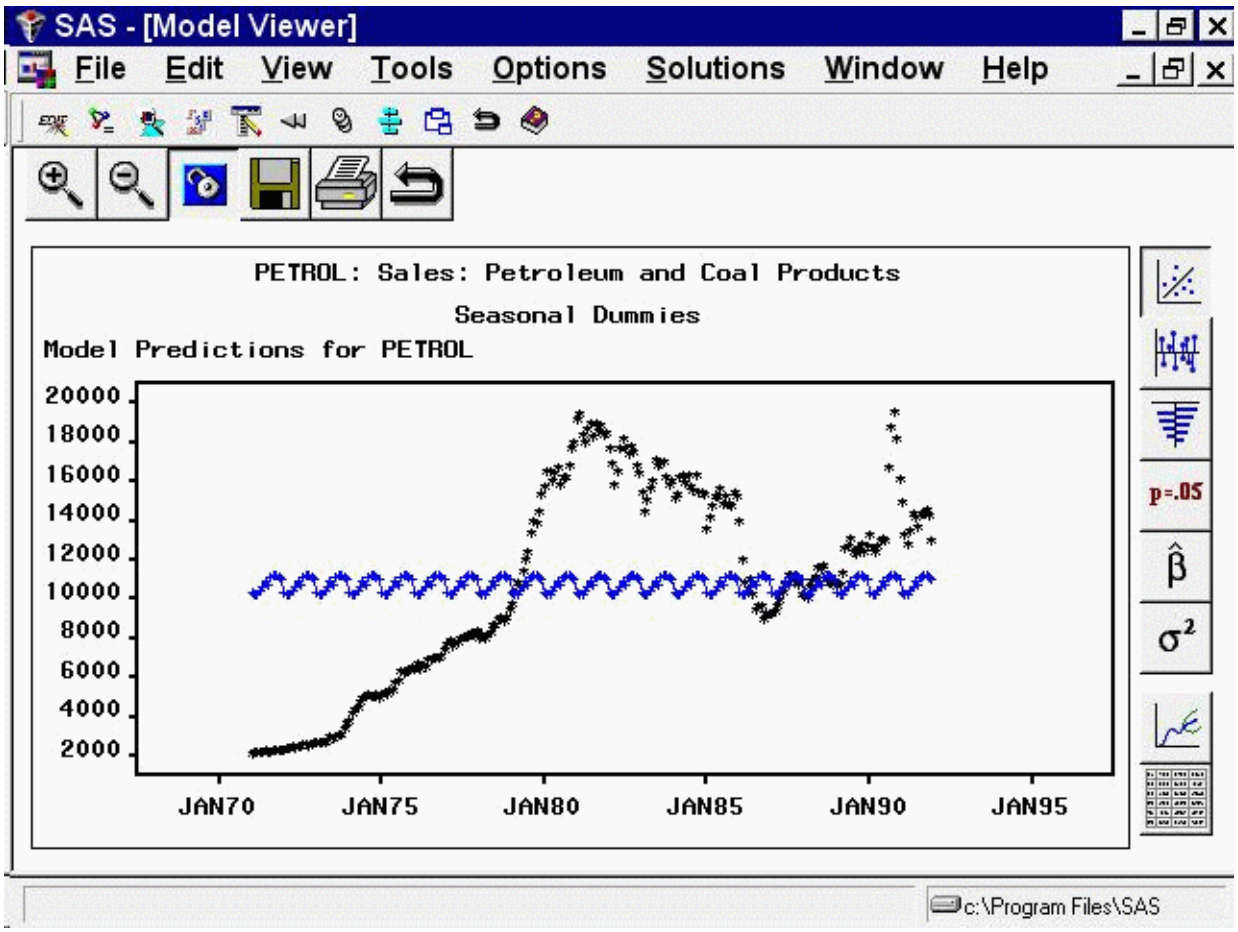
Figure 58.24 Seasonal Dummies Specified



Select the OK button. A model consisting of an intercept and 11 seasonal dummy variables is fit and added to the model list in the Develop Models window. This is effectively a mean model with a separate mean for each month.

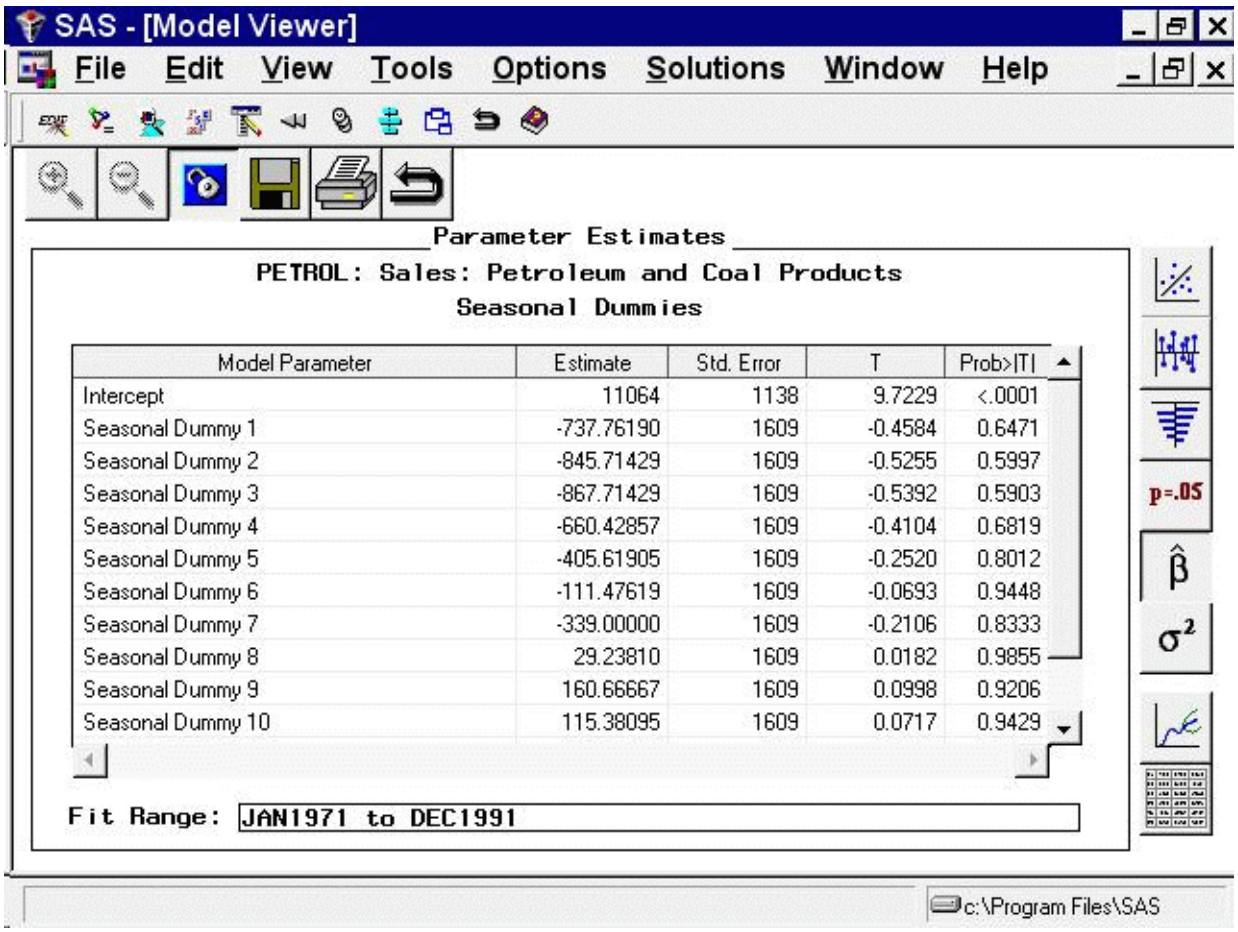
Now return to the Model Viewer, which displays a plot of the model predictions and actual series values, as shown in Figure 58.25. This is obviously a poor model for this series, but it serves to illustrate how seasonal dummy variables work.

Figure 58.25 Seasonal Dummies Model



Now select the parameter estimates icon, the fifth from the top on the vertical toolbar. This displays the `Parameter Estimates` table, as shown in Figure 58.26.

Figure 58.26 Parameter Estimates for Seasonal Dummies Model



Since the data for this example are monthly, the Seasonal Dummies option added 11 seasonal dummy variables. These include a dummy regressor variable that is 1.0 for January and 0 for other months, a regressor that is 1.0 only for February, and so forth through November.

Because the model includes an intercept, no dummy variable is added for December. The December effect is measured by the intercept, while the effect of other seasons is measured by the difference between the intercept and the estimated regression coefficient for the season's dummy variable.

The same principle applies for other data frequencies: the "Seasonal Dummy 1" parameter always refers to the first period in the seasonal cycle; and, when an intercept is present in the model, there is no seasonal dummy parameter for the last period in the seasonal cycle.

References

Box, G. E. P., and Jenkins, G. M. (1976). *Time Series Analysis: Forecasting and Control*. Rev. ed. San Francisco: Holden-Day.

Pankratz, A. (1991). *Forecasting with Dynamic Regression Models*. New York: John Wiley & Sons.

Chapter 59
Command Reference

Contents

TSVIEW Command and Macro **3771**
Syntax 3771
Examples 3772
FORECAST Command and Macro **3772**
Syntax 3773
Examples 3776

TSVIEW Command and Macro

The TSVIEW command invokes the Time Series Viewer. This is a component of the Time Series Forecasting System that can also be used as a standalone graphical viewer for any time series data set or view. See the section "Time Series Viewer Window" in Chapter 60, "Window Reference," for more information.

The TSVIEW command must be specified from the command line or an SCL program. If you need to submit from the program editor, use the %TSVIEW macro instead. You can use the macro within a data step program, but you must submit it within the SAS windowing environment.

If the TSVIEW command or %TSVIEW macro is issued without arguments, the Series Selection window appears to enable you to select an input data set and series. This is equivalent to selecting "Time Series Viewer" from the Analysis submenu of the Solutions menu. By specifying the DATA= and VAR= arguments, you can bring up the Time Series Viewer window directly. The ID= and INTERVAL= arguments are useful when the system cannot determine them automatically from the data.

Syntax

The TSVIEW command has the following form:

TSVIEW *[options]* **;**

The %TSVIEW macro has the following form:

%TSVIEW *[(option, . . . , option)]* **;**

The following options can be specified for the command and the macro.

DATA=*data set name*
specifies the name of the SAS data set containing the input data.

VAR=*time series variable name*
specifies the series variable name. It must be a numeric variable contained in the data set.

ID=*time id variable name*
specifies the time ID variable name for the data set. If the ID= option is not specified, the system attempts to locate the variables named DATE, DATETIME, and TIME in the data set specified by the DATA= option.

INTERVAL=*interval name*
specifies the time ID interval between observations in the data set.

Examples

TSVIEW Command

```
tsview data=sashelp.air var=air
tsview data=dept.prod var=units id=period interval=qtr
```

%TSVIEW Macro

```
%tsview( data=sashelp.air, var=air);
%tsview( data=dept.prod, var=units, id=period, interval=qtr);
```

FORECAST Command and Macro

The FORECAST command invokes the Time Series Forecasting System. The command must be specified from the command line or an SCL program. If you need to submit from the program editor, use the %FORECAST macro instead. You can use the macro within a data step program, but you must submit it within the SAS windowing environment.

If the FORECAST command or %FORECAST macro is issued without arguments, the Time Series Forecasting window appears. This is equivalent to selecting "Time Series Forecasting System" from the `Analysis` submenu of the Solutions menu.

Using the arguments, it is possible to do the following:

- Bring up the system with information already filled into some of the fields
- Bring up the system starting at a different window than the default Time Series Forecasting window

- Run the system in unattended mode so that a task such as creating a forecast data set is accomplished without any user interaction. By submitting such commands repeatedly from a SAS/AF or SAS/EIS application, it is possible to do "batch" processing for many data sets or by-group processing for many subsets of a data set. You can create a project in unattended mode and later open it for inspection interactively. You can also create a project interactively in order to set options, fit a model, or edit the list of models, and then use this project later in unattended mode.

The `Forecast Command Builder`, a point-and-click SAS/AF application, makes it easy to specify, run, save, and rerun forecasting jobs by using the FORECAST command. To use it, enter the following on the command line (not the program editor):

```
%FCB
```

or

```
AF C=SASHELP.FORCAST.FORCCMD.FRAME.
```

Syntax

The FORECAST command has the following form:

FORECAST *[options]* **;**

The %FORECAST macro has the following form:

%FORECAST *[(option, . . . , option)]* **;**

The following options can be specified for the command and the macro.

PROJECT=*project name*
specifies the name of the SAS catalog entry in which forecasting models and other results are stored and from which previously stored results are loaded into the forecasting system.

DATA=*data set name*
specifies the name of the SAS data set containing the input data.

VAR=*time series variable name*
specifies the series variable name. It must be a numeric variable contained in the data set.

ID=*time id variable name*
specifies the time ID variable name for the data set. If the ID= option is not specified, the system attempts to locate the variables named DATE, DATETIME, and TIME in the data set specified by the DATA= option. However, it is recommended that you specify the time ID variable whenever you are using the ENTRY= argument.

INTERVAL=*interval name*
specifies the time ID interval between observations in the data set. Commonly used intervals are `year`, `semiyear`, `qtr`, `month`, `semimonth`, `week`, `weekday`, `day`, `hour`, `minute`, and `second`. See Chapter 5, "Date Intervals, Formats, and Functions," for information about more complex interval specifications. If the INTERVAL= option is not specified, the system attempts to determine the interval based on the time ID variable. However, it is recommended that you specify the interval whenever you are using the ENTRY= argument.

STAT=*statistic*

specifies the name of the goodness-of-fit statistic to be used as the model selection criterion. The default is RMSE. Valid names are

`sse`	sum of square error
`mse`	mean square error
`rmse`	root mean square error
`mae`	mean absolute error
`mape`	mean absolute percent error
`aic`	Akaike information criterion
`sbc`	Schwarz Bayesian information criterion
`rsquare`	R-square
`ajdrsq`	adjusted R-square
`rwrsq`	random walk R-square
`arsq`	Amemiya's adjusted R-square
`apc`	Amemiya's prediction criterion

CLIMIT=*integer*

specifies the level of the confidence limits to be computed for the forecast. This integer represents a percentage; for example, 925 indicates 92.5% confidence limits. The default is 95—that is, 95% confidence limits.

HORIZON=*integer*

specifies the number of periods into the future for which forecasts are computed. The default is 12 periods. The maximum is 9999.

ENTRY=*name*

The name of an entry point into the system. Valid names are

`main`	starts the system at the Time Series Forecasting window (default).
`devmod`	starts the system at the Develop Models window.
`viewmod`	starts the system at the Model Viewer window. Specify a project that contains a forecasting model by using the PROJECT= option. If a project containing a model is not specified, the message "No forecasting model to view" appears.
`viewser`	starts the system at the Time Series Viewer window.
`autofit`	runs the system in unattended mode, fitting a forecasting model automatically and saving it in a project. If PROJECT= is not specified, the default project name SASUSER.FMSPROJ.PROJ is used.
`forecast`	runs the system in unattended mode to generate a forecast data set. The name of this data set is specified by the OUT= parameter. If OUT= is not specified, a window appears to prompt for the name and label of the output data set. If PROJECT= is not specified, the default project name SASUSER.FMSPROJ.PROJ is used. If the project does not exist or does not contain a forecasting model for the specified

series, automatic model fitting is performed and the forecast is computed by using the automatically selected model. If the project exists and contains a forecasting model for the specified series, the forecast is computed by using this model. If the series covers a different time range than it did when the project was created, use the REFIT or REEVAL keyword to reset the time ranges.

OUT=*argument*
specifies one or two-level name of a SAS data set in which forecasts are saved. Use in conjunction with ENTRY=FORECAST. If omitted, the system prompts for the name of the forecast data set.

KEEP=*argument*
specifies the number of models to keep in the project when automatic model fitting is performed. This corresponds to `Models to Keep` in the Automatic Model Selection Options window. A value greater than 9 indicates that all models are kept. The default is 1.

DIAG=YES|NO
specifies which models to search with regard to series diagnostics. DIAG= YES causes the automatic model selection process to search only over those models that are consistent with the series diagnostics. DIAG= NO causes the automatic model selection process to search over all models in the selection list, without regard for the series diagnostics. This corresponds to `Models to Fit` in the Automatic Model Selection Options window. The default is YES.

REFIT=*keyword*
(for macro usage) refits a previously saved forecasting model by using the current fit range; that is, it reestimates the model parameters. Refitting also causes the model to be reevaluated (statistics of fit recomputed), and it causes the time ranges to be reset if the data range has changed (for example, if new observations have been added to the series). This keyword has no effect if you do not use the PROJECT= argument to reference an existing project containing a forecasting model. Use the REFIT keyword if you have added new data to the input series and you want to refit the forecasting model and update the forecast by using the new time ranges. Be sure to use the same project, data set, and series names that you used previously.

REEVAL=*keyword*
(for macro usage) reevaluates a previously saved forecasting model by using the current evaluation range; that is, it recomputes the statistics of fit. Reevaluating also causes the time ranges to be reset if the data range has changed (for example, if new observations have been added to the series). It does not refit the model parameters. This keyword has no effect if you also specify REFIT, or if you do not use the PROJECT= argument to reference an existing project containing a forecasting model. Use the REEVAL keyword if you have added new data to the input series and want to update your forecast by using a previously fit forecasting model and the same project, data set, and series names that you used previously.

Examples

FORECAST Command

The following command opens the Time Series Forecasting window with the data set name and series name filled in. The time ID variable is also filled in since the data set contains the variable DATE. The interval is filled in because the system recognizes that the observations are monthly.

```
forecast data=sashelp.air var=air
```

The following command opens the Time Series Forecasting window with the project, data set name, series, time ID, and interval fields filled in, assuming that the project SAMPROJ was previously saved either interactively or by using unattended mode as depicted below. Previously fit models appear when the Develop Models or Manage Projects window is opened.

```
forecast project=samproj
```

The following command runs the system in unattended mode, fitting a model automatically, storing it in the project SAMPROJ in the default catalog SASUSER.FMSPROJ, and placing the forecasts in the data set WORK.SAMPOUT.

```
forecast data=sashelp.workers var=electric id=date interval=month
project=samproj entry=forecast out=sampout
```

The following command assumes that a new month's data have been added to the data set from the previous example and that an updated forecast is needed that uses the previously fit model. Time ranges are automatically updated to include the new data since the REEVAL keyword is included. Substitute REFIT for REEVAL if you want the system to reestimate the model parameters.

```
forecast data=sashelp.workers var=electric id=date interval=month
project=samproj entry=forecast out=sampout reeval
```

The following command opens the model viewer with the project created in the previous example and with 99 percent confidence limits in the forecast graph.

```
forecast data=sashelp.workers var=electric id=date interval=month
project=samproj entry=viewmod climit=99
```

The final example illustrates using unattended mode with an existing project that has been defined interactively. In this example, the goal is to add a model to the model selection list, to specify that all models in that list be fit, and that all models which are fit successfully be retained.

First open the Time Series Forecasting window and specify a new project name, WORKPROJ. Then select `Develop Models`, choosing SASHELP.WORKERS as the data set and MASONRY as the series. Now select "Model Selection List" from the Options menu. In the Model Selection List window, click `Actions`, then `Add`, and then `ARIMA Model`. Define the model `ARIMA(0,1,0)(0,1,0)s NOINT` by setting the differencing

value to 1 under both `ARIMA Options` and `Seasonal ARIMA Options`. Select `OK` to save the model and `OK` to close the Model Selection List window. Now select "Automatic Fit" from the Options menu. In the Automatic Model Selection Options window, select "All autofit models in selection list" in the Models to fit radio box, select "All models" from the `Models to keep` combo box, and then click `OK` to close the window. Select "Save Project" from the File menu, and then close the Develop Models window and the Time Series Forecasting window. You now have a project with a new model added to the selection list, options set for automatic model fitting, and one series selected but no models fit.

Now enter the command:

```
forecast data=sashelp.workers var=electric id=date interval=month
project=workproj entry=forecast out=workforc
```

The system runs in unattended mode to update the project and create the forecast data set WORKFORC. Check the messages in the Log window to find out if the run was successful and which model was selected for forecasting. To see the forecast data set, issue the command `viewtable WORKFORC`. To see the contents of the project, open the Time Series Forecasting window, open the project WORKPROJ, and select "Manage Projects." You will see that the variable ELECTRIC was added to the project and has a forecasting model. Select this row in the table and then select `List Models` from the Tools menu. You will see that all of the models in the selection list which fit successfully are there, including the new model you added to the selection list.

%FORECAST Macro

This example demonstrates the use of the %FORECAST macro to start the Time Series Forecasting System from a SAS program submitted from the Editor window. The SQL procedure is used to create a view of a subset of a products data set. Then the %FORECAST macro is used to produce forecasts.

```
proc sql;
   create view selprod as
   select * from products
   where type eq 'A'
   order by date;
run;

%forecast(data=selprod, var=amount, id=date, interval=day,
     entry=forecast, out=typea, project=proda, refit= );
```

Chapter 60
Window Reference

Contents

Overview **3780**
Adjustments Selection Window **3780**
AR/MA Polynomial Specification Window **3781**
ARIMA Model Specification Window **3783**
ARIMA Process Specification Window **3785**
Automatic Model Fitting Window **3787**
Automatic Model Fitting Results Window **3791**
Table Contents 3791
Automatic Model Selection Options Window **3794**
Custom Model Specification Window **3795**
Data Set Selection Window **3798**
Default Time Ranges Window **3800**
Develop Models Window **3801**
Differencing Specification Window **3808**
Dynamic Regression Specification Window **3809**
Dynamic Regressors Selection Window **3810**
Error Model Options Window **3811**
External Forecast Model Specification Window **3812**
Factored ARIMA Model Specification Window **3813**
Forecast Combination Model Specification Window **3815**
Forecasting Project File Selection Window **3817**
Forecast Options Window **3818**
Intervention Specification Window **3819**
Interventions for Series Window **3820**
Manage Forecasting Project Window **3822**
Model Fit Comparison Window **3828**
Model List Window **3829**
Model Selection Criterion Window **3833**
Model Selection List Editor Window **3834**
Model Viewer Window **3838**
Models to Fit Window **3843**
Polynomial Specification Window **3844**
Produce Forecasts Window **3845**
Regressors Selection Window **3850**
Save Data As **3851**
Save Graph As **3852**

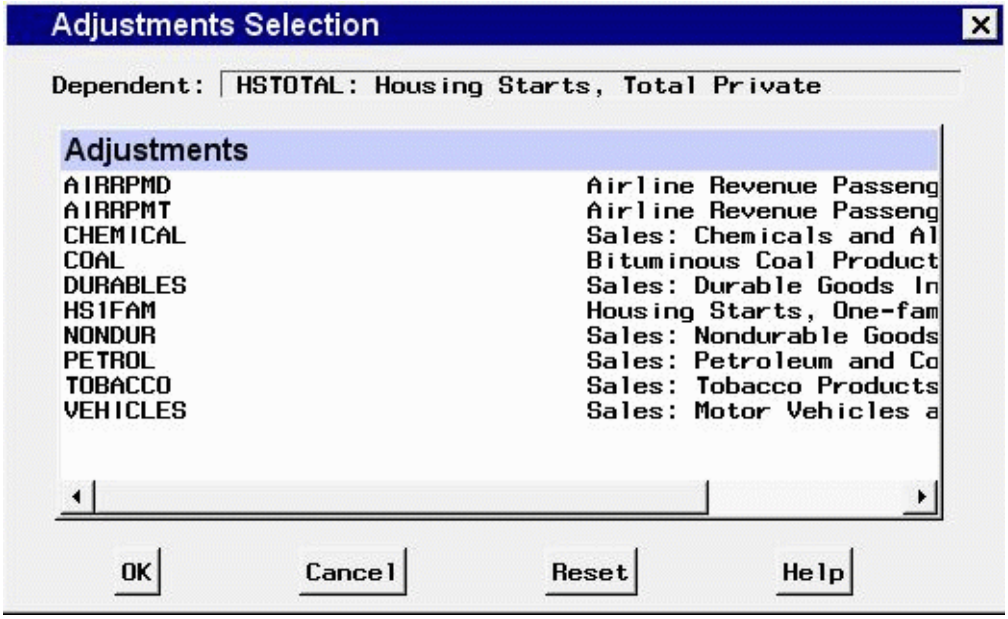
Seasonal ARIMA Model Options Window 3853
Series Diagnostics Window 3854
Series Selection Window 3855
Series to Process Window 3858
Series Viewer Transformations Window 3859
Smoothing Model Specification Window 3861
Smoothing Weight Optimization Window 3863
Statistics of Fit Selection Window 3864
Time ID Creation – 1,2,3 Window 3865
Time ID Creation from Several Variables Window 3866
Time ID Creation from Starting Date Window 3867
Time ID Creation Using Informat Window 3868
Time ID Variable Specification Window 3869
Time Ranges Specification Window 3870
Time Series Forecasting Window 3872
Time Series Simulation Window 3874
Time Series Viewer Window 3876

Overview

This chapter provides a reference to the various windows of the Time Series Forecasting System. The windows are presented in alphabetical order by name. Each section describes the purpose of the window, how to open it, its controls, fields, and menus. For windows that have their own menus, there is a description of each menu item under the heading "Menu Bar." These windows also have a toolbar with icons that duplicate the more commonly used menu items. Each icon has a *screen tip*: a brief description that appears when you hover the mouse cursor over the icon. If you don't see the screen tips, open the SAS Preferences window, under the Options submenu of the Tools menu. Select the View tab and make sure the "Screen tips" check box is checked.

Adjustments Selection Window

Use the Adjustments Selection window to select input variables for use as adjustments to the forecasts and add them to the Predictors list. Invoke this window from the pop-up menu that appears when you select the Add button of the ARIMA Model Specification window or Custom Model Specification window. For more information, see the "Adjustments" section in Chapter 58, "Using Predictor Variables."



Controls and Fields

`Dependent`
: is the name and variable label of the current series.

`Adjustments`
: is a table that lists the names and labels in the input data set available for selection as adjustments. The variables you select are highlighted. Selecting a highlighted row again deselects that variable.

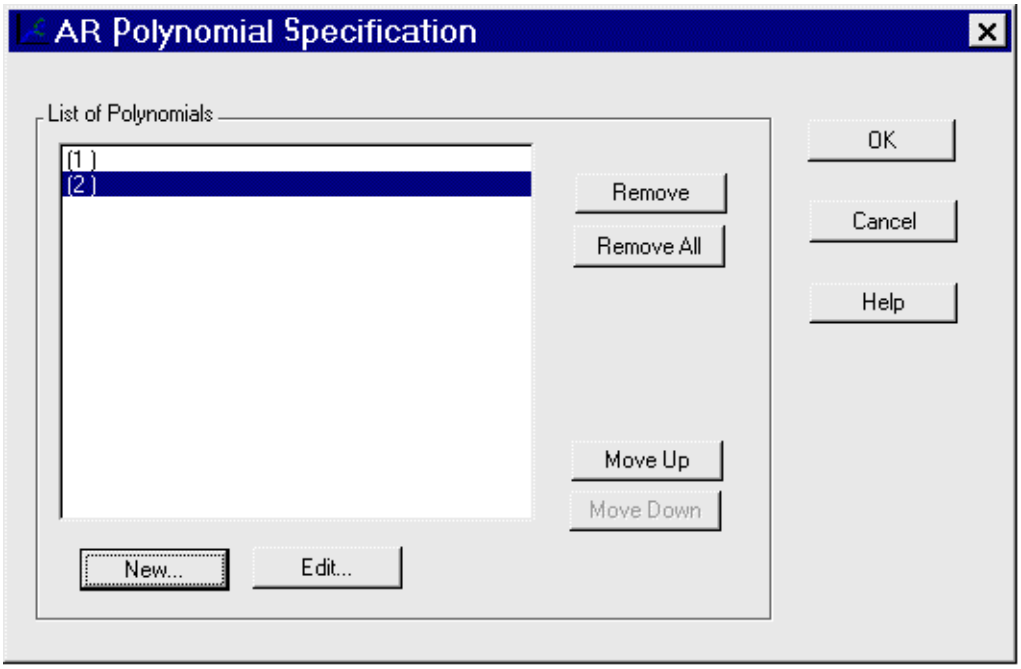
`OK`
: closes the Adjustments Selection window and adds the selected variables as adjustments in the model.

`Cancel`
: closes the window without adding any adjustments.

`Reset`
: resets all selections to their initial values upon entry to the window.

AR/MA Polynomial Specification Window

Use these windows to specify the autoregressive and moving-average terms in a factored ARIMA model. Access the AR Polynomial Specification window from the Set button next to the Autoregressive term in the Factored ARIMA Model Specification window. Access the MA Polynomial Specification window from the Set button next to the Moving Average term.



Controls and Fields

`List of Polynomials`

Lists the polynomials that have been specified. Each polynomial is represented by a comma-delimited list of lag values enclosed in parentheses.

`New`
Opens the Polynomial Specification window to add a new polynomial to the model.

`Edit`
Opens the Polynomial Specification window to edit a polynomial that has been selected. If no polynomial is selected, this button is unavailable.

`Remove`
Removes a selected polynomial from the list. If none are selected, this button is unavailable.

`Remove All`
Clears the list of polynomials.

`Move Up`
Moves a selected polynomial up one position in the list. If no polynomial is selected, or the first one is selected, this button is unavailable.

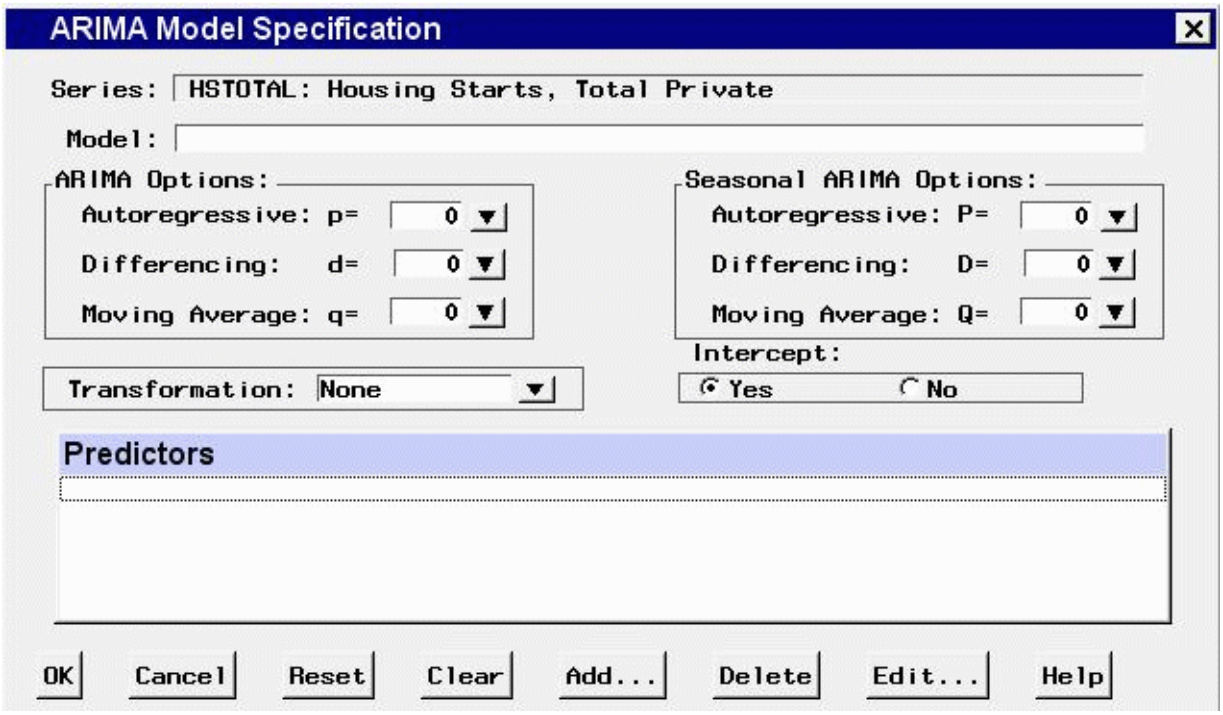
`Move Down`
Moves a selected polynomial down one position in the list. If no polynomial is selected, or the last one is selected, this button is unavailable.

`OK`
Closes the window and returns the specified list of polynomials to the Factored ARIMA Model Specification window.

`Cancel`
Closes the window and discards any changes made to the list of polynomials.

ARIMA Model Specification Window

Use the ARIMA Model Specification window to specify and fit an ARIMA model with or without predictor effects as inputs. Access it from the Develop Models menu, where it is invoked from the Fit Model item under Edit in the menu bar, or from the pop-up menu when you click an empty area of the model table.



Controls and Fields

`Series`
is the name and variable label of the current series.

`Model`
is a descriptive label for the model that you specify. You can type a label in this field or allow the system to provide a label. If you leave the label blank, a label is generated automatically based on the options you specify.

`ARIMA Options`
specify the orders of the ARIMA model. You can either type in a value or click the arrow to select from a list.

`Autoregressive`
defines the order of the autoregressive part of the model.

Differencing
: defines the order of simple differencing—for example, first difference or second difference.

Moving Average
: defines the order of the moving-average part of the model.

Seasonal ARIMA Options
: specify the orders of the seasonal part of the ARIMA model. You can either type in a value or click the arrow to select from a list.

  Autoregressive
  : defines the order of the seasonal autoregressive part of the model.

  Differencing
  : defines the order of seasonal differencing—for example, first difference or second difference at the seasonal lags.

  Moving Average
  : defines the order of the seasonal moving-average part of the model.

Transformation
: defines the series transformation for the model. When a transformation is specified, the ARIMA model is fit to the transformed series, and forecasts are produced by applying the inverse transformation to the ARIMA model predictions. The available transformations are: Log, Logistic, Square Root, Box-Cox, and None.

Intercept
: specify whether a mean or intercept parameter is included in the ARIMA model. By default, the Intercept option is set to No when the model includes differencing and Yes when there is no differencing.

Predictors
: lists the predictor effects included as inputs in the model.

OK
: closes the ARIMA Model Specification window and fits the model.

Cancel
: closes the ARIMA Model Specification window without fitting the model. Any options you specified are lost.

Reset
: resets all options to their initial values upon entry to the ARIMA Model Specification window. This might be useful when editing an existing model specification; otherwise, Reset has the same function as Clear.

Clear
: resets all options to their default values.

Add
: opens a menu of types of predictors to add to the Predictors list.

Delete
: deletes the selected (highlighted) entry from the Predictors list.

Edit
: edits the selected (highlighted) entry in the Predictors list.

Mouse Button Actions

You can select or deselect entries in the Predictors list by clicking them. The selected (highlighted) predictor effect is acted on by the Delete and Edit buttons. Double-clicking on a predictor in the list invokes an appropriate edit action for that predictor.

If you right-click an entry in the Predictors list, the system displays the following menu of actions that encompass the features of the Add, Delete, and Edit buttons.

`Add Linear Trend`
: adds a Linear Trend item to the Predictors list.

`Add Trend Curve`
: opens a menu of different time trend curves and adds the curve you select to the Predictors list. Certain trend curve specifications also set the Transformation field.

`Add Regressors`
: opens the Regressors Selection window to enable you to select other series in the input data set as regressors to predict the dependent series and add them to the Predictors list.

`Add Adjustments`
: opens the Adjustments Selection window to enable you to select other series in the input data set for use as adjustments to the forecasts and add them to the Predictors list.

`Add Dynamic Regressor`
: opens the Dynamic Regressor Selection window to enable you to select a series in the input data set as a predictor of the dependent series and also specify a transfer function model for the effect of the predictor series.

`Add Interventions`
: opens the Interventions for Series window to enable you to define and select intervention effects and add them to the Predictors list.

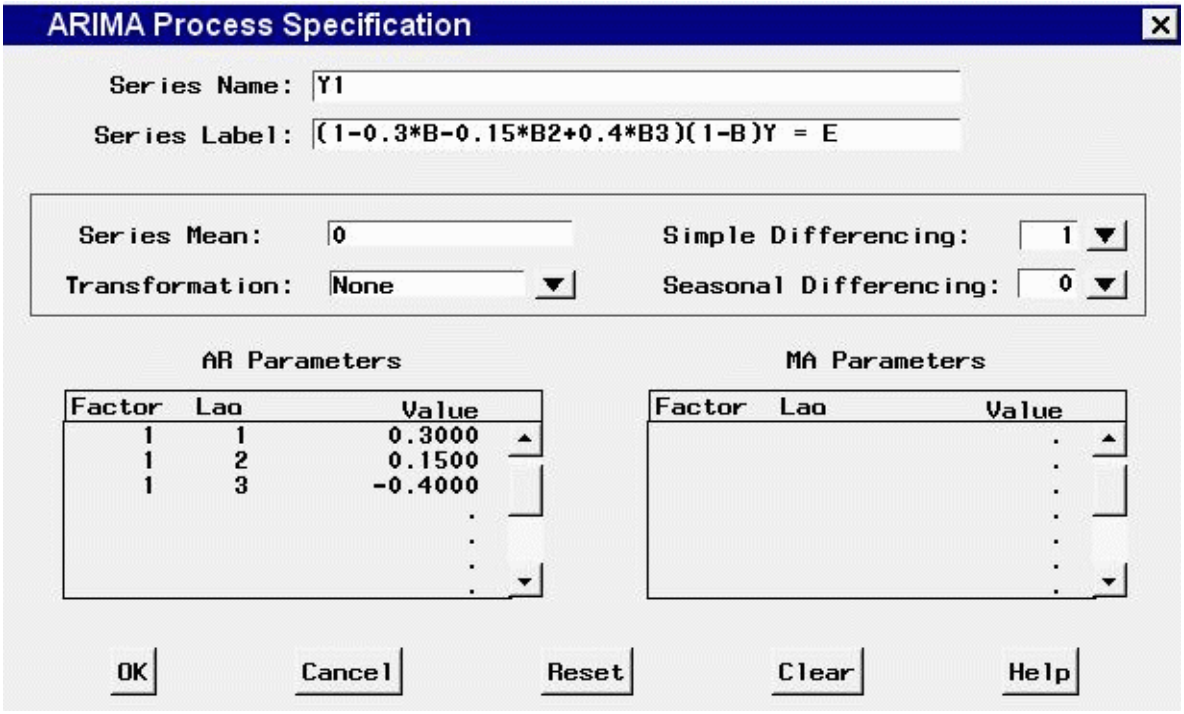
`Add Seasonal Dummies`
: adds a Seasonal Dummies predictor item to the Predictors list.

`Edit Predictor`
: edits the selected (highlighted) entry in the Predictors list.

`Delete Predictors`
: deletes the selected (highlighted) entry from the Predictors list.

ARIMA Process Specification Window

Use the ARIMA Process Specification window to define ARIMA processes for simulation. Invoke this window from the Add Series button in the Time Series Simulation window.



Controls and Fields

`Series Name`
: is the variable name for the series to be simulated.

`Series Label`
: is the variable label for the series to be simulated.

`Series Mean`
: is the mean of the simulated series.

`Transformation`
: defines the series transformation.

`Simple Differencing`
: is the order of simple differencing for the series.

`Seasonal Differencing`
: is the order of seasonal differencing for the series.

`AR Parameters`
: is a table of autoregressive terms for the simulated ARIMA process. Enter a value for Factor, Lag, and Value for each term of the AR part of the process you want to simulate. For a non-factored AR model, make the Factor values the same for all terms. For a factored AR model, use different Factor values to group the terms into the factors.

`MA Parameters`
: is a table of moving-average terms for the simulated ARIMA process. Enter a value for Factor, Lag, and Value for each term of the MA part of the process you want to simulate. For a non-factored MA model, make the Factor values the same for all terms. For a factored MA model, use different Factor values to group the terms into the factors.

`OK`
closes the ARIMA Process Specification window and adds the specified process to the Series to Generate list in the Time Series Simulation window.

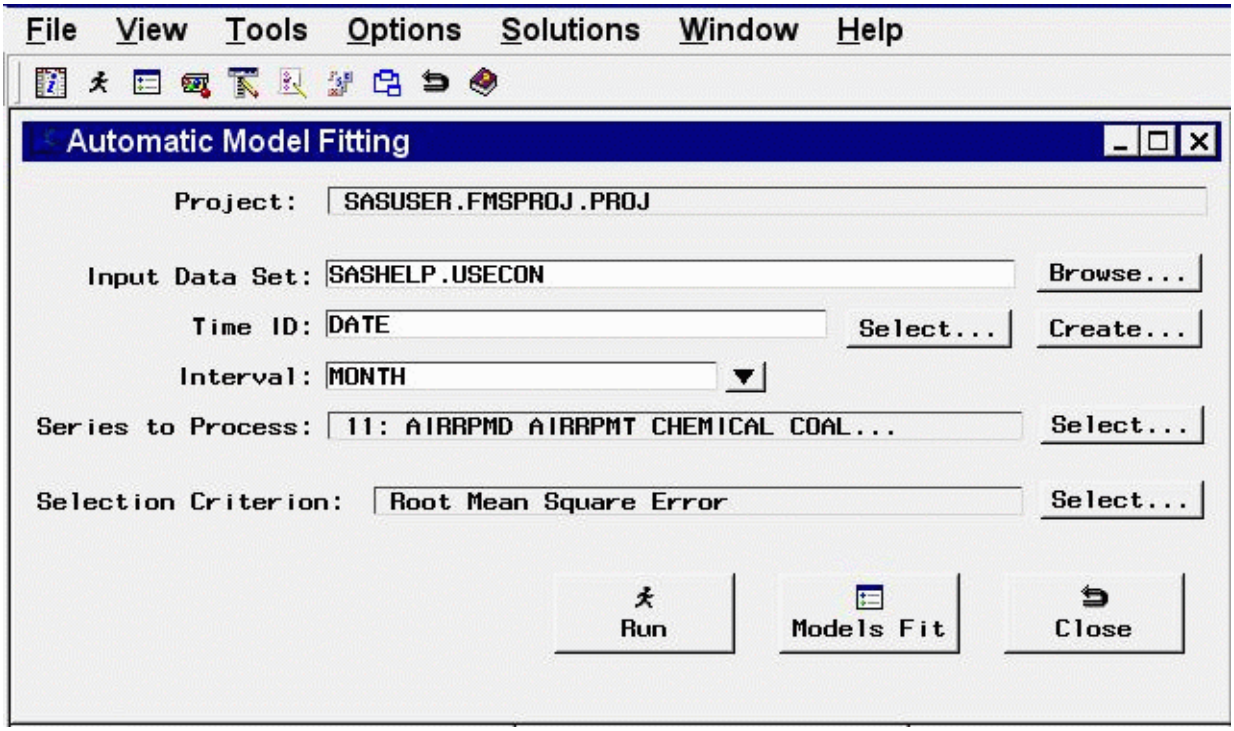
`Cancel`
closes the window without adding to the Series to Generate list. Any options you specified are lost.

`Reset`
resets all the fields to their initial values upon entry to the window.

`Clear`
resets all the fields to their default values.

Automatic Model Fitting Window

Use the Automatic Model Fitting window to perform automatic model selection on all series or selected series in an input data set. Invoke this window by using the Fit Models Automatically button on the Time Series Forecasting window. Note that you can also perform automatic model fitting, one series at a time, from the Develop Models window.



Controls and Fields

`Project`
the name of the SAS catalog entry in which the results of the model search process are stored.

`Input Data Set`
is the name of the current input data set. You can type in a one-level or two-level data set name here.

`Browse` button
opens the Data Set Selection window for selecting an input data set.

`Time ID`
is the name of the ID variable for the input data set. You can type in the variable name here or use the Select or Create button.

time ID `Select` button
opens the Time ID Variable Specification window.

time ID `Create` button
opens a menu of choices of methods for creating a time ID variable for the input data set. Use this feature if the input data set does not already contain a valid time ID variable.

`Interval`
is the time interval between observations (data frequency) in the current input data set. You can type in an interval name or select one by using the combo box pop-up menu.

`Series to Process`
indicates the number and names of time series variables for which forecasting model selection will be applied.

Series to Process `Select` button
opens the Series to Process window to let you select the series for which you want to fit models.

`Selection Criterion`
shows the goodness-of-fit statistic that will be used to determine the best fitting model for each series.

Selection Criterion `Select` button
opens the Model Selection Criterion window to enable you to select the goodness-of-fit statistic that will be used to determine the best fitting model for each series.

`Run` button
begins the automatic model fitting process.

`Models Fit` button
opens the Automatic Model Fitting Results window to display the models fit during the current invocation of the Automatic Model Fitting window. The results appear automatically when model fitting is complete, but this button enables you to redisplay the results window.

`Close` button
Closes the Automatic Model Fitting window.

Menu Bar

`File`

`Import Data`
is available if you license SAS/Access software. It opens an Import Wizard, which you can use to import your data from an external spreadsheet or data base to a SAS data set for use in the Time Series Forecasting System.

`Export Data`
: is available if you license SAS/Access software. It opens an Export Wizard, which you can use to export a SAS data set, such as a forecast data set created with the Time Series Forecasting System, to an external spreadsheet or data base.

`Print Setup`
: opens the Print Setup window, which allows you to access your operating system print setup.

`Close`
: closes the Automatic Model Fitting window.

`View`

`Input Data Set`
: opens a Viewtable window to browse the current input data set.

`Models Fit`
: opens Automatic Model Fitting Results window to show the forecasting models fit during the current invocation of the Automatic Model Fitting window. This is the same as the Models Fit button.

`Tools`

`Fit Models`
: performs the automatic model selection process for the selected series. This is the same as the Run button.

`Options`

`Default Time Ranges`
: opens the Default Time Ranges window to enable you to control how the system sets the time ranges for series.

`Model Selection List`
: opens the Model Selection List editor window. Use this action to control the forecasting models considered by the automatic model selection process and displayed in the Models to Fit window.

`Model Selection Criterion`
: opens the Model Selection Criterion window, which presents a list of goodness-of-fit statistics and enables you to select the fit statistic that is displayed in the table and used by the automatic model selection process to determine the best fitting model. This action is the same as the Selection Criterion Select button.

`Statistics of Fit`
: opens the Statistics of Fit Selection window, which presents a list of statistics that the system can display. Use this action to customize the list of statistics shown in the Statistics of Fit table and available for selection in the Model Selection Criterion menu.

`Forecast Options`
: opens the Forecast Options window, which enables you to control the widths of forecast confidence limits and control the kind of predicted values computed for models that include series transformations.

`Forecast Data Set`
see Produce Forecasts window.

`Alignment of Dates`

`Beginning`
aligns dates that the system generates to identify forecast observations in output data sets to the beginning of the time intervals.

`Middle`
aligns dates that the system generates to identify forecast observations in output data sets to the midpoints of the time intervals.

`End`
aligns dates that the system generates to identify forecast observations in output data sets to the end of the time intervals.

`Automatic Fit`
opens the Automatic Model Selection Options window, which enables you to control the number of models retained by the automatic model selection process and whether the models considered for automatic selection are subset according to the series diagnostics.

`Tool Bar Type`

`Image Only`
displays the toolbar items as icons without text.

`Label Only`
displays the toolbar items as text without icon images.

`Both`
displays the toolbar items with both text and icon images.

`Include Interventions`
controls whether intervention effects defined for the current series are automatically added as predictors to the models considered by the automatic selection process. A check mark or filled check box next to this item indicates that the option is turned on.

`Print Audit Trail`
prints to the SAS log information about the models fit by the system. A check mark or filled check box next to this item indicates that the audit option is turned on.

`Show Source Statements`
controls whether SAS statements submitted by the forecasting system are printed in the SAS log. When the Show Source Statements option is selected, the system sets the SAS system option SOURCE before submitting SAS statements; otherwise, the system uses the NOSOURCE option. Note that only some of the functions performed by the forecasting system are accomplished by submitting SAS statements. A check mark or filled check box next to this item indicates that the option is turned on.

Automatic Model Fitting Results Window

This resizable window displays the models fit by the most recent invocation of the Automatic Model Fitting window. It appears automatically after Automatic Model Fitting runs, and can be opened repeatedly from that window by using the Models Fit button or by selecting Models Fit from the View menu. Once you exit the Automatic Model Fitting window, the Automatic Model Fitting Results window cannot be opened again until you fit additional models by using Automatic Model Fitting.

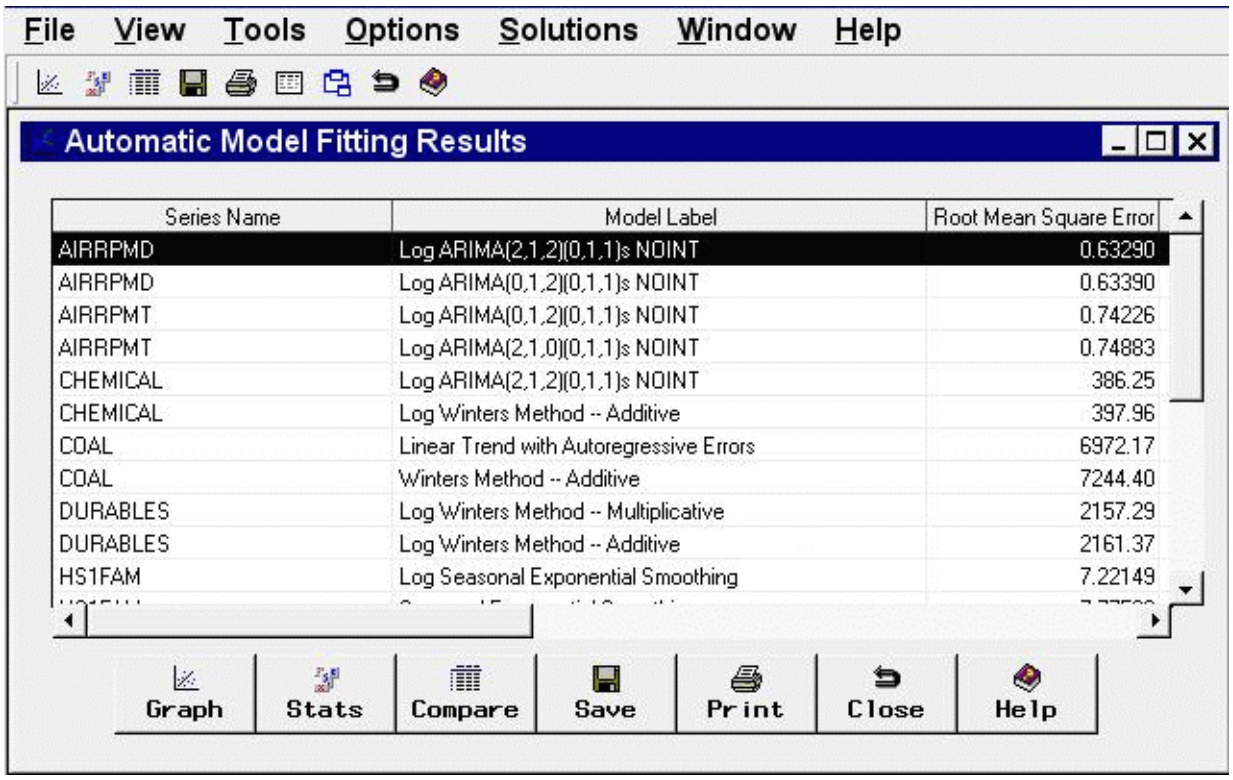


Table Contents

The results table displays the series name in the first column and the model label in the second column. If you have chosen to retain more than one model by using the Automatic Model Selection Options window, more than one row appears in the table for each series; that is, there is a row for each model fit. If you have already fit models to the same series before invoking the Automatic Model Fitting window, those models do not appear here, since the Automatic Model Fitting Results window is intended to show the results of the current operation of Automatic Model Fitting. To see all models that have been fit, use the Manage Projects window.

The third column of the table shows the values of the current model selection criterion statistic. Additional columns show the values of other fit statistics. The set of statistics shown are selectable by using the Statistics of Fit Selection window.

The table can be sorted by any column other than Series Name by clicking on the column heading.

Controls and Fields

`Graph`
opens the Model Viewer window on the model currently selected in the table.

`Stats`
opens the Statistics of Fit Selection window. This controls the set of goodness-of-fit statistics displayed in the table and in other parts of the Time Series Forecasting System.

`Compare`
opens the Model Fit Comparison window for the series currently selected in the table. This button is unavailable if the currently selected row in the table represents a series for which fewer than two models have been fit.

`Save`
opens an output data set dialog, enabling you to specify a SAS data set to which the contents of the table is saved. Note that this operation saves what you see in the table. If you want to save the models themselves for use in a future session, use the Manage Projects window.

`Print`
prints the contents of the table.

`Close`
closes the window and returns to the Automatic Model Fitting window.

Menu Bar

`File`

`Save`
opens an output data set dialog, enabling you to specify a SAS data set to which the contents of the table is saved. This is the same as the Save button.

`Print`
prints the contents of the table. This is the same as the Print button.

`Import Data`
is available if you license SAS/Access software. It opens an Import Wizard, which you can use to import your data from an external spreadsheet or data base to a SAS data set for use in the Time Series Forecasting System.

`Export Data`
is available if you license SAS/Access software. It opens an Export Wizard, which you can use to export a SAS data set, such as a forecast data set created with the Time Series Forecasting System, to an external spreadsheet or data base.

`Print Setup`
opens the Print Setup window, which allows you to access your operating system print setup.

`Close`
closes the window and returns to the Automatic Model Fitting window.

`View`

`Model Predictions`
opens the Model Viewer to display a predicted and actual plot for the currently highlighted model.

`Prediction Errors`
opens the Model Viewer to display the prediction errors for the currently highlighted model.

`Prediction Error Autocorrelations`
opens the Model Viewer to display the prediction error autocorrelations, partial autocorrelations, and inverse autocorrelations for the currently highlighted model.

`Prediction Error Tests`
opens the Model Viewer to display graphs of white noise and stationarity tests on the prediction errors of the currently highlighted model.

`Parameter Estimates`
opens the Model Viewer to display the parameter estimates table for the currently highlighted model.

`Statistics of Fit`
opens the Model Viewer window to display goodness-of-fit statistics for the currently highlighted model.

`Forecast Graph`
opens the Model Viewer to graph the forecasts for the currently highlighted model.

`Forecast Table`
opens the Model Viewer to display forecasts for the currently highlighted model in a table.

`Tools`

`Compare Models`
opens the Model Fit Comparison window to display fit statistics for selected pairs of forecasting models. This item is unavailable until you select a series in the table for which the automatic model fitting run selected two or more models.

`Options`

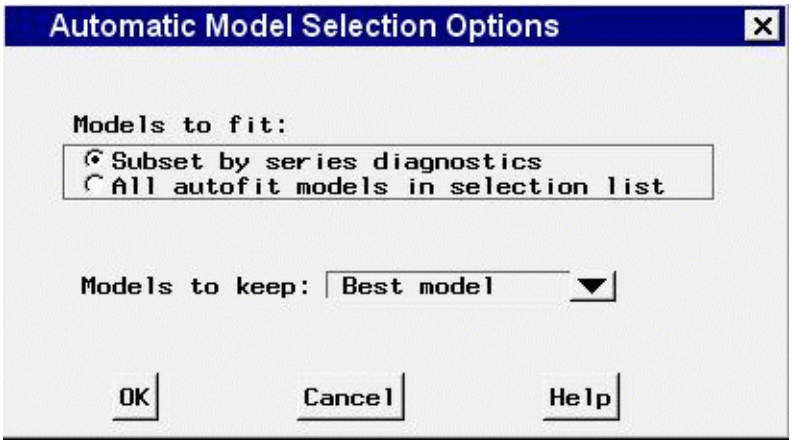
`Statistics of Fit`
opens the Statistics of Fit Selection window. This is the same as the Stats button.

`Column Labels`
selects long or short column labels for the table. Long column labels are used by default.

`ID Columns`
freezes or unfreezes the series and model columns. By default they are frozen so that they remain visible when you scroll the table horizontally to view other columns.

Automatic Model Selection Options Window

Use the Automatic Model Selection Options window to control the automatic selection process. This window is available from the Automatic Fit item of the Options menu in the Develop Models window, Automatic Model Fitting window, and Produce Forecasts window.



Controls and Fields

`Models to fit`

`Subset by series diagnostics`
when selected, causes the automatic model selection process to search only over those models consistent with the series diagnostics.

`All models in selection list`
when selected, causes the automatic model selection process to search over all models in the search list, without regard for the series diagnostics.

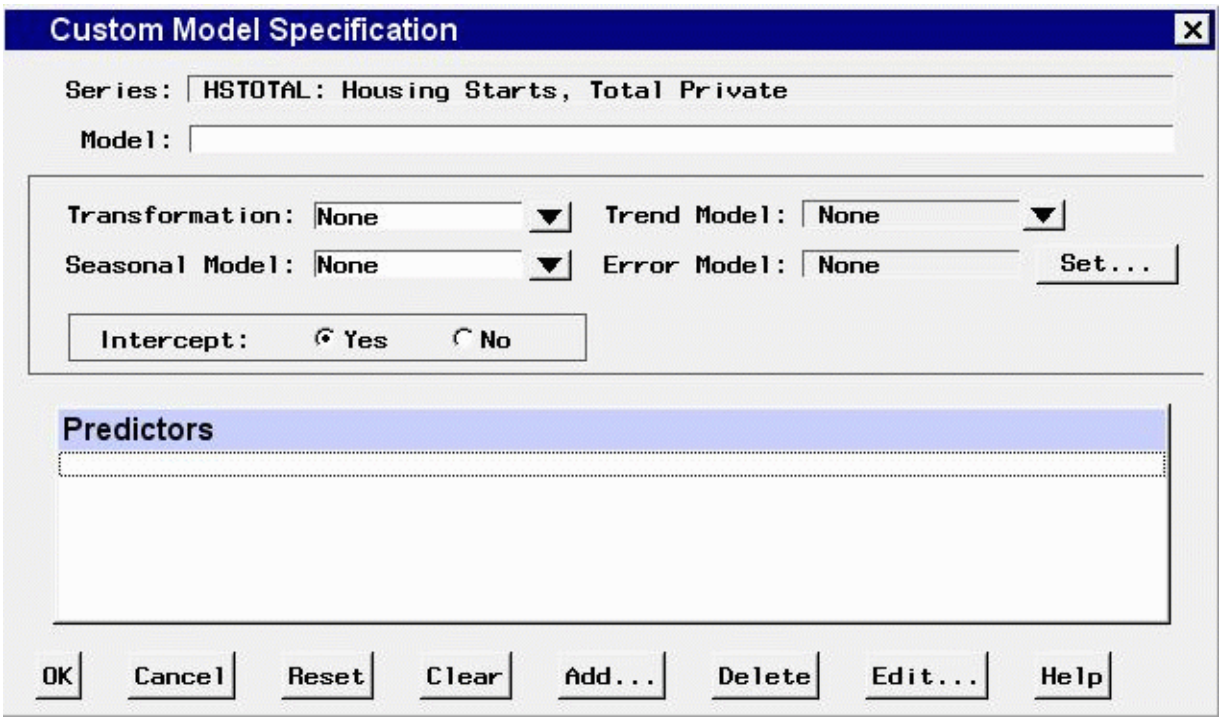
`Models to keep`
specifies how many of the models tried by the automatic model selection process are retained and added to the model list for the series. You can specify the best fitting model only, the best *n* models, where *n* can be 1 through 9, or all models tried.

`OK`
closes the window and saves the automatic model selection options you specified.

`Cancel`
closes the window without changing the automatic model selection options.

Custom Model Specification Window

Use the Custom Model Specification window to specify and fit an ARIMA model with or without predictor effects as inputs. Access it from the Develop Models window, where it is invoked from the Fit Model item under the Edit menu, or from the pop-up menu when you click an empty area of the model table.



Controls and Fields

`Series`
: is the name and variable label of the current series.

`Model`
: is a descriptive label for the model that you specify. You can type a label in this field or allow the system to provide a label. If you leave the label blank, a label is generated automatically based on the options you specify.

`Transformation`
: defines the series transformation for the model. When a transformation is specified, the model is fit to the transformed series, and forecasts are produced by applying the inverse transformation to the resulting forecasts. The following transformations are available:

  `Log`
  : specifies a logarithmic transformation.

  `Logistic`
  : specifies a logistic transformation.

`Square Root`
specifies a square root transformation.

`Box-Cox`
specifies a Box-Cox transform and opens a window to specify the Box-Cox λ parameter.

`None`
specifies no series transformation.

`Trend Model`
controls the model options to model and forecast the series trend. Select from the following:

`Linear Trend`
adds a Linear Trend item to the Predictors list.

`Trend Curve`
brings of a menu of different time trend curves and adds the curve you select to the Predictors list.

`First Difference`
specifies differencing the series.

`Second Difference`
specifies second-order differencing of the series.

`None`
specifies no model for the series trend.

`Seasonal Model`
controls the model options to model and forecast the series seasonality. Select from the following:

`Seasonal ARIMA`
opens the Seasonal ARIMA Model Options window to enable you to specify an ARIMA model for the seasonal pattern in the series.

`Seasonal Difference`
specifies differencing the series at the seasonal lag.

`Seasonal Dummy Regressors`
adds a Seasonal Dummies predictor item to the Predictors list.

`None`
specifies no seasonal model.

`Error Model`
displays the current settings of the autoregressive and moving-average terms, if any, for modeling the prediction error autocorrelation pattern in the series.

`Set` button
opens the Error Model Options window to enable you to set the autoregressive and moving-average terms for modeling the prediction error autocorrelation pattern in the series.

`Intercept`
specifies whether a mean or intercept parameter is included in the model. By default, the Intercept option is set to No when the model includes differencing and set to Yes when there is no differencing.

`Predictors`
is a list of the predictor effects included as inputs in the model.

`OK`
closes the Custom Model Specification window and fits the model.

`Cancel`
closes the Custom Model Specification window without fitting the model. Any options you specified are lost.

`Reset`
resets all options to their initial values upon entry to the Custom Model Specification window. This might be useful when editing an existing model specification; otherwise, Reset has the same function as Clear.

`Clear`
resets all options to their default values.

`Add`
opens a menu of types of predictors to add to the Predictors list. Select from the following:

`Linear Trend`
adds a Linear Trend item to the Predictors list.

`Trend Curve`
opens a menu of different time trend curves and adds the curve you select to the Predictors list.

`Regressors`
opens the Regressors Selection window to enable you to select other series in the input data set as regressors to predict the dependent series and add them to the Predictors list.

`Adjustments`
opens the Adjustments Selection window to enable you to select other series in the input data set for use as adjustments to the forecasts and add them to the Predictors list.

`Dynamic Regressor`
opens the Dynamic Regressor Selection window to enable you to select a series in the input data set as a predictor of the dependent series and also specify a transfer function model for the effect of the predictor series.

`Interventions`
opens the Interventions for Series window to enable you to define and select intervention effects and add them to the Predictors list.

`Seasonal Dummies`
adds a Seasonal Dummies predictor item to the Predictors list. This is unavailable if the series interval is not one which has a seasonal cycle.

`Delete`
deletes the selected (highlighted) entry from the Predictors list.

`Edit`
edits the selected (highlighted) entry in the Predictors list.

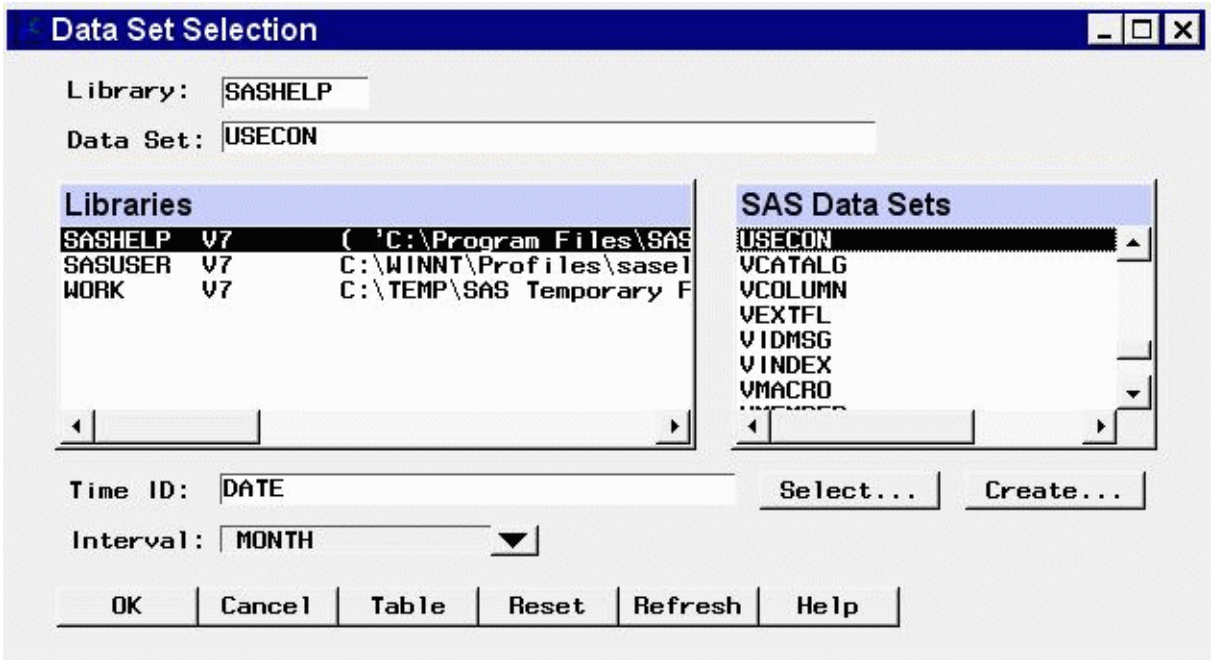
Mouse Button Actions

You can select or deselect entries in the Predictors list by clicking them. The selected (highlighted) predictor effect is acted on by the Delete and Edit buttons. Double-clicking on a predictor in the list invokes an appropriate edit action for that predictor.

If you right-click an entry in the Predictors list and press the right mouse button, the system displays a menu of actions that encompass the features of the Add, Delete, and Edit buttons.

Data Set Selection Window

Use this resizable window to select a data set to process by specifying a library and a SAS data set or view. These selections can be made by typing, by selecting from lists, or by a combination of the two. In addition, you can control the time ID variable and time interval, and you can browse the data set.



Access this window by using the Browse button to the right of the Data Set field in the Time Series Forecasting, Automatic Model Fitting, and Produce Forecasts windows. It functions in the same way as the Series Selection window, except that it does not allow you to select or view a time series variable.

Controls and Fields

`Library`
: is a SAS libname assigned within the current SAS session. If you know the libname associated with the data set of interest, you can type it in this field. If it is a valid choice, it will appear in the libraries list and will be highlighted. The SAS Data Sets list will be populated with data sets associated with that libname. See also Libraries under Selection Lists.

`Data Set`
: is the name of a SAS data set (data file or data view) that resides under the selected libname. If you know the name, you can type it in and press Return. If it is a valid choice, it will appear in the SAS Data Sets list and will be highlighted.

`Time ID`
: is the name of the ID variable for the selected input data set. To specify the ID variable, you can type the ID variable name in this field or select the control arrows to the right of the field.

Time ID `Select` button
: opens the Time ID Variable Specification window.

Time ID `Create` button
: opens a menu of methods for creating a time ID variable for the input data set. Use this feature if the data set does not already contain a valid time ID variable.

`Interval`
: is the time interval between observations (data frequency) in the selected data set. If the interval is not automatically identified by the system, you can type in the interval name or select it from a list by clicking the combo box arrow. For more information about intervals, see Chapter 5, "Date Intervals, Formats, and Functions," in this book.

`OK`
: closes the Data Set Selection window and makes the selected data set the current input data set.

`Cancel`
: closes the window without applying any selections made.

`Table`
: opens a Viewtable window for browsing the selected data set.

`Reset`
: resets the fields to their initial values upon entry to the window.

`Refresh`
: updates all fields and lists in the window. If you assign a new libname without exiting the Data Set Selection window, use the refresh action to update the Libraries list so that it will include the newly assigned libname.

Selection Lists

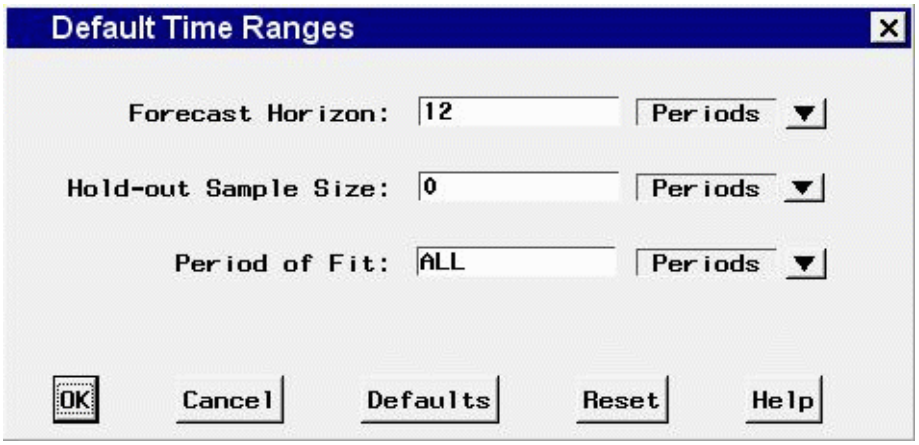
`Libraries`
: displays a list of currently assigned libnames. You can select a libname by clicking it with the left mouse button, which is equivalent to typing its name in the Library field.

  If you cannot locate the library or directory you are interested in, go to the SAS Explorer window, select "New" from the File menu, then select "Library" and "OK." This opens the New Library window. You also assign a libname by submitting a libname statement from the Editor window. Select the Refresh button to make the new libname available in the libraries list.

`SAS Data Sets`
: displays a list of the SAS data sets (data files or data views) contained in the selected library. You can select one of these by clicking with the left mouse button, which is equivalent to typing its name in the Data set field. You can double-click a data set name to select it and exit the window.

Default Time Ranges Window

Use the Default Time Ranges window to control how the period of fit and evaluation and the forecasting horizon are determined for each series when you do not explicitly set these ranges for a particular series. Invoke this window from the Options menu of the Develop Models, Automatic Model Fitting, Produce Forecasts, and Manage Forecasting Project windows. The settings you make in this window affect subsequently selected series; they do not alter the time ranges of series you have already selected.



Controls and Fields

`Forecast Horizon`
specifies the forecast horizon as either a number of periods or years from the last nonmissing data value or as a fixed date. You can type a number or date value in this field. Date value must be entered in a form recognized by a SAS date informat. (See *SAS Language Reference: Concepts* for information about SAS date informats.)

`Forecast Horizon Units`
indicates whether the value in the forecast horizon field represents periods or years or a date. Click the arrow and select one from the pop-up list.

`Hold-out Sample Size`
specifies that a number of observations, number of years, or percent of the data at the end of the data range be used for the period of evaluation with the remainder of data used as the period of fit.

`Hold-out Sample Size Units`
indicates whether the hold-out sample size represents periods or years or percent of data range.

`Period of Fit`
specifies how much of the data range for a series is to be used as the period of fit for models fit to the series. ALL indicates that all the available data is used. You can specify a number of periods, number of years, or a fixed date, depending on the value of the units field to the right. When you specify a date, the start of the period of fit is the specified date or the first nonmissing series value, whichever is more recent. Date value must be entered in a form recognized by a SAS date informat. (See *SAS Language Reference: Concepts* for information about SAS date informats.) When you specify the number of

periods or years, the start of the period of fit is computed as the date that number of periods or years from the end of the data.

`Period of Fit Units`
indicates whether the period-of-fit value represents periods or years or a date.

`OK`
closes the window and stores the specified changes.

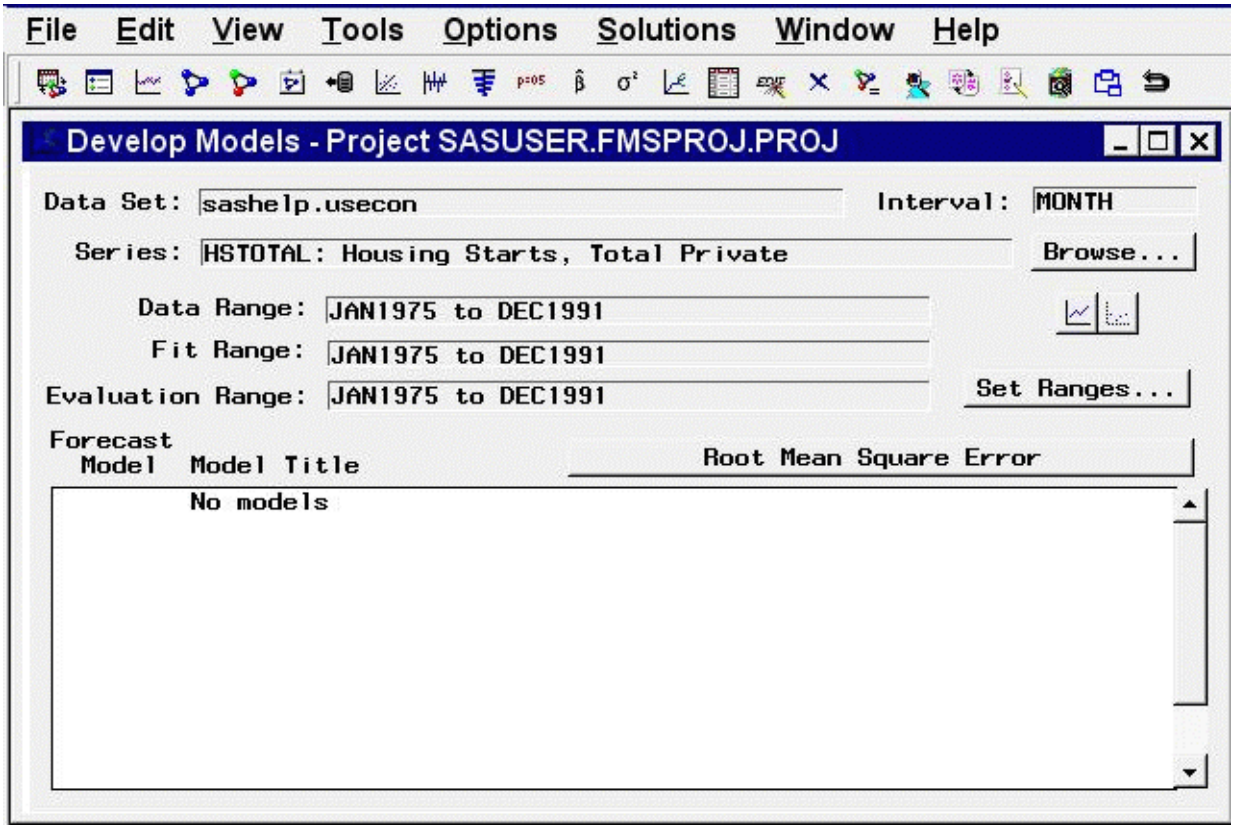
`Cancel`
closes the window without saving changes. Any options you specified are lost.

`Defaults`
resets all options to their default values.

`Reset`
resets the options to their initial values upon entry to the window.

Develop Models Window

This resizable window provides access to all of the Forecasting System's interactive model fitting and graphical tools. Use it to fit forecasting models to an individual time series and choose the best model to use to produce the final forecasts of the series. Invoke this window by using the Develop Models button on the Time Series Forecasting window.



Controls and Fields

`Data Set`
: is the name of the current input data set.

`Interval`
: is the time interval (data frequency) for the input data set.

`Series`
: is the variable name and label of the current time series.

`Browse` button
: opens the Series Selection window to enable you to change the current input data set or series.

`Data Range`
: is the date of the first and last nonmissing data values available for the current series in the input data set.

`Fit Range`
: is the current period of fit setting. This is the range of data that will be used to fit models to the series.

`Evaluation Range`
: is the current period of evaluation setting. This is the range of data that will be used to calculate the goodness-of-fit statistics for models fit to the series.

`Set Ranges` button
: opens the Time Ranges Specification window to enable you to change the fit range or evaluation range. Note: A new fit range is applied when new models are fit or when existing models are refit. A new evaluation range is applied when new models are fit or when existing models are refit or reevaluated. Changing the ranges does not automatically refit or reevaluate any models in the table: Use the Refit Models or Reevaluate Models items under the Edit menu.

`View Series Graphically` icon
: opens the Time Series Viewer window to display plots of the current series.

`View Selected Model Graphically` icon
: opens the Model Viewer to display graphs and tables for the currently highlighted model.

`Forecast Model`
: is the column of the model table that contains check boxes to select which model is used to produce the final forecasts for the current series.

`Model Title`
: is the column of the model table that contains the descriptive labels of the forecasting models fit to the current series.

`Root Mean Square Error` (or other statistic name) button
: is the button above the right side of the table. It displays the name of the current model selection criterion: a statistic that measures how well each model in the table fits the values of the current series for observations within the evaluation range. Clicking this button opens the Model Selection Criterion window to let you to select a different statistic. When you select a statistic, the model table the Develop Models window is updated to show current values of that statistic.

Menu Bar

`File`

`New Project`
opens a dialog that lets you create a new project, assign it a name and description, and make it the active project.

`Open Project`
opens a dialog that lets you select and load a previously saved project.

`Save Project`
saves the current state of the system (including all the models fit to a series) to the current project catalog entry.

`Save Project as`
saves the current state of the system with a prompt for the name of the catalog entry in which to store the information.

`Clear Project`
clears the system, deleting all the models for all series.

`Save Forecast`
writes forecasts from the currently highlighted model to an output data set.

`Save Forecast As`
prompts for an output data set name and saves the forecasts from the currently highlighted model.

`Output Forecast Data Set`
opens a dialog for specifying the default data set used when you select “Save Forecast.”

`Import Data`
is available if you license SAS/Access software. It opens an Import Wizard, which you can use to import your data from an external spreadsheet or data base to a SAS data set for use in the Time Series Forecasting System.

`Export Data`
is available if you license SAS/Access software. It opens an Export Wizard, which you can use to export a SAS data set, such as a forecast data set created with the Time Series Forecasting System, to an external spreadsheet or data base.

`Print Setup`
opens the Print Setup window, which enables you to access your operating system print setup.

`Close`
closes the Develop Models window and returns to the main window.

Edit

Fit Model

Automatic Fit
invokes the automatic model selection process.

Select From List
opens the Models to Fit window.

Smoothing Model
opens the Smoothing Model Specification window.

ARIMA Model
opens the ARIMA Model Specification window.

Custom Model
opens the Custom Model Specification window.

Combine Forecasts
opens the Forecast Combination Model Specification window.

External Forecasts
opens the External Forecast Model Specification window.

Edit Model
enables you to modify the specification of the currently highlighted model in the table and fit the modified model. The new model replaces the current model in the table.

Delete Model
deletes the currently highlighted model from the model table.

Refit Models

All Models
refits all models in the table by using data within the current fit range.

Selected Model
refits the currently highlighted model by using data within the current fit range.

Reevaluate Models

All Models
recomputes statistics of fit for all models in the table by using data within the current evaluation range.

Selected Model
recomputes statistics of fit for the currently highlighted model by using data within the current evaluation range.

View

Project
opens the Manage Forecasting Project window.

`Data Set`
: opens a Viewtable window to display the current input data set.

`Series`
: opens the Time Series Viewer window to display plots of the current series. This is the same as the View Series Graphically icon.

`Model Predictions`
: opens the Model Viewer to display a predicted versus actual plot for the currently highlighted model. This is the same as the View Selected Model Graphically icon.

`Prediction Errors`
: opens the Model Viewer to display the prediction errors for the currently highlighted model.

`Prediction Error Autocorrelations`
: opens the Model Viewer to display the prediction error autocorrelations, partial autocorrelations, and inverse autocorrelations for the currently highlighted model.

`Prediction Error Tests`
: opens the Model Viewer to display graphs of white noise and stationarity tests on the prediction errors of the currently highlighted model.

`Parameter Estimates`
: opens the Model Viewer to display the parameter estimates table for the currently highlighted model.

`Statistics of Fit`
: opens the Model Viewer window to display goodness-of-fit statistics for the currently highlighted model.

`Forecast Graph`
: opens the Model Viewer to graph the forecasts for the currently highlighted model.

`Forecast Table`
: opens the Model Viewer to display forecasts for the currently highlighted model in a table.

`Tools`

`Diagnose Series`
: opens the Series Diagnostics window to determine the kinds of forecasting models appropriate for the current series.

`Define Interventions`
: opens the Interventions for Series window to enable you to edit or add intervention effects for use in modeling the current series.

`Sort Models`
: sorts the models in the table by the values of the currently displayed fit statistic.

`Compare Models`
: opens the Model Fit Comparison window to display fit statistics for selected pairs of forecasting models. This is unavailable if there are fewer than two models in the table.

Generate Data
: opens the Time Series Simulation window. This window enables you to simulate ARIMA time series processes and is useful for educational exercises or testing the system.

Options

Time Ranges
: opens the Time Ranges Specification window to enable you to change the fit and evaluation time ranges and the forecast horizon. This action is the same as the Set Ranges button.

Default Time Ranges
: opens the Default Time Ranges window to enable you to control how the system sets the time ranges for series when you do not explicitly set time ranges with the Time Ranges Specification window. Settings made by using this window do not affect series you are already working with; they take effect when you select a new series.

Model Selection List
: opens the Model Selection List editor window. Use this action to edit the set of forecasting models considered by the automatic model selection process and displayed by the Models to Fit window.

Model Selection Criterion
: opens the Model Selection Criterion window, which presents a list of goodness-of-fit statistics and enables you to select the fit statistic that is displayed in the table and used by the automatic model selection process to determine the best fitting model. This action is the same as clicking the button above the table which displays the name of the current model selection criterion.

Statistics of Fit
: opens the Statistics of Fit Selection window, which presents a list of statistics that the system can display. Use this action to customize the list of statistics shown in the Model Viewer, Automatic Model Fitting Results, and Model Fit Comparison windows and available for selection in the Model Selection Criterion menu.

Forecast Options
: opens the Forecast Options window, which enables you to control the widths of forecast confidence limits and control the kind of predicted values computed for models that include series transformations.

Alignment of Dates

Beginning
: aligns dates that the system generates to identify forecast observations in output data sets to the beginning of the time intervals.

Middle
: aligns dates that the system generates to identify forecast observations in output data sets to the midpoints of the time intervals.

End
: aligns dates that the system generates to identify forecast observations in output data sets to the end of the time intervals.

`Automatic Fit`
: opens the Automatic Model Selection Options window, which enables you to control the number of models retained by the automatic model selection process and whether the models considered for automatic selection are subset according to the series diagnostics.

`Include Interventions`
: controls whether intervention effects defined for the current series are automatically added as predictors to the models considered by the automatic selection process and displayed by the Models to Fit window. When the Include Interventions option is selected, the series interventions are also automatically added to the predictors list when you specify a model in the ARIMA and Custom Models Specification windows. A check mark or filled check box next to this item indicates that the option is turned on.

`Print Audit Trail`
: prints to the SAS log information about the models fit by the system. A check mark or filled check box next to this item indicates that the audit option is turned on.

`Show Source Statements`
: Controls whether SAS statements submitted by the forecasting system are printed in the SAS log. When the Show Source Statements option is selected, the system sets the SAS system option SOURCE before submitting SAS statements; otherwise, the system uses the NOSOURCE option. Note that only some of the functions performed by the forecasting system are accomplished by submitting SAS statements. A check mark or filled check box next to this item indicates that the option is turned on.

Left Mouse Button Actions for the Model Table

When the cursor is over the description of a model in the table, the left mouse button selects (highlights) or deselects that model. On some computer systems, you can double-click to open the Model Viewer window for the selected model.

When the cursor is over an empty part of the model table, the left mouse button opens a menu of model fitting choices. These choices are the same as those in the Fit Model submenu of the Edit menu.

Right Mouse Button Actions for the Model Table

When a model in the table is selected, the right mouse opens a menu of actions that apply to the highlighted model. The actions available in this menu are as follows.

`View Model`
: opens the Model Viewer for the selected model. This action is the same as the View Model Graphically icon.

`View Parameter Estimates`
: opens the Model Viewer to display the parameter estimates table for the currently highlighted model. This is the same as the Parameter Estimates item in the View menu.

`View Statistics of Fit`
: opens the Model Viewer to display a table of goodness-of-fit statistics for the currently highlighted model. This is the same as the Statistics of Fit item in the View menu.

`Edit Model`
: enables you to modify the specification of the currently highlighted model in the table and fit the modified model. This is the same as the Edit Model item in the Edit menu.

`Refit Model`
: refits the highlighted model by using data within the current fit range. This is the same as the Selected Model item under the Refit Models submenu of the Edit menu.

`Reevaluate Model`
: reevaluates the highlighted model by using data within the evaluation fit range. This is the same as the Selected Model item under the Reevaluate Models submenu of the Edit menu.

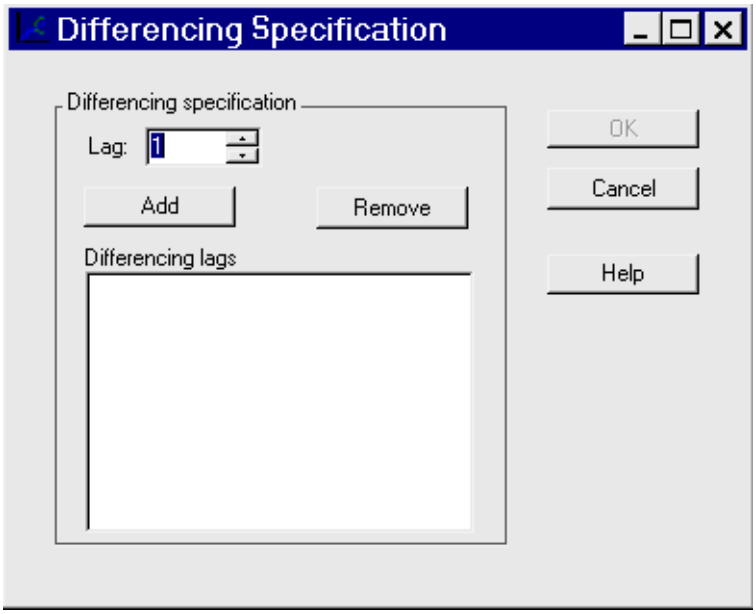
`Delete Model`
: deletes the currently highlighted model from the model table. This is the same as the Delete Model item under the Edit menu.

`View Forecasts`
: opens the Model Viewer to display the forecasts for the currently highlighted model. This is the same as the Forecast Graph item under the View menu.

When the model list is empty or when no model is selected, the right mouse button opens the same menu of model fitting actions as the left mouse button.

Differencing Specification Window

Use the Differencing Specification window to specify the list of differencing lags `d=(lag, ..., lag)` in a factored ARIMA model. To specify a first difference, add the value 1 (`d=(1)`). To specify a second difference (difference twice at lag 1), add the value 1 again (`d=(1,1)`). For first differencing at lags 1 and 12, use the values 1 and 12 (`d=(1,12)`).



Controls and Fields

`Lag`
specifies a lag value to add to the list. Type in a positive integer or select one by clicking the spin box arrows. Duplicates are allowed.

`Add`
adds the value in the `Lag` spin box to the list of differencing lags.

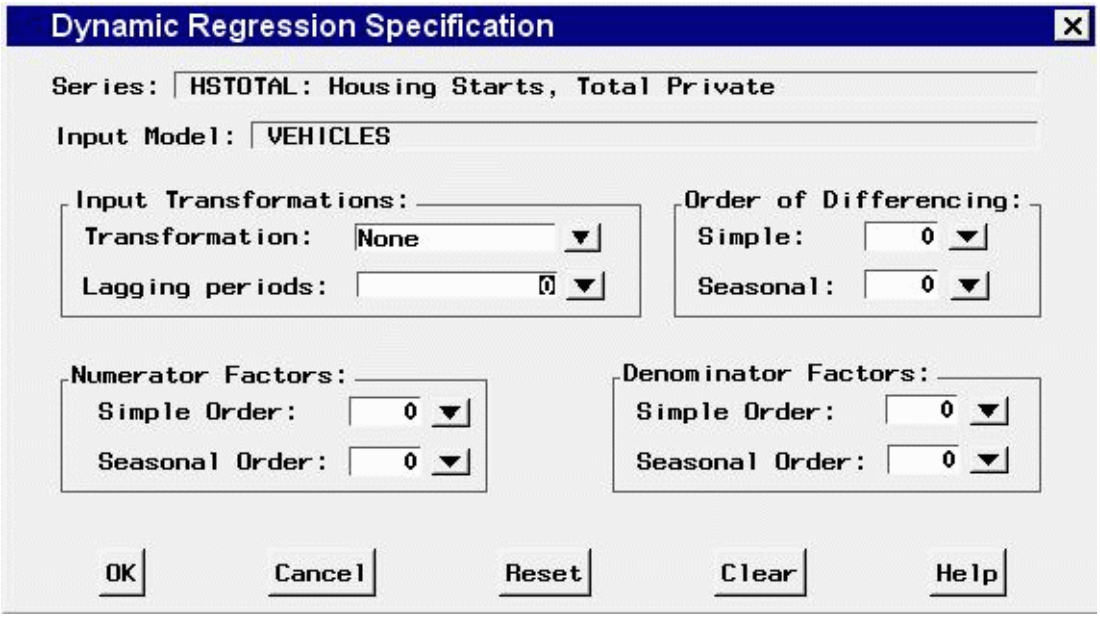
`Remove`
deletes a selected lag from the list of differencing lags.

`OK`
closes the window and returns the specified list to the Factored ARIMA Model Specification window.

`Cancel`
closes the window and discards any lags added to the list.

Dynamic Regression Specification Window

Use the Dynamic Regression Specification window to specify a dynamic regression or transfer function model for the effect of the predictor variable. It is invoked from the Dynamic Regressors Selection window.



Controls and Fields

`Series`
is the name and variable label of the current series.

`Input Model`
is a descriptive label for the dynamic regression model. You can type a label in this field or allow the

system to provide the label. If you leave the label blank, a label is generated automatically based on the options you specify. When no options are specified, the label is the name and variable label of the predictor variable.

`Input Transformation`
displays the transformation specified for the predictor variable. When a transformation is specified, the transfer function model is fit to the transformed input variable.

`Lagging periods`
is the pure delay in the effect of the predictor, *l*.

`Simple Order of Differencing`
is the order of differencing, *d*. Set this field to 1 to use the changes in the predictor variable.

`Seasonal Order of Differencing`
is the order of seasonal differencing, *D*. Set this field to 1 to difference the predictor variable at the seasonal lags—for example, to use the year-over-year or week-over-week changes in the predictor variable.

`Simple Order Numerator Factors`
is the order of the numerator factor of the transfer function, *p*.

`Seasonal Order Numerator Factors`
is the order of the seasonal numerator factor of the transfer function, *P*.

`Simple Order Denominator Factors`
is the order of the denominator factor of the transfer function, *q*.

`Seasonal Order Denominator Factors`
is the order of the seasonal denominator factor of the transfer function, *Q*.

`OK`
closes the window and adds the dynamic regression model specified to the model predictors list.

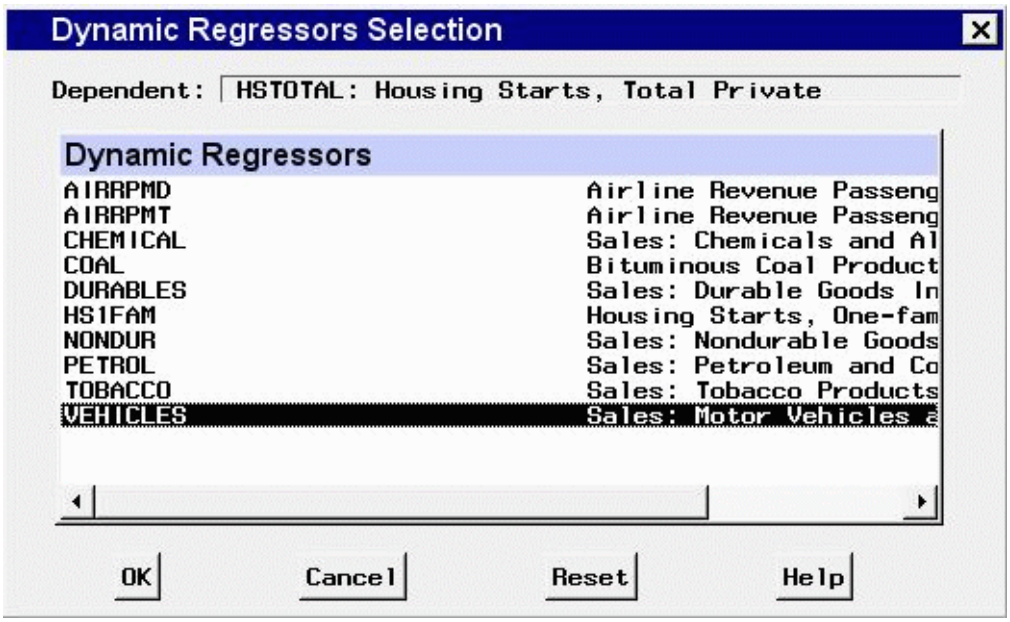
`Cancel`
closes the window without adding the dynamic regression model. Any options you specified are lost.

`Reset`
resets all options to their initial values upon entry to the window. This might be useful when editing a predictor specification; otherwise, Reset has the same function as Clear.

`Clear`
resets all options to their default values.

Dynamic Regressors Selection Window

Use the Dynamic Regressors Selection window to select an input variable as a dynamic regressor. Access this window from the pop-up menu which appears when you select the Add button of the ARIMA Model Specification window or Custom Model Specification window.



Controls and Fields

`Dependent`
: is the name and variable label of the current series.

`Dynamic Regressors`
: is a table listing the variables in the input data set. Select one variable in this list as the predictor series.

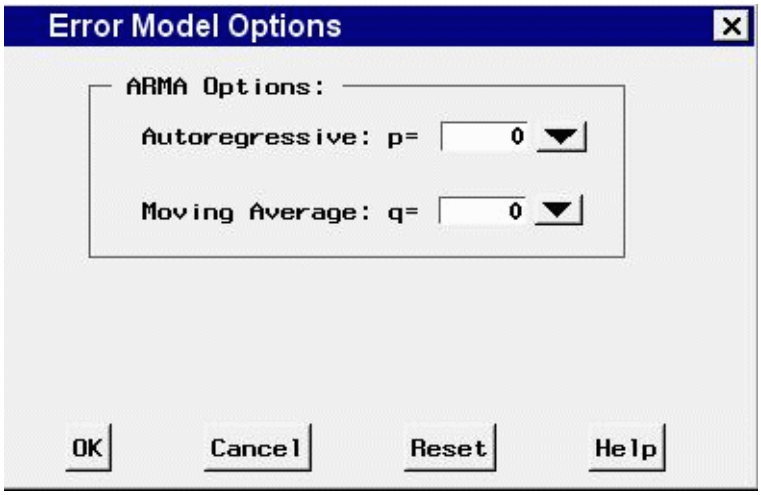
`OK`
: opens the Dynamic Regression Specification window for you to specify the form of the dynamic regression for the selected predictor series, and then closes the Dynamic Regressors Selection window and adds the specified dynamic regression to the model predictors list.

`Cancel`
: closes the window without adding the dynamic regression model. Any options you specified are lost.

`Reset`
: resets all options to their initial values upon entry to the window.

Error Model Options Window

Use the Error Model Options window to specify the autoregressive and moving-average orders for the residual autocorrelation part of a model defined by using the Custom Model Specification window. Access it by using the Set button of that window.



Controls and Fields

`ARIMA Options`
: Use these combo boxes to specify the orders of the ARIMA model. You can either type in a value or click the combo box arrow to select from a pop-up list.

  `Autoregressive`
  : defines the order of the autoregressive part of the model.

  `Moving Average`
  : defines the order of the moving-average term.

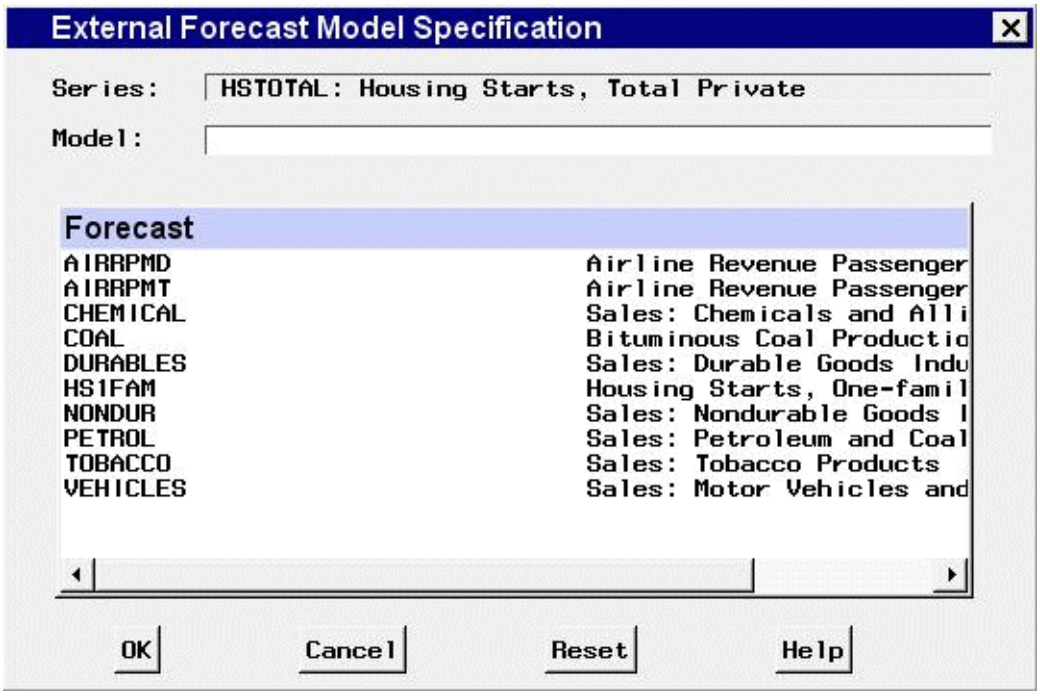
`OK`
: closes the Error Model Options window and returns to the Custom Model Specification window.

`Cancel`
: closes the Error Model Options window and returns to the Custom Model Specification window, discarding any changes made.

`Reset`
: resets all options to their initial values upon entry to the window.

External Forecast Model Specification Window

Use the External Forecast Model Specification window to add to the current project forecasts produced externally to the Time Series Forecasting System. To add an external forecast, select a variable from the selection list and choose the OK button. The name of the selected variable will be added to the list of models fit, and the values of this variable will be used as the forecast. For more information, see “Incorporating Forecasts from Other Sources” in the “Specifying Forecasting Models” chapter.



Controls and Fields

`OK`
: closes the window and adds the external forecast to the project.

`Cancel`
: closes the window without adding an external forecast to the project.

`Reset`
: deselects any selection made in the selection list.

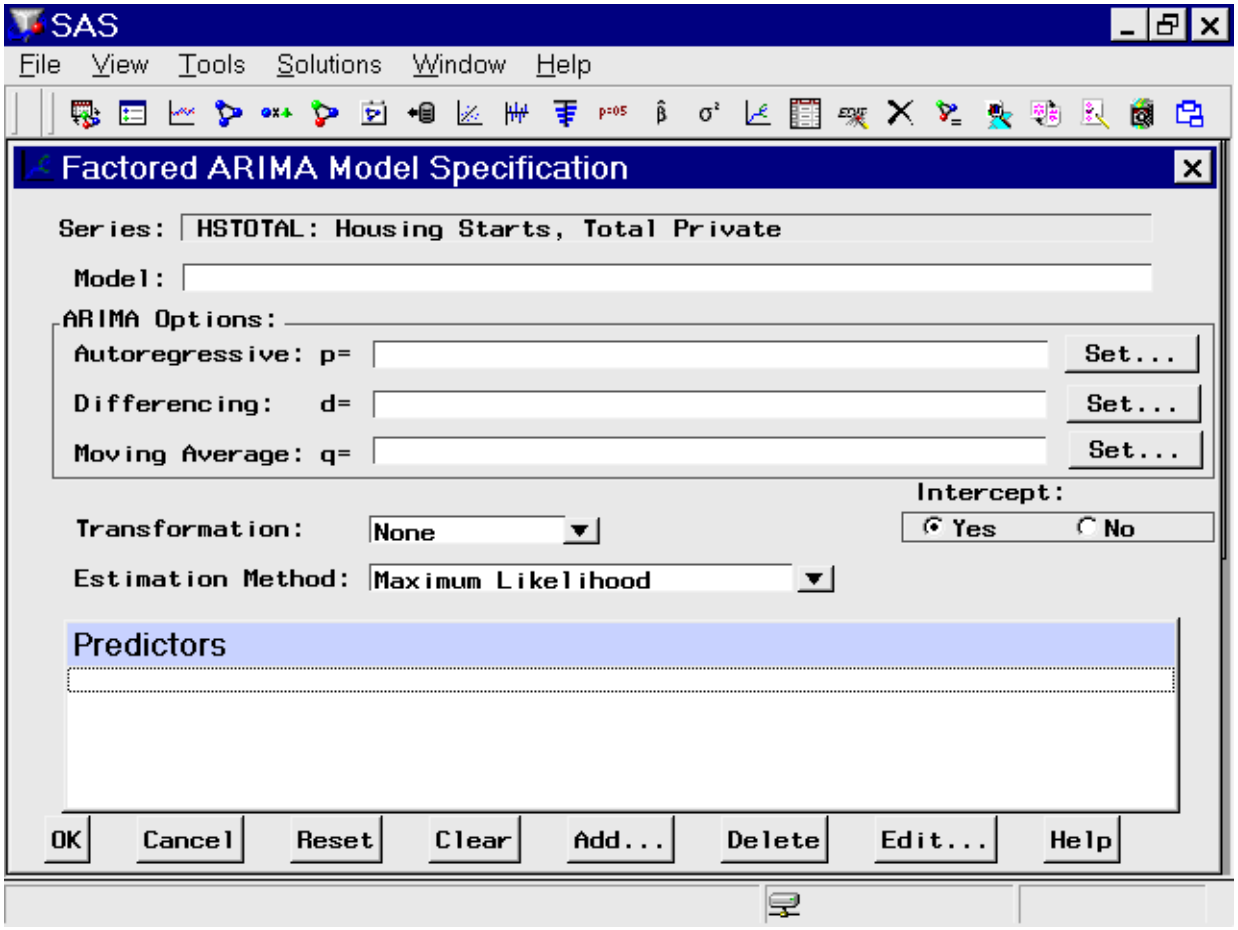
Factored ARIMA Model Specification Window

Use the ARIMA Model Specification window to specify an ARIMA model by using the notation:

```
p = (lag, ..., lag) ...(lag, ..., lag)
d = (lag, ..., lag)
q = (lag, ..., lag) ...(lag, ..., lag)
```

where p, d, and q represent autoregressive, differencing, and moving-average terms, respectively.

Access it from the Develop Models menu, where it is invoked from the Fit Model item under Edit in the menu bar, or from the pop-up menu when you click an empty area of the model table.



The Factored ARIMA Model Specification window is identical to the ARIMA Model Specification window, except that the p, d, and q terms are specified in a more general and less limited way. Only those controls and fields that differ from the ARIMA Model Specification window are described here.

Controls and Fields

`Model`
is a descriptive label for the model. You can type a label in this field or allow the system to provide a label. If you leave the label blank, a label is generated automatically based on the p, d, and q terms that you specify. For example, if you specify `p=(1,2,3), d=(1), q=(12)` and no intercept, the model label is `ARIMA p=(1,2,3) d=(1) q=(12) NOINT`. For monthly data, this is equivalent to the model `ARIMA(3,1,0)(0,0,1)s NOINT` as specified in the ARIMA Model Specification window or the Custom Model Specification window.

`ARIMA Options`
Specifies the ARIMA model in terms of the autoregressive lags (p), differencing lags (d), and moving-average lags (q).

`Autoregressive`
defines the autoregressive part of the model. Select the Set button to open the AR Polynomial Specification window, where you can add any set of autoregressive lags grouped into any number of factors.

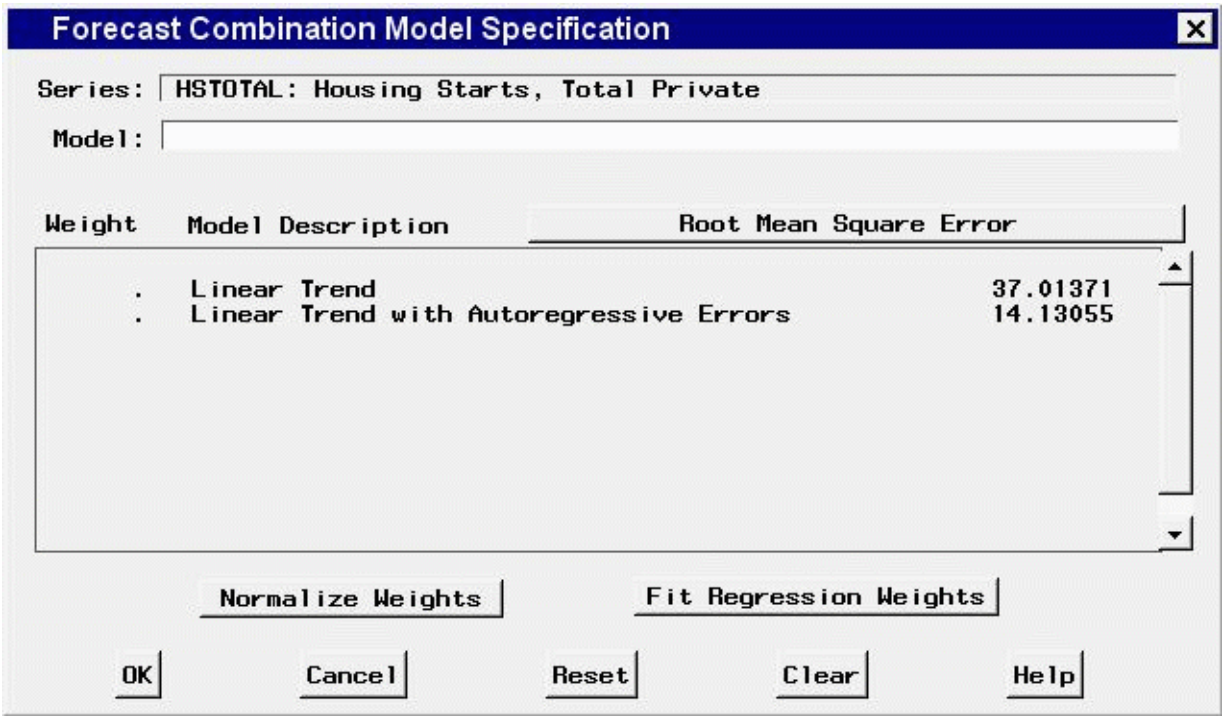
`Differencing`
: specifies differencing to be applied to the input data. Select the Set button to open the Differencing Specification window, where you can specify any set of differencing lags.

`Moving Average`
: defines the moving-average part of the model. Select the Set button to open the MA Polynomial Specification window, where you can add any set of moving-average lags grouped into any number of factors.

`Estimation Method`
: specifies the method used to estimate the model parameters. The Conditional Least Squares and Unconditional Least Squares methods generally require fewer computing resources and are more likely to succeed in fitting complex models. The Maximum Likelihood method requires more resources but provides a better fit in some cases. See also Estimation Details in Chapter 8, "The ARIMA Procedure."

Forecast Combination Model Specification Window

Use the Forecast Combination Model Specification window to produce forecasts by averaging the forecasts of two or more forecasting models. The specified combination of models is added to the model list for the series. Access this window from the Develop Models window whenever two or more models have been fit to the current series. It is invoked by selecting Combine Forecasts from the Fit Model submenu of the Edit menu, or from the pop-up menu which appears when you click an empty part of the model table.



Controls and Fields

`Series`
is the name and variable label of the current series.

`Model`
is a descriptive label for the model that you specify. You can type a label in this field or allow the system to provide a label. If you leave the label blank, a label is generated automatically based on the options you specify.

`Weight`
is a column of the forecasting model table that contains the weight values for each model. The forecasts for the combined model are computed as a weighted average of the predictions from the models in the table that use these weights. Models with missing weight values are not included in the forecast combination. You can type weight values in these fields or you can use other features of the window to set the weights.

`Model Description`
is a column of the forecasting model table that contains the descriptive labels of the forecasting models fit to the current series that are available for combination.

`Root Mean Square Error` (or other statistic name) button
is the button above the right side of the table. It displays the name of the current model selection criterion: a statistic that measures how well each model in the table fits the values of the current series for observations within the evaluation range. Clicking this button opens the Model Selection Criterion window to enable you to select a different statistic.

`Normalize Weights` button
replaces each nonmissing value in the Weights column with the current value divided by the sum of the weights. The resulting weights are proportional to original weights and sum to 1.

`Fit Regression Weights` button
computes weight values for the models in the table by regressing the series on the predictions from the models. The values in the Weights column are replaced by the estimated coefficients produced by this linear regression. If some weight values are nonmissing and some are missing, only models with nonmissing weight values are included in the regression. If all weights are missing, all models are used.

`OK`
closes the Forecast Combination Model Specification window and fits the model.

`Cancel`
closes the Forecast Combination Model Specification window without fitting the model. Any options you specified are lost.

`Reset`
resets all options to their initial values upon entry to the Forecast Combination Model Specification window. This might be useful when editing an existing model specification; otherwise, Reset has the same function as Clear.

`Clear`
resets all options to their default values.

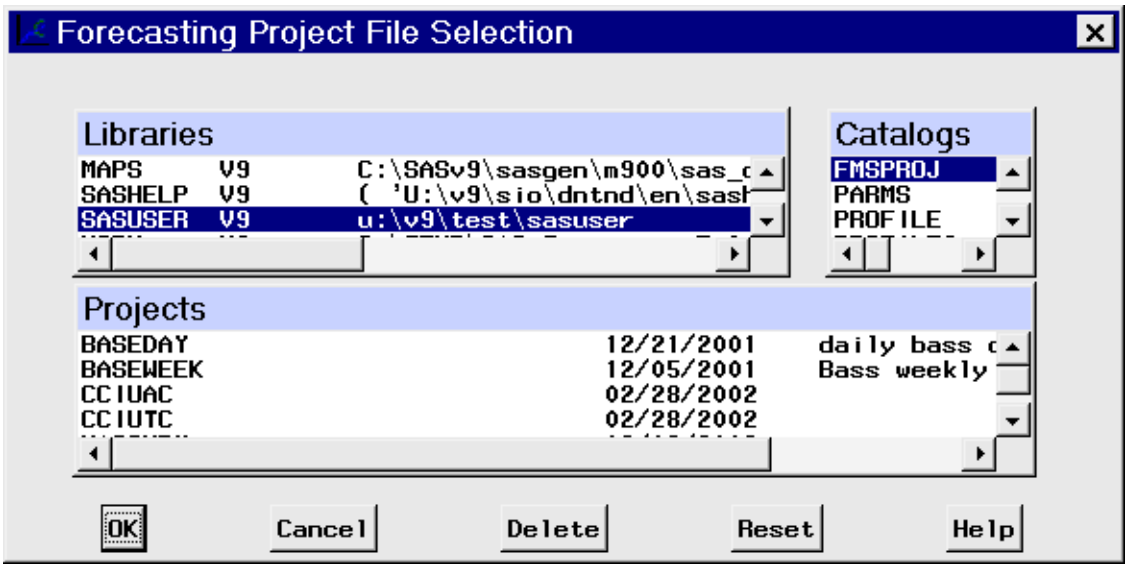
Mouse Button Actions

You can select or deselect models for inclusion in the combination model by positioning the mouse cursor over the model description and pressing the left mouse button. When you select a model in this way, the weights are automatically updated.

The newly selected model is given a weight equal to the average weight of the previously selected models, and all the nonmissing weights are normalized to sum to 1. When you use the mouse to remove a model from the combination, the weight of the deselected model is set to missing and the remaining nonmissing weights are normalized to sum to 1.

Forecasting Project File Selection Window

Use the Forecasting Project File Selection window to locate and load a previously stored forecasting project. Access it from the project Browse button of the Manage Forecasting Project window or the Time Series Forecasting window or from the Open Project item under the File menu of the Develop Models window.



Selection Lists

`Libraries`
: is a list of currently assigned libraries. When you select a library from this list, the catalogs in that library are shown in the catalog selection list.

`Catalogs`
: is a list of catalogs contained in the currently selected library. When you select a catalog from this list, any forecasting project entries stored in that catalog are shown in the projects selection list.

`Projects`
: is a list of forecasting project entries contained in the currently selected catalog.

Controls and Fields

`OK`
: closes the window and opens the selected project.

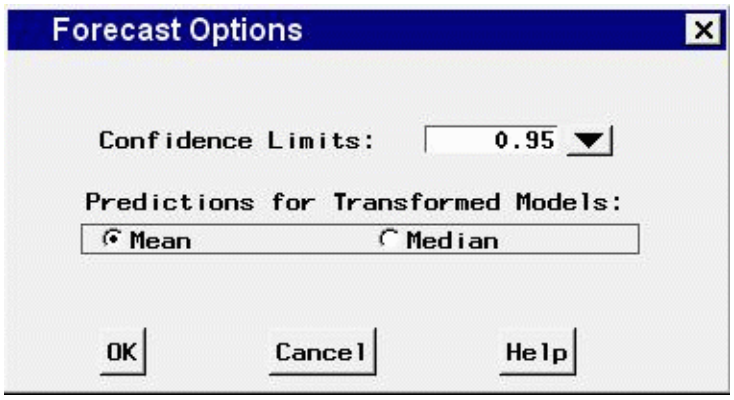
`Cancel`
: closes the window without selecting a project.

`Delete`
: deletes the selected project file.

`Reset`
: restores selections to those which were set before the window was opened.

Forecast Options Window

Use the Forecast Options window to set options to control how forecasts and confidence limits are computed. It is available from the Forecast Options item in the Options menu of the Develop Models window, Automatic Model Fitting window, Produce Forecasts, and Manage Projects windows.



Controls and Fields

`Confidence Limits`
: specifies the size of the confidence limits for the forecast values. For example, a value of 0.95 specifies 95% confidence intervals. You can type in a number or select from the pop-up list.

`Predictions for transformed models`
: controls how forecast values are computed for models that employ a series transformation. See the section Predictions for Transformed Models in Chapter 61, "Forecasting Process Details," for more information. The values are as follows.

  `Mean`
  : specifies that forecast values be predictions of the conditional mean of the series.

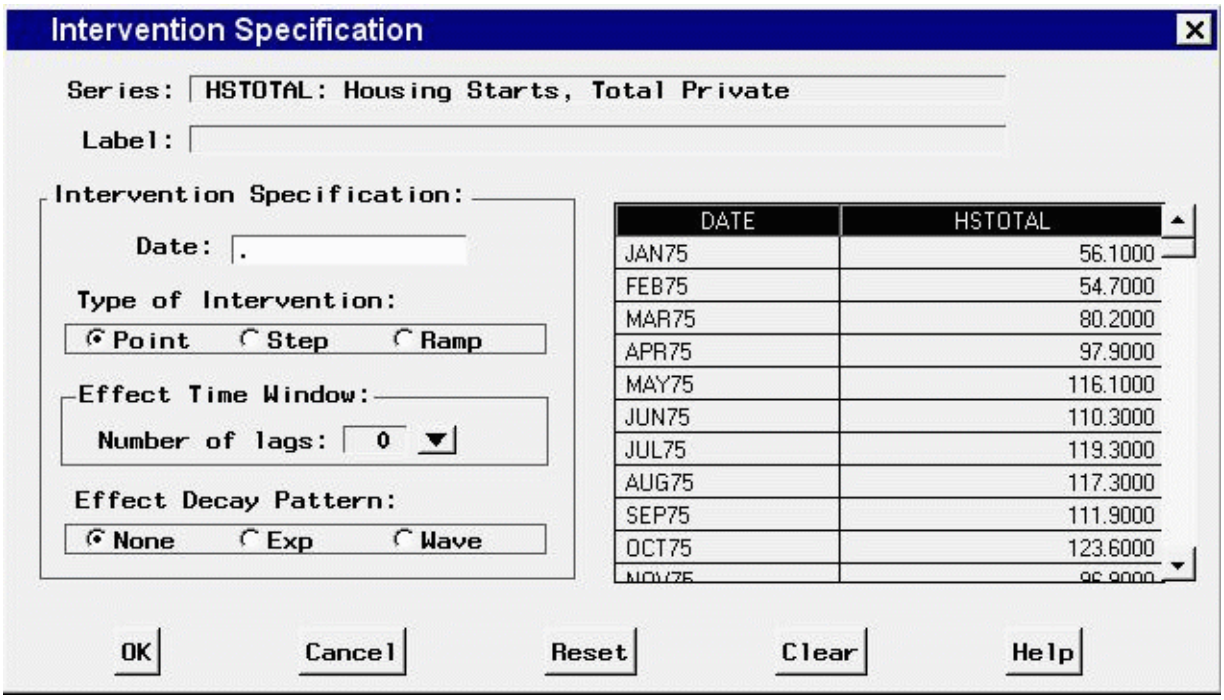
  `Median`
  : specifies that forecast values be predictions of the conditional median of the series.

`OK`
closes the window and saves the option settings you specified.

`Cancel`
closes the window without changing the forecast options. Any options you specified are lost.

Intervention Specification Window

Use the Intervention Specification window to specify intervention effects to model the impact on the series of unusual events. Access it from the Intervention for Series window. For more information, see the section "Interventions" on page 3755.



Controls and Fields

`Series`
is the name and variable label of the current series.

`Label`
is a descriptive label for the intervention effect that you specify. You can type a label in this field or allow the system to provide the label. If you leave the label blank, a label is generated automatically based on the options you specify.

`Date`
is the date that the intervention occurs. You can type a date value in this field, or you can set the date by selecting a row of the data table on the right side of the window.

`Type of Intervention`

`Point`
specifies that the intervention variable is zero except for the specified date.

`Step`
specifies that the intervention variable is zero before the specified date and a constant 1.0 after the date.

`Ramp`
specifies that the intervention variable is an increasing linear function of time after the date of the intervention and zero before the intervention date.

`Number of lags`
specifies the numerator order for the transfer function model for the intervention effect. Select a value from the pop-up list.

`Effect Decay Pattern`
specifies the denominator order for the transfer function model for the intervention effect. The value "Exp" specifies a single lag denominator factor; the value "Wave" specifies a two-lag denominator factor.

`OK`
closes the window and adds the intervention effect specified to the series interventions list.

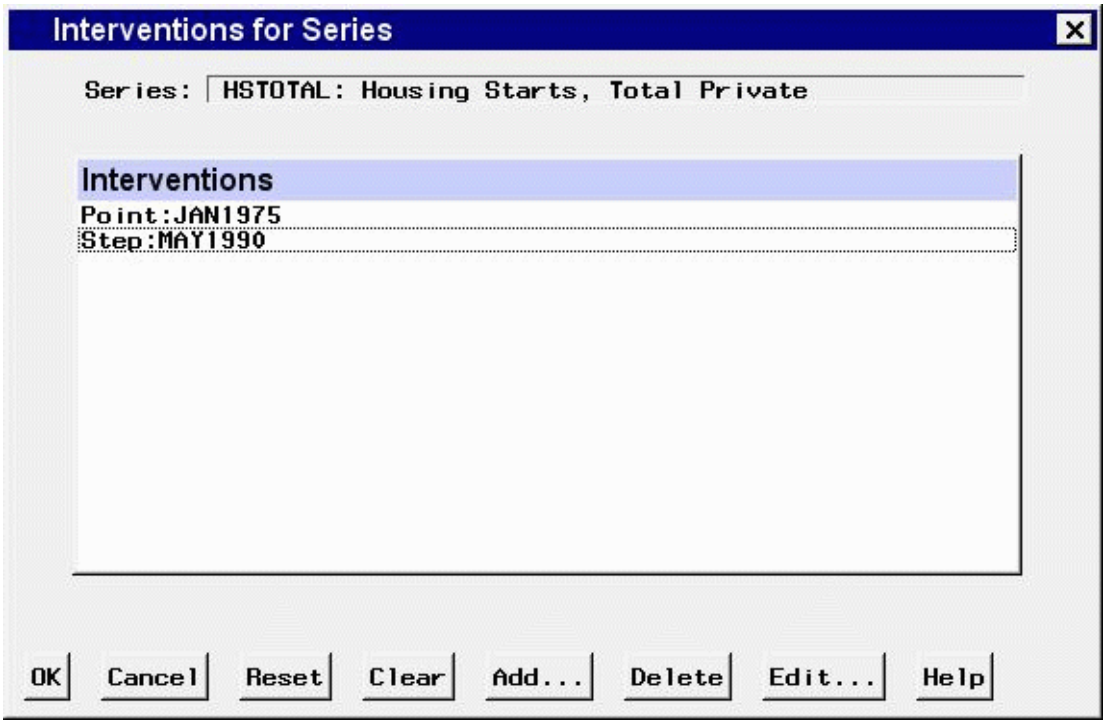
`Cancel`
closes the window without adding the intervention. Any options you specified are lost.

`Reset`
resets all options to their initial values upon entry to the window. This might be useful when editing an intervention specification; otherwise, Reset has the same function as Clear.

`Clear`
resets all options to their default values.

Interventions for Series Window

Use the Interventions for Series window to create and edit a list of intervention effects to model the impact on the series of unusual events and to select intervention effects as predictors for forecasting models. Access it from the Add button pop-up menu of the ARIMA Model Specification or Custom Model Specification window, or by selecting Define Interventions from the Tools in the Develop Models window. For more information, see the section "Interventions" on page 3755.



Controls and Fields

`Series`
: is the name and variable label of the current series.

`OK`
: closes the window. If you access this window from the ARIMA Model Specification window or the Custom Model Specification window, any interventions that are selected (highlighted) in the list are added to the model. If you access this window from the Tools menu, all interventions in the list are saved for the current series.

`Cancel`
: closes the window without returning a selection or changing the interventions list. Any options you specified are lost.

`Reset`
: resets the list as it was on entry to the window.

`Clear`
: deletes all interventions from the list.

`Add`
: opens the Intervention Specification window to specify a new intervention effect and add it to the list.

`Delete`
: deletes the currently selected (highlighted) entries from the list.

`Edit`
: opens the Intervention Specification window to edit the currently selected (highlighted) intervention.

Mouse Button Actions

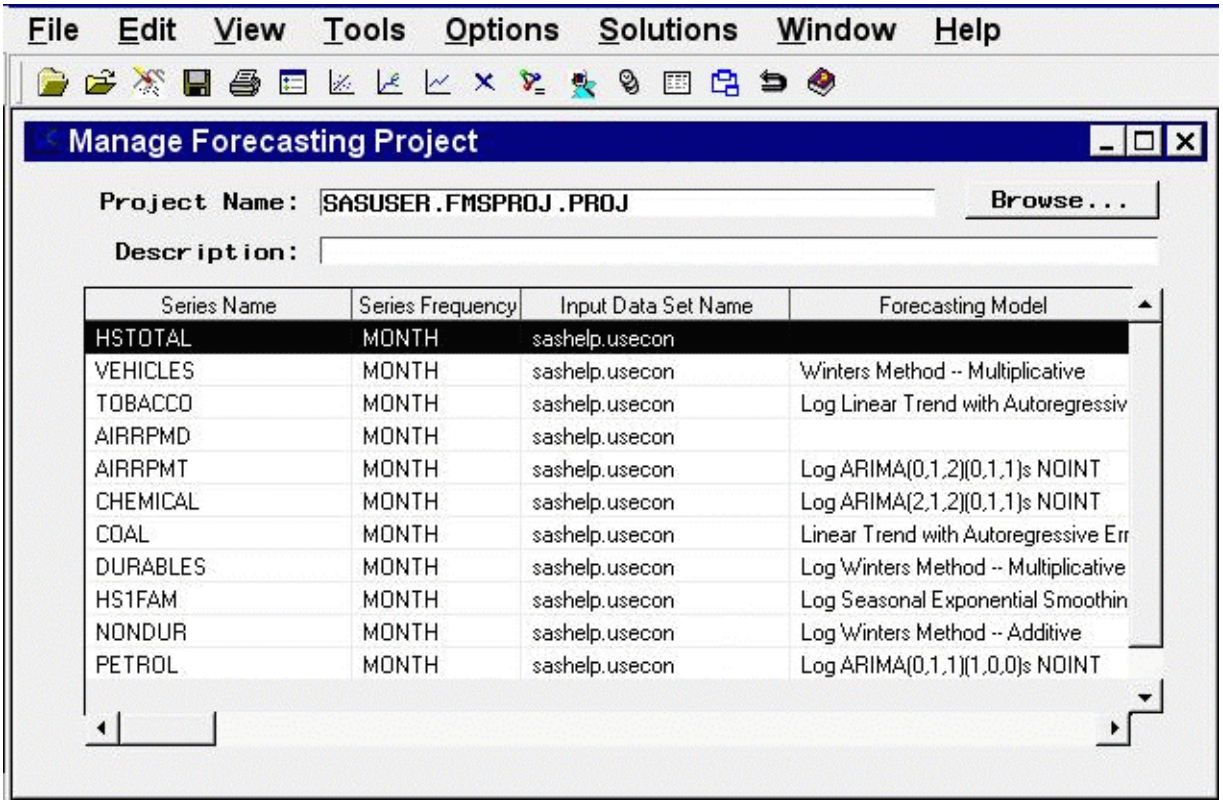
To select or deselect interventions, position the mouse cursor over the intervention's label in the Interventions list and press the left mouse button.

When you position the mouse cursor in the Interventions list and press the right mouse button, a menu containing the actions Add, Delete, and Edit appears. These actions are the same as the Add, Delete, and Edit buttons.

Double-clicking on an intervention in the list invokes an Edit action for that intervention specification.

Manage Forecasting Project Window

Use this resizable window to work with collections of series, models, and options called *projects*. The window contains a project name, a description field, and a table of information about all the series for which you have fit forecasting models. Access it by using the Manage Projects button on the Time Series Forecasting window.



Controls and Fields

`Project Name`
: is the name of the SAS catalog entry in which forecasting models and other results will be stored and from which previously stored results are loaded into the forecasting system. You can specify the project

by typing a SAS catalog entry name in this field or by selecting the Browse button to the right of this field. If you specify the name of an existing catalog entry, the information in the project file is loaded. If you specify a one-level name, it is assumed to be the name of a project in the "fmsproj" catalog in the "sasuser" library. For example, typing `samproj` is equivalent to typing `sasuser.fmsproj.samproj`.

project `Browse` button
opens the Forecasting Project File Selection window to enable you to select and load the project from a list of previously stored project files.

`Description`
is a descriptive label for the forecasting project. The description you type in this field will be stored with the catalog entry shown in the Project field if you save the project.

Series List Table

The table of series for which forecasting models have been fit contains the following columns.

`Series Name`
is the name of the time series variable represented in the given row of the table.

`Series Frequency`
is the time interval (data frequency) for the time series.

`Input Data Set Name`
is the input data set that provided the data for the series.

`Forecasting Model`
is the descriptive label for the forecasting model selected for the series.

`Statistic Name`
is the statistic of fit for the forecasting model selected for the series.

`Number of Models`
is the total number of forecasting models fit to the series. If there is more than one model for a series, use the Model List window to see a list of models.

`Series Label`
is the variable label for the series.

`Time ID Variable Name`
is the time ID variable for the input data set for the series.

`Series Data Range`
is the time range of the nonmissing values of the series.

`Model Fit Range`
is the period of fit used for the series.

`Model Evaluation Range`
is the evaluation period used for the series.

`Forecast Range`
is the forecast period set for the series.

Menu Bar

File

New
: opens a dialog which lets you create a new project, assign it a name and description, and make it the active project.

Open
: opens a dialog that lets you select and load a previously saved project.

Close
: closes the Manage Forecasting Project window and returns to the main window.

Save
: saves the current state of the system (including all the models fit to a series) to the current project catalog entry.

Save As
: saves the current state of the system with a prompt for the name of the catalog entry in which to store the information.

Save to Data Set
: saves the current project file information in a SAS data set. The contents of the data set are the same as the information displayed in the series list table.

Delete
: deletes the current project file.

Import Data
: is available if you license SAS/Access software. It opens an Import Wizard, which you can use to import your data from an external spreadsheet or data base to a SAS data set for use in the Time Series Forecasting System.

Export Data
: is available if you license SAS/Access software. It opens an Export Wizard, which you can use to export a SAS data set, such as a forecast data set created with the Time Series Forecasting System, to an external spreadsheet or data base.

Print
: prints the current project file information.

Print Setup
: opens the Print Setup window, which allows you to access your operating system print setup.

Edit

Delete Series
: deletes all models for the selected (highlighted) row of the table and removes the series from the project.

`Clear`
resets the system, deleting all series and models from the project.

`Reset`
restores the Manage Forecasting Project window to its initial state.

`View`

`Data Set`
opens a Viewtable window to display the input data set for the selected (highlighted) series.

`Series`
opens the Time Series Viewer window to display plots of the selected (highlighted) series.

`Model`
opens the Model Viewer window to show the current forecasting model for the selected series.

`Forecast`
opens the Model Viewer to display plots of the forecasts produced by the forecasting model for the selected (highlighted) series.

`Tools`

`Diagnose Series`
opens the Series Diagnostics window to perform the automatic series diagnostic process to determine the kinds of forecasting models appropriate for the selected (highlighted) series.

`List Models`
opens the Model List window for the selected (highlighted) series, which displays a list of all the models that you fit for the series. This action is the same as double-clicking the mouse on the table row.

`Generate Data`
opens the Time Series Simulation window. This window enables you to simulate ARIMA time series processes and is useful for educational exercises or testing the system.

`Refit Models`

`All Series`
refits all the models for all the series in the project by using data within the current fit range.

`Selected Series`
refits all the models for the currently highlighted series by using data within the current fit range.

Reevaluate Models

All Series
reevaluates all the models for all the series in the project by using data within the current evaluation fit range.

Selected Series
reevaluates all the models for the currently highlighted series by using data within the current evaluation range.

Options

Time Ranges
opens the Time Ranges Specification window to enable you to change the fit and evaluation time ranges and the forecast horizon.

Default Time Ranges
opens the Default Time Ranges window to enable you to control how the system sets the time ranges for series when you do not explicitly set time ranges with the Time Ranges Specification window. Settings made by using this window do not affect series you are already working with; they take effect when you select a new series.

Model Selection List
opens the Model Selection List editor window. Use this to edit the set of forecasting models considered by the automatic model selection process and displayed by the Models to Fit window.

Statistics of Fit
opens the Statistics of Fit Selection window, which controls which of the available statistics will be displayed.

Forecast Options
opens the Forecast Options window, which enables you to control the widths of forecast confidence limits and control the kind of predicted values computed for models that include series transformations.

Column Labels
enables you to set long or short column labels. Long labels are used by default.

Include Interventions
controls whether intervention effects defined for the current series are automatically added as predictors to the models considered by the automatic selection process and displayed by the Model Selection List editor window. When the Include Interventions option is selected, the series interventions are also automatically added to the predictors list when you specify a model in the ARIMA and Custom Models Specification windows.

Print Audit Trail
prints to the SAS log information about the models fit by the system. A check mark or filled check box next to this item indicates that the audit option is turned on.

Show Source Statements
controls whether SAS statements submitted by the forecasting system are printed in the SAS log. When the Show Source Statements option is selected, the system sets the SAS system option

SOURCE before submitting SAS statements; otherwise, the system uses the NOSOURCE option. Note that only some of the functions performed by the forecasting system are accomplished by submitting SAS statements. A check mark or filled check box next to this item indicates that the option is turned on.

Left Mouse Button Actions

If you select a series in the table by positioning the cursor over the table row and clicking with the left mouse button once, that row of the table is highlighted. Menu bar actions such as Delete Series will apply to the highlighted row of the table.

If you select a series in the table by positioning the cursor over the table row and double-clicking with the left mouse button, the system opens the Model List window for that series, which displays a list of all the models that you fit for the series. This is the same as the List Models action under Tools in the menu bar.

Right Mouse Button Actions

Clicking the right mouse button invokes a pop-up menu of actions applicable to the highlighted series. The actions in this menu are as follows.

`Delete Series`
: deletes the highlighted series and its models from the project. This is the same as Delete Series in the Edit menu.

`Refit All Models`
: refits all models attached to the highlighted series by using data within the current fit range. This is the same as the Selected Series item under Refit Models in the Tools menu.

`Reevaluate All Models`
: reevaluates all models attached to the highlighted series by using data within the current evaluation range. This is the same as the Selected Series item under Reevaluate Models in the Tools menu.

`List Models`
: invokes the Model List window. This is the same as List Models under the Tools menu.

`View Series`
: opens the Time Series Viewer window to display plots of the highlighted series. This is the same as the Series item under the View menu.

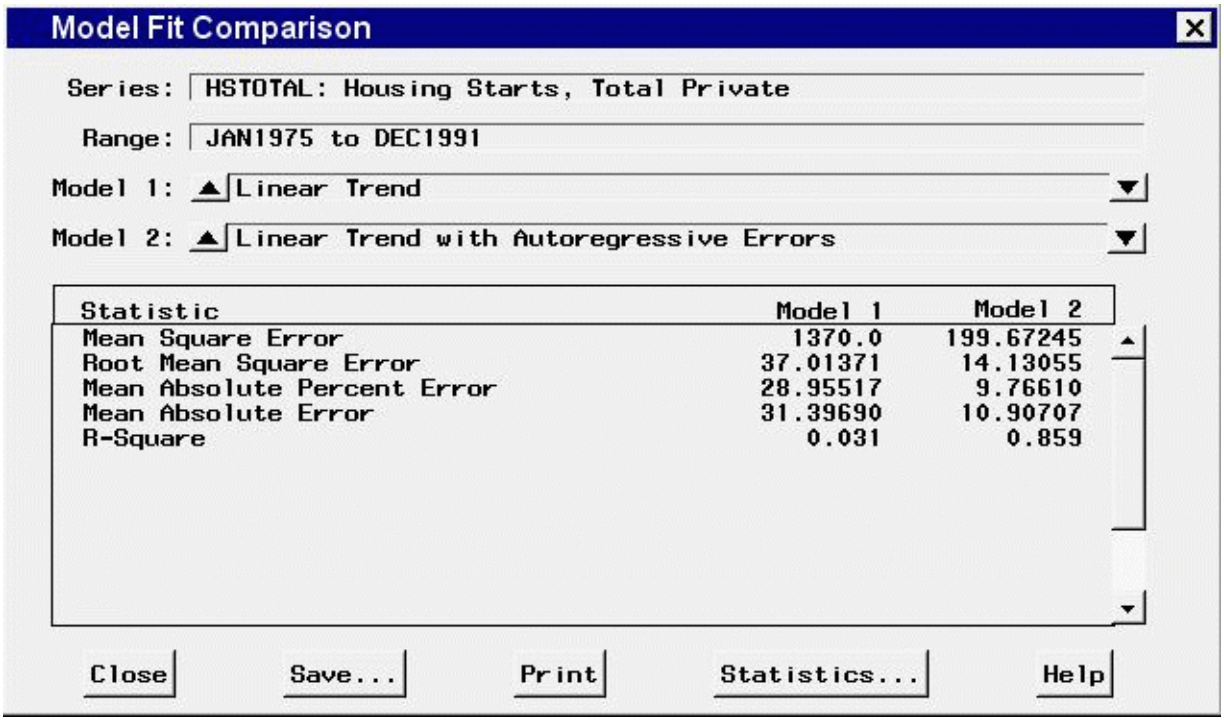
`View Forecasting Model`
: invokes the Model Viewer window to display the forecasting model for the highlighted series. This is the same as the Model item under the View menu.

`View Forecast`
: opens the Model Viewer window to display the forecasts for the highlighted series. This is the same as the Forecast item under the View menu.

`Refresh`
: updates information shown in the Manage Forecasting Project window.

Model Fit Comparison Window

Use the Model Fit Comparison window to compare goodness-of-fit statistics for any two models fit to the current series. Access it from the Tools menu of the Develop Models window and the Automatic Model Fitting Results window whenever two or more models have been fit to the series.



Controls and Fields

`Series`
identifies the current time series variable.

`Range`
displays the starting and ending dates of the series data range.

`Model 1`
shows the model currently identified as Model 1.

`Model 1` upward arrow button
enables you to change the model identified as Model 1 if it is not already the first model in the list of models associated with the series. Select this button to cycle upward through the list of models.

`Model 1` downward arrow button
: enables you to change the model identified as Model 1 if it is not already the last model in the list of models. Select this button to cycle downward through the list of models.

`Model 2`
: shows the model currently identified as Model 2.

`Model 2` upward arrow button
: enables you to change the model identified as Model 2 if it is not already the first model in the list of models associated with the series. Select this button to cycle upward through the list of models.

`Model 2` downward arrow button
: enables you to change the model identified as Model 2 if it is not already the last model in the list of models. Select this button to cycle downward through the list of models.

`Close`
: closes the Model Fit Comparison window.

`Save`
: opens a dialog for specifying the name and label of a SAS data set to which the statistics will be saved. The data set will contain all available statistics and their values for Model 1 and Model 2, as well as a flag variable that is set to 1 for those statistics that were displayed.

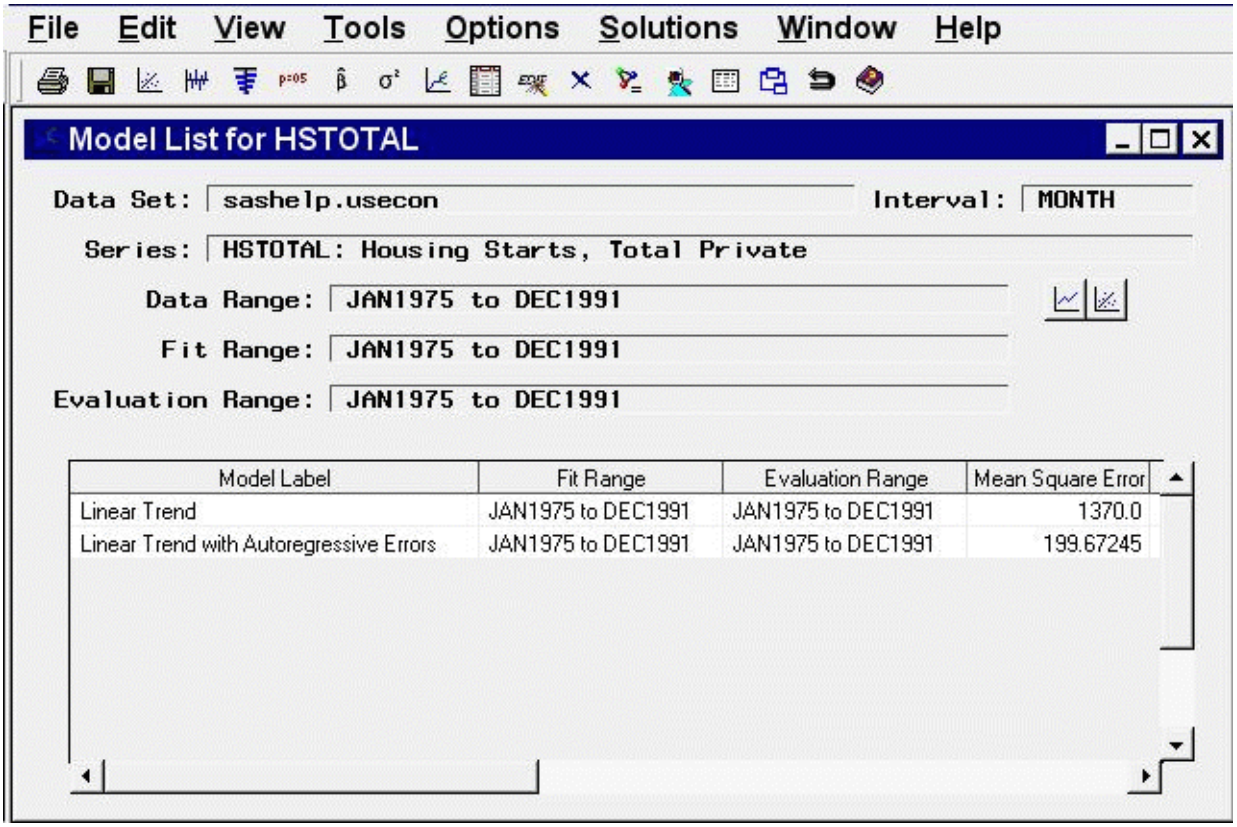
`Print`
: prints the contents of the table to the SAS Output window. If you find that the contents do not appear immediately in the Output window, you need to set scrolling options. Select "Preferences" under the Options submenu of the Tools menu. In the Preferences window, select the Advanced tab, then set output scroll lines to a number greater than zero.

  If you want to route the contents to a printer, go to the Output window and select "Print" from the File menu.

`Statistics`
: opens the Statistics of Fit Selection window for controlling which statistics are displayed.

Model List Window

This resizable window shows all of the models that have been fit to a particular series in a project. Access it from the Manage Forecasting Project window by selecting a series in the series list table and choosing "List Models" from the Tools menu or by double-clicking the series.



Controls and Fields

`Data Set`
: is the name of the current input data set.

`Interval`
: is the time interval (data frequency) for the input data set.

`Series`
: is the variable name and label of the current time series.

`Data Range`
: is the date of the first and last nonmissing data values available for the current series in the input data set.

`Fit Range`
: is the current period of fit setting. This is the range of data that will be used to fit models to the series. It might be different from the fit ranges shown in the table, which were in effect when the models were fit.

`Evaluation Range`
: is the current period of evaluation setting. This is the range of data that will be used to calculate the goodness-of-fit statistics for models fit to the series. It might be different from the evaluation ranges shown in the table, which were in effect when the models were fit.

`View Series Graphically` icon
: opens the Time Series Viewer window to display plots of the current series.

`View Model Graphically` icon
: opens the Model Viewer to display graphs and tables for the currently highlighted model.

Model List Table

The table of models fit to the series contains columns that show the model label, the fit range and evaluation range used to fit the models, and all of the currently selected fit statistics. You can change the selection of fit statistics by using the Statistics of Fit Selection window.

Click on column headings to sort the table by a particular column. If a model is highlighted, clicking with the right mouse button invokes a pop-up menu that provides actions applicable to the highlighted model. It includes the following items.

`View Model`
: opens the Model Viewer on the selected model. This is the same as "Model Predictions" under the View menu.

`View Parameter Estimates`
: opens the Model Viewer to display the parameter estimates table for the currently highlighted model. This is the same as "Parameter Estimates" under the View menu.

`View Statistics of Fit`
: opens the Model Viewer to display the statistics of fit table for the currently highlighted model. This is the same as "Statistics of FIt" under the View menu.

`Edit Model`
: opens the appropriate model specification window for changing the attributes of the highlighted model and fitting the modified model.

`Refit Model`
: refits the highlighted model using the current fit range.

`Reevaluate Model`
: reevaluates the highlighted model using the current evaluation range.

`Delete Model`
: deletes the highlighted model from the project.

`View Forecasts`
: opens the Model Viewer to show the forecasts for the highlighted model. This is the same as "Forecast Graph" under the View menu.

Menu Bar

`File`

`Save`
: opens a dialog which lets you save the contents of the table to a specified SAS data set.

`Import Data`
: is available if you license SAS/Access software. It opens an Import Wizard, which you can use to import your data from an external spreadsheet or data base to a SAS data set for use in the Time Series Forecasting System.

`Export Data`
: is available if you license SAS/Access software. It opens an Export Wizard, which you can use to export a SAS data set, such as a forecast data set created with the Time Series Forecasting System, to an external spreadsheet or data base.

`Print`
: sends the contents of the table to a printer as defined through Print Setup.

`Print Setup`
: opens the Print Setup window, which allows you to access your operating system print setup.

`Close`
: closes the window and returns to the Manage Forecasting Projects window.

`Edit`

`Edit Model`
: enables you to modify the specification of the currently highlighted model in the table and fit the modified model. The new model replaces the current model in the table.

`Refit Model`
: refits the currently highlighted model using data within the current fit range.

`Reevaluate Model`
: recomputes statistics of fit for the currently highlighted model using data within the current evaluation range.

`Delete Model`
: deletes the currently highlighted model from the model table.

`Reset`
: restores the contents of the Model List window to the state initially displayed.

`View`

`Series`
: opens the Time Series Viewer window to display plots of the current series. This is the same as the View Series Graphically icon.

`Model Predictions`
: opens the Model Viewer to display a predicted and actual plot for the currently highlighted model. This is the same as the View Model Graphically icon.

`Prediction Errors`
: opens the Model Viewer to display the prediction errors for the currently highlighted model.

`Prediction Error Autocorrelations`
: opens the Model Viewer to display the prediction error autocorrelations, partial autocorrelations, and inverse autocorrelations for the currently highlighted model.

`Prediction Error Tests`
: opens the Model Viewer to display graphs of white noise and stationarity tests on the prediction errors of the currently highlighted model.

`Parameter Estimates`
: opens the Model Viewer to display the parameter estimates table for the currently highlighted model.

`Statistics of Fit`
: opens the Model Viewer window to display goodness-of-fit statistics for the currently highlighted model.

`Forecast Graph`
: opens the Model Viewer to graph the forecasts for the currently highlighted model.

`Forecast Table`
: opens the Model Viewer to display forecasts for the currently highlighted model in a table.

`Options`

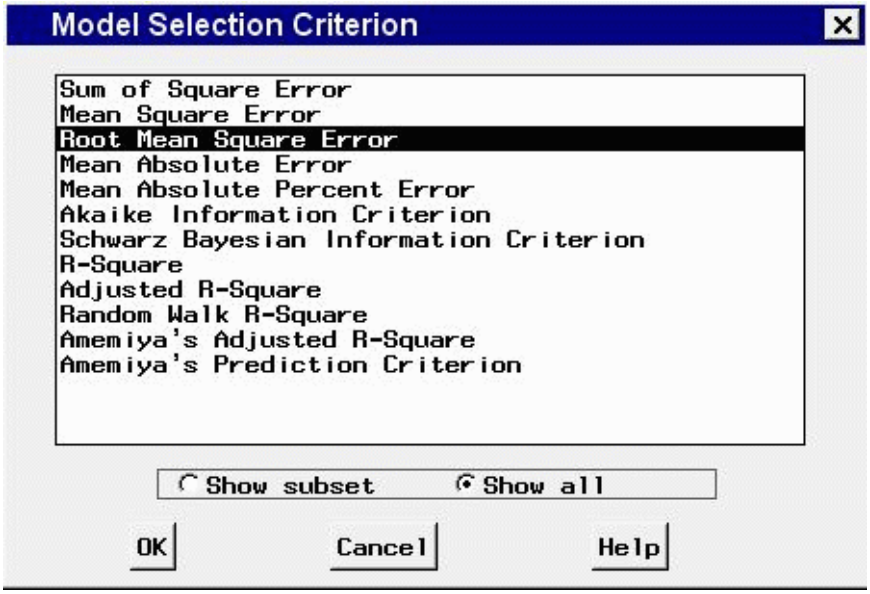
`Statistics of Fit`
: opens the Statistics of Fit Selection window, which presents a list of statistics that the system can display. Use this action to customize the list of statistics shown in the Model Viewer, Automatic Model Fitting Results, and Model Fit Comparison windows and available for selection in the Model Selection Criterion menu.

`Column Labels`
: enables you to set long or short column labels. Long labels are used by default.

Model Selection Criterion Window

Use the Model Selection Criterion window to select the model selection criterion statistic used by the automatic selection process to determine the best fitting forecasting model. Model selection criterion statistics are a subset of those shown in the Statistics of Fit Selection window, since some statistics of fit, such as number of observations, are not useful for model selection.

This window is available from the Model Selection Criterion item of the Options menu of the Develop Models window, Automatic Model Fitting window, and Produce Forecasts window.



Controls and Fields

`Show subset`
: when selected, lists only those model selection criterion statistics that are selected in the Statistics of Fit Selection window.

`Show all`
: when selected, lists all available model selection criterion statistics.

`OK`
: closes the window and sets the model selection criterion to the statistic you specified.

`Cancel`
: closes the window without changing the model selection criterion.

Model Selection List Editor Window

Use the Model Selection List Editor window to edit the model selection list, including adding your own custom models, and to specify which models in the list are to be used in the automatic fitting process. Access it from the Options menu in the Develop Models, Automatic Model Fitting window, Produce Forecasts, and Manage Projects windows.

The window initially displays the current model list for your project. You can modify this set of models in several ways:

- Open one or more alternate model lists to replace or append to the current model list. These can be either model lists included with the software or model lists previously saved by you or other users.

- Turn the autofit option on or off for individual models. Those that are not flagged for autofit will be available by using the Models to Fit window but not by using automatic model fitting.
- Delete models from the list that are not needed for your project.
- Reorder the models in the list.
- Edit models in the list.
- Create a new empty list.
- Add new models to the list.

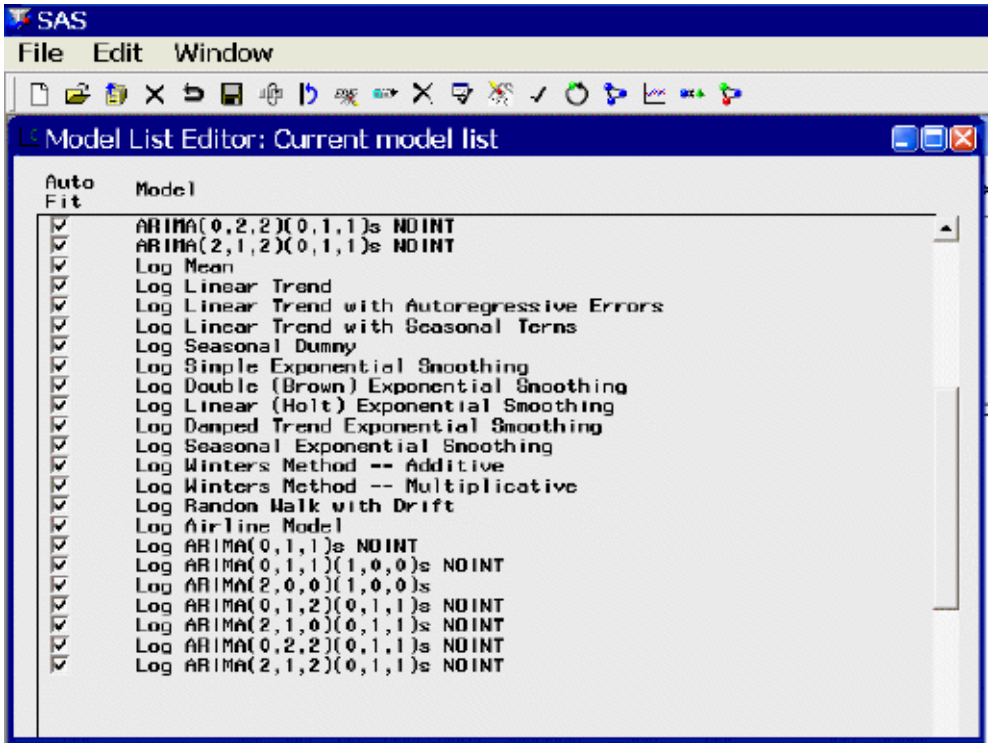
Having modified the current model list, you can save it for future use in several ways:

- Save it in a catalog so it can be opened later in the Model Selection List Editor.
- Save it as the user default to be used automatically when new projects are created.
- Select *close* to close the Model Selection List Editor and attach the modified model selection list to the current project.
- Select *cancel* to close the Model Selection List Editor without changing the current project's model selection list.

Since model selection lists are not bound to specific data sources, care must be taken when including data-specific features such as interventions and regressors. When you add an ARIMA, Factored ARIMA, or Custom model to the list, you can add regressors by selecting from the variables in the current data set. If there is no current data set, you will be prompted to specify a data set so you can select regressors from the series it contains.

If you use a model list that has models with a particular regressor name on a data set that does not contain a series of that name, model fitting will fail. However, you can make global changes to the regressor names in the model list by using *Set regressor names*. For example, you might use the list of dynamic regression models found in the sashelp.forcast catalog. It uses the regressor name "price." If your regressor series is named "x," you can specify "price" as the current regressor name and "x" as the "change to" name. The change will be applied to all models in the list that contain the specified regressor name.

Interventions cannot be defined for models defined from the Model Selection List Editor. However, you can define interventions by using the Intervention Specification Window and apply them to your models by turning on the Include Interventions option.



Auto Fit

The auto fit column of check boxes enables you to eliminate some of the models from being used in the automatic fitting process without having to delete them from the list. By default, all models are checked, meaning that they are all used for automatic fitting.

Model

This column displays the descriptions of all models in the model selection list. You can select one or more models by clicking them. Selected models are highlighted and become the object of the actions Edit, Move, and Delete.

Menu Bar

`File`

`New`
creates a new empty model selection list.

`Open`
opens a dialog for selecting one or more existing model selection lists to open. If you select multiple lists, they are all opened at once as a concatenated list. This helps you build large specialized model lists quickly by mixing and matching various existing lists such as the various ARIMA model lists included in SASHELP.FORCAST. By default, the lists you open replace the current model list. Select the “append” radio button if you want to append them to the current model list.

`Open System Default`
opens the default model list supplied with the product.

`Cancel`
exits the window without applying any changes to the current project's model selection list.

`Close`
closes the window and applies any changes made to the project's model selection list.

`Save`
opens a dialog for saving the edited model selection list in a catalog of your choice.

`Save as User Default`
saves your edited model list as a default list for new projects. The location of this saved list is shown on the message line. When you create new projects, the system searches for this model list and uses it if it is found. If it is not found, the system uses the original default model list supplied with the product.

`Edit`

`Reset`
restores the list to its initial state when the window was invoked.

`Add Model`
enables you to add new models to the selection list. You can use the Smoothing Model Specification window, the ARIMA Model Specification window, the Factored ARIMA Model Specification window, or the Custom Model Specification window.

`Edit Selected`
opens the appropriate model specification window for changing the attributes of the highlighted model and adding the modified model to the selection list. The original model is not deleted.

`Move Selected`
enables you to reorder the models in the list. Select one or more models, then select Move Selected from the menu or toolbar. A note appears on the message line: "Select the row after which the selected models are to be moved." Then select any unhighlighted row in the table. The selected models will be moved after this row.

`Delete`
deletes any highlighted models from the list. This item is not available if no models are selected.

`Set Regressor Names`
opens a dialog for changing all occurrences of a given regressor name in the models of the current model selection list to a name that you specify.

`Select All`
selects all models in the list.

`Clear Selections`
deselects all models in the list.

`Select All for Autofit`
checks the autofit check boxes of all models in the list.

`Clear Autofit Selections`
deselects the autofit check boxes of all models in the list.

Mouse Button Actions

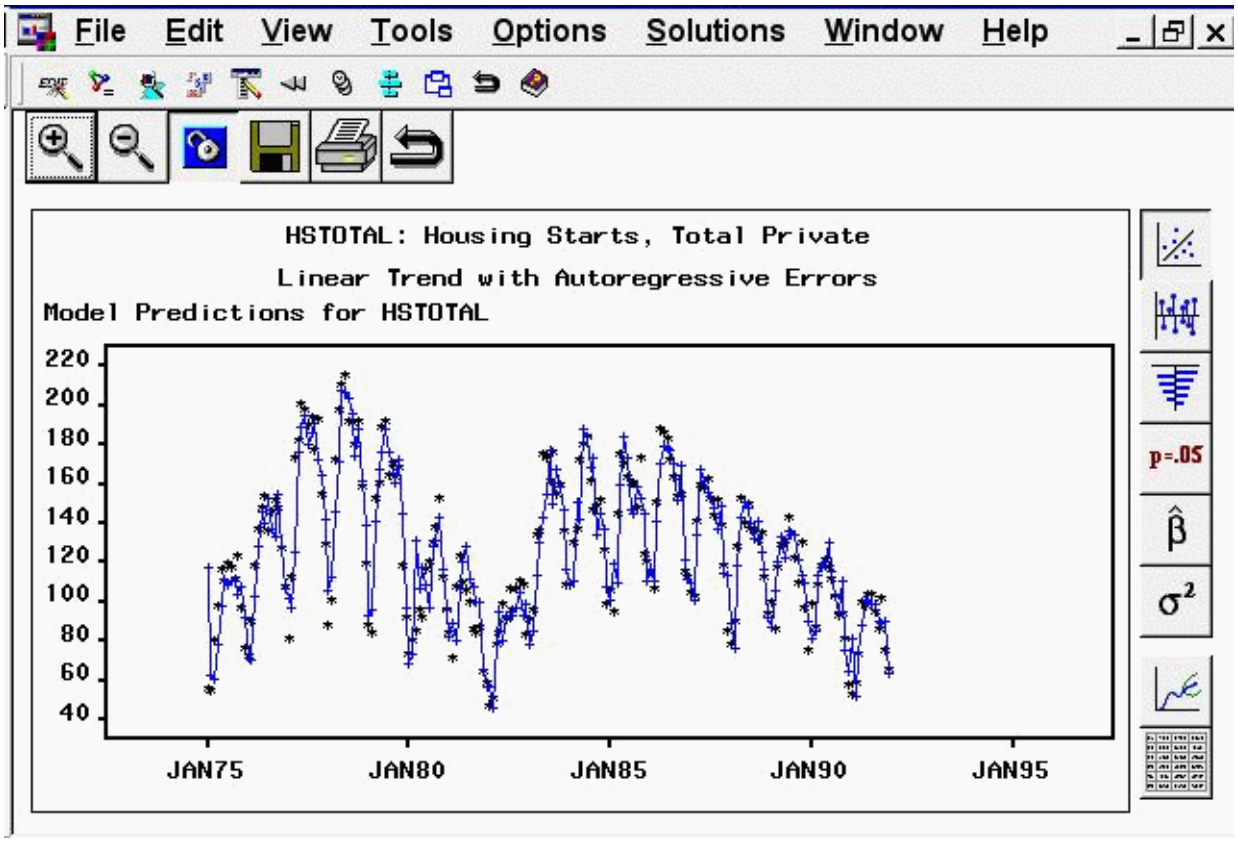
Clicking any model description in the table selects (highlights) that model. Clicking the same model again deselects it. Multiple selections are allowed.

Clicking the auto fit check box in any row toggles the associated model's eligibility for use in automatic model fitting.

Right-clicking the right mouse button opens a pop-up menu.

Model Viewer Window

This resizable window provides plots and tables of actual values, model predictions, forecasts, and related statistics. The various plots and tables available are referred to as *views*. The section "View Selection Icons" on page 3839 explains how to change the view.



You can access Model Viewer in a number of ways, including the View Model Graphically icon of the Develop Models and Model List windows, the Graph button of the Automatic Model Fitting Results window, and the Model item under the View menu in the Manage Forecasting Project window. In addition, you can go directly to a selected view in the Model Viewer window by selecting Model Predictions, Prediction Errors, Statistics of Fit, Prediction Error Autocorrelations, Prediction Error Tests, Parameter Estimates, Forecast Graph, or Forecast Table from the View menu or corresponding toolbar icon or pop-up menu item in the Develop Models, Model List, or Automatic Model Fitting Results windows.

The state of the Model Viewer window is controlled by the current model and the currently selected view. You can resize this window, and you can use other windows without closing the Model Viewer window. By default, the Model Viewer window is automatically updated to display the new model when you switch to working with another model (that is, when you highlight a different model). You can unlink the Model Viewer window from the current model selection by selecting the Link/Unlink icon from the window's horizontal toolbar. See "Link/Unlink" in the section "Toolbar Icons" on page 3839.

For more information, see the section "Model Viewer" on page 3654.

Toolbar Icons

The Model Viewer window contains a horizontal row of icons called the Toolbar. Corresponding menu items appear under various menus. The function of each icon is explained in the following list.

`Zoom in`
: In the Model Predictions, Prediction Errors, and Forecast Graph views, the Zoom In action changes the mouse cursor into cross hairs that you can use with the left mouse button to define a region of the graph to zoom in on. In the Prediction Error Autocorrelations and Prediction Error Tests views, Zoom In reduces the number of lags displayed.

`Zoom out`
: reverses the previous Zoom In action.

`Link/Unlink viewer`
: disconnects or connects the Model Viewer window to the model table (Develop Models window, Model List window, or Automatic Model Fitting Results window). When the viewer is linked, selecting another model in the model table causes the model viewer to be updated to show the selected model. When the Viewer is unlinked, selecting another model does not affect the viewer. This feature is useful for comparing two or more models graphically. You can display a model of interest in the Model Viewer, unlink it, then select another model and open another Model Viewer window for that model. Position the viewer windows side by side for convenient comparisons of models, or use the Next Viewer icon or F12 function key to switch between them.

`Save`
: saves the contents of the Model Viewer window. By default, an HTML page is created. This enables you to display graphs and tables by using the Results Viewer or publish them on the Web or your intranet. See also "Save Graph As" and "Save Data As" under "Menu Bar" below.

`Print`
: prints the contents of the viewer window.

`Close`
: closes the Model Viewer window and returns to the window from which it was invoked.

View Selection Icons

At the right hand side of the Model Viewer window is a vertical toolbar to select the view—that is, the kind of plot or table that the viewer displays. Corresponding menu items appear under View in the menu bar. The function of each icon is explained in the following list.

`Model Predictions`
: displays a plot of actual series values and model predictions over time. Click individual points in the graph to get a display of the type (actual or predicted), ID value, and data value in the upper right corner of the window.

`Prediction Errors`
: displays a plot of model prediction errors (residuals) over time. Click individual points in the graph to get a display of the prediction error value in the upper right corner of the window.

`Prediction Error Autocorrelations`
: displays horizontal bar charts of the sample autocorrelation, partial autocorrelation, and inverse autocorrelation functions for the model prediction errors. Overlaid line plots represent confidence limits computed at plus and minus two standard errors. Click any of the bars to display its value.

`Prediction Error Tests`
: displays horizontal bar charts that represent results of white noise and stationarity tests on the model prediction errors. The first bar chart shows the significance probability of the Ljung-Box chi-square statistic computed on autocorrelations up to the given lag. Longer bars favor rejection of the null hypothesis that the series is white noise. Click any of the bars to display an interpretation.

  The second bar chart shows tests of stationarity of the model prediction errors, where longer bars favor the conclusion that the series is stationary. Each bar displays the significance probability of the augmented Dickey-Fuller unit root test to the given autoregressive lag. Long bars represent higher levels of significance against the null hypothesis that the series contains a unit root. For seasonal data, a third bar chart appears for seasonal root tests. Click on any of the bars to display an interpretation.

`Parameter Estimates`
: displays a table showing model parameter estimates along with standard errors and *t* tests for the null hypothesis that the parameter is zero.

`Statistics of Fit`
: displays a table of statistics of fit for the selected model. The set of statistics shown can be changed by using the Statistics of Fit item under Options in the menu bar.

`Forecast Graph`
: displays a plot of actual and predicted values for the series data range, followed by a horizontal reference line and forecasted values with confidence limits. Click individual points in the graph to get a display of the type, date/time, and value of the data point in the upper right corner of the window.

`Forecast Table`
: displays a data table with columns containing the date/time, actual, predicted, error (residual), lower confidence limit, and upper confidence limit values, together with any predictor series.

Menu Bar

`File`

`Save Graph`
: saves the plot displayed in viewer window as a SAS/GRAPH grseg catalog entry. When the current view is a table, this menu item is not available. See also "Save" in the section "Toolbar Icons" on page 3839. If a graphics catalog entry name has not already been specified, this action functions like "Save Graph As."

`Save Graph As`
saves the current graph as a SAS/GRAPH grseg catalog entry in a SAS catalog that you specify and/or as an Output Delivery System (ODS) object. By default, an HTML page is created, with the graph embedded as a gif image.

`Save Data`
saves the data displayed in the viewer window in a SAS data set, where applicable.

`Save Data As`
saves the data in a SAS data set that you specify and/or as an Output Delivery System (ODS) object. By default, an HTML page is created, with the data displayed as a table.

`Import Data`
is available if you license SAS/Access software. It opens an Import Wizard, which you can use to import your data from an external spreadsheet or data base to a SAS data set for use in the Time Series Forecasting System.

`Export Data`
is available if you license SAS/Access software. It opens an Export Wizard, which you can use to export a SAS data set, such as a forecast data set created with the Time Series Forecasting System, to an external spreadsheet or data base.

`Print Graph`
prints the contents of the viewer window if the current view is a graph. This is the same as the Print toolbar icon. If the current view is a table, this menu item is not available.

`Print Data`
prints the data displayed in the viewer window, where applicable.

`Print Setup`
opens the Print Setup window, which allows you to access your operating system print setup.

`Print Preview`
opens a preview window to show how your plots will appear when printed.

`Close`
closes the Model Viewer window and returns to the window from which it was invoked.

`Edit`

`Edit Model`
enables you to modify the specification of the current model and to fit the modified model, which is then displayed in the viewer.

`Refit Model`
refits the current model by using data within the current fit range. This action also causes the ranges to be reset if the data range has changed.

`Reevaluate Model`
reevaluates the current model by using data within the current evaluation range. This action also causes the ranges to be reset if the data range has changed.

`View`
See "View Selection Icons" on page 3839. It describes each of the items available under "View," except "Zoom Way Out."

Zoom Way Out
: zooms the plot out as far as it will go, undoing all prior zoom in operations.

Tools

Link Viewer
: See “Link/Unlink” in the section “Toolbar Icons” on page 3839.

Options

Time Ranges
: opens the Time Ranges Specification window to enable you to change the period of fit, period of evaluation, or forecast horizon to be applied to subsequently fit models.

Statistics of Fit
: opens the Statistics of Fit Selection window, which presents a list of statistics that the system can display. Use this action to customize the list of statistics shown in the statistics of fit table and available for selection in the Model Selection Criterion menu.

Forecast Options
: opens the Forecast Options window, which enables you to control the widths of forecast confidence limits and control the kind of predicted values computed for models that include series transformations.

Residual Plot Options
: Provides a choice of four methods of computing prediction errors for models which include a data transformation.

  Prediction Errors
  : computes the difference between the transformed series actual values and model predictions.

  Normalized Prediction Errors
  : computes prediction errors in normalized form.

  Model Residuals
  : computes the difference between the untransformed series values and the untransformed model predictions.

  Normalized Model Residuals
  : computes model residuals in normalized form.

Number of Lags
: opens a window to enable you to specify the number of lags shown in the Prediction Error Autocorrelations and Prediction Error Tests views. You can also use the Zoom In and Zoom Out actions to control the number of lags displayed.

Correlation Probabilities
: controls whether the bar charts in the Prediction Error Autocorrelations view represent significance probabilities or values of the correlation coefficient. A check mark or filled check box next to this item indicates that significance probabilities are displayed. In each case the bar graph horizontal axis label changes accordingly.

`Include Interventions`
: controls whether intervention effects defined for the current series are automatically added as predictors to the models considered by the automatic selection process. A check mark or filled check box next to this item indicates that the option is turned on.

`Print Audit Trail`
: prints to the SAS log information about the models fit by the system. A check mark or filled check box next to this item indicates that the audit option is turned on.

`Show Source Statements`
: controls whether SAS statements submitted by the forecasting system are printed in the SAS log. When the Show Source Statements option is selected, the system sets the SAS system option SOURCE before submitting SAS statements; otherwise, the system uses the NOSOURCE option. Note that only some of the functions performed by the forecasting system are accomplished by submitting SAS statements. A check mark or filled check box next to this item indicates that the option is turned on.

Mouse Button Actions

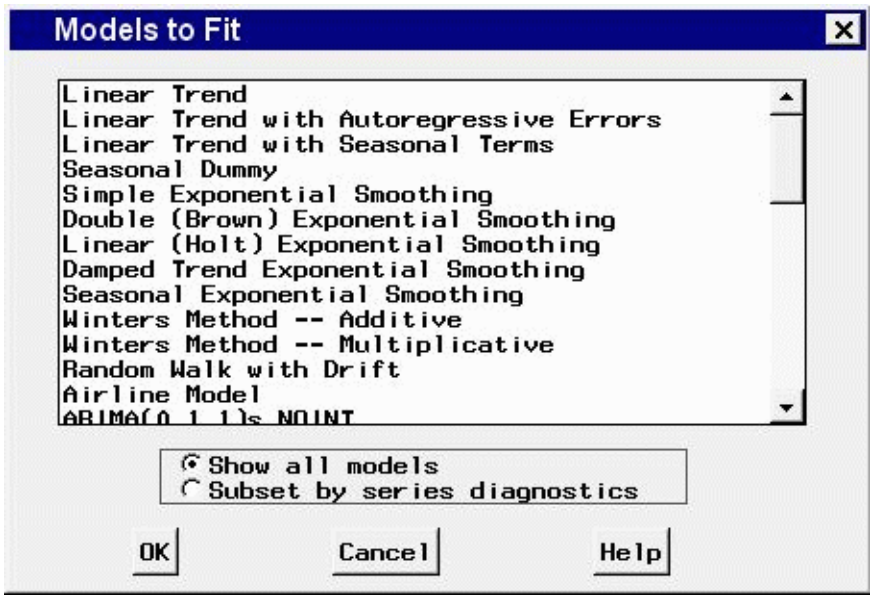
You can examine the data values of individual points in the Model Predictions, Model Prediction Errors, and Forecast Graph views of the Model Viewer by clicking the point. The date/time and data values as well as the type (actual, predicted, and so forth) are displayed in a box that appears in the upper right corner of the Viewer window. Click the mouse elsewhere or select any action to dismiss the data box.

Similarly, you can display values in the Prediction Error Autocorrelations view by clicking any of the bars. Clicking bars in the Prediction Error Tests view displays a Recommendation for Current View window which explains the test represented by the bar.

When you select the Zoom In action in the Predicted Values, Model Prediction Errors, and Forecasted Values views, you can use the mouse to define a region of the graph to zoom. Position the mouse cursor at one corner of the region, press the left mouse button, and move the mouse cursor to the opposite corner of the region while holding the left mouse button down. When you release the mouse button, the plot is redrawn to show an expanded view of the data within the region you selected.

Models to Fit Window

Use the Models to Fit window to fit models by choosing them from the current model selection list. Access it by using “Fit Models from List” under the Fit Model submenu of the Edit menu in the Develop Models window, or the pop-up menu that appears when you click an empty area of the model table in the Develop Models window. If you want to alter the list of models that appears here, use the Model Selection List editor window.



To select a model to be fit, use the left mouse button. To select more than one model to fit, drag with the mouse, or select the first model, then press the shift key while selecting the last model. For noncontiguous selections, press the control key while selecting with the mouse. To begin fitting the models, double-click the last selection or select the OK button.

If series diagnostics have been performed, the radio box is available. If the Subset by series diagnostics radio button is selected, only those models in the selection list that fit the diagnostic criteria will be shown for selection. If you want to choose models that do not fit the diagnostic criteria, select the Show all models button.

Controls and Fields

`Show all models`
: when selected, lists all available models, regardless of the setting of the series diagnostics options.

`Subset by series diagnostics`
: when selected, lists only the available models that are consistent with the series diagnostics options.

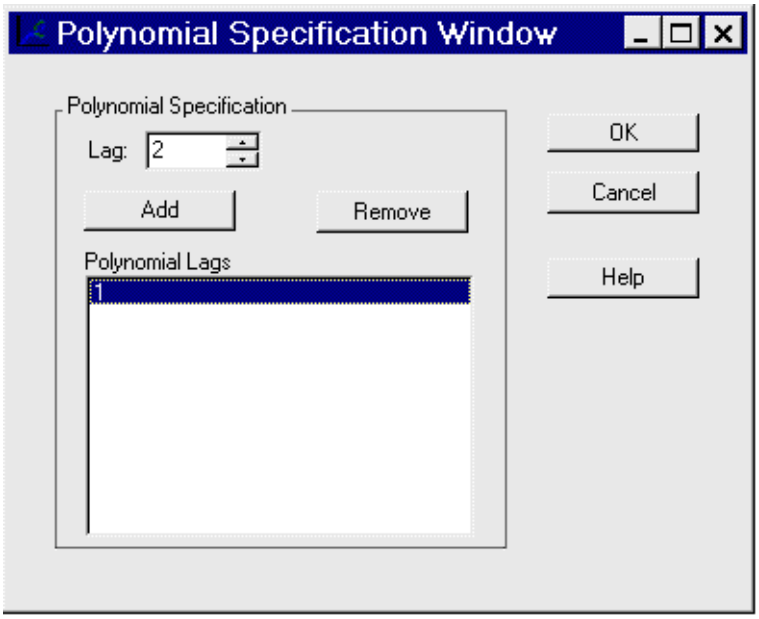
`OK`
: closes the Models to Fit window and fits the selected models.

`Cancel`
: closes the window without fitting any models. Any selections you made are lost.

Polynomial Specification Window

Use the Polynomial Specification window to add a polynomial to an ARIMA model. The set of lags defined here become a polynomial factor, denoted by a list of lags in parentheses, when you select “OK.” If you accessed this window from the AR Polynomial Specification window, then it is added to the autoregressive

part of the model. If you accessed it from the MA Polynomial Specification window, it is added to the moving-average part of the model.



Controls and Fields

`Lag`
: specifies a lag value to add to the list. Type in a positive integer or select one by clicking the spin box arrows.

`Add`
: adds the value in the Lag spin box to the list of polynomial lags. Duplicate values are not allowed.

`Remove`
: deletes a selected lag from the list of polynomial lags.

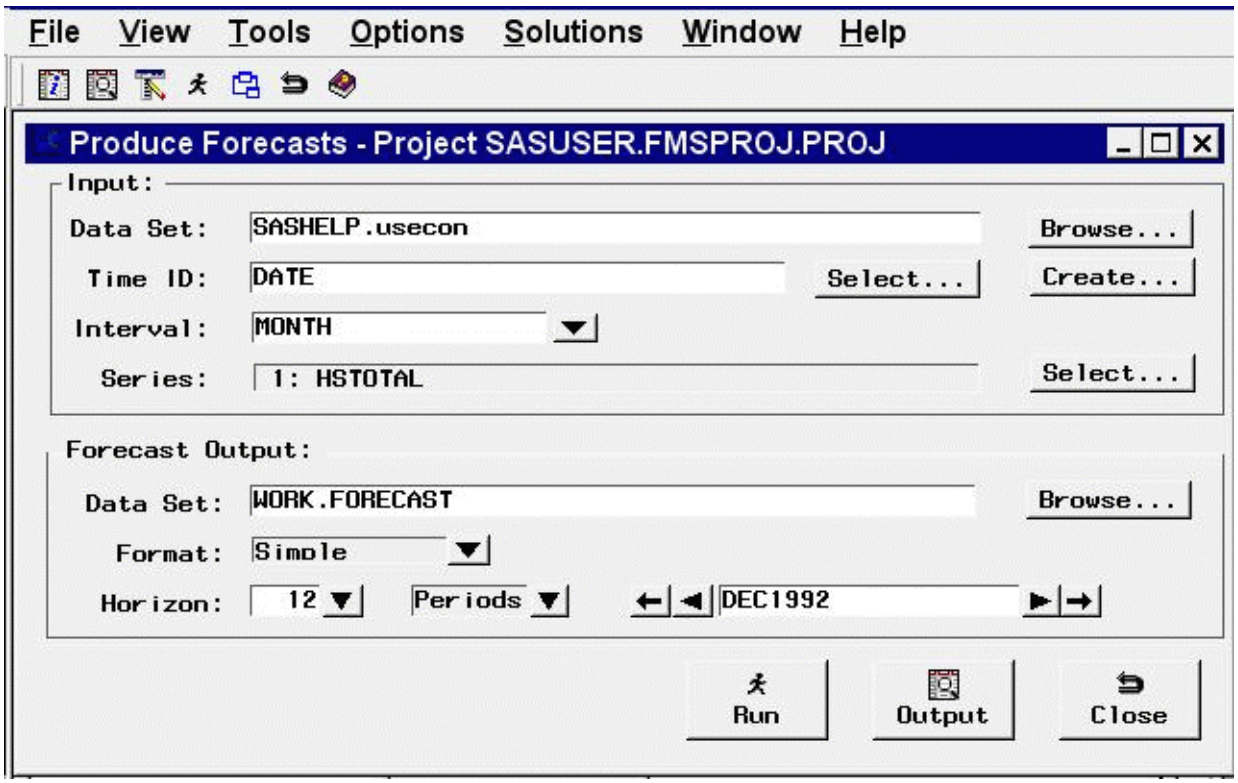
`Polynomial Lags`
: is a list of unique integers that represent lags to be added to the model.

`OK`
: closes the window and returns the specified polynomial to the AR or MA polynomial specification window.

`Cancel`
: closes the window and discards any polynomial lags added to the list.

Produce Forecasts Window

Use the Produce Forecasts window to produce forecasts for the series in the current input data set for which you have fit forecasting models. Access it by using the Produce Forecasts button of the Time Series Forecasting window.



Controls and Fields

Input `Data Set`
: is the name of the current input data set. To specify the input data set, you can type a one-level or two-level SAS data set name in this field or select the Browse button to the right of the field.

Input data set `Browse` button
: opens the Data Set Selection window to enable you to select the input data set.

`Time ID`
: is the name of the time ID variable for the input data set. To specify this variable, you can type the ID variable name in this field or use the Select button.

Time ID `Select` button
: opens the Time ID Variable Specification window.

`Create` button
: opens a menu of choices of methods for creating a time ID variable for the input data set. Use this feature if the input data set does not already contain a valid time ID variable.

`Interval`
: is the time interval between observations (data frequency) in the current input data set. If the interval is not automatically filled in by the system, you can type in an interval name here, or select one from the pop-up list.

`Series`
: indicates the number and names of time series variables for which forecasts will be produced.

Series `Select` button
: opens the Series to Process window to let you select the series for which you want to produce forecasts.

Forecast Output `Data Set`
: is the name of the output data set that will contain the forecasts. Type the name of the output data set in this field or click the Browse button.

Forecast Output `Browse` button
: opens a dialog to let you locate an existing data set to which to save the forecasts.

`Format`
: enables you to select one of three formats for the forecast data set:

  `Simple`
  : specifies the simple format for the output data set. The data set contains the time ID variable and the forecast variables and contains one observation per time period. Observations for earlier time periods contain actual values copied from the input data set; later observations contain the forecasts.

  `Interleaved`
  : specifies the interleaved format for the output data set. The data set contains the time ID variable, the variable TYPE, and the forecast variables. There are several observations per time period, with the meaning of each observation identified by the TYPE variable.

  `Concatenated`
  : specifies the concatenated format for the output data set. The data set contains the variable SERIES, the time ID variable, and the variables ACTUAL, PREDICT, ERROR, LOWER, and UPPER. There is one observation per time period per forecast series. The variable SERIES contains the name of the forecast series, and the data set is sorted by SERIES and DATE.

`Horizon`
: is the number of periods or years to forecast beyond the end of the input data range. To specify the forecast horizon, you can type a value in this field or select one from the pop-up list.

`Horizon periods`
: selects the units to apply to the horizon. By default, the horizon value represents number of periods. For example, if the interval is month, the horizon represents the number of months to forecast. Depending on the interval, you can also select weeks or years, so that the horizon is measured in those units.

`Horizon date`
: is the ending date of the forecast horizon. You can type in a date that uses a form recognized by a SAS date informat, or you can increment or decrement the date shown by using the left and right arrows. The outer arrows change the date by a larger amount than the inner arrows. The date field and the horizon field reset each other, so you can use either one to specify the forecasting horizon.

`Run` button
: produces forecasts for the selected series and stores the forecasts in the specified output SAS data set.

`Output` button
: opens a Viewtable window to display the output data set. This button becomes available once the forecasts have been written to the data set.

`Close` button
: closes the Produce Forecasts window and returns to the Time Series Forecasting window.

Menu Bar

File

Import Data
: is available if you license SAS/Access software. It opens an Import Wizard, which you can use to import your data from an external spreadsheet or data base to a SAS data set for use in the Time Series Forecasting System.

Export Data
: is available if you license SAS/Access software. It opens an Export Wizard, which you can use to export a SAS data set, such as a forecast data set created with the Time Series Forecasting System, to an external spreadsheet or data base.

Print Setup
: opens the Print Setup window, which allows you to access your operating system print setup.

Close
: closes the Produce Forecasts window and returns to the Time Series Forecasting window.

View

Input Data Set
: opens a Viewtable window to browse the current input data set.

Output Data Set
: opens a Viewtable window to browse the output data set. This is the same as the Output button.

Tools

Produce Forecasts
: produces forecasts for the selected series and stores the forecasts in the specified output SAS data set. This is the same as the Run button.

Options

Default Time Ranges
: opens the Default Time Ranges window to enable you to control how the system sets the time ranges when new series are selected.

Model Selection List
: opens the Model Selection List editor window. Use this to edit the set of forecasting models considered by the automatic model selection process and displayed by the Models to Fit window.

Model Selection Criterion
: opens the Model Selection Criterion window, which presents a list of goodness-of-fit statistics and enables you to select the fit statistic that is displayed in the table and used by the automatic model selection process to determine the best fitting model.

`Statistics of Fit`
: opens the Statistics of Fit Selection window, which presents a list of statistics that the system can display. Use this action to customize the list of statistics shown in the Statistics of Fit table and available for selection in the Model Selection Criterion window.

`Forecast Options`
: opens the Forecast Options window, which enables you to control the widths of forecast confidence limits and control the kind of predicted values computed for models that include series transformations.

`Forecast Data Set`
: enables you to select one of three formats for the forecast data set. See `Format`, which is described previously in this section.

`Alignment of Dates`

`Beginning`
: aligns dates that the system generates to identify forecast observations in output data sets to the beginning of the time intervals.

`Middle`
: aligns dates that the system generates to identify forecast observations in output data sets to the midpoints of the time intervals.

`End`
: aligns dates that the system generates to identify forecast observations in output data sets to the end of the time intervals.

`Automatic Fit`
: opens the Automatic Model Selection Options window, which enables you to control the number of models retained by the automatic model selection process and whether the models considered for automatic selection are subset according to the series diagnostics.

`Set Toolbar Type`

`Image Only`
: displays the toolbar items as icons without text.

`Label Only`
: displays the toolbar items as text without icon images.

`Both`
: displays the toolbar items as both text and icon images.

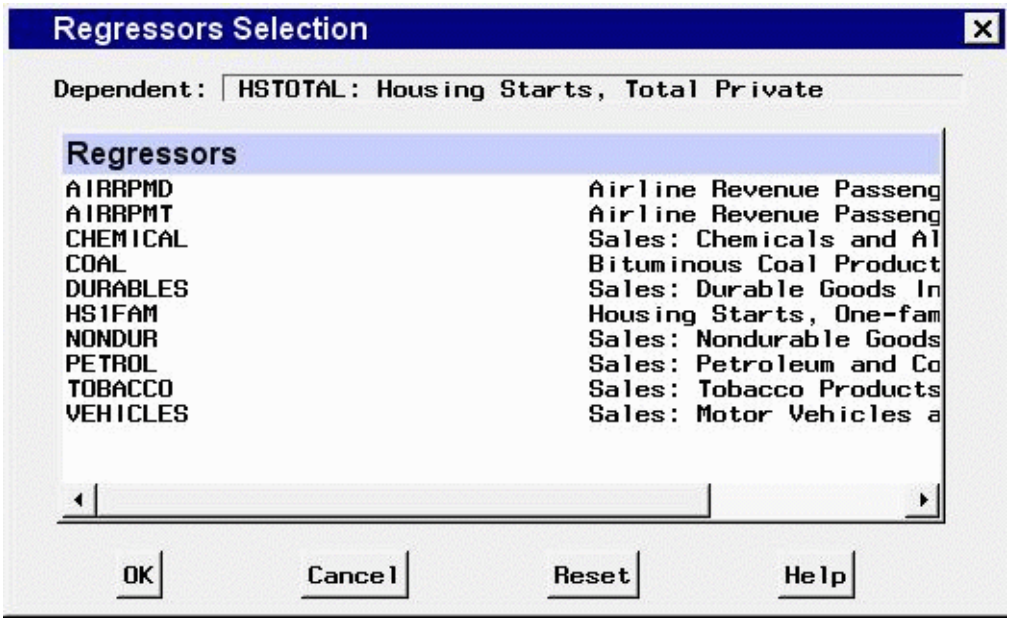
`Include Interventions`
: controls whether intervention effects defined for the current series are automatically added as predictors to the models considered by the automatic selection process. A check mark or filled check box next to this item indicates that the option is turned on.

`Print Audit Trail`
: prints to the SAS log information about the models fit by the system. A check mark or filled check box next to this item indicates that the audit option is turned on.

`Show Source Statements`
: controls whether SAS statements submitted by the forecasting system are printed in the SAS log. When the Show Source Statements option is selected, the system sets the SAS system option SOURCE before submitting SAS statements; otherwise, the system uses the NOSOURCE option. Note that only some of the functions performed by the forecasting system are accomplished by submitting SAS statements. A check mark or filled check box next to this item indicates that the option is turned on.

Regressors Selection Window

Use the Regressors Selection window to select one or more time series variables in the input data set to include as regressors in the forecasting model to predict the dependent series. Access it from the pop-up menu that appears when you select the Add button of the ARIMA Model Specification window or Custom Model Specification window.



Controls and Fields

`Dependent`
: is the name and variable label of the current series.

`Regressors`
: is a table listing the names and labels of the variables in the input data set available for selection as regressors. The variables that you select are highlighted. Selecting a highlighted row again deselects that variable.

`OK`
: closes the Regressors Selection window and adds the selected variables as regressors in the model.

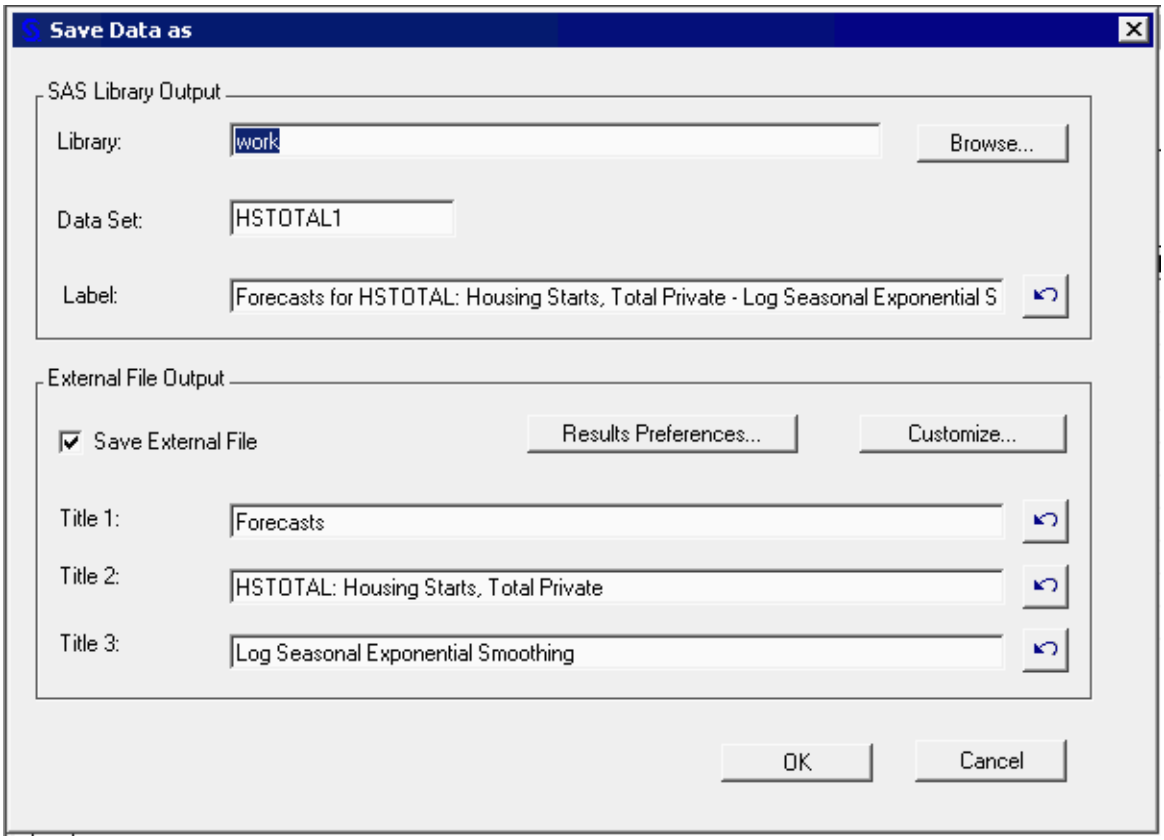
`Cancel`
: closes the window without adding any regressors. Any selections you made are lost.

`Reset`
resets all options to their initial values upon entry to the window.

Save Data As

Use Save Data As from the Time Series Viewer Window or the Model Viewer Window to save data displayed in a table to a SAS data set or external file.

Use Save Forecast As from the Develop Models Window to save forecasts and related data including the series name, model, and interval. It supports append mode, enabling you to accumulate the forecasts of multiple series in a single data set.



To save your data in a SAS data set, provide a library name or assign one by using the Browse button, then provide a data set name or accept the default. Enter a descriptive label for the data set in the Label field. Click OK to save the data set. If you specify an existing data set, it will be overwritten, except in the case of Save Forecast As.

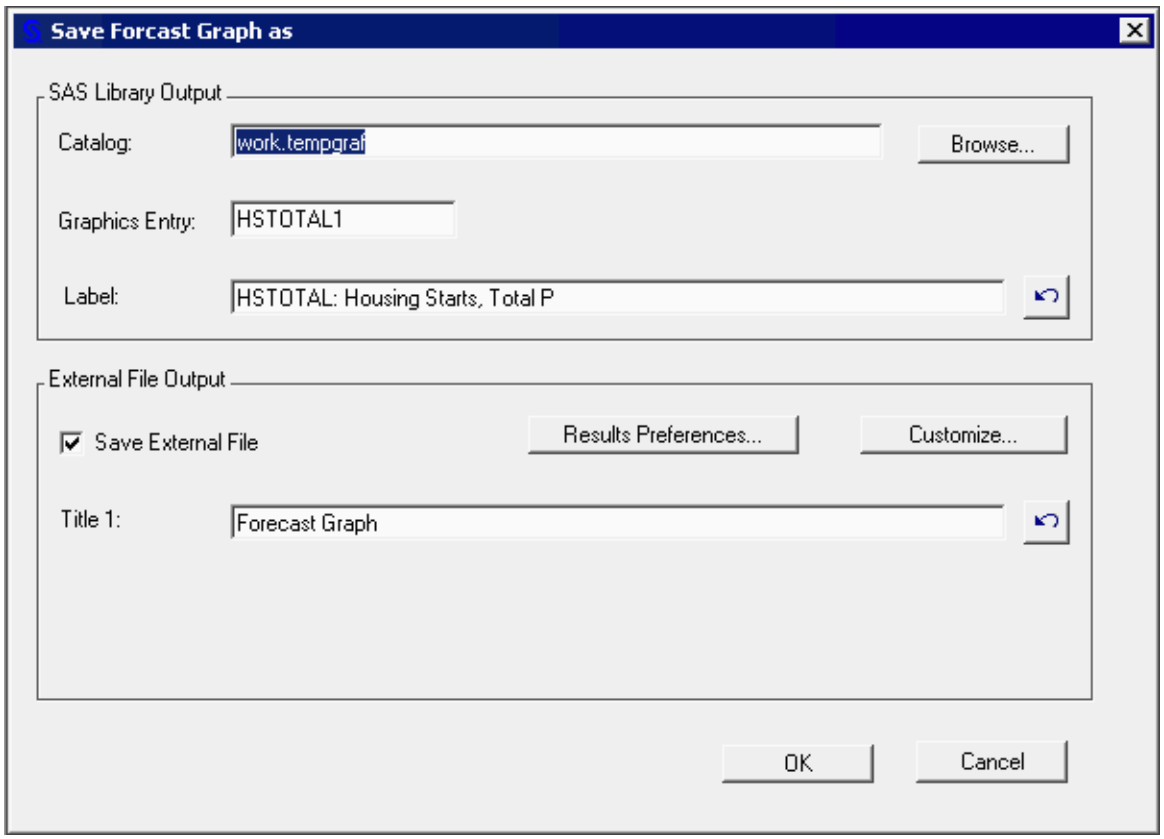
External file output takes advantage of the Output Delivery System (ODS) and is designed primarily for creating HTML tables for Web reporting. You can build a set of Web pages quickly and use the ODS Results window to view and organize them. To use this feature, check Save External File in the External File Output box. To set ODS options, click Results Preferences, then select the Results tab in the Preferences dialog.

If you have previously saved data of the current type, the system remembers your previous labels and titles. To reuse them, click the arrow button to the right of each of these window fields.

Use the Customize button if you need to specify the name of a custom macro that contains ODS statements. The default macro simply runs the PRINT procedure. A custom macro can be used to add PRINT procedure and/or ODS statements to customize the type and organization of output files produced.

Save Graph As

Use Save Graph As from the Time Series Viewer Window or the Model Viewer Window to save any of the graphs in a catalog or external file.



To save your graph as a grseg catalog entry, enter a two level name for the catalog or select Browse to open an Open dialog. Use it to select an existing library or assign a new one and then select a catalog to contain the graph. Click the Open button to open the catalog and close the dialog. Then enter a graphics entry name (eight characters or less) and a label or accept the defaults and click the OK button to save the graph.

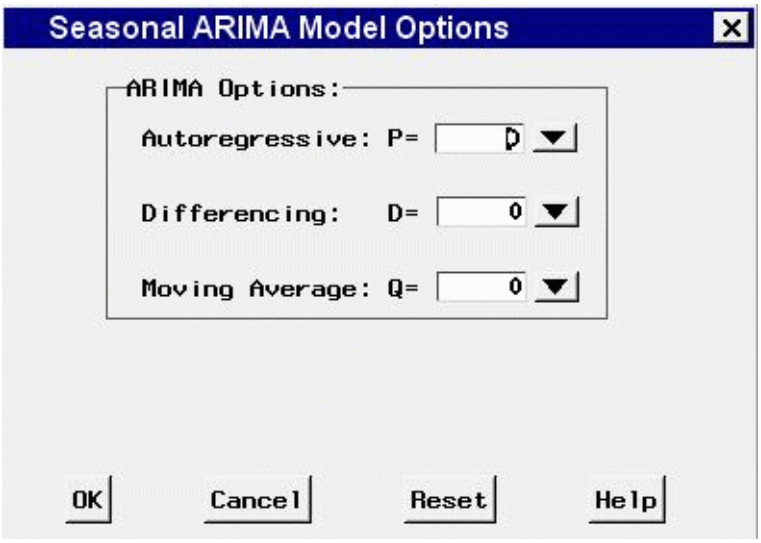
External file output takes advantage of the Output Delivery System (ODS) and is designed primarily for creating gif images and HTML for Web reporting. You can build a set of Web pages that contain graphs and use the Results window to view and organize them. To use this feature, check Save External File in the External File Output box. To set ODS options, click Results Preferences, then select the Results tab in the Preferences dialog.

If you have previously saved graphs of the current type, the system remembers your previous labels and titles. To reuse them, click the arrow button to the right of each of these window fields.

Use the Customize button if you need to specify the name of a custom macro that contains ODS statements. The default macro simply runs the GREPLAY procedure. Users familiar with ODS might want to add statements to the macro to customize the type and organization of output files produced.

Seasonal ARIMA Model Options Window

Use the Seasonal ARIMA Model Options window to specify the autoregressive, differencing, and moving-average orders for the seasonal part of a model defined by using the Custom Model Specification window. Access it by selecting "Seasonal ARIMA. . . " from the Seasonal Model combo box of that window.



Controls and Fields

`ARIMA Options`

Use these combo boxes to specify the orders of the ARIMA model. You can either type in a value or click the combo box arrow to select from a pop-up list.

`Autoregressive`

defines the order of the seasonal autoregressive part of the model.

`Differencing`

defines the order of seasonal differencing.

`Moving Average`

defines the order of the seasonal moving-average term.

`OK`

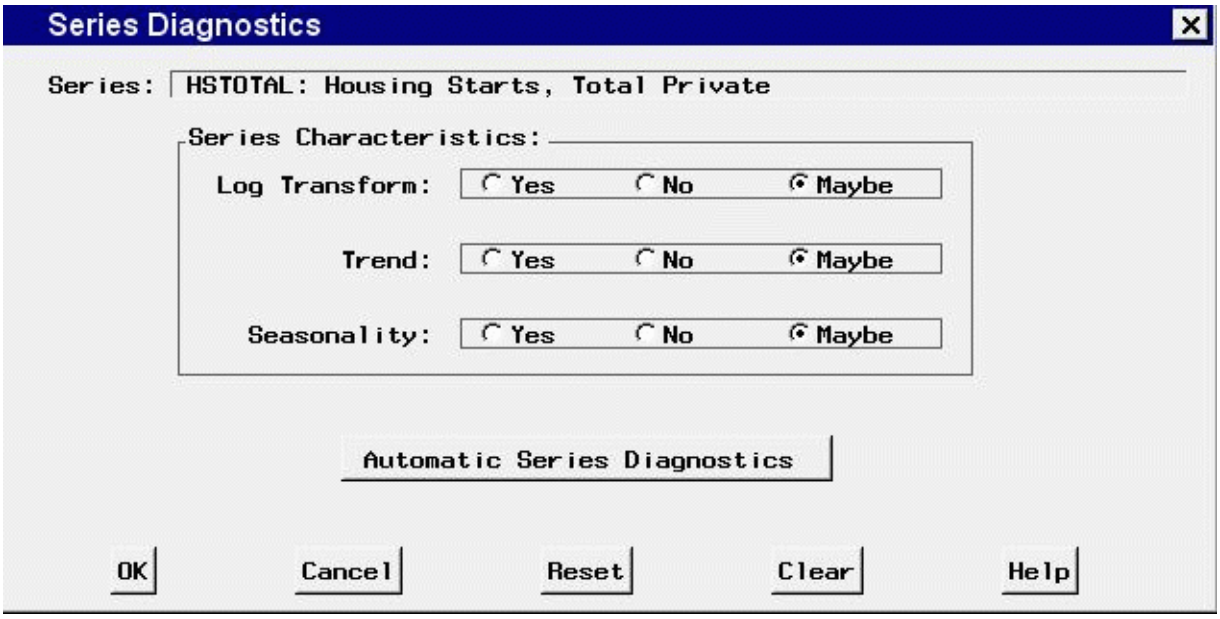
closes the Seasonal ARIMA Model Options window and returns to the Custom Model Specification window.

`Cancel`
: closes the Seasonal ARIMA Model Options window and returns to the Custom Model Specification window, discarding any changes made.

`Reset`
: resets all options to their initial values upon entry to the window.

Series Diagnostics Window

Use the Series Diagnostics window to set options to limit the kinds of forecasting models considered for the series according to series characteristics. Access it by selecting "Diagnose Series" from the Tools menu in the Develop Models, Manage Project, and Time Series Viewer window menu bars. You can let the system diagnose the series characteristics automatically or you can specify series characteristics according to your judgment by using the radio buttons.



For each of the options Log Transform, Trend, and Seasonality, the value "Yes" means that only models appropriate for series with that characteristic should be considered. The value "No" means that only models appropriate for series without that characteristic should be considered. The value "Maybe" means that models should be considered without regard for that characteristic.

Controls and Fields

`Series`
is the name and variable label of the current series.

`Series Characteristics`

`Log Transform`
specifies whether forecasting models with or without a logarithmic series transformation are appropriate for the series.

`Trend`
specifies whether forecasting models with or without a trend component are appropriate for the series.

`Seasonality`
specifies whether forecasting models with or without a seasonal component are appropriate for the series.

`Automatic Series Diagnostics`
performs the automatic series diagnostic process. The options Log Transform, Trend, and Seasonality are set according to the results of statistical tests.

`OK`
closes the Series Diagnostics window.

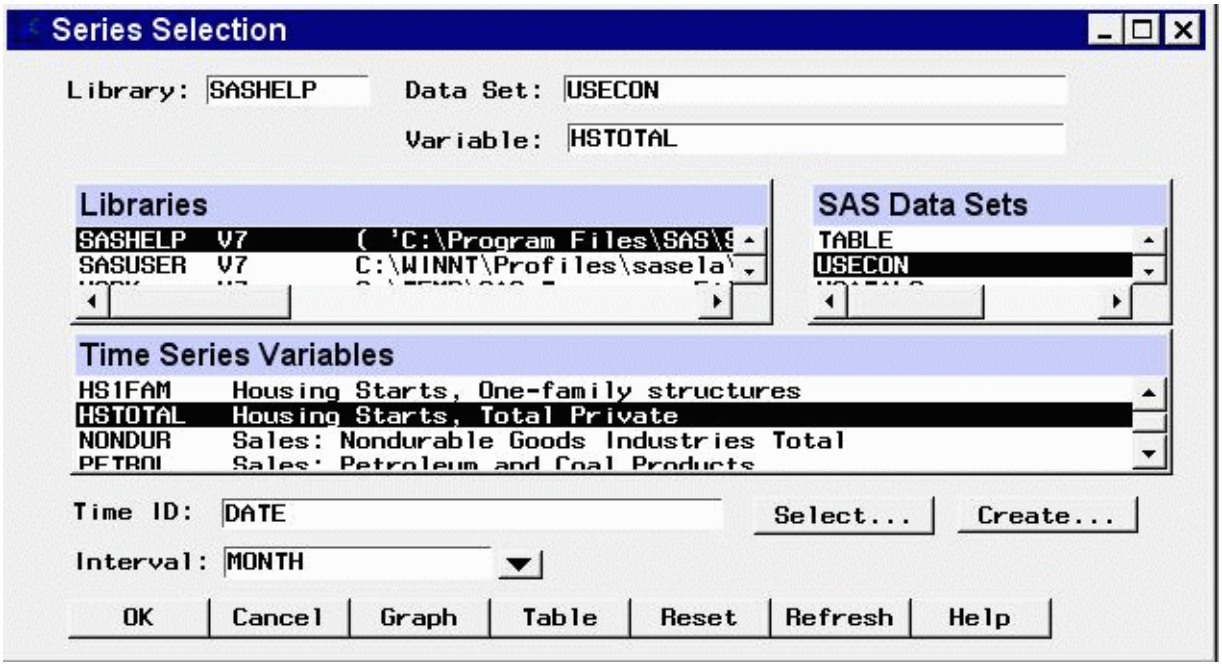
`Cancel`
closes the Series Diagnostics window without changing the series diagnostics options. Any options you specified are lost.

`Reset`
resets all options to their initial values upon entry to the Series Diagnostics window.

`Clear`
resets all options to their default values.

Series Selection Window

Use this resizable window to select a time series variable by specifying a library, a SAS data set or view, and a variable. These selections can be made by typing, by selecting from lists, or by a combination of the two. In addition, you can control the time ID variable and time interval, and you can browse the data set or view plots of the series from this window.



This window appears automatically when you select the View Series Graphically or Develop Models buttons in the Time Series Forecasting window and no series has been selected, and when you open the Time Series Viewer as a standalone tool. It is also invoked by using the Browse button in the Develop Models window.

The system requires that series names be unique for each frequency (interval) within the forecasting project. If you select a series from the current input data set that already exists in the project with the same interval but a different input data set name, the system warns you and gives you the option to cancel the selection, to refit all models associated with the series by using the data from the current input data set, to delete the models for the series, or to inherit the existing models.

Controls and Fields

`Library`
: is a SAS libname assigned within the current SAS session. If you know the libname associated with the data set of interest, you can type it in this field and press Return. If it is a valid choice, it will appear in the libraries list and will be highlighted. The SAS Data Sets list will be populated with data sets associated with that libname.

`Data Set`
: is the name of a SAS data set (data file or data view) that resides under the selected libname. If you know the name, you can type it in and press Return. If it is a valid choice, it will appear in the SAS Data Sets list and will be highlighted, and the Time Series Variables list will be populated with the numeric variables in the data set.

`Variable`
: is the name of a numeric variable contained in the selected data set. You can type the variable name in this field or you can select the variable with the mouse from the Time Series Variables list.

`Time ID`
: is the name of the ID variable for the input data set. To specify the ID variable, you can type the ID variable name in this field or click the Select button.

`Select` button
: opens the Time ID Variable Specification window to let you select an existing variable in the data set as the Time ID.

`Create` button
: opens a menu of methods for creating a time ID variable for the input data set. Use this feature if the data set does not already contain a valid time ID variable.

`Interval`
: is the time interval between observations (data frequency) in the selected data set. If the interval is not automatically filled in by the system, you can type in an interval name or select one from the pop-up list. For more information about intervals, see Chapter 5, "Date Intervals, Formats, and Functions," in this book.

`OK`
: This button is present when you have selected "Develop Models" from the Time Series Forecasting window. It closes the Series Selection window and makes the selected series the current series.

`Close`
: If you have selected the View Series Graphically icon from the Time Series Forecasting window, this button returns you to that window. If you have selected a series, it remains selected as the current series.

  If you are using the Time Series Viewer as a standalone application, this button closes the application.

`Cancel`
: This button is present when you have selected "Develop Models" from the Time Series Forecasting window. It closes the Series Selection window without applying any selections made.

`Reset`
: resets the fields to their initial values at entry to the window.

`Table`
: opens a Viewtable window for browsing the selected data set. This can assist you in locating the variable containing data you are looking for.

`Graph`
: opens the Time Series Viewer window to display the selected time series variable. You can switch to a different series in the Series Selection window without closing the Time Series Viewer window. Position the windows so they are both visible, or use the Next Viewer toolbar icon or F12 function key to switch between windows.

`Refresh`
: updates all fields and lists on the window. If you assign a new libname without exiting the Series Selection window, use the refresh action to update the Libraries list so that it will include the newly assigned libname. Also use the Refresh action to update the variables list if the input data set is changed.

Selection Lists

`Libraries`
: displays a list of currently assigned libnames. You can select a libname by clicking it, which is equivalent to typing its name in the Library field. If you cannot locate the library or directory you are interested in, go to the SAS Explorer window, select "New" from the File menu, then select "Library" and "OK." This opens the New Library dialog window. You also assign a libname by submitting a libname statement from the Editor window. Select the Refresh button to make the new libname available in the libraries list.

`SAS Data Sets`
: displays a list of the SAS data sets (data files or data views) located under the selected libname. You can select one of these by clicking it, which is equivalent to typing its name in the Data Set field.

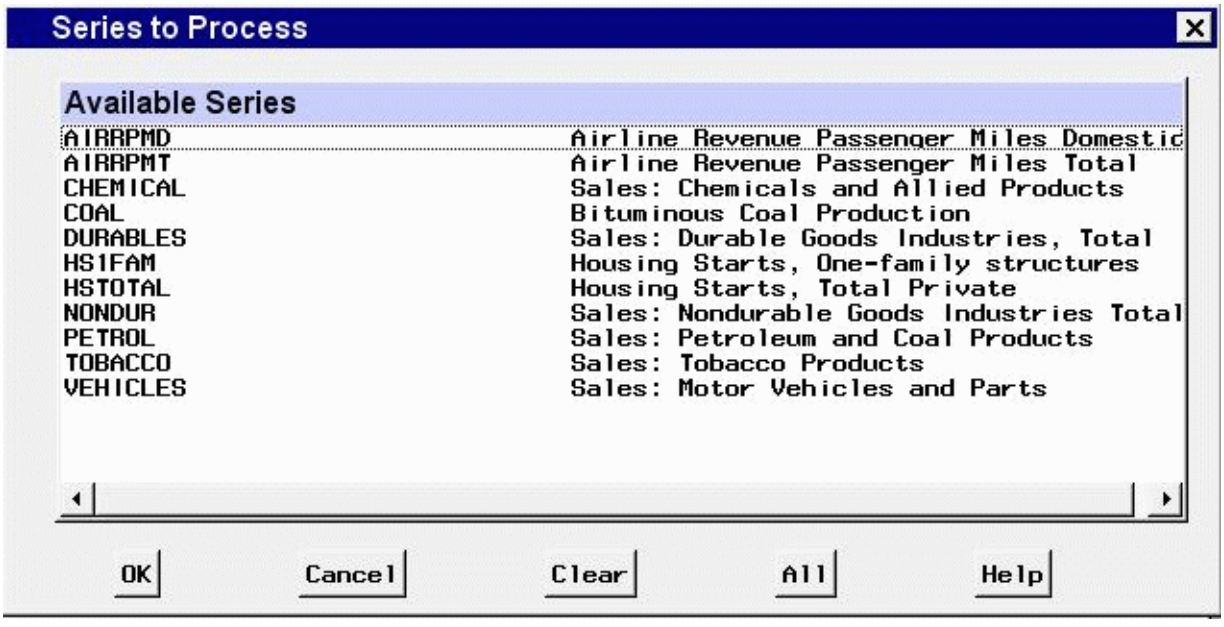
`Time Series Variables`
: displays a list of numeric variables contained within the selected data set. You can select one of these by clicking it, which is equivalent to typing its name in the Variable field. You can double-click a series to select it and exit the window.

Series to Process Window

Use the Series to Process window to select series for model fitting or forecasting. Access it by using the Select button in the Automatic Model Fitting and Produce Forecasts windows. Hold down the shift key or drag with the left mouse button for multiple selections. Use the control key for noncontiguous multiple selections. Once you make selections and select OK, the number of selected series and their names are listed in the Series to Process field of the calling window (with ellipses if not all the names will fit).

When invoked from Automatic Model Fitting, the Series to Process window shows all the numeric variables in the input data set except the time ID variable. These are the series which are currently available for model fitting.

When invoked from Produce Forecasts, the Series to Process window shows all the series in the input data set for which models have been fit. These are the series which are currently available for forecasting.



Controls and Fields

`OK`
: closes the window and applies the series selection(s) which have been made. At least one series must be selected.

`Cancel`
: closes the window, ignoring series selections which have been made, if any.

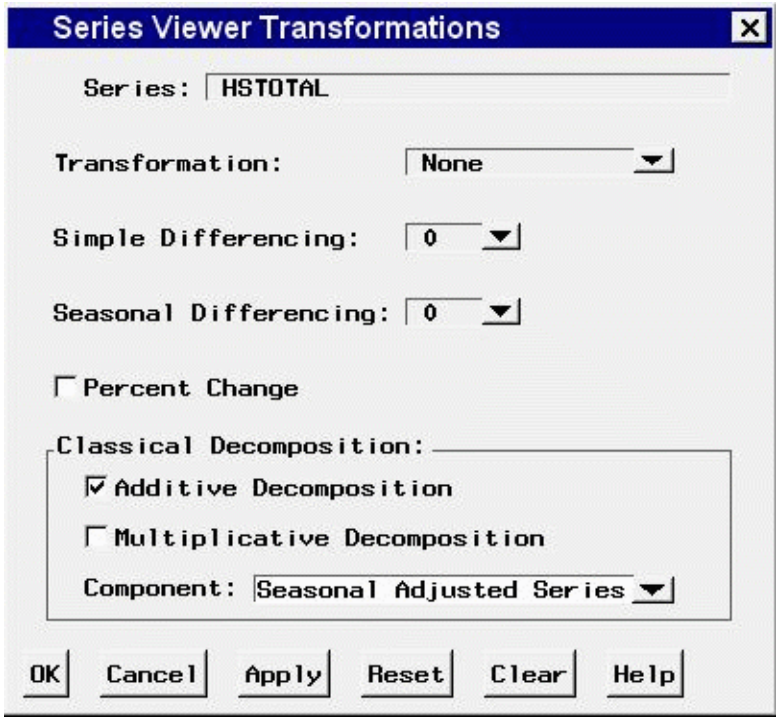
`Clear`
: deselects all series in the selection list.

`All`
: selects all series in the selection list.

Series Viewer Transformations Window

Use the Series Viewer Transformations window to view plots of transformations of the current series in the Time Series Viewer window. It provides a larger set of transformations than those available from the viewer window's toolbar. It is invoked by using "Other Transformations" under the Tools menu of the Time Series Viewer window. The options that you specify in this window are applied to the series displayed in the Time Series Viewer window when you select "OK" or "Apply."

Use the Apply button if you want to make repeated transformations to a series without having to close and reopen the Series Viewer Transformations window each time.



Controls and Fields

`Series`
: is the variable name for the current time series.

`Transformation`
: is the transformation applied to the time series displayed in the Time Series Viewer window. Select Log, Logistic, Square Root, Box-Cox, or none from the pop-up list.

`Simple Differencing`
: is the order of differencing applied to the time series displayed in the Time Series Viewer window. Select a number from 0 to 5 from the pop-up list.

`Seasonal Differencing`
: is the order of seasonal differencing applied to the time series displayed in the Time Series Viewer window. Select a number from 0 to 3 from the pop-up list.

`Percent Change`
: is a check box that if selected displays the series in terms of percent change from the previous period.

`Additive Decomposition`
: is a check box that produces a display of a selected series component derived by using additive decomposition.

`Multiplicative Decomposition`
: is a check box that produces a display of a selected series component derived using multiplicative decomposition.

`Component`
: selects a series component to display when either additive or multiplicative decomposition is turned on. You can display the seasonally adjusted component, the trend-cycle component, the seasonal component, or the irregular component—that is, the residual that remains after removal of the other components. The heading in the viewer window shows which component is currently displayed.

`OK`
: applies the transformation options you selected to the series displayed in the Time Series Viewer window and closes the Series Viewer Transformations window.

`Cancel`
: closes the Series Viewer Transformations window without changing the series displayed by the Time Series Viewer window.

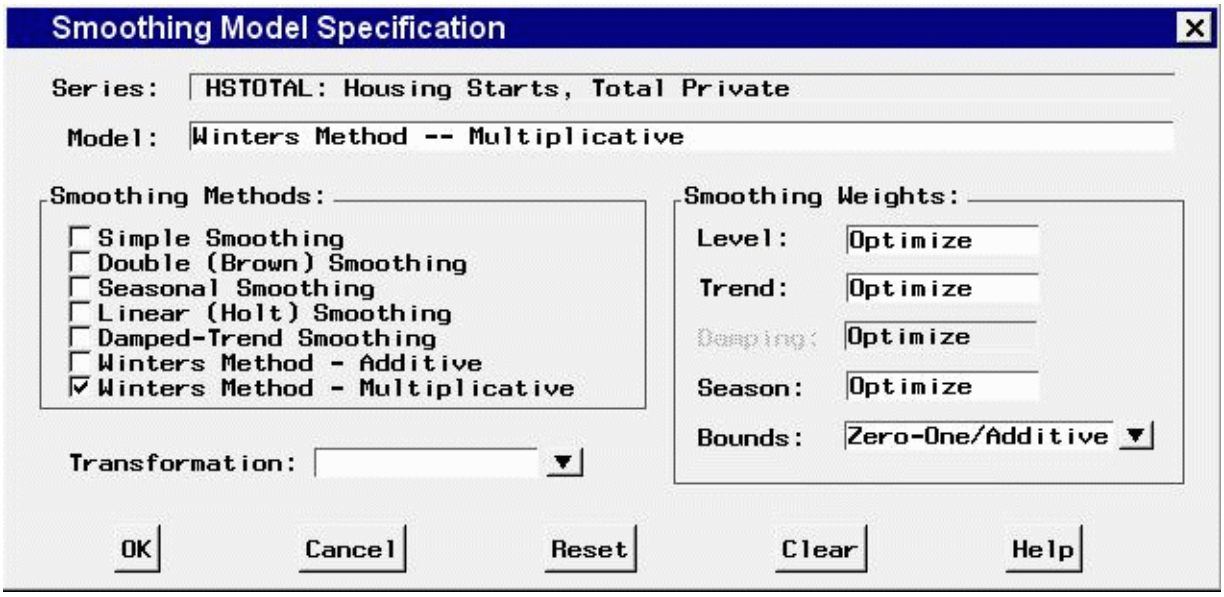
`Apply`
: applies the transformation options you selected to the series displayed in the Time Series Viewer window without closing the Series Viewer Transformations window.

`Reset`
: resets the transformation options to their initial values upon entry to the Series Viewer Transformations window.

`Clear`
: resets the transformation options to their default values (no transformations).

Smoothing Model Specification Window

Use the Smoothing Model Specification window to specify and fit exponential smoothing and Winters method models. Access it from the Develop Models window by using the Fit Model submenu of the Edit menu or from the pop-up menu when you click an empty area of the model table.



Controls and Fields

`Series`
is the name and variable label of the current series.

`Model`
is a descriptive label for the model that you specify. You can type a label in this field or allow the system to provide a label. If you leave the label blank, a label is generated automatically based on the options you specify.

`Smoothing Methods`

`Simple Smoothing`
specifies simple (single) exponential smoothing.

`Double (Brown) Smoothing`
specifies double exponential smoothing by using Brown's one parameter model (single exponential smoothing applied twice).

`Seasonal Smoothing`
specifies seasonal exponential smoothing. (This is like Winters method with the trend term omitted.)

`Linear (Holt) Smoothing`
specifies exponential smoothing of both the series level and trend (Holt's two parameter model).

`Damped-Trend Smoothing`
specifies exponential smoothing of both the series level and trend with a trend damping weight.

`Winters Method - Additive`
specifies Winters method with additive seasonal factors.

`Winters Method - Multiplicative`
specifies Winters method with multiplicative seasonal factors.

`Smoothing Weights`
displays the values used for the smoothing weights. By default, the Smoothing Weights fields are set to "optimize," which means that the system will compute the weight values that best fit the data. You can also type smoothing weight values in these fields.

`Level`
is the smoothing weight used for the level of the series.

`Trend`
is the smoothing weight used for the trend of the series.

`Damping`
is the smoothing weight used by the damped-trend method to damp the forecasted trend towards zero as the forecast horizon increases.

`Season`
is the smoothing weight used for the seasonal factors in Winters method and seasonal exponential smoothing.

`Transformation`
displays the series transformation specified for the model. When a transformation is specified, the model is fit to the transformed series, and forecasts are produced by applying the inverse transformation to the model predictions. Select `Log`, `Logistic`, `Square Root`, `Box-Cox`, or `None` from the pop-up list.

`Bounds`
displays the constraints imposed on the fitted smoothing weights. Select one of the following from the pop-up list:

`Zero-One/Additive`
sets the smoothing weight optimization region to the intersection of the region bounded by the intervals from zero (0.001) to one (0.999) and the additive invertible region. This is the default.

`Zero-One Boundaries`
sets the smoothing weight optimization region to the region bounded by the intervals from zero (0.001) to one (0.999).

`Additive Invertible`
sets the smoothing weight optimization region to the additive invertible region.

`Unrestricted`
sets the smoothing weight optimization region to be unbounded.

`Custom`
opens the `Smoothing Weights` window to enable you to customize the constraints for smoothing weights optimization.

`OK`
: closes the Smoothing Model Specification window and fits the model you specified.

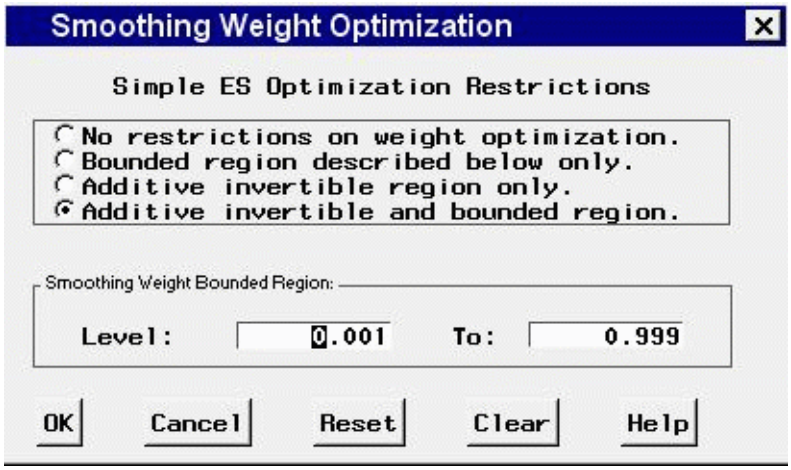
`Cancel`
: closes the Smoothing Model Specification window without fitting the model. Any options you specified are lost.

`Reset`
: resets all options to their initial values upon entry to the window. This might be useful when editing an existing model specification; otherwise, Reset has the same function as Clear.

`Clear`
: resets all options to their default values.

Smoothing Weight Optimization Window

Use the Smoothing Weight Optimization window to specify constraints for the automatic fitting of smoothing weights for exponential smoothing and Winters method models. Access it from the Smoothing Models Specification window when you select "Custom" in the "Bounds" combo box.



Controls and Fields

`No restrictions`
: when selected, specifies unrestricted smoothing weights.

`Bounded region`
: when selected, restricts the fitted smoothing weights to be within the bounds that you specify with the "Smoothing Weight Bounded Region" options.

`Additive invertible region`
: when selected, restricts the fitted smoothing weights to be within the additive invertible region of the parameter space of the ARIMA model equivalent to the smoothing model. (See the section "Smoothing Models" on page 3890 for details.)

`Additive invertible and bounded region`
: when selected, restricts the fitted smoothing weights to be both within the additive invertible region and within bounds that you specify.

`Smoothing Weight Bounded Region`
: is a group of numeric entry fields that enable you to specify lower and upper limits on the fitted value of each smoothing weight. The fields that appear in this part of the window depend on the kind of smoothing model that you specified.

`OK`
: closes the window and sets the options that you specified.

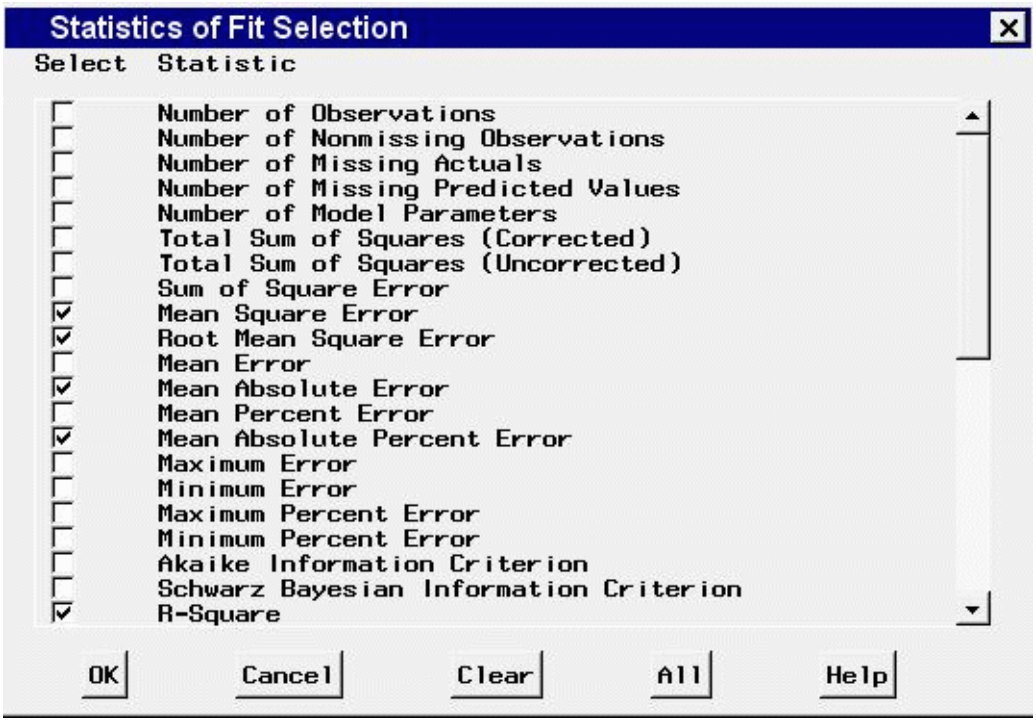
`Cancel`
: closes the window without changing any options. Any values you specified are lost.

`Reset`
: resets all options to their initial values upon entry to the window.

`Clear`
: resets all options to their default values.

Statistics of Fit Selection Window

Use the Statistics of Fit Selection window to specify which of the available goodness-of-fit statistics are reported for models you fit and are available for selection as the model selection criterion used by the automatic selection process. This window is available under the Options menu in the Develop Models, Automatic Model Fitting, Produce Forecasts, and Model List windows, and from the Statistics button of the Model Fit Comparison window and Automatic Model Fitting results windows.



Controls and Fields

`Select Statistics Table`
: list the available statistics. Select a row of the table to select or deselect the statistic shown in that row.

`OK`
: closes the window and applies the selections made.

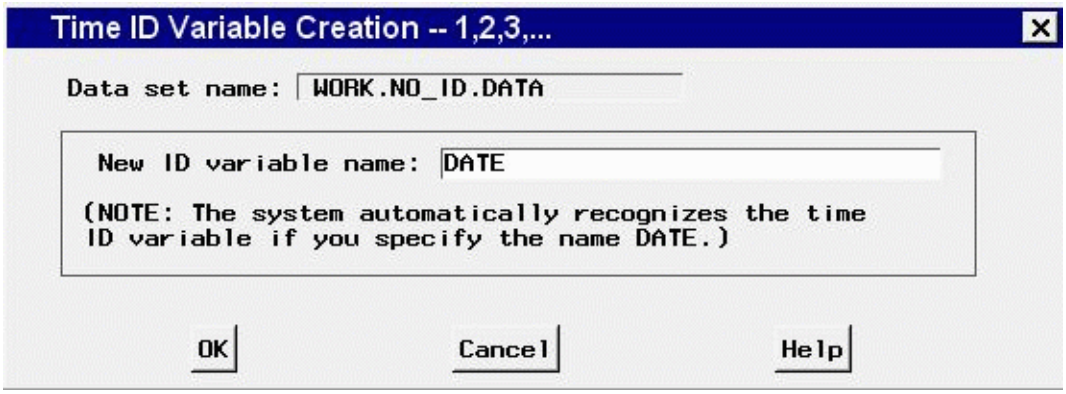
`Cancel`
: closes the window without applying any selections.

`Clear`
: deselects all statistics.

`All`
: selects all statistics.

Time ID Creation – 1,2,3 Window

Use the Time ID Creation – 1,2,3 window to add a time ID variable to an input data set with observation numbers as the ID values. The interval for the series will be 1. Use this approach if the data frequency does not match any of the system's date or date-time intervals, or if other methods of assigning a time ID do not work. To access this window, select "Create from observation numbers" from the Create pop-up list in any window where you can select a Time ID variable. For more information, see Chapter 5, "Date Intervals, Formats, and Functions," in this book.



Controls and Fields

`Data set name`
: is the name of the input data set.

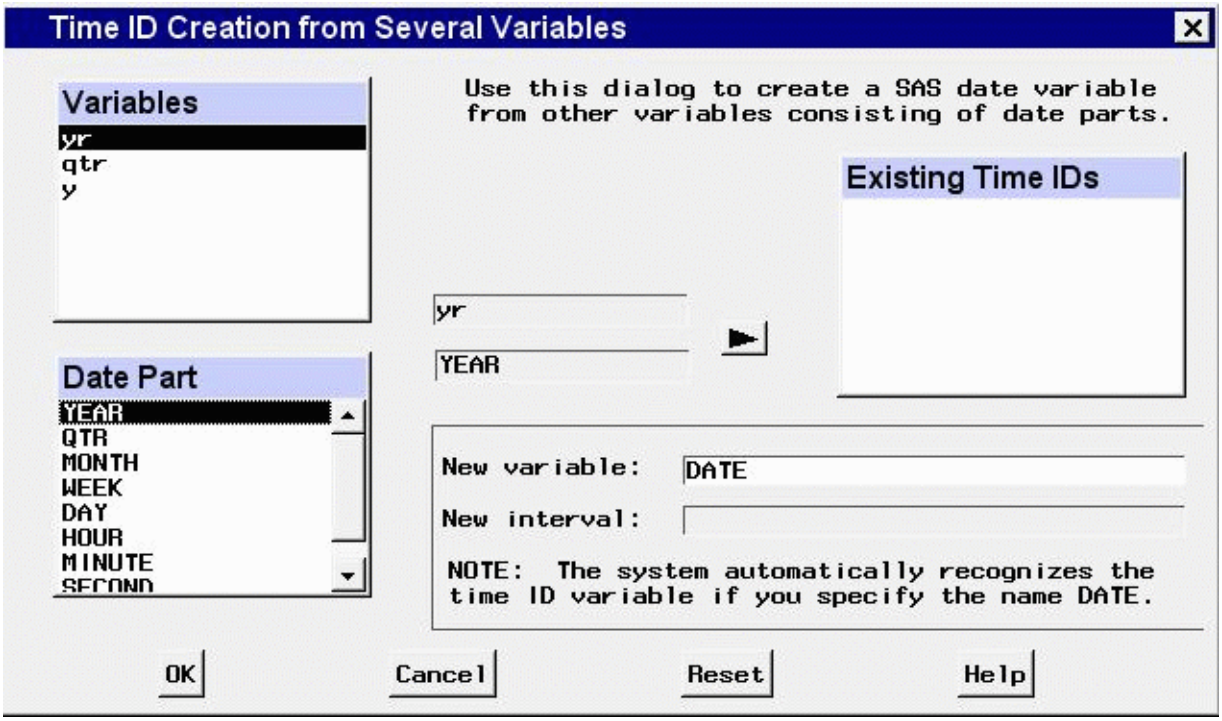
`New ID variable name`
: is the name of the time ID variable to be created. You can type any valid SAS variable name in this field.

`OK`
: closes the window and proceeds to the next step in the time ID creation process.

`Cancel`
closes the window without creating a Time ID variable. Any options you specified are lost.

Time ID Creation from Several Variables Window

Use the Time ID Creation from Several Variables window to add a SAS date valued time ID variable to an input data set when the input data set already contains several dating variables, such as day, month, and year. To access this window, select "Create from existing variables" from the Create pop-up list in any window where you can select a Time ID variable. For more information, see Chapter 55, "Creating Time ID Variables."



Controls and Fields

`Variables`
is a list of variables in the input data set. Select existing ID variables from this list.

`Date Part`
is a list of date parts that you can specify for the selected ID variable. For each ID variable that you select from the Variables list, select the Date Part value that describes what the values of the ID variable represent.

`arrow button`
moves the selected existing ID variable and date part specification to the "Existing Time IDs" list. Once you have done this, you can select another ID variable from the Variables list.

`New variable`
is the name of the time ID variable to be created. You can type any valid SAS variable name in this field.

`New interval`
is the time interval between observations in the input data set implied by the date part ID variables you have selected.

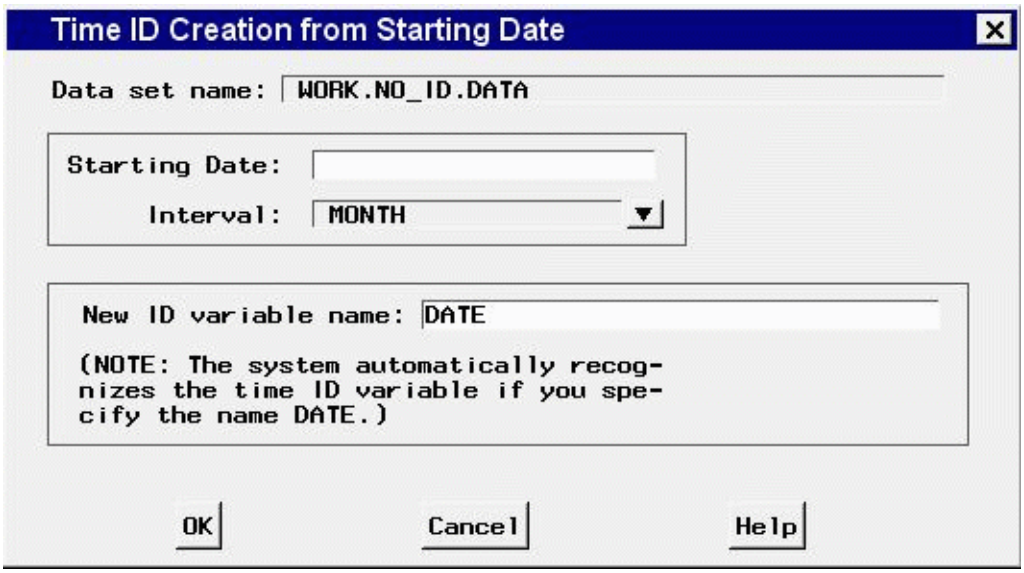
`OK`
closes the window and proceeds to the next step in the time ID creation process.

`Cancel`
closes the window without creating a time ID. Any options you specified are lost.

`Reset`
resets the options to their initial values upon entry to the window.

Time ID Creation from Starting Date Window

Use the Time ID Creation from Starting Date window to add a SAS date valued time ID variable to an input data set. This is a convenient way to add a time ID of any interval as long as you know the starting date of the series. To access this window, select "Create from starting date and frequency" from the Create pop-up list in any window where you can select a Time ID variable. For more information, see Chapter 55, "Creating Time ID Variables."



Controls and Fields

`Data set name`
is the name of the input data set.

`Starting Date`
: is the starting date for the time series in the data set. Enter a date value in this field, using a form recognizable by a SAS date informat, for example, 1998:1, feb1997, or 03mar1998.

`Interval`
: is the time interval between observations in the data set. Select an interval from the pop-up list.

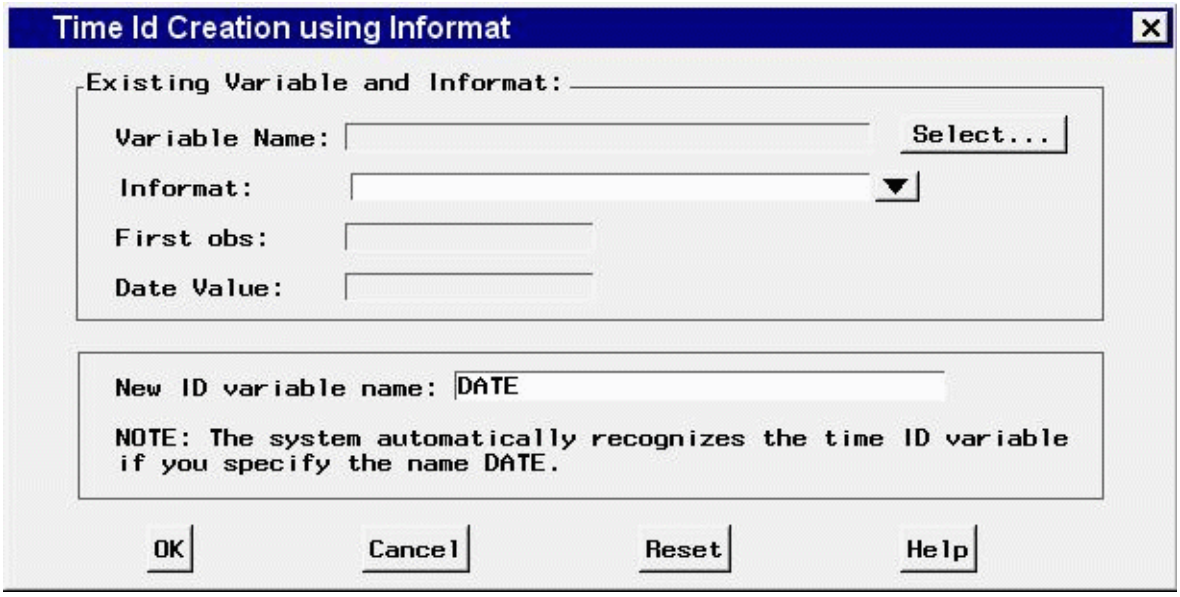
`New ID variable name`
: is the name of the time ID variable to be created. You can type any valid SAS variable name in this field.

`OK`
: closes the window and proceeds to the next step in the time ID creation process.

`Cancel`
: closes the window without changing the input data set. Any options you specified are lost.

Time ID Creation Using Informat Window

Use the Time ID Creation using Informat window to add a SAS date valued time ID variable to an input data set. Use this window if your data set contains a date variable that is stored as a character string. Using the appropriate SAS date informat, the date string is read in and used to create a date or date-time variable. To access this window, select "Create from existing variable/informat" from the Create pop-up list in any window where you can select a Time ID variable.



Controls and Fields

`Variable Name`
: is the name of an existing ID variable in the input data set. Click the Select button to select a variable.

`Select button`
: opens a list of variables in the input data set for you to select from.

`Informat`
: is a SAS date or datetime informat for reading date or datetime value from the values of the specified existing ID variable. You can type in an informat or select one from the pop-up list.

`First Obs`
: is the value of the variable you selected from the first observation in the data set, displayed here for convenience.

`Date Value`
: is the SAS date or datetime value read from the first observation value that uses the informat that you specified.

`New ID variable name`
: is the name of the time ID variable to be created. You can type any valid SAS variable name in this field.

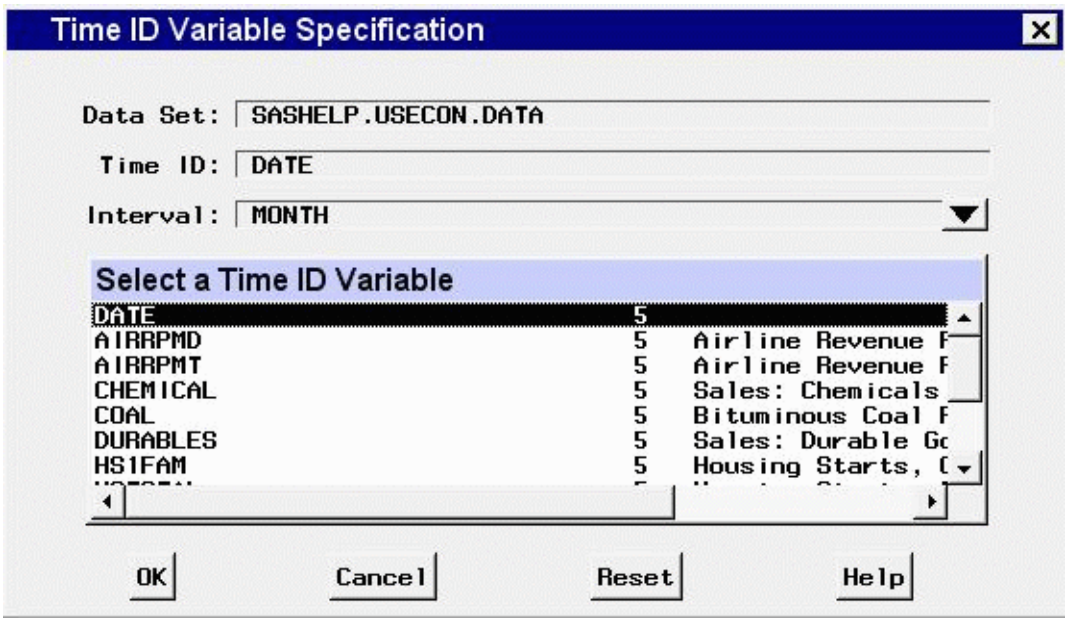
`OK`
: closes the window and proceeds to the next step in the time ID creation process.

`Cancel`
: closes the window without changing the input data set. Any options you specified are lost.

`Reset`
: resets the options to their initial values upon entry to the window.

Time ID Variable Specification Window

Use the Time ID Variable Specification window to specify a variable in the input data set that contains the SAS date or datetime value of each observation. You do not need to use this window if your time ID variable is named `date`, `time`, or `datetime`, since these are picked up automatically. Invoke the window from the Select button to the right of the Time ID field in the Data Set Selection, Automatic Model Fitting, Produce Forecasts, Series Selection, and Time Series Forecasting windows.



Controls and Fields

`Data Set`
is the name of the current input data set.

`Time ID`
is the name of the currently selected Time ID variable, if any.

`Interval`
is the time interval between observations (data frequency) in the input data set.

`Select a Time ID Variable`
is a selection list of variables in the input set. Select one variable to assign it as the Time ID variable.

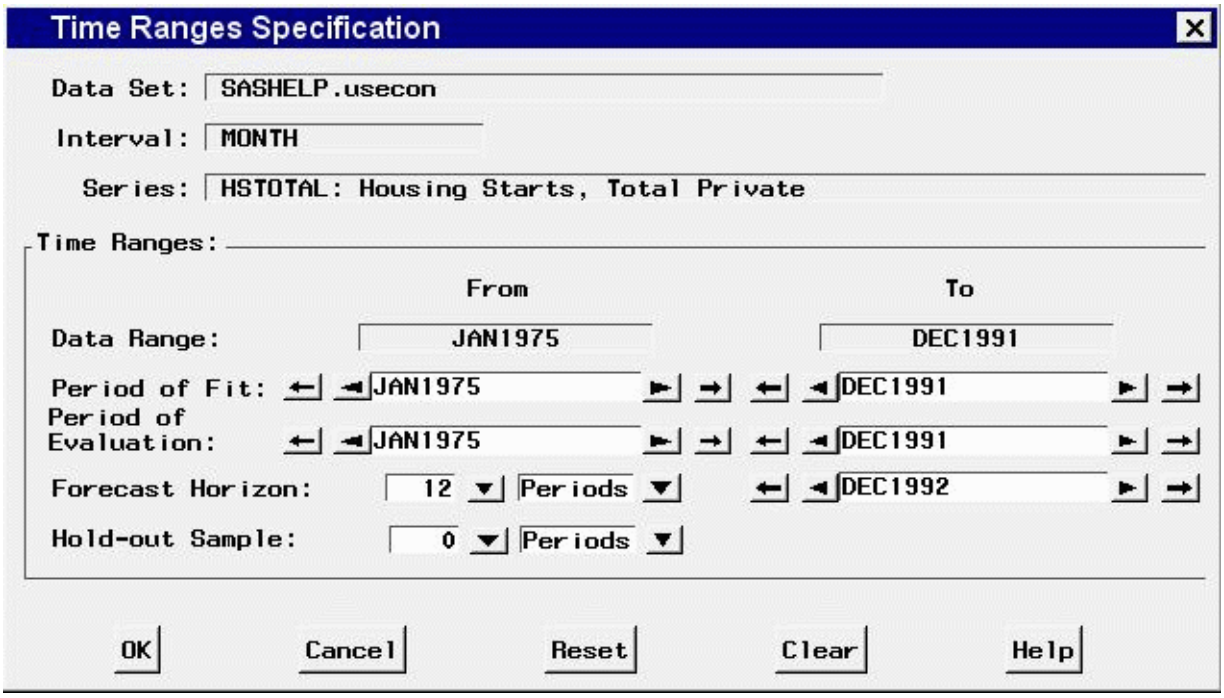
`OK`
closes the window and retains the selection made, if it is a valid time ID.

`Cancel`
closes the window and ignores any selection made.

`Reset`
restores the time ID variable to the one assigned when the window was initially opened, if any.

Time Ranges Specification Window

Use the Time Ranges Specification window to control the period of fit and evaluation and the forecasting horizon. Invoke this window from the Options menu in the Develop Models, Manage Forecasting Project, and Model Viewer windows or the Set Ranges button in the Develop Models window.



Controls and Fields

`Data Set`
: is the name of the current input data set.

`Interval`
: is the time interval (data frequency) for the input data set.

`Series`
: is the variable name and label of the current time series.

`Data Range`
: gives the date of the first and last nonmissing data values available for the current series in the input data set.

`Period of Fit`
: gives the starting and ending dates of the period of fit. This is the time range used for estimating model parameters. By default, it is the same as the data range. You can type dates in these fields, or you can use the arrow buttons to the left and right of the date fields to decrement or increment the date values shown. Date values must be entered in a form recognized by a SAS date informat. (See *SAS Language Reference: Concepts* for information about SAS date informats.) The inner arrows increment by periods, the outer arrows increment by larger amounts, depending on the data interval.

`Period of Evaluation`
: gives the starting and ending dates of the period of evaluation. This is the time range used for evaluating models in terms of statistics of fit. By default, it is the same as the data range. You can type dates in these fields, or you can use the control arrows to the left and right of the date fields to decrement or increment the date values shown. Date values must be entered in a form recognized by a SAS date informat. (See *SAS Language Reference: Concepts* for information about SAS date informats.) The inner arrows increment by periods, the outer arrows increment by larger amounts, depending on the data interval.

`Forecast Horizon`
: is the forecasting horizon expressed as a number of forecast periods or number of years (or number of weeks for daily data). You can type a number or select one from the pop-up list. The ending date for the forecast period is automatically updated when you change the number of forecasts periods.

`Forecast Horizon – Units`
: indicates whether the Forecast Horizon value represents periods or years (or weeks for daily data).

`Forecast Horizon Date Value`
: is the date of the last forecast observation. You can type a date in this field, or you can use the arrow buttons to the left and right of the date field to decrement or increment the date values shown. Date values must be entered in a form recognized by a SAS date informat. (See *SAS Language Reference: Concepts* for information about SAS date informats.) The Forecast Horizon is automatically updated when you change the ending date for the forecast period.

`Hold-out Sample`
: specifies that a number of observations or years (or weeks) of data at the end of the data range are used for the period of evaluation with the remainder of data used as the period of fit. You can type a number in this field or select one from the pop-up list. When the hold-out sample value is changed, the Period of Fit and Period of Evaluation ranges are changed to reflect the hold-out sample specification.

`Hold-out Sample – Units`
: indicates whether the hold-out sample field represents periods or years (or weeks for daily data).

`OK`
: closes the window and stores the specified changes.

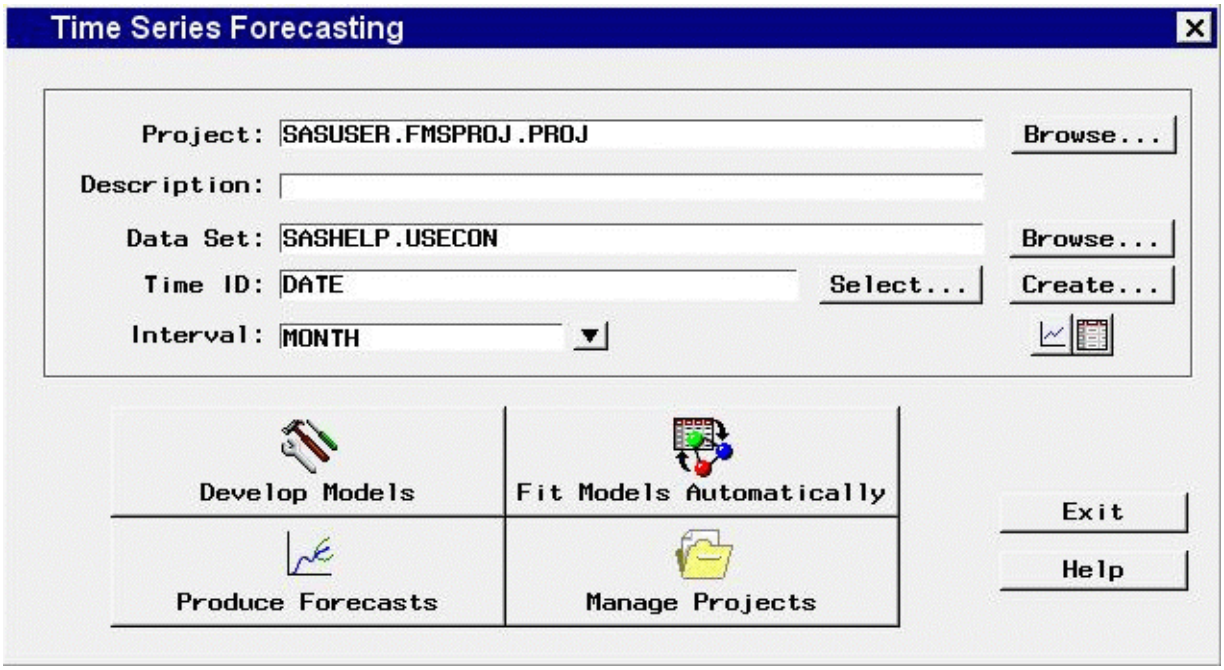
`Cancel`
: closes the window without saving changes. Any options you specified are lost.

`Reset`
: resets the options to their initial values upon entry to the window.

`Clear`
: resets all options to their default values.

Time Series Forecasting Window

The Time Series Forecasting window is the main application window that appears when you invoke the Time Series Forecasting System. It enables you to specify a project file and an input data set and provides access to the other windows described in this chapter.



Controls and Fields

`Project`
: is the name of the SAS catalog entry in which forecasting models and other results will be stored and from which previously stored results are loaded into the forecasting system. You can specify the project by typing a SAS catalog entry name in this field or by selecting the Browse button to right of this field. If you specify the name of an existing catalog entry, the information in the project file is loaded. If you specify a one-level name, the catalog name is assumed to be `fmsproj` and the library is assumed to be `sasuser`. For example, `samproj` is equivalent to `sasuser.fmsproj.samproj`.

Project `Browse` button
: opens the Forecasting Project File Selection window to enable you to select and load the project from a list of previously stored projects.

`Description`
: is a descriptive label for the forecasting project. The description you type in this field will be stored with the catalog entry shown in the Project field.

`Data Set`
: is the name of the current input data set. To specify the input data set, you can type the data set name in this field or use the Browse button to the right of the field.

Data set `Browse` button
: opens the Data Set Selection window to enable you to select the input data set.

`Time ID`
: is the name of the ID variable for the input data set. To specify the ID variable, you can type the ID variable name in this field or use the Select button. If the time ID variable is named `date`, `time`, or `datetime`, it is automatically picked up by the system.

`Select` button
: opens the Time ID Variable Specification window.

`Create` button
: opens a menu of choices of methods for creating a time ID variable for the input data set. Use this feature if the input data set does not already contain a valid time ID variable.

`Interval`
: is the time interval between observations (data frequency) in the current input data set. If the interval is not automatically filled in, you can type an interval name or select one from the pop-up list. For more information about intervals, see the section "Time Series Data Sets, ID Variables, and Time Intervals" on page 3622.

`View Series Graphically` icon
: opens the Time Series Viewer window to display plots of series in the current input data set.

`View Data as a Table`
: opens a Viewtable window for browsing the selected input data set.

`Develop Models`
: opens the Develop Models window to enable you to fit forecasting models to individual time series and choose the best models to use to produce the final forecasts of each series.

`Fit Models Automatically`
: opens the Automatic Model Fitting window for applying the automatic model selection process to all series or to selected series in an input data set.

`Produce Forecast`
: opens the Produce Forecasts window for producing forecasts for the series in the current input data set for which you have fit forecasting models.

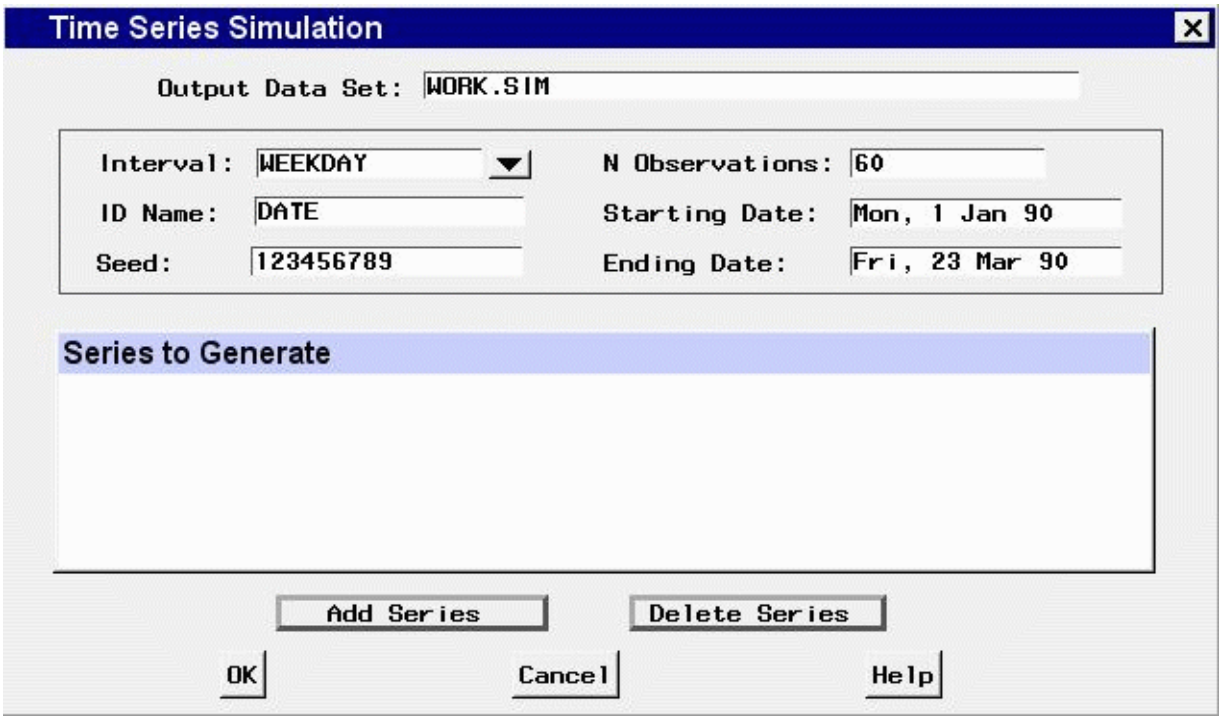
`Manage Projects`
: opens the Manage Forecasting Project window for viewing or editing information stored in projects.

`Exit`
: closes the Time Series Forecasting system.

`Help`
: accesses the help system.

Time Series Simulation Window

Use the Time Series Simulation window to create a data set of simulated series generated by ARIMA processes. Access this window from the Tools menu in the Develop Models and Manage Forecasting Project windows.



Controls and Fields

`Output Data Set`
: is the name of the data set to be created. Type in a one-level or two-level SAS data set name.

`Interval`
: is the time interval between observations (data frequency) in the simulated data set. Type in an interval name or select one from the pop-up list.

`Seed`
: is the seed for the random number generator used to produce the simulated time series.

`N Observations`
: is the number of time periods to simulate.

`Starting Date`
: is the starting date for the simulated observations. Type in a date in a form recognizable by a SAS data informat, for example, 1998:1, feb1997, or 03mar1998.

`Ending Date`
: is the ending date for the simulated observations. Type in a date in a form recognizable by a SAS data informat.

`Series to Generate`
: is the list of variable names and ARIMA processes to simulate.

`Add Series`
: opens the ARIMA Process Specification window to enable you to add entries to the Series to Generate list.

`Delete Series`
: deletes selected (highlighted) entries from the Series to Generate list.

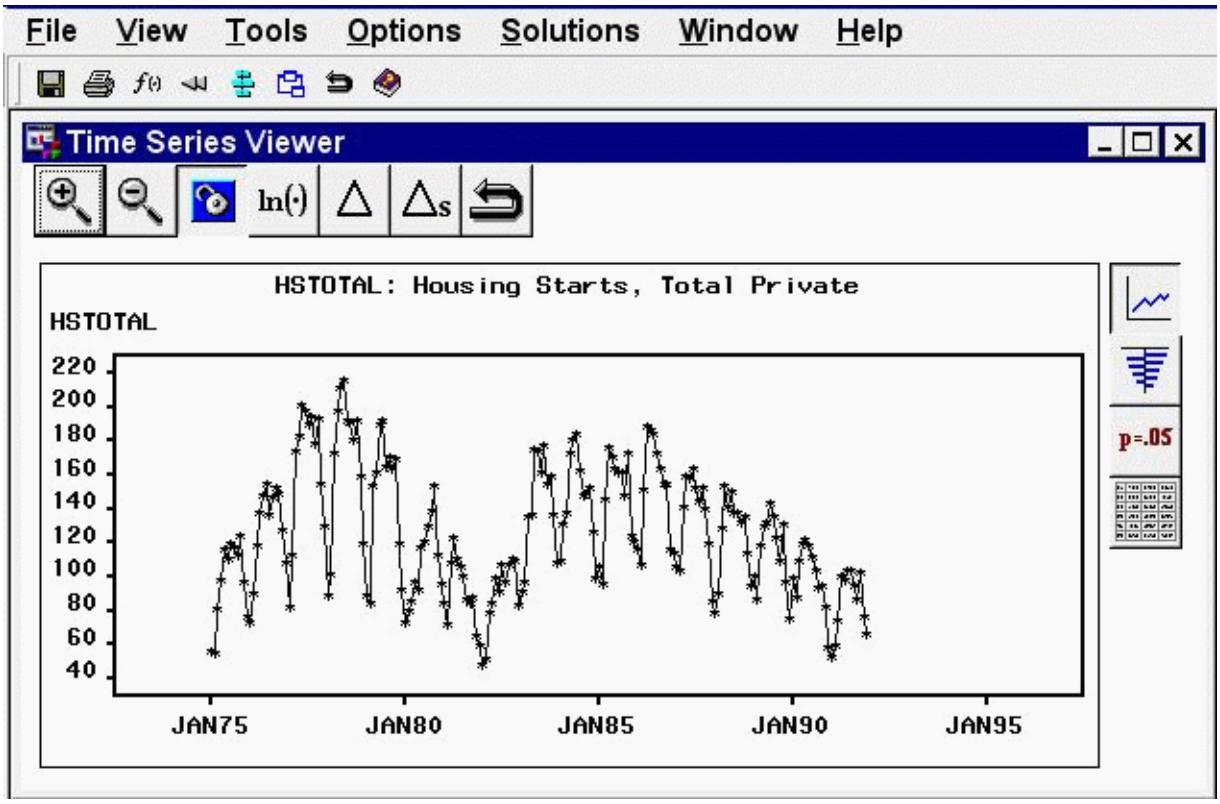
`OK`
: closes the Time Series Simulation window and performs the specified simulations and creates the specified data set.

`Cancel`
: closes the window without creating a simulated data set. Any options you specified are lost.

Time Series Viewer Window

Use the Time Series Viewer window to explore time series data using plots, transformations, statistical tests, and tables. It is available as a standalone application and as part of the Time Series Forecasting System. To use it as a standalone application, select it from the Analysis submenu of the Solutions menu, or use the `tsview` command (see Chapter 59, "Command Reference," in this book). To use it within the Time Series Forecasting System, select the View Series Graphically icon in the Time Series Forecasting, Develop Models, or Model List window, or select "Series" from the View menu of the Develop Models, Manage Project, or Model List window.

The various plots and tables available are referred to as *views*. The section "View Selection Icons" on page 3839 explains how to change the view.



The state of the Time Series Viewer window is controlled by the current series, the current series transformation specification, and the currently selected view. You can resize this window, and you can use other windows without closing the Time Series Viewer window. You can explore a number of series conveniently

by keeping the Series Selection window open. Each time you make a selection, the viewer window is updated to show the selected series. Keep both windows visible, or switch between them by using the Next Viewer toolbar icon or the F12 function key.

You can open multiple Time Series Viewer windows. This enables you to "freeze"a plot so you can come back to it later, or compare two plots side by side on your screen. To do this, unlink the viewer by using the Link/Unlink icon on the window's toolbar or the corresponding item in the Tools menu. While the viewer window remains unlinked, it is not updated when other selections are made in the Series Selection window. Instead, when you select a series and click the Graph button, a new Time Series Viewer window is invoked. You can continue this process to open as many viewer windows as you want. The Next Viewer icon and corresponding F12 function key are useful for navigating between windows when they are not simultaneously visible on your screen.

A wide range of series transformations is available. Basic transformations are available from the window's horizontal toolbar, and others are available by selecting "Other Transformations" from the Tools menu.

Horizontal Tool Bar

The Time Series Viewer window contains a horizontal toolbar with the following icons:

`Zoom in`
: changes the mouse cursor into cross hairs that you can use with the left mouse button to drag out a region of the time series plot to zoom in on. In the Autocorrelations view and the White Noise and Stationarity Tests view, Zoom In reduces the number of lags displayed.

`Zoom out`
: reverses the previous Zoom In action and expands the time range of the plot to show more of the series. In the Autocorrelations view and the White Noise and Stationarity Tests view, Zoom Out increases the number of lags displayed.

`Link/Unlink viewer`
: disconnects or connects the Time Series Viewer window to the window in which the series was selected. When the Viewer is linked, it always shows the current series. If you select another series, linked Viewers are updated. Unlinking a Viewer freezes its current state, and changing the current series has no effect on the Viewer's display. The View Series action creates a new Series Viewer window if there is no linked Viewer. By using the unlink feature, you can open several Time Series Viewer windows and display several different series simultaneously.

`Log Transform`
: applies a log transform to the current view. This can be combined with other transformations; the current transformations are shown in the title.

`Difference`
: applies a simple difference to the current view. This can be combined with other transformations; the current transformations are shown in the title.

`Seasonal Difference`
: applies a seasonal difference to the current view. For example, if the data are monthly, the seasonal cycle is one year. Each value has subtracted from it the value from one year previous. This can be combined with other transformations; the current transformations are shown in the title.

`Close`
: closes the Time Series Viewer window and returns to the window from which it was invoked.

Vertical Toolbar View Selection Icons

At the right-hand side of the Time Series Viewer window is a vertical toolbar used to select the kind of plot or table that the Viewer displays.

`Series`
: displays a plot of series values over time.

`Autocorrelations`
: displays plots of the sample autocorrelations, partial autocorrelation, and inverse autocorrelation functions for the series, with lines overlaid at plus and minus two standard errors.

`White Noise and Stationarity Tests`
: displays horizontal bar charts that represent results of white noise and stationarity tests. The first bar chart shows the significance probability of the Ljung-Box chi-square statistic computed on autocorrelations up to the given lag. Longer bars favor rejection of the null hypothesis that the series is white noise. Click any of the bars to display an interpretation.

  The second bar chart shows tests of stationarity, where longer bars favor the conclusion that the series is stationary. Each bar displays the significance probability of the augmented Dickey-Fuller unit root test to the given autoregressive lag. Long bars represent higher levels of significance against the null hypothesis that the series contains a unit root. For seasonal data, a third bar chart appears for seasonal root tests. Click any of the bars to display an interpretation.

`Data Table`
: displays a data table containing the values in the input data set.

Menu Bar

`File`

`Save Graph`
: saves the current plot as a SAS/GRAPH grseg catalog entry in a default or most recently specified catalog. This item is unavailable in the Data Table view.

`Save Graph as`
: saves the current graph as a SAS/GRAPH grseg catalog entry in a SAS catalog that you specify and/or as an Output Delivery System (ODS) object. By default, an HTML page is created, with the graph embedded as a gif image. This item is unavailable in the Data Table view.

`Save Data`
: saves the data displayed in the viewer window to an output SAS data set. This item is unavailable in the Series view.

`Save Data as`
: saves the data in a SAS data set that you specify and/or as an Output Delivery System (ODS) object. By default, an HTML page is created, with the data displayed as a table.

`Import Data`
: is available if you license SAS/Access software. It opens an Import Wizard, which you can use to import your data from an external spreadsheet or data base to a SAS data set for use in the Time Series Forecasting System.

`Export Data`
: is available if you license SAS/Access software. It opens an Export Wizard, which you can use to export a SAS data set, such as a forecast data set created with the Time Series Forecasting System, to an external spreadsheet or data base.

`Print Graph`
: prints the plot displayed in the viewer window. This item is unavailable in the Data Table view.

`Print Data`
: prints the data displayed in the viewer window. This item is unavailable in the Series view.

`Print Setup`
: opens the Print Setup window, which allows you to access your operating system print setup.

`Print Preview`
: opens a preview window to show how your plots will look when printed.

`Close`
: closes the Time Series Viewer window and returns to the window from which it was invoked.

`View`

`Series`
: displays a plot of series values over time. This is the same as the Series icon in the vertical toolbar.

`Autocorrelations`
: displays plots of the sample autocorrelation, partial autocorrelation, and inverse autocorrelation functions for the series. This is the same as the Autocorrelations icon in the vertical toolbar.

`White Noise and Stationarity Tests`
: displays horizontal bar charts representing results of white noise and stationarity tests. This is the same as the White Noise and Stationarity Tests icon in the vertical toolbar.

`Data Table`
: displays a data table containing the values in the input data set. This is the same as the Data Table icon in the vertical toolbar.

`Zoom In`
: zooms the display. This is the same as the Zoom In icon in the window's horizontal toolbar.

`Zoom Out`
: undoes the last zoom in action. This is the same as the Zoom Out icon in the window's horizontal toolbar.

`Zoom Way Out`
: reverses all previous Zoom In actions and expands the time range of the plot to show all of the series, or shows the maximum number of lags in the Autocorrelations View or the White Noise and Stationarity Tests view.

`Tools`

`Log Transform`
: applies a log transformation. This is the same as the Log Transform icon in the window's horizontal toolbar.

Difference
: applies simple differencing. This is the same as the Difference icon in the window's horizontal toolbar.

Seasonal Difference
: applies seasonal differencing. This is the same as the Seasonal Difference icon in the window's horizontal toolbar.

Other Transformations
: opens the Series Viewer Transformations window to enable you to apply a wide range of transformations.

Diagnose Series
: opens the Series Diagnostics window to determine the kinds of forecasting models appropriate for the current series.

Define Interventions
: opens the Interventions for Series window to enable you to edit or add intervention effects for use in modeling the current series.

Link Viewer
: connects or disconnects the Time Series Viewer window to the window from which series are selected. This is the same as the Link item in the window's horizontal toolbar.

Options

Number of Lags
: opens a window to let you specify the number of lags shown in the Autocorrelations view and the White Noise and Stationarity Tests view. You can also use the Zoom In and Zoom Out actions to control the number of lags displayed.

Correlation Probabilities
: controls whether the bar charts in the Autocorrelations view represent significance probabilities or values of the correlation coefficient. A check mark or filled check box next to this item indicates that significance probabilities are displayed. In each case the bar graph horizontal axis label changes accordingly.

Mouse Button Actions

You can examine the data value and date of individual points in the Series view by clicking them. The date and value are displayed in a box that appears in the upper right corner of the Viewer window. Click the mouse elsewhere or select any action to dismiss the data box.

You can examine the values of the bars and confidence limits at different lags in the Autocorrelations view by clicking individual bars in the vertical bar charts.

You can display an interpretation of the tests in the White Noise and Stationarity Tests view by clicking the bars.

When you select the Zoom In action, you can use the mouse to define a region of the graph to take a closer look at. Position the mouse cursor at one corner of the region, press the left mouse button, and move the mouse cursor to the opposite corner of the region while holding the left mouse button down. When you release the mouse button, the plot is redrawn to show an expanded view of the data within the region you selected.

Chapter 61
Forecasting Process Details

Contents

Forecasting Process Summary **3883**
Parameter Estimation 3884
Model Evaluation 3884
Forecasting 3886
Forecast Combination Models 3888
External or User-Supplied Forecasts 3888
Adjustments 3888
Series Transformations 3889
Smoothing Models **3890**
Smoothing Model Calculations 3890
Missing Values 3892
Predictions and Prediction Errors 3892
Smoothing Weights 3892
Equations for the Smoothing Models 3894
ARIMA Models **3901**
Notation for ARIMA Models 3902
Predictor Series **3905**
Time Trend Curves 3905
Intervention Effects 3906
Seasonal Dummy Inputs 3908
Series Diagnostic Tests **3908**
Statistics of Fit **3909**
References **3911**

This chapter provides computational details on several aspects of the Time Series Forecasting System.

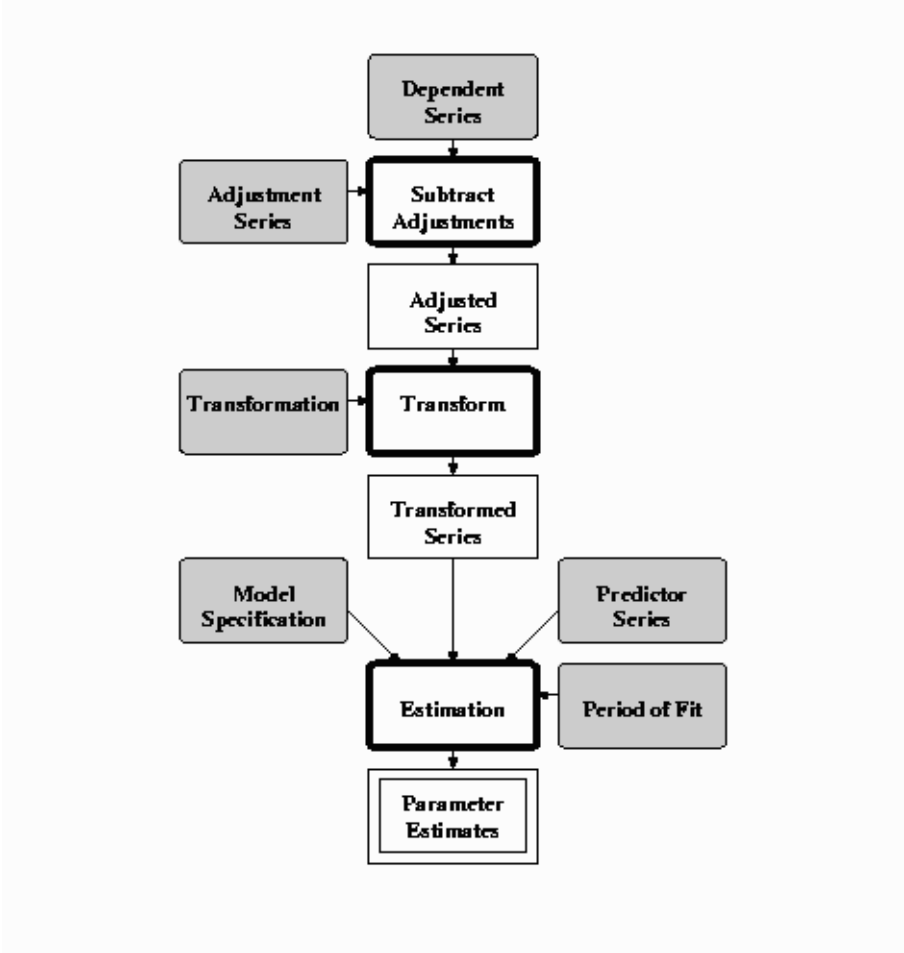
Forecasting Process Summary

This section summarizes the forecasting process.

Parameter Estimation

The parameter estimation process for ARIMA and smoothing models is described graphically in Figure 61.1.

Figure 61.1 Model Fitting Flow Diagram

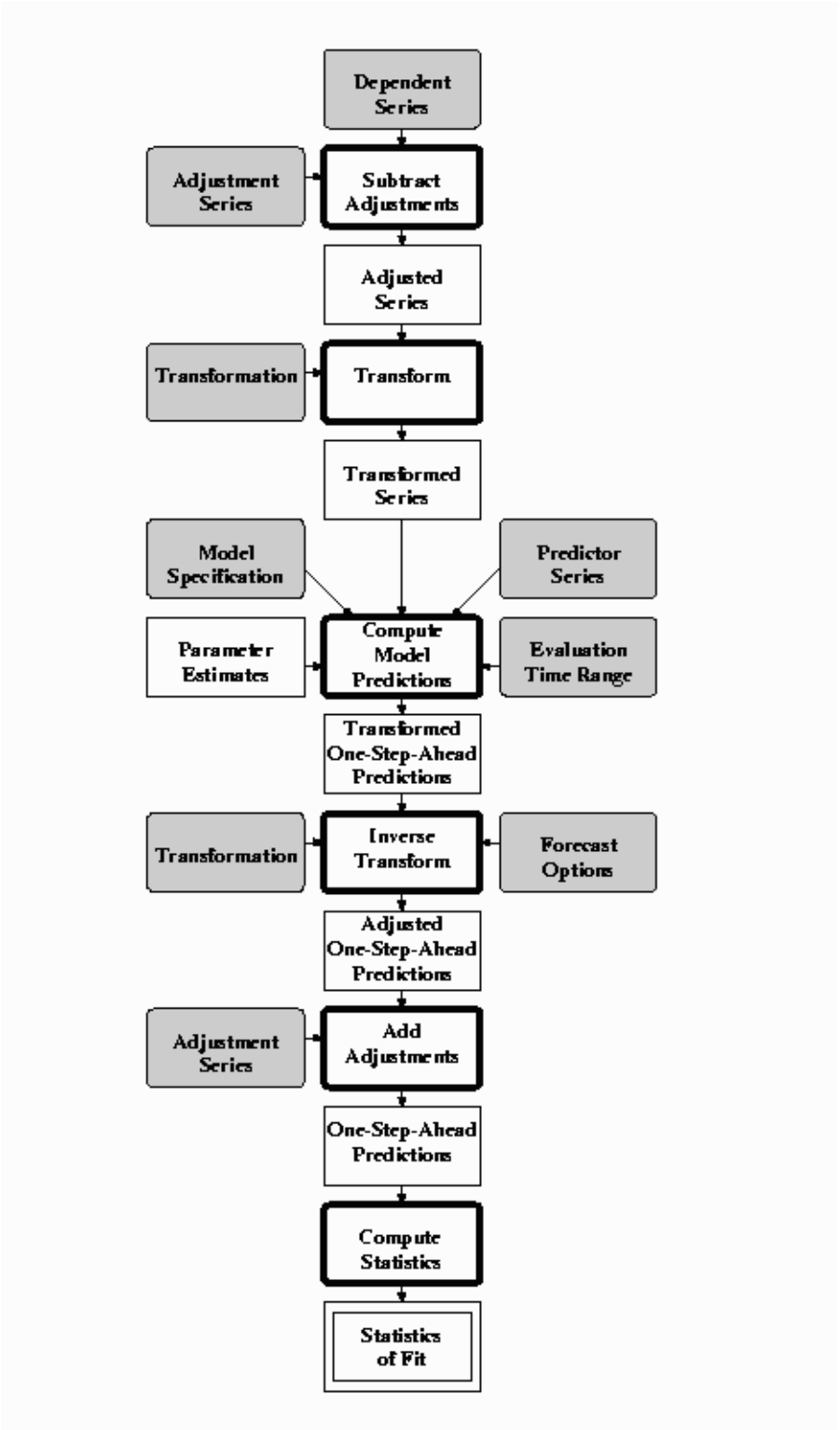


The specification of smoothing and ARIMA models is described in Chapter 56, "Specifying Forecasting Models." Computational details for these kinds of models are provided in the following sections "Smoothing Models" on page 3890 and "ARIMA Models" on page 3901. The results of the parameter estimation process are displayed in the Parameter Estimates table of the Model Viewer windows along with the estimate of the model variance and the final smoothing state.

Model Evaluation

The model evaluation process is described graphically in Figure 61.2.

Figure 61.2 Model Evaluation Flow Diagram

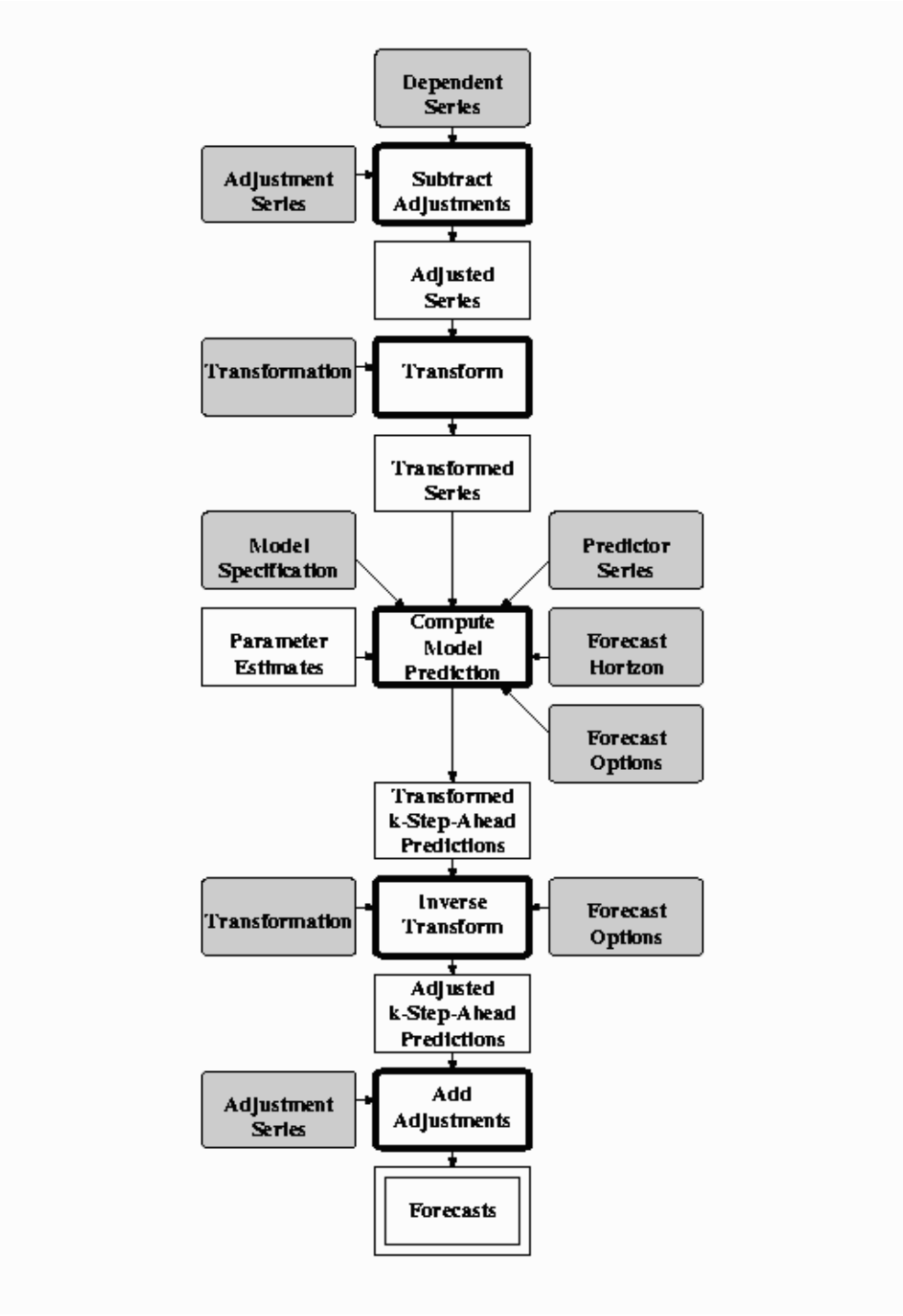


Model evaluation is based on the one-step-ahead prediction errors for observations within the period of evaluation. The one-step-ahead predictions are generated from the model specification and parameter estimates. The predictions are inverse transformed (median or mean) and adjustments are removed. The prediction errors (the difference of the dependent series and the predictions) are used to compute the statistics of fit, which are described in the section "Series Diagnostic Tests" on page 3908. The results generated by the evaluation process are displayed in the Statistics of Fit table of the Model Viewer window.

Forecasting

The forecasting generation process is described graphically in Figure 61.3.

Figure 61.3 Forecasting Flow Diagram



The forecasting process is similar to the model evaluation process described in the preceding section, except that *k*-step-ahead predictions are made from the end of the data through the specified forecast horizon, and prediction standard errors and confidence limits are calculated. The forecasts and confidence limits are displayed in the Forecast plot or table of the Model Viewer window.

Forecast Combination Models

This section discusses the computation of predicted values and confidence limits for forecast combination models. See Chapter 56, "Specifying Forecasting Models," for information about how to specify forecast combination models and their combining weights.

Given the response time series $\{y_t : 1 \leq t \leq n\}$ with previously generated forecasts for the m component models, a combined forecast is created from the component forecasts as follows:

Predictions:	$\hat{y}_t = \sum_{i=1}^{m} w_i \hat{y}_{i,t}$
Prediction Errors:	$\hat{e}_t = y_t - \hat{y}_t$

where $\hat{y}_{i,t}$ are the forecasts of the component models and w_i are the combining weights.

The estimate of the root mean square prediction error and forecast confidence limits for the combined forecast are computed by assuming independence of the prediction errors of the component forecasts, as follows:

Standard Errors:	$\hat{\sigma}_t = \sqrt{\sum_{i=1}^{m} w_i^2 \hat{\sigma}_{i,t}^2}$
Confidence Limits:	$\pm \hat{\sigma}_t Z_{\alpha/2}$

where $\hat{\sigma}_{i,t}$ are the estimated root mean square prediction errors for the component models, α is the confidence limit width, $1 - \alpha$ is the confidence level, and $Z_{\alpha/2}$ is the $\frac{\alpha}{2}$ quantile of the standard normal distribution.

Since, in practice, there might be positive correlation between the prediction errors of the component forecasts, these confidence limits may be too narrow.

External or User-Supplied Forecasts

This section discusses the computation of predicted values and confidence limits for external forecast models.

Given a response time series y_t and external forecast series $\hat{y}_t$, the prediction errors are computed as $\hat{e}_t = y_t - \hat{y}_t$ for those t for which both y_t and $\hat{y}_t$ are nonmissing. The mean squared error (MSE) is computed from the prediction errors.

The variance of the k-step-ahead prediction errors is set to k times the MSE. From these variances, the standard errors and confidence limits are computed in the usual way. If the supplied predictions contain so many missing values within the time range of the response series that the MSE estimate cannot be computed, the confidence limits, standard errors, and statistics of fit are set to missing.

Adjustments

Adjustment predictors are subtracted from the response time series prior to model parameter estimation, evaluation, and forecasting. After the predictions of the adjusted response time series are obtained from the forecasting model, the adjustments are added back to produce the forecasts.

If y_t is the response time series and $X_{i,t}$, $1 \leq i \leq m$ are m adjustment predictor series, then the adjusted response series w_t is

$$w_t = y_t - \sum_{i=1}^{m} X_{i,t}$$

Parameter estimation for the model is performed by using the adjusted response time series w_t. The forecasts $\hat{w}_t$ of w_t are adjusted to obtain the forecasts $\hat{y}_t$ of y_t.

$$\hat{y}_t = \hat{w}_t + \sum_{i=1}^{m} X_{i,t}$$

Missing values in an adjustment series are ignored in these computations.

Series Transformations

For pure ARIMA models, transforming the response time series can aid in obtaining stationary noise series. For general ARIMA models with inputs, transforming the response time series or one or more of the input time series can provide a better model fit. Similarly, the fit of smoothing models can improve when the response series is transformed.

There are four transformations available, for strictly positive series only. Let $y_t > 0$ be the original time series, and let w_t be the transformed series. The transformations are defined as follows:

Log — is the logarithmic transformation,

$$w_t = \ln(y_t)$$

Logistic — is the logistic transformation,

$$w_t = \ln(c y_t / (1 - c y_t))$$

where the scaling factor c is

$$c = (1 - 10^{-6}) 10^{-\text{ceil}(\log_{10}(max(y_t)))}$$

and $\text{ceil}(x)$ is the smallest integer greater than or equal to x.

Square Root — is the square root transformation,

$$w_t = \sqrt{y_t}$$

Box Cox — is the Box-Cox transformation,

$$w_t = \begin{cases} \frac{y_t^{\lambda} - 1}{\lambda}, & \lambda \neq 0 \\ \ln(y_t), & \lambda = 0 \end{cases}$$

Parameter estimation is performed by using the transformed series. The transformed model predictions and confidence limits are then obtained from the transformed time series and these parameter estimates.

The transformed model predictions $\hat{w}_t$ are used to obtain either the minimum mean absolute error (MMAE) or minimum mean squared error (MMSE) predictions $\hat{y}_t$, depending on the setting of the forecast options. The model is then evaluated based on the residuals of the original time series and these predictions. The transformed model confidence limits are inverse-transformed to obtain the forecast confidence limits.

Predictions for Transformed Models

Since the transformations described in the previous section are monotonic, applying the inverse-transformation to the transformed model predictions results in the *median* of the conditional probability density function at each point in time. This is the minimum mean absolute error (MMAE) prediction.

If $w_t = \mathrm{F}(y_t)$ is the transform with inverse-transform $y_t = \mathrm{F}^{-1}(w_t)$, then

$$\text{median}(\hat{y}_t) = \mathrm{F}^{-1}(E\,[w_t]) = \mathrm{F}^{-1}(\hat{w}_t)$$

The minimum mean squared error (MMSE) predictions are the *mean* of the conditional probability density function at each point in time. Assuming that the prediction errors are normally distributed with variance σ_t^2, the MMSE predictions for each of the transformations are as follows:

Log is the conditional expectation of inverse-logarithmic transformation,

$$\hat{y}_t = E\left[e^{w_t}\right] = \exp\left(\hat{w}_t + \sigma_t^2/2\right)$$

Logistic is the conditional expectation of inverse-logistic transformation,

$$\hat{y}_t = E\left[\frac{1}{c(1 + exp(-w_t))}\right]$$

where the scaling factor $c = (1 - e^{-6})10^{-\text{ceil}(\log_{10}(\max(y_t)))}$.

Square Root is the conditional expectation of the inverse-square root transformation,

$$\hat{y}_t = E\left[w_t^2\right] = \hat{w}_t^2 + \sigma_t^2$$

Box Cox is the conditional expectation of the inverse Box-Cox transformation,

$$\hat{y}_t = \begin{cases} E\left[(\lambda w_t + 1)^{1/\lambda}\right], & \lambda \neq 0 \\ E\left[e^{w_t}\right] = \exp(\hat{w}_t + \frac{1}{2}\sigma_t^2), & \lambda = 0 \end{cases}$$

The expectations of the inverse logistic and Box-Cox ($\lambda \neq 0$) transformations do not generally have explicit solutions and are computed by using numerical integration.

Smoothing Models

This section details the computations performed for the exponential smoothing and Winters method forecasting models.

Smoothing Model Calculations

The descriptions and properties of various smoothing methods can be found in Gardner (1985); Chatfield (1978); Bowerman and O'Connell (1979). The following section summarizes the smoothing model computations.

Given a time series $\{Y_t : 1 \leq t \leq n\}$, the underlying model assumed by the smoothing models has the following (additive seasonal) form:

$$Y_t = \mu_t + \beta_t t + s_p(t) + \epsilon_t$$

where

μ_t	represents the time-varying mean term.
β_t	represents the time-varying slope.
$s_p(t)$	represents the time-varying seasonal contribution for one of the p seasons.
ϵ_t	are disturbances.

For smoothing models without trend terms, $\beta_t = 0$; and for smoothing models without seasonal terms, $s_p(t) = 0$. Each smoothing model is described in the following sections.

At each time t, the smoothing models estimate the time-varying components described above with the *smoothing state*. After initialization, the smoothing state is updated for each observation using the *smoothing equations*. The smoothing state at the last nonmissing observation is used for predictions.

Smoothing State and Smoothing Equations

Depending on the smoothing model, the *smoothing state* at time t consists of the following:

L_t is a smoothed level that estimates μ_t.

T_t is a smoothed trend that estimates β_t.

S_{t-j}, $j = 0, \ldots, p-1$, are seasonal factors that estimate $s_p(t)$.

The smoothing process starts with an initial estimate of the smoothing state, which is subsequently updated for each observation by using the *smoothing equations*.

The smoothing equations determine how the smoothing state changes as time progresses. Knowledge of the smoothing state at time $t-1$ and that of the time series value at time t uniquely determine the smoothing state at time t. The *smoothing weights* determine the contribution of the previous smoothing state to the current smoothing state. The smoothing equations for each smoothing model are listed in the following sections.

Smoothing State Initialization

Given a time series $\{Y_t : 1 \leq t \leq n\}$, the smoothing process first computes the smoothing state for time $t = 1$. However, this computation requires an initial estimate of the smoothing state at time $t = 0$, even though no data exists at or before time $t = 0$.

An appropriate choice for the initial smoothing state is made by backcasting from time $t = n$ to $t = 1$ to obtain a prediction at $t = 0$. The initialization for the backcast is obtained by regression with constant and linear terms and seasonal dummies (additive or multiplicative) as appropriate for the smoothing model. For models with linear or seasonal terms, the estimates obtained by the regression are used for initial smoothed trend and seasonal factors; however, the initial smoothed level for backcasting is always set to the last observation, Y_n.

The smoothing state at time $t = 0$ obtained from the backcast is used to initialize the smoothing process from time $t = 1$ to $t = n$ (Chatfield and Yar 1988).

For models with seasonal terms, the smoothing state is normalized so that the seasonal factors S_{t-j} for $j = 0, \ldots, p-1$ sum to zero for models that assume additive seasonality and average to one for models (such as Winters method) that assume multiplicative seasonality.

Missing Values

When a missing value is encountered at time t, the smoothed values are updated using the *error-correction form* of the smoothing equations with the one-step-ahead prediction error, e_t, set to zero. The missing value is estimated using the one-step-ahead prediction at time $t-1$, that is $\hat{Y}_{t-1}(1)$ (Aldrin and Damsleth 1989). The error-correction forms of each of the smoothing models are listed in the following sections.

Predictions and Prediction Errors

Predictions are made based on the last known smoothing state. Predictions made at time t for k steps ahead are denoted $\hat{Y}_t(k)$ and the associated prediction errors are denoted $e_t(k) = Y_{t+k} - \hat{Y}_t(k)$. The *prediction equation* for each smoothing model is listed in the following sections.

The *one-step-ahead predictions* refer to predictions made at time $t-1$ for one time unit into the future—that is, $\hat{Y}_{t-1}(1)$. The *one-step-ahead prediction errors* are more simply denoted $e_t = e_{t-1}(1) = Y_t - \hat{Y}_{t-1}(1)$. The one-step-ahead prediction errors are also the model residuals, and the sum of squares of the one-step-ahead prediction errors is the objective function used in smoothing weight optimization.

The *variance of the prediction errors* are used to calculate the confidence limits (Sweet 1985; McKenzie 1986; Yar and Chatfield 1990; Chatfield and Yar 1991). The equations for the variance of the prediction errors for each smoothing model are listed in the following sections.

Note: $var(\epsilon_t)$ is estimated by the mean square of the one-step-ahead prediction errors.

Smoothing Weights

Depending on the smoothing model, the smoothing weights consist of the following:

α	is a level smoothing weight.
γ	is a trend smoothing weight.
δ	is a seasonal smoothing weight.
ϕ	is a trend damping weight.

Larger smoothing weights (less damping) permit the more recent data to have a greater influence on the predictions. Smaller smoothing weights (more damping) give less weight to recent data.

Specifying the Smoothing Weights

Typically the smoothing weights are chosen to be from zero to one. (This is intuitive because the weights associated with the past smoothing state and the value of current observation would normally sum to one.) However, each smoothing model (except Winters Method—Multiplicative Version) has an ARIMA equivalent. Weights chosen to be within the ARIMA additive-invertible region will guarantee stable predictions (Archibald 1990; Gardner 1985). The ARIMA equivalent and the additive-invertible region for each smoothing model are listed in the following sections.

Optimizing the Smoothing Weights

Smoothing weights are determined so as to minimize the sum of squared, one-step-ahead prediction errors. The optimization is initialized by choosing from a predetermined grid the initial smoothing weights that result in the smallest sum of squared, one-step-ahead prediction errors. The optimization process is highly dependent on this initialization. It is possible that the optimization process will fail due to the inability to obtain stable initial values for the smoothing weights (Greene 1993; Judge et al. 1980), and it is possible for the optimization to result in a local minima.

The optimization process can result in weights to be chosen outside both the zero-to-one range and the ARIMA additive-invertible region. By restricting weight optimization to additive-invertible region, you can obtain a local minimum with stable predictions. Likewise, weight optimization can be restricted to the zero-to-one range or other ranges. It is also possible to fix certain weights to a specific value and optimize the remaining weights.

Standard Errors

The standard errors associated with the smoothing weights are calculated from the Hessian matrix of the sum of squared, one-step-ahead prediction errors with respect to the smoothing weights used in the optimization process.

Weights Near Zero or One

Sometimes the optimization process results in weights near zero or one.

For simple or double (Brown) exponential smoothing, a level weight near zero implies that simple differencing of the time series might be appropriate.

For linear (Holt) exponential smoothing, a level weight near zero implies that the smoothed trend is constant and that an ARIMA model with deterministic trend might be a more appropriate model.

For damped-trend linear exponential smoothing, a damping weight near one implies that linear (Holt) exponential smoothing might be a more appropriate model.

For Winters method and seasonal exponential smoothing, a seasonal weight near one implies that a nonseasonal model might be more appropriate and a seasonal weight near zero implies that deterministic seasonal factors might be present.

Equations for the Smoothing Models

Simple Exponential Smoothing

The model equation for simple exponential smoothing is

$$Y_t = \mu_t + \epsilon_t$$

The smoothing equation is

$$L_t = \alpha Y_t + (1 - \alpha) L_{t-1}$$

The error-correction form of the smoothing equation is

$$L_t = L_{t-1} + \alpha e_t$$

(Note: For missing values, $e_t = 0$.)

The k-step prediction equation is

$$\hat{Y}_t(k) = L_t$$

The ARIMA model equivalency to simple exponential smoothing is the ARIMA(0,1,1) model

$$(1 - B)Y_t = (1 - \theta B)\epsilon_t$$
$$\theta = 1 - \alpha$$

The moving-average form of the equation is

$$Y_t = \epsilon_t + \sum_{j=1}^{\infty} \alpha \epsilon_{t-j}$$

For simple exponential smoothing, the additive-invertible region is

$$\{0 < \alpha < 2\}$$

The variance of the prediction errors is estimated as

$$var(e_t(k)) = var(\epsilon_t) \left[1 + \sum_{j=1}^{k-1} \alpha^2 \right] = var(\epsilon_t)(1 + (k-1)\alpha^2)$$

Double (Brown) Exponential Smoothing

The model equation for double exponential smoothing is

$$Y_t = \mu_t + \beta_t t + \epsilon_t$$

The smoothing equations are

$$L_t = \alpha Y_t + (1-\alpha)L_{t-1}$$
$$T_t = \alpha(L_t - L_{t-1}) + (1-\alpha)T_{t-1}$$

This method can be equivalently described in terms of two successive applications of simple exponential smoothing:

$$S_t^{[1]} = \alpha Y_t + (1-\alpha)S_{t-1}^{[1]}$$
$$S_t^{[2]} = \alpha S_t^{[1]} + (1-\alpha)S_{t-1}^{[2]}$$

where $S_t^{[1]}$ are the smoothed values of Y_t, and $S_t^{[2]}$ are the smoothed values of $S_t^{[1]}$. The prediction equation then takes the form:

$$\hat{Y}_t(k) = (2 + \alpha k/(1-\alpha))S_t^{[1]} - (1 + \alpha k/(1-\alpha))S_t^{[2]}$$

The error-correction forms of the smoothing equations are

$$L_t = L_{t-1} + T_{t-1} + \alpha e_t$$
$$T_t = T_{t-1} + \alpha^2 e_t$$

(Note: For missing values, $e_t = 0$.)

The k-step prediction equation is

$$\hat{Y}_t(k) = L_t + ((k-1) + 1/\alpha)T_t$$

The ARIMA model equivalency to double exponential smoothing is the ARIMA(0,2,2) model,

$$(1-B)^2 Y_t = (1-\theta B)^2 \epsilon_t$$
$$\theta = 1 - \alpha$$

The moving-average form of the equation is

$$Y_t = \epsilon_t + \sum_{j=1}^{\infty} (2\alpha + (j-1)\alpha^2)\epsilon_{t-j}$$

For double exponential smoothing, the additive-invertible region is

$$\{0 < \alpha < 2\}$$

The variance of the prediction errors is estimated as

$$var(e_t(k)) = var(\epsilon_t)\left[1 + \sum_{j=1}^{k-1} (2\alpha + (j-1)\alpha^2)^2\right]$$

Linear (Holt) Exponential Smoothing

The model equation for linear exponential smoothing is

$$Y_t = \mu_t + \beta_t t + \epsilon_t$$

The smoothing equations are

$$L_t = \alpha Y_t + (1-\alpha)(L_{t-1} + T_{t-1})$$
$$T_t = \gamma(L_t - L_{t-1}) + (1-\gamma)T_{t-1}$$

The error-correction form of the smoothing equations is

$$L_t = L_{t-1} + T_{t-1} + \alpha e_t$$
$$T_t = T_{t-1} + \alpha\gamma e_t$$

(Note: For missing values, $e_t = 0$.)

The k-step prediction equation is

$$\hat{Y}_t(k) = L_t + kT_t$$

The ARIMA model equivalency to linear exponential smoothing is the ARIMA(0,2,2) model,

$$(1-B)^2 Y_t = (1 - \theta_1 B - \theta_2 B^2)\epsilon_t$$
$$\theta_1 = 2 - \alpha - \alpha\gamma$$
$$\theta_2 = \alpha - 1$$

The moving-average form of the equation is

$$Y_t = \epsilon_t + \sum_{j=1}^{\infty} (\alpha + j\alpha\gamma)\epsilon_{t-j}$$

For linear exponential smoothing, the additive-invertible region is

$$\{0 < \alpha < 2\}$$
$$\{0 < \gamma < 4/\alpha - 2\}$$

The variance of the prediction errors is estimated as

$$var(e_t(k)) = var(\epsilon_t)\left[1 + \sum_{j=1}^{k-1} (\alpha + j\alpha\gamma)^2\right]$$

Damped-Trend Linear Exponential Smoothing

The model equation for damped-trend linear exponential smoothing is

$$Y_t = \mu_t + \beta_t t + \epsilon_t$$

The smoothing equations are

$$L_t = \alpha Y_t + (1-\alpha)(L_{t-1} + \phi T_{t-1})$$
$$T_t = \gamma(L_t - L_{t-1}) + (1-\gamma)\phi T_{t-1}$$

The error-correction form of the smoothing equations is

$$L_t = L_{t-1} + \phi T_{t-1} + \alpha e_t \; T_t = \phi T_{t-1} + \alpha\gamma e_t$$

(Note: For missing values, $e_t = 0$.)

The k-step prediction equation is

$$\hat{Y}_t(k) = L_t + \sum_{i=1}^{k} \phi^i T_t$$

The ARIMA model equivalency to damped-trend linear exponential smoothing is the ARIMA(1,1,2) model,

$$(1-\phi B)(1-B)Y_t = (1-\theta_1 B - \theta_2 B^2)\epsilon_t$$
$$\theta_1 = 1 + \phi - \alpha - \alpha\gamma\phi$$
$$\theta_2 = (\alpha - 1)\phi$$

The moving-average form of the equation (assuming $|\phi| < 1$) is

$$Y_t = \epsilon_t + \sum_{j=1}^{\infty} (\alpha + \alpha\gamma\phi(\phi^j - 1)/(\phi - 1))\epsilon_{t-j}$$

For damped-trend linear exponential smoothing, the additive-invertible region is

$$\{0 < \alpha < 2\}$$
$$\{0 < \phi\gamma < 4/\alpha - 2\}$$

The variance of the prediction errors is estimated as

$$var(e_t(k)) = var(\epsilon_t)\left[1 + \sum_{j=1}^{k-1} (\alpha + \alpha\gamma\phi(\phi^j - 1)/(\phi - 1))^2\right]$$

Seasonal Exponential Smoothing

The model equation for seasonal exponential smoothing is

$$Y_t = \mu_t + s_p(t) + \epsilon_t$$

The smoothing equations are

$$L_t = \alpha(Y_t - S_{t-p}) + (1-\alpha)L_{t-1}$$
$$S_t = \delta(Y_t - L_t) + (1-\delta)S_{t-p}$$

The error-correction form of the smoothing equations is

$$L_t = L_{t-1} + \alpha e_t$$
$$S_t = S_{t-p} + \delta(1-\alpha)e_t$$

(Note: For missing values, $e_t = 0$.)

The k-step prediction equation is

$$\hat{Y}_t(k) = L_t + S_{t-p+k}$$

The ARIMA model equivalency to seasonal exponential smoothing is the ARIMA(0,1,p+1)(0,1,0)$_p$ model,

$$(1-B)(1-B^p)Y_t = (1-\theta_1 B - \theta_2 B^p - \theta_3 B^{p+1})\epsilon_t$$
$$\theta_1 = 1-\alpha$$
$$\theta_2 = 1-\delta(1-\alpha)$$
$$\theta_3 = (1-\alpha)(\delta-1)$$

The moving-average form of the equation is

$$Y_t = \epsilon_t + \sum_{j=1}^{\infty} \psi_j \epsilon_{t-j}$$

$$\psi_j = \begin{cases} \alpha & \text{for } j \bmod p \neq 0 \\ \alpha + \delta(1-\alpha) & \text{for } j \mod p = 0 \end{cases}$$

For seasonal exponential smoothing, the additive-invertible region is

$$\{\max(-p\alpha, 0) < \delta(1-\alpha) < (2-\alpha)\}$$

The variance of the prediction errors is estimated as

$$var(e_t(k)) = var(\epsilon_t)\left[1 + \sum_{j=1}^{k-1} \psi_j^2\right]$$

Multiplicative Seasonal Smoothing

In order to use the multiplicative version of seasonal smoothing, the time series and all predictions must be strictly positive.

The model equation for the multiplicative version of seasonal smoothing is

$$Y_t = \mu_t s_p(t) + \epsilon_t$$

The smoothing equations are

$$L_t = \alpha(Y_t/S_{t-p}) + (1-\alpha)L_{t-1}$$
$$S_t = \delta(Y_t/L_t) + (1-\delta)S_{t-p}$$

The error-correction form of the smoothing equations is

$$L_t = L_{t-1} + \alpha e_t/S_{t-p}$$
$$S_t = S_{t-p} + \delta(1-\alpha)e_t/L_t$$

(Note: For missing values, $e_t = 0$.)

The k-step prediction equation is

$$\hat{Y}_t(k) = L_t S_{t-p+k}$$

The multiplicative version of seasonal smoothing does not have an ARIMA equivalent; however, when the seasonal variation is small, the ARIMA additive-invertible region of the additive version of seasonal described in the preceding section can approximate the stability region of the multiplicative version.

The variance of the prediction errors is estimated as

$$var(e_t(k)) = var(\epsilon_t)\left[\sum_{i=0}^{\infty}\sum_{j=0}^{p-1}(\psi_{j+ip}S_{t+k}/S_{t+k-j})^2\right]$$

where ψ_j are as described for the additive version of seasonal method, and $\psi_j = 0$ for $j \geq k$.

Winters Method—Additive Version

The model equation for the additive version of Winters method is

$$Y_t = \mu_t + \beta_t t + s_p(t) + \epsilon_t$$

The smoothing equations are

$$L_t = \alpha(Y_t - S_{t-p}) + (1-\alpha)(L_{t-1} + T_{t-1})$$
$$T_t = \gamma(L_t - L_{t-1}) + (1-\gamma)T_{t-1}$$
$$S_t = \delta(Y_t - L_t) + (1-\delta)S_{t-p}$$

The error-correction form of the smoothing equations is

$$L_t = L_{t-1} + T_{t-1} + \alpha e_t$$
$$T_t = T_{t-1} + \alpha\gamma e_t$$
$$S_t = S_{t-p} + \delta(1-\alpha)e_t$$

(Note: For missing values, $e_t = 0$.)

The k-step prediction equation is

$$\hat{Y}_t(k) = L_t + kT_t + S_{t-p+k}$$

The ARIMA model equivalency to the additive version of Winters method is the ARIMA(0,1,p+1)(0,1,0)$_p$ model,

$$(1-B)(1-B^p)Y_t = \left[1 - \sum_{i=1}^{p+1} \theta_i B^i\right]\epsilon_t$$

$$\theta_j = \begin{cases} 1-\alpha-\alpha\gamma & j = 1 \\ -\alpha\gamma & 2 \le j \le p-1 \\ 1-\alpha\gamma-\delta(1-\alpha) & j = p \\ (1-\alpha)(\delta-1) & j = p+1 \end{cases}$$

The moving-average form of the equation is

$$Y_t = \epsilon_t + \sum_{j=1}^{\infty} \psi_j \epsilon_{t-j}$$

$$\psi_j = \begin{cases} \alpha + j\alpha\gamma & \text{for} j \mod p \neq 0 \\ \alpha + j\alpha\gamma + \delta(1-\alpha), & \text{for} j \mod p = 0 \end{cases}$$

For the additive version of Winters method (Archibald 1990), the additive-invertible region is

$$\{\max(-p\alpha, 0) < \delta(1-\alpha) < (2-\alpha)\}$$
$$\{0 < \alpha\gamma < 2-\alpha-\delta(1-\alpha)(1-cos(\vartheta)\}$$

where ϑ is the smallest nonnegative solution to the equations listed in Archibald (1990).

The variance of the prediction errors is estimated as

$$var(e_t(k)) = var(\epsilon_t)\left[1 + \sum_{j=1}^{k-1} \psi_j^2\right]$$

Winters Method—Multiplicative Version

In order to use the multiplicative version of Winters method, the time series and all predictions must be strictly positive.

The model equation for the multiplicative version of Winters method is

$$Y_t = (\mu_t + \beta_t t)s_p(t) + \epsilon_t$$

The smoothing equations are

$$L_t = \alpha(Y_t/S_{t-p}) + (1-\alpha)(L_{t-1} + T_{t-1})$$
$$T_t = \gamma(L_t - L_{t-1}) + (1-\gamma)T_{t-1}$$
$$S_t = \delta(Y_t/L_t) + (1-\delta)S_{t-p}$$

The error-correction form of the smoothing equations is

$$L_t = L_{t-1} + T_{t-1} + \alpha e_t/S_{t-p}$$
$$T_t = T_{t-1} + \alpha\gamma e_t/S_{t-p}$$
$$S_t = S_{t-p} + \delta(1-\alpha)e_t/L_t$$

NOTE: For missing values, $e_t = 0$.

The k-step prediction equation is

$$\hat{Y}_t(k) = (L_t + kT_t)S_{t-p+k}$$

The multiplicative version of Winters method does not have an ARIMA equivalent; however, when the seasonal variation is small, the ARIMA additive-invertible region of the additive version of Winters method described in the preceding section can approximate the stability region of the multiplicative version.

The variance of the prediction errors is estimated as

$$var(e_t(k)) = var(\epsilon_t)\left[\sum_{i=0}^{\infty}\sum_{j=0}^{p-1}(\psi_{j+ip}S_{t+k}/S_{t+k-j})^2\right]$$

where ψ_j are as described for the additive version of Winters method and $\psi_j = 0$ for $j \geq k$.

ARIMA Models

Autoregressive integrated moving-average (ARIMA) models predict values of a dependent time series with a linear combination of its own past values, past errors (also called shocks or innovations), and current and past values of other time series (predictor time series).

The Time Series Forecasting System uses the ARIMA procedure of SAS/ETS software to fit and forecast ARIMA models. The maximum likelihood method is used for parameter estimation. Refer to Chapter 8, "The ARIMA Procedure," for details of ARIMA model estimation and forecasting.

This section summarizes the notation used for ARIMA models.

Notation for ARIMA Models

A dependent time series that is modeled as a linear combination of its own past values and past values of an error series is known as a (pure) ARIMA model.

Nonseasonal ARIMA Model Notation

The order of an ARIMA model is usually denoted by the notation ARIMA(p,d,q), where

p	is the order of the autoregressive part.
d	is the order of the differencing (rarely should $d > 2$ be needed).
q	is the order of the moving-average process.

Given a dependent time series $\{Y_t : 1 \leq t \leq n\}$, mathematically the ARIMA model is written as

$$(1 - B)^d Y_t = \mu + \frac{\theta(B)}{\phi(B)} a_t$$

where

t	indexes time.
μ	is the mean term.
B	is the backshift operator; that is, $BX_t = X_{t-1}$.
$\phi(B)$	is the autoregressive operator, represented as a polynomial in the back shift operator: $\phi(B) = 1 - \phi_1 B - \ldots - \phi_p B^p$.
$\theta(B)$	is the moving-average operator, represented as a polynomial in the back shift operator: $\theta(B) = 1 - \theta_1 B - \ldots - \theta_q B^q$.
a_t	is the independent disturbance, also called the random error.

For example, the mathematical form of the ARIMA(1,1,2) model is

$$(1 - B)Y_t = \mu + \frac{(1 - \theta_1 B - \theta_2 B^2)}{(1 - \phi_1 B)} a_t$$

Seasonal ARIMA Model Notation

Seasonal ARIMA models are expressed in factored form by the notation ARIMA(p,d,q)(P,D,Q)$_s$, where

P	is the order of the seasonal autoregressive part.
D	is the order of the seasonal differencing (rarely should $D > 1$ be needed).
Q	is the order of the seasonal moving-average process.
s	is the length of the seasonal cycle.

Given a dependent time series $\{Y_t : 1 \leq t \leq n\}$, mathematically the ARIMA seasonal model is written as

$$(1 - B)^d (1 - B^s)^D Y_t = \mu + \frac{\theta(B)\theta_s(B^s)}{\phi(B)\phi_s(B^s)} a_t$$

where

$\phi_s(B^s)$	is the seasonal autoregressive operator, represented as a polynomial in the back shift operator: $\phi_s(B^s) = 1 - \phi_{s,1}B^s - \ldots - \phi_{s,P}B^{sP}$
$\theta_s(B^s)$	is the seasonal moving-average operator, represented as a polynomial in the back shift operator: $\theta_s(B^s) = 1 - \theta_{s,1}B^s - \ldots - \theta_{s,Q}B^{sQ}$

For example, the mathematical form of the ARIMA(1,0,1)(1,1,2)$_{12}$ model is

$$(1 - B^{12})Y_t = \mu + \frac{(1 - \theta_1 B)(1 - \theta_{s,1}B^{12} - \theta_{s,2}B^{24})}{(1 - \phi_1 B)(1 - \phi_{s,1}B^{12})}a_t$$

Abbreviated Notation for ARIMA Models

If the differencing order, autoregressive order, or moving-average order is zero, the notation is further abbreviated as

I(d)(D)$_s$	integrated model or ARIMA(*0,d,0*)(*0,D,0*)
AR(p)(P)$_s$	autoregressive model or ARIMA(*p,0,0*)(*P,0,0*)
IAR(p,d)(P,D)$_s$	integrated autoregressive model or ARIMA(*p,d,0*)(*P,D,0*)$_s$
MA(q)(Q)$_s$	moving average model or ARIMA(*0,0,q*)(*0,0,Q*)$_s$
IMA(d,q)(D,Q)$_s$	integrated moving average model or ARIMA(*0,d,q*)(*0,D,Q*)$_s$
ARMA(p,q)(P,Q)$_s$	autoregressive moving-average model or ARIMA(*p,0,q*)(*P,0,Q*)$_s$.

Notation for Transfer Functions

A transfer function can be used to filter a predictor time series to form a dynamic regression model.

Let Y_t be the dependent series, let X_t be the predictor series, and let $\Psi(B)$ be a linear filter or transfer function for the effect of X_t on Y_t. The ARIMA model is then

$$(1 - B)^d(1 - B^s)^D Y_t = \mu + \Psi(B)(1 - B)^d(1 - B^s)^D X_t + \frac{\theta(B)\theta_s(B^s)}{\phi(B)\phi_s(B^s)}a_t$$

This model is called a *dynamic regression* of Y_t on X_t.

Nonseasonal Transfer Function Notation

Given the ith predictor time series $\{X_{i,t} : 1 \leq t \leq n\}$, the transfer function is written as

$$\mathrm{Dif}(d_i)\mathrm{Lag}(k_i)\mathrm{N}(q_i)/\mathrm{D}(p_i)$$

where

d_i	is the simple order of the differencing for the ith predictor time series, $(1-B)^{d_i} X_{i,t}$ (rarely should $d_i > 2$ be needed).
k_i	is the pure time delay (lag) for the effect of the ith predictor time series, $X_{i,t} B^{k_i} = X_{i,t-k_i}$.
p_i	is the simple order of the denominator for the ith predictor time series.
q_i	is the simple order of the numerator for the ith predictor time series.

The mathematical notation used to describe a transfer function is

$$\Psi_i(B) = \frac{\omega_i(B)}{\delta_i(B)}(1-B)^{d_i} B^{k_i}$$

where

B	is the backshift operator; that is, $BX_t = X_{t-1}$.
$\delta_i(B)$	is the denominator polynomial of the transfer function for the ith predictor time series: $\delta_i(B) = 1 - \delta_{i,1}B - \ldots - \delta_{i,p_i} B^{p_i}$.
$\omega_i(B)$	is the numerator polynomial of the transfer function for the ith predictor time series: $\omega_i(B) = 1 - \omega_{i,1}B - \ldots - \omega_{i,q_i} B^{q_i}$.

The numerator factors for a transfer function for a predictor series are like the MA part of the ARMA model for the noise series. The denominator factors for a transfer function for a predictor series are like the AR part of the ARMA model for the noise series. Denominator factors introduce exponentially weighted, infinite distributed lags into the transfer function.

For example, the transfer function for the ith predictor time series with

$k_i = 3$	time lag is 3
$d_i = 1$	simple order of differencing is one
$p_i = 1$	simple order of the denominator is one
$q_i = 2$	simple order of the numerator is two

would be written as [Dif(1)Lag(3)N(2)/D(1)]. The mathematical notation for the transfer function in this example is

$$\Psi_i(B) = \frac{(1 - \omega_{i,1}B - \omega_{i,2}B^2)}{(1 - \delta_{i,1}B)}(1-B)B^3$$

Seasonal Transfer Function Notation

The general transfer function notation for the ith predictor time series $X_{i,t}$ with seasonal factors is [Dif(d_i)(D_i)$_s$ Lag(k_i) N(q_i)(Q_i)$_s$/ D(p_i)(P_i)$_s$] where

D_i	is the seasonal order of the differencing for the ith predictor time series (rarely should $D_i > 1$ be needed).
P_i	is the seasonal order of the denominator for the ith predictor time series (rarely should $P_i > 2$ be needed).

Q_i	is the seasonal order of the numerator for the *i*th predictor time series, (rarely should $Q_i > 2$ be needed).
s	is the length of the seasonal cycle.

The mathematical notation used to describe a seasonal transfer function is

$$\Psi_i(B) = \frac{\omega_i(B)\omega_{s,i}(B^s)}{\delta_i(B)\delta_{s,i}(B^s)}(1-B)^{d_i}(1-B^s)^{D_i}B^{k_i}$$

where

$\delta_{s,i}(B^s)$	is the denominator seasonal polynomial of the transfer function for the *i*th predictor time series: $\delta_{s,i}(B) = 1 - \delta_{s,i,1}B - \ldots - \delta_{s,i,P_i}B^{sP_i}$
$\omega_{s,i}(B^s)$	is the numerator seasonal polynomial of the transfer function for the *i*th predictor time series: $\omega_{s,i}(B) = 1 - \omega_{s,i,1}B - \ldots - \omega_{s,i,Q_i}B^{sQ_i}$

For example, the transfer function for the *i*th predictor time series $X_{i,t}$ whose seasonal cycle $s = 12$ with

$d_i = 2$	simple order of differencing is two
$D_i = 1$	seasonal order of differencing is one
$q_i = 2$	simple order of the numerator is two
$Q_i = 1$	seasonal order of the numerator is one

would be written as [Dif(2)(1)$_s$ N(2)(1)$_s$]. The mathematical notation for the transfer function in this example is

$$\Psi_i(B) = (1 - \omega_{i,1}B - \omega_{i,2}B^2)(1 - \omega_{s,i,1}B^{12})(1-B)^2(1-B^{12})$$

Note: In this case, [Dif(2)(1)$_s$ N(2)(1)$_s$] = [Dif(2)(1)$_s$Lag(0)N(2)(1)$_s$/D(0)(0)$_s$].

Predictor Series

This section discusses time trend curves, seasonal dummies, interventions, and adjustments.

Time Trend Curves

When you specify a time trend curve as a predictor in a forecasting model, the system computes a predictor series that is a deterministic function of time. This variable is then included in the model as a regressor, and the trend curve is fit to the dependent series by linear regression, in addition to other predictor series.

Some kinds of nonlinear trend curves are fit by transforming the dependent series. For example, the exponential trend curve is actually a linear time trend fit to the logarithm of the series. For these trend curve

specifications, the series transformation option is set automatically, and you cannot independently control both the time trend curve and transformation option.

The computed time trend variable is included in the output data set in a variable named in accordance with the trend curve type. Let t represent the observation count from the start of the period of fit for the model, and let X_t represent the value of the time trend variable at observation t within the period of fit. The names and definitions of these variables are as follows. (Note: These deterministic variables are reserved variable names.)

Linear trend	variable name _LINEAR_, with $X_t = t - c$
Quadratic trend	variable name _QUAD_, with $X_t = (t - c)^2$. Note that a quadratic trend implies a linear trend as a special case and results in two regressors: _QUAD_ and _LINEAR_.
Cubic trend	variable name _CUBE_, with $X_t = (t - c)^3$. Note that a cubic trend implies a quadratic trend as a special case and results in three regressors: _CUBE_, _QUAD_, and _LINEAR_.
Logistic trend	variable name _LOGIT_, with $X_t = t$. The model is a linear time trend applied to the logistic transform of the dependent series. Thus, specifying a logistic trend is equivalent to specifying the logistic series transformation and a linear time trend. A logistic trend predictor can be used only in conjunction with the logistic transformation, which is set automatically when you specify logistic trend.
Logarithmic trend	variable name _LOG_, with $X_t = ln(t)$
Exponential trend	variable name _EXP_, with $X_t = t$. The model is a linear time trend applied to the logarithms of the dependent series. Thus, specifying an exponential trend is equivalent to specifying the log series transformation and a linear time trend. An exponential trend predictor can be used only in conjunction with the log transformation, which is set automatically when you specify exponential trend.
Hyperbolic trend	variable name _HYP_, with $X_t = 1/t$
Power curve trend	variable name _POW_, with $X_t = ln(t)$. The model is a logarithmic time trend applied to the logarithms of the dependent series. Thus, specifying a power curve is equivalent to specifying the log series transformation and a logarithmic time trend. A power curve predictor can be used only in conjunction with the log transformation, which is set automatically when you specify a power curve trend.
EXP(A+B/TIME) trend	variable name _ERT_, with $X_t = 1/t$. The model is a hyperbolic time trend applied to the logarithms of the dependent series. Thus, specifying this trend curve is equivalent to specifying the log series transformation and a hyperbolic time trend. This trend curve can be used only in conjunction with the log transformation, which is set automatically when you specify this trend.

Intervention Effects

Interventions are used for modeling events that occur at specific times. That is, they are known changes that affect the dependent series or outliers.

The *i*th intervention series is included in the output data set with variable name _INTV*i* _, which is a reserved variable name.

Point Interventions

The point intervention is a one-time event. The *i*th intervention series $X_{i,t}$ has a point intervention at time t_{int} when the series is nonzero only at time t_{int}—that is,

$$X_{i,t} = \begin{cases} 1, & t = t_{int} \\ 0, & otherwise \end{cases}$$

Step Interventions

Step interventions are continuing, and the input time series flags periods after the intervention. For a step intervention, before time t_{int}, the *i*th intervention series $X_{i,t}$ is zero and then steps to a constant level thereafter—that is,

$$X_{i,t} = \begin{cases} 1, & t \geq t_{int} \\ 0, & otherwise \end{cases}$$

Ramp Interventions

A ramp intervention is a continuing intervention that increases linearly after the intervention time. For a ramp intervention, before time t_{int}, the *i*th intervention series $X_{i,t}$ is zero and increases linearly thereafter—that is, proportional to time.

$$X_{i,t} = \begin{cases} t - t_{int}, & t \geq t_{int} \\ 0, & otherwise \end{cases}$$

Intervention Effect

Given the *i*th intervention series $X_{i,t}$, you can define how the intervention takes effect by filters (transfer functions) of the form

$$\Psi_i(B) = \frac{1 - \omega_{i,1}B - \ldots - \omega_{i,q_i}B^{q_i}}{1 - \delta_{i,1}B - \ldots - \delta_{i,p_i}B^{p_i}}$$

where B is the backshift operator $By_t = y_{t-1}$.

The denominator of the transfer function determines the decay pattern of the intervention effect, whereas the numerator terms determine the size of the intervention effect time window.

For example, the following intervention effects are associated with the respective transfer functions.

Immediately	$\Psi_i(B) = 1$
Gradually	$\Psi_i(B) = 1/(1 - \delta_{i,1}B)$
1 lag window	$\Psi_i(B) = 1 - \omega_{i,1}B$
3 lag window	$\Psi_i(B) = 1 - \omega_{i,1}B - \omega_{i,2}B^2 - \omega_{i,3}B^3$

Intervention Notation

The notation used to describe intervention effects has the form *type* :t_{int} $(q_i)/(p_i)$, where *type* is point, step, or ramp; t_{int} is the time of the intervention (for example, OCT87); q_i is the transfer function numerator order; and p_i is the transfer function denominator order. If $q_i = 0$, the part "(q_i)" is omitted; if $p_i = 0$, the part "$/(p_i)$" is omitted.

In the Intervention Specification window, the `Number of Lags` option specifies the transfer function numerator order q_i, and the `Effect Decay Pattern` option specifies the transfer function denominator order p_i. In the `Effect Decay Pattern options`, values and resulting p_i are: `None`, $p_i = 0$; `Exp`, $p_i = 1$; `Wave`, $p_i = 2$.

For example, a step intervention with date 08MAR90 and effect pattern `Exp` is denoted "Step:08MAR90/(1)" and has a transfer function filter $\Psi_i(B) = 1/(1 - \delta_1 B)$. A ramp intervention immediately applied on 08MAR90 is denoted "Ramp:08MAR90" and has a transfer function filter $\Psi_i(B) = 1$.

Seasonal Dummy Inputs

For a seasonal cycle of length s, the seasonal dummy regressors include

$$\{X_{i,t} : 1 \le i \le (s-1), 1 \le t \le n\}$$

for models that include an intercept term and

$$\{X_{i,t} : 1 \le i \le s, 1 \le t \le n\}$$

for models that exclude an intercept term. Each element of a seasonal dummy regressor is either zero or one, based on the following rule:

$$X_{i,t} = \begin{cases} 1, & \text{when } i = t \mod s \\ 0, & \text{otherwise} \end{cases}$$

Note that if the model includes an intercept term, the number of seasonal dummy regressors is one less than s to ensure that the linear system is full rank.

The seasonal dummy variables are included in the output data set with variable names prefixed with "SDUMMYi" and sequentially numbered. They are reserved variable names.

Series Diagnostic Tests

This section describes the diagnostic tests that are used to determine the kinds of forecasting models appropriate for a series.

The series diagnostics are a set of heuristics that provide recommendations on whether or not the forecasting model should contain a log transform, trend terms, and seasonal terms. These recommendations are used by the automatic model selection process to restrict the model search to a subset of the model selection list. (You can disable this behavior by using the Automatic Model Selection Options window.)

The tests that are used by the series diagnostics do not always produce the correct classification of the series. They are intended to accelerate the process of searching for a good forecasting model for the series, but you should not rely on them if finding the very best model is important to you.

If you have information about the appropriate kinds of forecasting models (perhaps from studying the plots and autocorrelations shown in the Series Viewer window), you can set the series diagnostic flags in the Series Diagnostics window. Select the YES, NO, or MAYBE values for the `Log Transform`, `Trend`, and `Seasonality` options in the Series Diagnostics window as you think appropriate.

The series diagnostics tests are intended as a heuristic tool only, and no statistical validity is claimed for them. These tests might be modified and enhanced in future releases of the Time Series Forecasting System. The testing strategy is as follows:

1. **Log transform test.** The log test fits a high-order autoregressive model to the series and to the log of the series and compares goodness-of-fit measures for the prediction errors of the two models. If this test finds that log transforming the series is suitable, the `Log Transform` option is set to YES, and the subsequent diagnostic tests are performed on the log transformed series.

2. **Trend test.** The resultant series is tested for presence of a trend by using an augmented Dickey-Fuller test and a random walk with drift test. If either test finds that the series appears to have a trend, the `Trend` option is set to YES, and the subsequent diagnostic tests are performed on the differenced series.

3. **Seasonality test.** The resultant series is tested for seasonality. A seasonal dummy model with AR(1) errors is fit and the joint significance of the seasonal dummy estimates is tested. If the seasonal dummies are significant, the AIC statistic for this model is compared to the AIC for and AR(1) model without seasonal dummies. If the AIC for the seasonal model is lower than that of the nonseasonal model, the `Seasonal` option is set to YES.

Statistics of Fit

This section explains the goodness-of-fit statistics reported to measure how well different models fit the data. The statistics of fit for the various forecasting models can be viewed or stored in a data set by using the Model Viewer window.

Statistics of fit are computed by using the actual and forecasted values for observations in the period of evaluation. One-step forecasted values are used whenever possible, including the case when a hold-out sample contains no missing values. If a one-step forecast for an observation cannot be computed due to missing values for previous series observations, a multi-step forecast is computed, using the minimum number of steps as the previous nonmissing values in the data range permit.

The various statistics of fit reported are as follows. In these formulas, n is the number of nonmissing observations and k is the number of fitted parameters in the model.

Number of Nonmissing Observations.
The number of nonmissing observations used to fit the model.

Number of Observations.
The total number of observations used to fit the model, including both missing and nonmissing observations.

Number of Missing Actuals.
The number of missing actual values.

Number of Missing Predicted Values.
The number of missing predicted values.

Number of Model Parameters.
The number of parameters fit to the data. For combined forecast, this is the number of forecast components.

Total Sum of Squares (Uncorrected).
The total sum of squares for the series, SST, uncorrected for the mean: $\sum_{t=1}^{n} y_t^2$.

Total Sum of Squares (Corrected).
The total sum of squares for the series, SST, corrected for the mean: $\sum_{t=1}^{n} (y_t - \overline{y})^2$, where $\overline{y}$ is the series mean.

Sum of Square Errors.
The sum of the squared prediction errors, SSE. $SSE = \sum_{t=1}^{n} (y_t - \hat{y}_t)^2$, where $\hat{y}$ is the one-step predicted value.

Mean Squared Error.
The mean squared prediction error, MSE, calculated from the one-step-ahead forecasts. $MSE = \frac{1}{n} SSE$. This formula enables you to evaluate small hold-out samples.

Root Mean Squared Error.
The root mean square error (RMSE), $\sqrt{MSE}$.

Mean Absolute Percent Error.
The mean absolute percent prediction error (MAPE), $\frac{100}{n} \sum_{t=1}^{n} |(y_t - \hat{y}_t)/y_t|$.
The summation ignores observations where $y_t = 0$.

Mean Absolute Error.
The mean absolute prediction error, $\frac{1}{n} \sum_{t=1}^{n} |y_t - \hat{y}_t|$.

R-Square.
The R^2 statistic, $R^2 = 1 - SSE/SST$. If the model fits the series badly, the model error sum of squares, *SSE*, can be larger than *SST* and the R^2 statistic will be negative.

Adjusted R-Square.
The adjusted R^2 statistic, $1 - (\frac{n-1}{n-k})(1 - R^2)$.

Amemiya's Adjusted R-Square.
Amemiya's adjusted R^2, $1 - (\frac{n+k}{n-k})(1 - R^2)$.

Random Walk R-Square.
The random walk R^2 statistic (Harvey's R^2 statistic by using the random walk model for comparison), $1 - (\frac{n-1}{n}) SSE/RWSSE$, where $RWSSE = \sum_{t=2}^{n} (y_t - y_{t-1} - \mu)^2$, and $\mu = \frac{1}{n-1} \sum_{t=2}^{n} (y_t - y_{t-1})$.

Akaike's Information Criterion.
Akaike's information criterion (AIC), $n \ln(MSE) + 2k$.

Schwarz Bayesian Information Criterion.
Schwarz Bayesian information criterion (SBC or BIC),
$n \ln(MSE) + k \ln(n)$.

Amemiya's Prediction Criterion.
Amemiya's prediction criterion, $\frac{1}{n}SST(\frac{n+k}{n-k})(1-R^2) = (\frac{n+k}{n-k})\frac{1}{n}SSE$.

Maximum Error.
The largest prediction error.

Minimum Error.
The smallest prediction error.

Maximum Percent Error.
The largest percent prediction error, $100 \max((y_t - \hat{y}_t)/y_t)$. The summation ignores observations where $y_t = 0$.

Minimum Percent Error.
The smallest percent prediction error, $100 \min((y_t - \hat{y}_t)/y_t)$. The summation ignores observations where $y_t = 0$.

Mean Error.
The mean prediction error, $\frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)$.

Mean Percent Error.
The mean percent prediction error, $\frac{100}{n}\sum_{t=1}^{n}\frac{(y_t-\hat{y}_t)}{y_t}$. The summation ignores observations where $y_t = 0$.

References

Akaike, H. (1974). "A New Look at the Statistical Model Identification." *IEEE Transactions on Automatic Control* AC-19:716–723.

Aldrin, M., and Damsleth, E. (1989). "Forecasting Non-seasonal Time Series with Missing Observations." *Journal of Forecasting* 8:97–116.

Anderson, T. W. (1971). *The Statistical Analysis of Time Series*. New York: John Wiley & Sons.

Ansley, C. F. (1979). "An Algorithm for the Exact Likelihood of a Mixed Autoregressive–Moving Average Process." *Biometrika* 66:59–65.

Ansley, C. F., and Newbold, P. (1980). "Finite Sample Properties of Estimators for Autoregressive Moving-Average Models." *Journal of Econometrics* 13:159–183.

Archibald, B. C. (1990). "Parameter Space of the Holt-Winters' Model." *International Journal of Forecasting* 6:199–209.

Bartolomei, S. M., and Sweet, A. L. (1989). "A Note on the Comparison of Exponential Smoothing Methods for Forecasting Seasonal Series." *International Journal of Forecasting* 5:111–116.

Bhansali, R. J. (1980). "Autoregressive and Window Estimates of the Inverse Correlation Function." *Biometrika* 67:551–566.

Bowerman, B. L., and O'Connell, R. T. (1979). *Time Series and Forecasting: An Applied Approach*. North Scituate, MA: Duxbury Press.

Box, G. E. P., and Cox, D. R. (1964). "An Analysis of Transformations." *Journal of the Royal Statistical Society, Series B* 26:211–234.

Box, G. E. P., and Jenkins, G. M. (1976). *Time Series Analysis: Forecasting and Control*. Rev. ed. San Francisco: Holden-Day.

Box, G. E. P., and Tiao, G. C. (1975). "Intervention Analysis with Applications to Economic and Environmental Problems." *Journal of the American Statistical Association* 70:70–79.

Brocklebank, J. C., and Dickey, D. A. (1986). *SAS System for Forecasting Time Series*. 1986 ed. Cary, NC: SAS Institute Inc.

Brown, R. G. (1962). *Smoothing, Forecasting, and Prediction of Discrete Time Series*. New York: Prentice-Hall.

Brown, R. G., and Meyer, R. F. (1961). "The Fundamental Theorem of Exponential Smoothing." *Operations Research* 9:673–685.

Chatfield, C. (1978). "The Holt-Winters Forecasting Procedure." *Journal of the Royal Statistical Society, Series C* 27:264–279.

Chatfield, C., and Prothero, D. L. (1973). "Box-Jenkins Seasonal Forecasting: Problems in a Case Study." *Journal of the Royal Statistical Society, Series A* 136:295–315.

Chatfield, C., and Yar, M. (1988). "Holt-Winters Forecasting: Some Practical Issues." *The Statistician* 37:129–140.

Chatfield, C., and Yar, M. (1991). "Prediction Intervals for Multiplicative Holt-Winters." *International Journal of Forecasting* 7:31–37.

Cogger, K. O. (1974). "The Optimality of General-Order Exponential Smoothing." *Operations Research* 22:858–867.

Cox, D. R. (1961). "Prediction by Exponentially Weighted Moving Averages and Related Methods." *Journal of the Royal Statistical Society, Series B* 23:414–422.

Davidson, J. (1981). "Problems with the Estimation of Moving Average Models." *Journal of Econometrics* 16:295.

Dickey, D. A., and Fuller, W. A. (1979). "Distribution of the Estimators for Autoregressive Time Series with a Unit Root." *Journal of the American Statistical Association* 74:427–431.

Dickey, D. A., Hasza, D. P., and Fuller, W. A. (1984). "Testing for Unit Roots in Seasonal Time Series." *Journal of the American Statistical Association* 79:355–367.

Fair, R. C. (1986). "Evaluating the Predictive Accuracy of Models." In *Handbook of Econometrics*, vol. 3, edited by Z. Griliches, and M. D. Intriligator, 1979–1995. New York: North-Holland.

Fildes, R. (1979). "Quantitative Forecasting—the State of the Art: Extrapolative Models." *Journal of the Operational Research Society* 30:691–710.

Fuller, W. A. (1976). *Introduction to Statistical Time Series*. New York: John Wiley & Sons.

Gardner, E. S., Jr. (1984). "The Strange Case of the Lagging Forecasts." *Interfaces* 14:47–50.

Gardner, E. S., Jr. (1985). "Exponential Smoothing: The State of the Art." *Journal of Forecasting* 4:1–38.

Granger, C. W. J., and Newbold, P. (1977). *Forecasting Economic Time Series*. New York: Academic Press.

Greene, W. H. (1993). *Econometric Analysis*. 2nd ed. New York: Macmillan.

Hamilton, J. D. (1994). *Time Series Analysis*. Princeton, NJ: Princeton University Press.

Harvey, A. C. (1981). *Time Series Models*. New York: John Wiley & Sons.

Harvey, A. C. (1984). "A Unified View of Statistical Forecasting Procedures." *Journal of Forecasting* 3:245–275.

Hopewood, W. S., McKeown, J. C., and Newbold, P. (1984). "Time Series Forecasting Models Involving Power Transformations." *Journal of Forecasting* 3:57–61.

Jones, R. H. (1980). "Maximum Likelihood Fitting of ARMA Models to Time Series with Missing Observations." *Technometrics* 22:389–396.

Judge, G. G., Griffiths, W. E., Hill, R. C., and Lee, T.-C. (1980). *The Theory and Practice of Econometrics*. New York: John Wiley & Sons.

Ledolter, J., and Abraham, B. (1984). "Some Comments on the Initialization of Exponential Smoothing." *Journal of Forecasting* 3:79–84.

Ljung, G. M., and Box, G. E. P. (1978). "On a Measure of Lack of Fit in Time Series Models." *Biometrika* 65:297–303.

Makridakis, S. G., Wheelwright, S. C., and McGee, V. E. (1983). *Forecasting: Methods and Applications*. 2nd ed. New York: John Wiley & Sons.

McKenzie, E. (1984). "General Exponential Smoothing and the Equivalent ARMA Process." *Journal of Forecasting* 3:333–344.

McKenzie, E. (1986). "Error Analysis for Winters' Additive Seasonal Forecasting System." *International Journal of Forecasting* 2:373–382.

Montgomery, D. C., and Johnson, L. A. (1976). *Forecasting and Time Series Analysis*. New York: McGraw-Hill.

Morf, M., Sidhu, G. S., and Kailath, T. (1974). "Some New Algorithms for Recursive Estimation on Constant Linear Discrete Time Systems." *IEEE Transactions on Automatic Control* 19:315–323.

Nelson, C. R. (1973). *Applied Time Series for Managerial Forecasting*. San Francisco: Holden-Day.

Newbold, P. (1981). "Some Recent Developments in Time Series Analysis." *International Statistical Review* 49:53–66.

Newton, H. J., and Pagano, M. (1983). "The Finite Memory Prediction of Covariance Stationary Time Series." *SIAM Journal on Scientific and Statistical Computing* 4:330–339.

Pankratz, A. (1983). *Forecasting with Univariate Box-Jenkins Models: Concepts and Cases*. New York: John Wiley & Sons.

Pankratz, A. (1991). *Forecasting with Dynamic Regression Models*. New York: John Wiley & Sons.

Pankratz, A., and Dudley, U. (1987). "Forecast of Power-Transformed Series." *Journal of Forecasting* 6:239–248.

Pearlman, J. G. (1980). "An Algorithm for the Exact Likelihood of a High-Order Autoregressive–Moving Average Process." *Biometrika* 67:232–233.

Priestley, M. B. (1981). *Spectral Analysis and Time Series*. London: Academic Press.

Roberts, S. A. (1982). "A General Class of Holt-Winters Type Forecasting Models." *Management Science* 28:808–820.

Schwarz, G. (1978). "Estimating the Dimension of a Model." *Annals of Statistics* 6:461–464.

Sweet, A. L. (1985). "Computing the Variance of the Forecast Error for the Holt-Winters Seasonal Models." *Journal of Forecasting* 4:235–243.

Winters, P. R. (1960). "Forecasting Sales by Exponentially Weighted Moving Averages." *Management Science* 6:324–342.

Woodfield, T. J. (1987). "Time Series Intervention Analysis Using SAS Software." In *Proceedings of the Twelfth Annual SAS Users Group International Conference*, 331–339. Cary, NC: SAS Institute Inc. http://www.sascommunity.org/sugi/SUGI87/Sugi-12-57%20Woodfield.pdf.

Yar, M., and Chatfield, C. (1990). "Prediction Intervals for the Holt-Winters Forecasting Procedure." *International Journal of Forecasting* 6:127–137.

Part V

Investment Analysis

Chapter 62
Overview

Contents

About Investment Analysis **3917**
Starting Investment Analysis **3918**
Getting Help **3918**
Using Help **3919**
Software Requirements **3919**

About Investment Analysis

The Investment Analysis system is an interactive environment for the time-value of money of a variety of investments:

- loans
- savings
- depreciations
- bonds
- generic cashflows

Various analyses are provided to help analyze the value of investment alternatives: time value, periodic equivalent, internal rate of return, benefit-cost ratio, and breakeven analysis.

These analyses can help answer a number of questions you may have about your investments:

- Which option is more profitable or less costly?
- Is it better to buy or rent?
- Are the extra fees for refinancing at a lower interest rate justified?
- What is the balance of this account after saving this amount periodically for so many years?
- How much is legally tax-deductible?
- Is this a reasonable price?

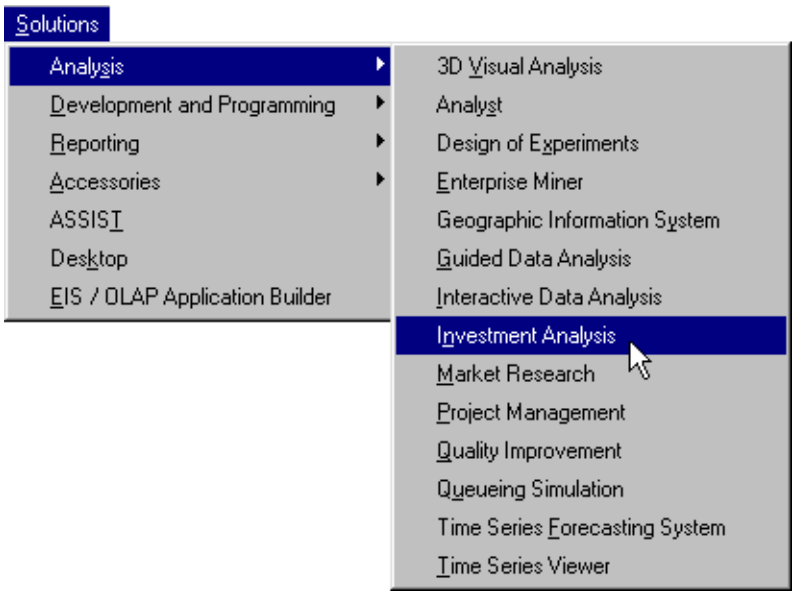
Investment Analysis can be beneficial to users in many industries for a variety of decisions:

- manufacturing: cost justification of automation or any capital investment, replacement analysis of major equipment, or economic comparison of alternative designs
- government: setting funds for services
- finance: investment analysis and portfolio management for fixed-income securities

Starting Investment Analysis

There are two ways to invoke Investment Analysis from the main SAS window. One way is to select **Solutions** → **Analysis** → **Investment Analysis** from the main SAS menu, as displayed in Figure 62.1.

Figure 62.1 Initializing Investment Analysis with the Menu Bar



The other way is to type **INVEST** into the toolbar's command prompt, as displayed in Figure 62.2.

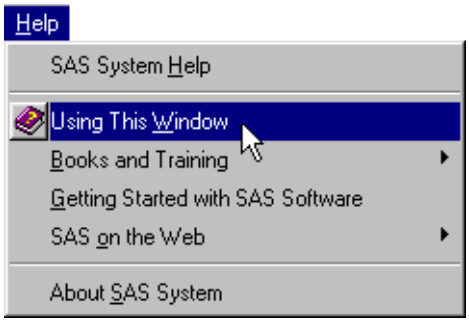
Figure 62.2 Initializing Investment Analysis with the Toolbar



Getting Help

You can get help in Investment Analysis in three ways. One way is to use the Help Menu, as displayed in Figure 62.3. This is the right-most menu item on the menu bar.

Figure 62.3 The Help Menu



Help buttons, as in Figure 62.4, provide another way to access help. Most dialog boxes provide help buttons in their lower-right corners.

Figure 62.4 A Help Button



Also, the toolbar has a button (see Figure 62.5) that invokes the help system. This is the right-most icon on the toolbar.

Figure 62.5 The Help Icon



Each of these methods invokes a browser that gives specific help for the active window.

Using Help

The chapters pertaining to Investment Analysis in this document typically have a section that introduces you to a menu and summarizes the options available through the menu. Such chapters then have sections titled Task and Dialog Box Guides. The Task section provides a description of how to perform many useful tasks. The Dialog Box Guide lists all dialog boxes pertinent to those tasks and gives a brief description of each element of each dialog box.

Software Requirements

Investment Analysis uses the following SAS software:

- Base SAS
- SAS/ETS
- SAS/GRAPH (optional, to view bond pricing and breakeven graphs)

Chapter 63
Portfolios

Contents

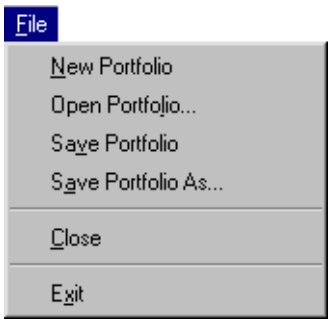
The File Menu **3921**
Tasks **3922**
Creating a New Portfolio 3922
Saving a Portfolio 3922
Opening an Existing Portfolio 3923
Saving a Portfolio to a Different Name 3924
Selecting Investments within a Portfolio 3924
Dialog and Utility Guide **3925**
Investment Analysis 3925
Menu Bar Options 3926
Right-Clicking within the Portfolio Area 3927

The File Menu

Investment Analysis stores portfolios as catalog entries. Portfolios contain a collection of investments, providing a structure to collect investments with a common purpose or goal (like a retirement or building fund portfolio). It may be advantageous also to collect investments into a common portfolio if they are competing investments you want to perform a comparative analysis upon. Within this structure you can perform computations and analyses on a collection of investments in a portfolio, just as you would perform them on a single investment.

Investment Analysis provides many tools to aid in your manipulation of portfolios through the **File** menu, shown in Figure 63.1.

Figure 63.1 File Menu



The **File** menu offers the following items:

New Portfolio creates an empty portfolio with a new name.

Open Portfolio opens the standard SAS Open dialog box where you select a portfolio to open.

Save Portfolio saves the current portfolio to its current name.

Save Portfolio As opens the standard SAS Save As dialog box where you supply a new portfolio name for the current portfolio.

Close closes Investment Analysis.

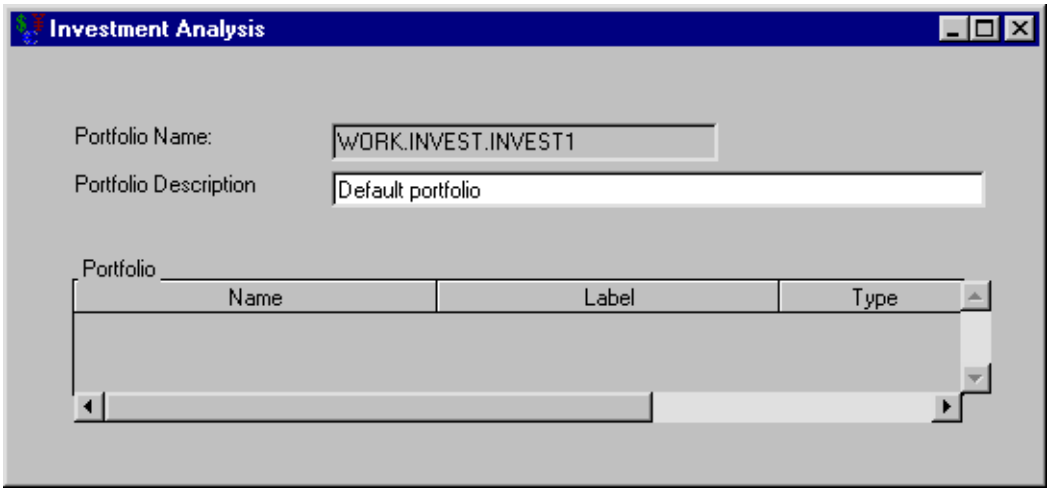
Exit closes SAS (Windows only).

Tasks

Creating a New Portfolio

From the Investment Analysis dialog box, select **File** → **New Portfolio**.

Figure 63.2 Creating a New Portfolio



The **Portfolio Name** is WORK.INVEST.INVEST1 as displayed in Figure 63.2, unless you have saved a portfolio to that name in the past. In that case, some other unused portfolio name is given to the new portfolio.

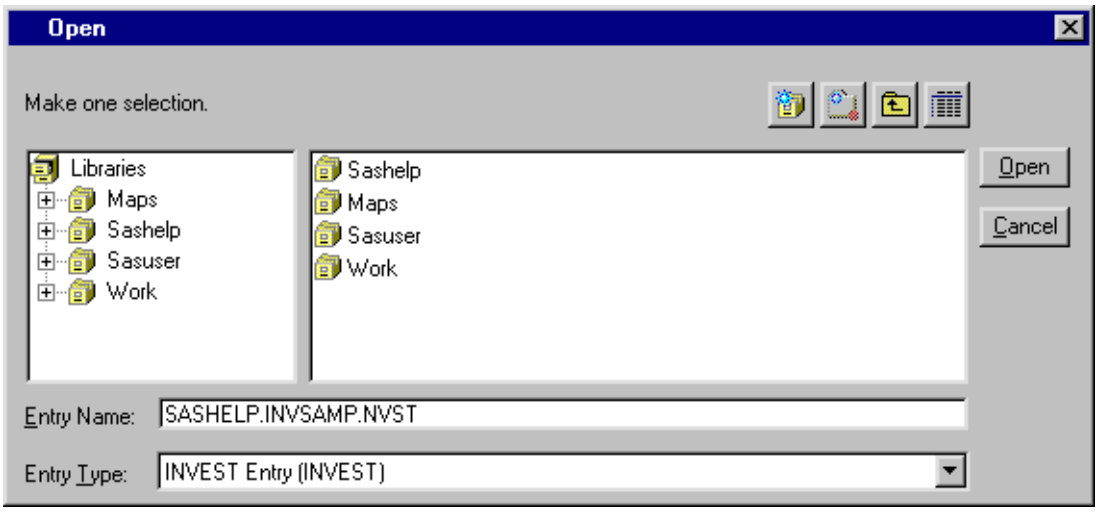
Saving a Portfolio

From the Investment Analysis dialog box, select **File** → **Save Portfolio**. The portfolio is saved to a catalog-entry with the name in the **Portfolio Name** box.

Opening an Existing Portfolio

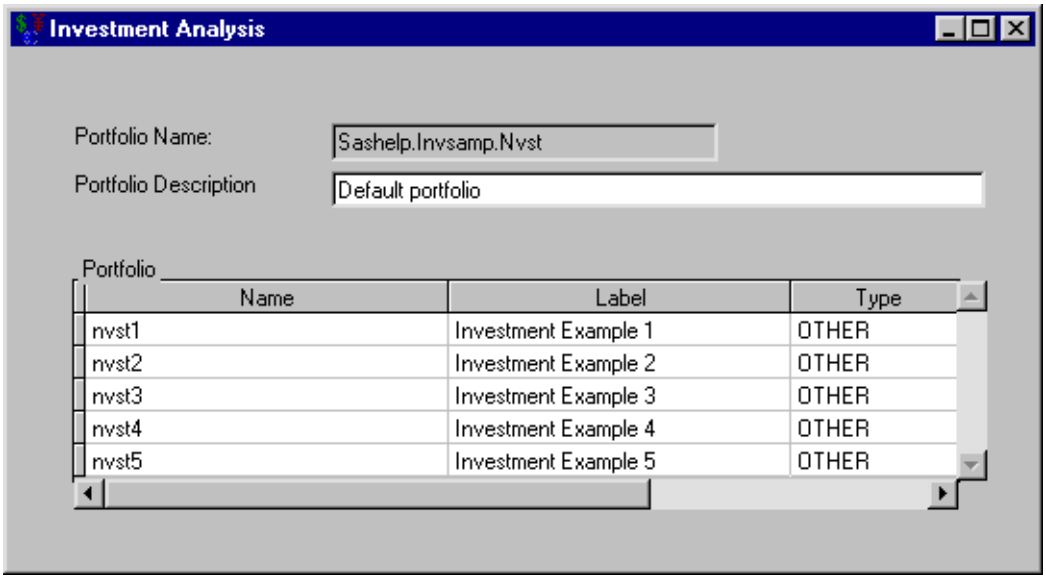
From the Investment Analysis dialog box, select **File** → **Open Portfolio**. This opens the standard SAS Open dialog box. You enter the name of a SAS portfolio to open in the **Entry Name** box. For example, enter SASHELP.INVSAMP.NVST as displayed in Figure 63.3.

Figure 63.3 Opening an Existing Portfolio



Click **Open** to load the portfolio. The portfolio should look like Figure 63.4.

Figure 63.4 The Opened Portfolio

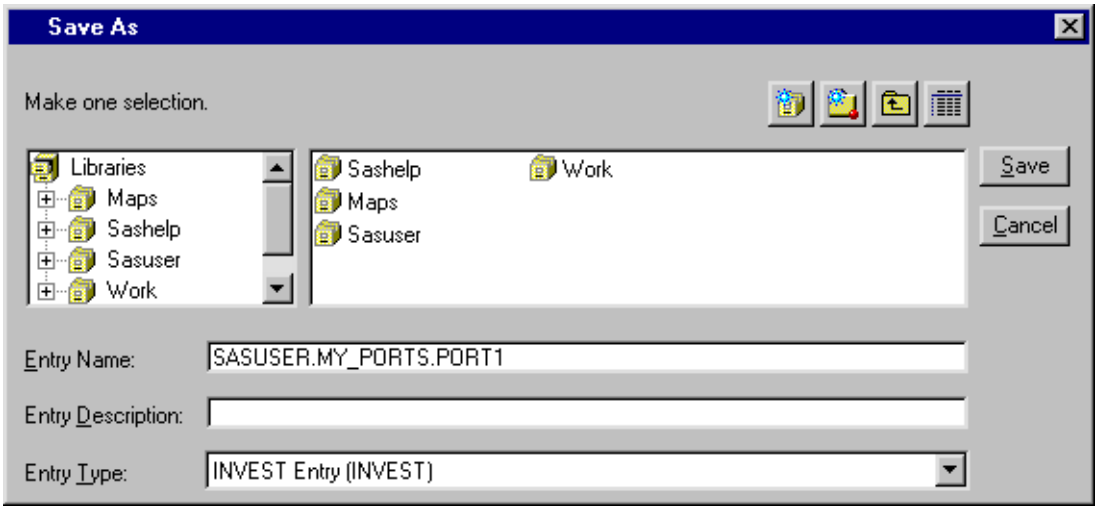


Saving a Portfolio to a Different Name

From the Investment Analysis dialog box, select **File** → **Save Portfolio As**.

This opens the standard SAS Save As dialog box. You can enter the name of a SAS portfolio into the **Entry Name** box. For example, enter SASUSER.MY_PORTS.PORT1, as in Figure 63.5.

Figure 63.5 Saving a Portfolio to a Different Name



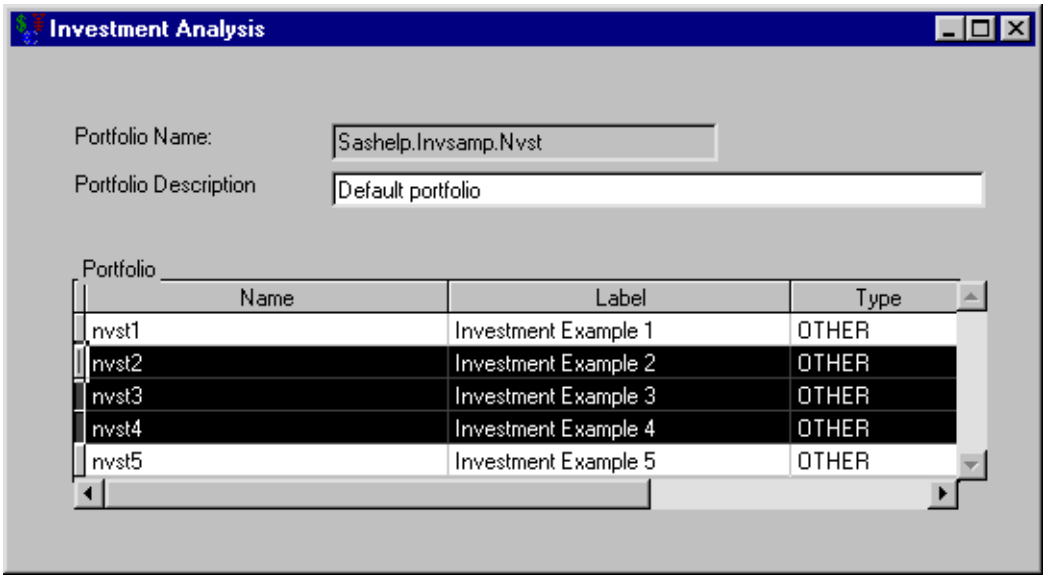
Click **Save** to save the portfolio.

Selecting Investments within a Portfolio

To select a single investment in an opened portfolio, click the investment in the Portfolio area within the Investment Analysis dialog box.

To select a list of adjacent investments, do the following: click the first investment, hold down SHIFT, and click the final investment. After the list of investments is selected, you can release the SHIFT key. The selected investments will appear highlighted as in Figure 63.6.

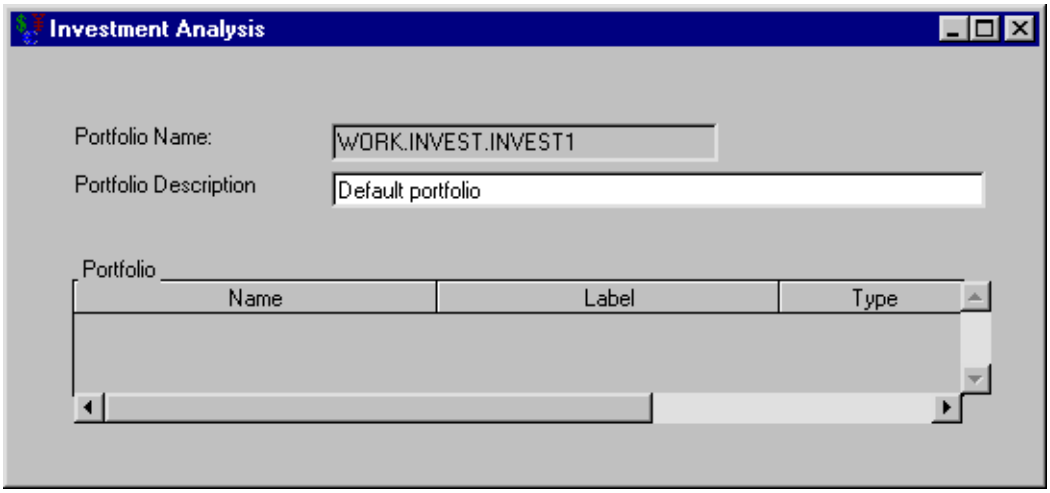
Figure 63.6 Selecting Investments within a Portfolio



Dialog and Utility Guide

Investment Analysis

Figure 63.7 Investment Analysis Dialog Box



Investment Portfolio Name holds the name of the portfolio. The name is of the form library.catalog_entry.portfolio. The default portfolio name is work.invest.invest1, as in Figure 63.7.

Portfolio Description provides a more descriptive explanation of the portfolio's contents. You can edit this description any time this dialog box is active.

The **Portfolio** area contains the list of investments comprising the particular portfolio. Each investment in the **Portfolio** area displays the following attributes:

Name is the name of the investment. It must be a valid SAS name. It is used to distinguish investments when performing analyses and computations.

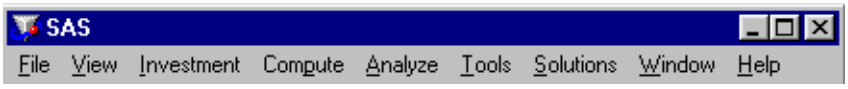
Label is a place where you can provide a more descriptive explanation of the investment.

Type is the type of investment, which is fixed when you create the investment. It is one of the following: LOAN, SAVINGS, DEPRECIATION, BOND, or OTHER.

Additional tools to aid in the management of your portfolio are available by selecting from the menu bar or by right-clicking within the **Portfolio** area.

Menu Bar Options

Figure 63.8 The Menu Bar



The menu bar (shown in Figure 63.8) provides many tools to aid in the management of portfolios and the investments that comprise them. The following menu items provide functionality particular to Investment Analysis:

File opens and saves portfolios.

Investment creates new investments within the portfolio.

Compute performs constant dollar, after tax, and currency conversion computations on generic cashflows.

Analyze analyzes investments to aid in decision-making.

Tools sets default values of inflation and income tax rates.

Right-Clicking within the Portfolio Area

Figure 63.9 Right-Clicking



After selecting an investment, right-clicking in the **Portfolio** area pops up a menu (see Figure 63.9) that offers the following options:

Edit opens the selected investment within the portfolio.

Duplicate creates a duplicate of the selected investment within the portfolio.

Delete removes the selected investment from the portfolio.

If you wish to perform one of these actions on a collection of investments, you must select a collection of investments (as described in the section "Selecting Investments within a Portfolio" on page 3924) before right-clicking.

Chapter 64
Investments

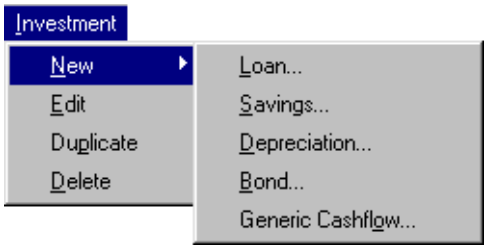
Contents

The Investment Menu **3930**
Tasks **3931**
Loan Tasks 3931
Specifying Savings Terms to Create an Account Summary 3937
Depreciation Tasks 3938
Bond Tasks 3941
Generic Cashflow Tasks 3945
Dialog Box Guide **3952**
Loan 3952
Loan Initialization Options 3953
Loan Prepayments 3955
Balloon Payments 3956
Rate Adjustment Terms 3956
Rounding Off 3958
Savings 3958
Depreciation 3960
Depreciation Table 3962
Bond 3962
Bond Analysis 3964
Bond Price 3965
Generic Cashflow 3966
Right-Clicking within Generic Cashflow's Cashflow Specification Area 3967
Flow Specification 3968
Forecast Specification 3969

The Investment Menu

Because there are many types of investments, a tool that manages and analyzes collections of investments must be robust and flexible. Providing specifications for four specific investment types and one generic type, Investment Analysis can model almost any real-world investment.

Figure 64.1 Investment Menu



The **Investment** menu, shown in Figure 64.1, offers the following items:

New → Loan opens the Loan dialog box. Loans are useful for acquiring capital to pursue various interests. Available terms include rate adjustments for variable rate loans, initialization costs, prepayments, and balloon payments.

New → Savings opens the Savings dialog box. Savings are necessary when planning for the future, whether for business or personal purposes. Account summary calculations available per deposit include starting balance, deposits, interest earned, and ending balance.

New → Depreciation opens the Depreciation dialog box. Depreciations are relevant in tax calculation. The available depreciation methods are Straight Line, Sum-of-years Digits, Depreciation Table, and Declining Balance. Depreciation Tables are necessary when depreciation calculations must conform to set yearly percentages. Declining Balance with conversion to Straight Line is also provided.

New → Bond opens the Bond dialog box. Bonds have widely varying terms depending on the issuer. Because bond issuers frequently auction their bonds, the ability to price a bond between the issue date and maturity date is desirable. Fixed-coupon bonds may be analyzed for the following: price versus yield-to-maturity, duration, and convexity. These are available at different times in the bond's life.

New → Generic Cashflow opens the Generic Cashflow dialog box. Generic cashflows are the most flexible investments. Only a sequence of date-amount pairs is necessary for specification. You can enter date-amount pairs and load values from SAS data sets to specify any type of investment. You can generate uniform, arithmetic, and geometric cashflows with ease. SAS's forecasting ability is available to forecast future cashflows as well. The new graphical display aids in visualization of the cashflow and enables the user to change the frequency of the cashflow view to aggregate and disaggregate the view.

Edit opens the specification dialog box for an investment selected within the portfolio.

Duplicate creates a duplicate of an investment selected within the portfolio.

Delete removes an investment selected from the portfolio.

If you want to edit, duplicate, or delete a collection of investments, you must select a collection of investments as described in the section "Selecting Investments within a Portfolio" on page 3924 before performing the menu-option.

Tasks

Loan Tasks

Suppose you want to buy a home that costs $100,000. You can make a down payment of $20,000. Hence, you need a loan of $80,000. You are able to acquire a 30-year loan at 7% interest starting January 1, 2000. Let's use Investment Analysis to specify and analyze this loan.

In the Investment Analysis dialog box, select **Investment** → **New** → **Loan** from the menu bar to open the Loan dialog box.

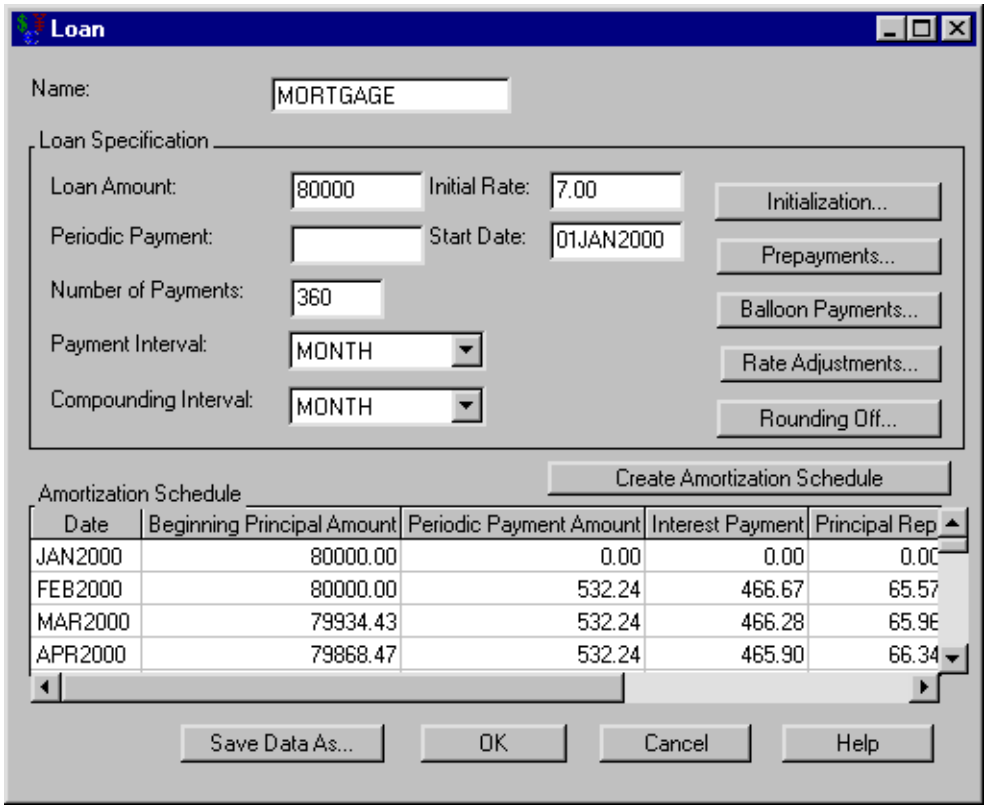
Specifying Loan Terms to Create an Amortization Schedule

You must specify the loan before generating the amortization table. To specify the loan, follow these steps:

1. Enter MORTGAGE for the **Name**.
2. Enter 80000 for the **Loan Amount**.
3. Enter 7 for the **Initial Rate**.
4. Enter 360 for the **Number of Payments**.
5. Enter 01JAN2000 for the **Start Date**.

After you have specified the loan, click **Create Amortization Schedule** to generate the amortization schedule displayed in Figure 64.2.

Figure 64.2 Creating an Amortization Schedule

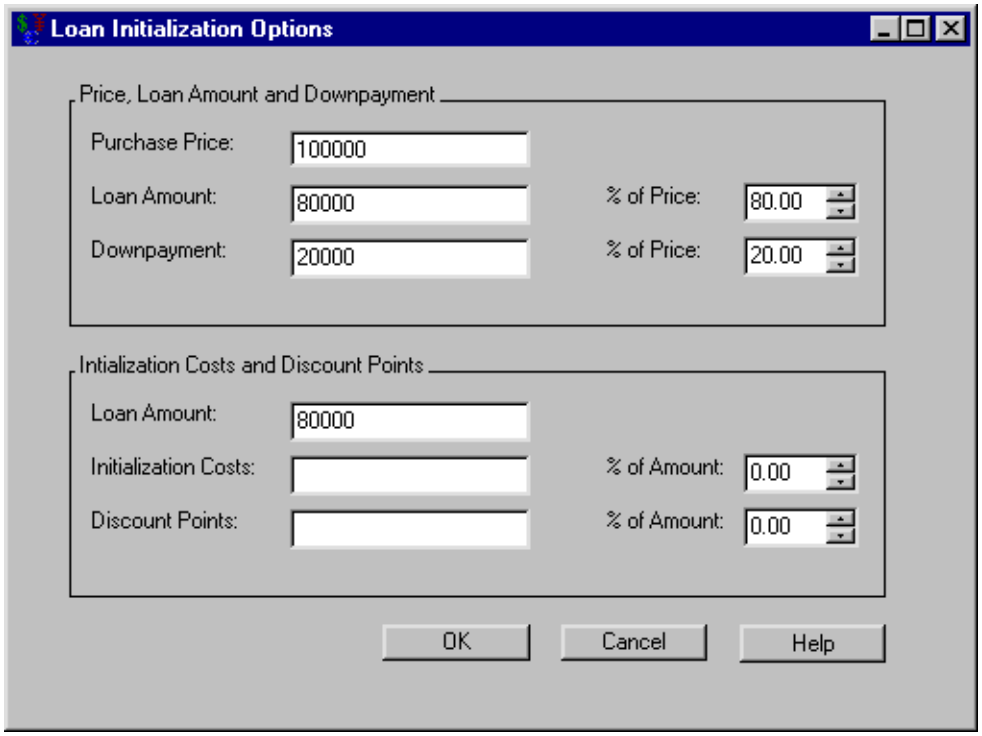


Storing Other Loan Terms

Let's include information concerning the purchase price and downpayment. These terms are not necessary to specify the loan, but it may be advantageous to store such information with the loan.

Consider the loan described in the section "Loan Tasks" on page 3931. In the Loan dialog box (Figure 64.2) click **Initialization** to open the Loan Initialization Options dialog box, where you can specify the down payment, initialization costs, and discount points. To specify the down payment, enter 100000 for the **Purchase Price**, as shown in Figure 64.3.

Figure 64.3 Including the Purchase Price



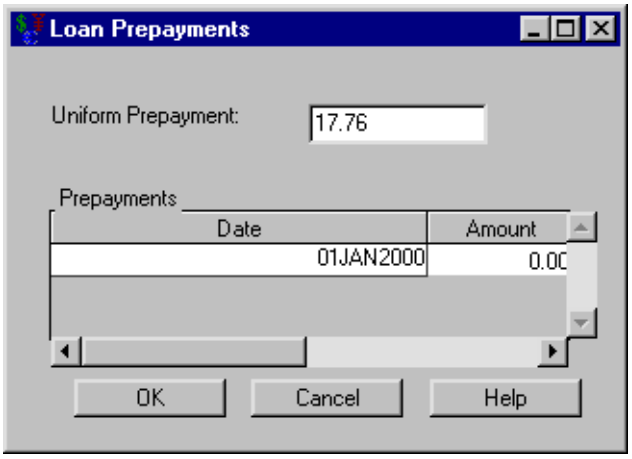
Click **OK** to return to the Loan dialog box.

Adding Prepayments

Now let's observe the effect of prepayments on the loan. Consider the loan described in the section "Loan Tasks" on page 3931. You must pay a minimum of $532.24 each month to keep up with payments. However, let's say you dislike entering this amount in your checkbook. You would rather pay $550.00 to keep the arithmetic simpler. This would constitute a uniform prepayment of $17.76 each month.

In the Loan dialog box, click **Prepayments**, which opens the Loan Prepayments dialog box shown in Figure 64.4.

Figure 64.4 Specifying the Loan Prepayments

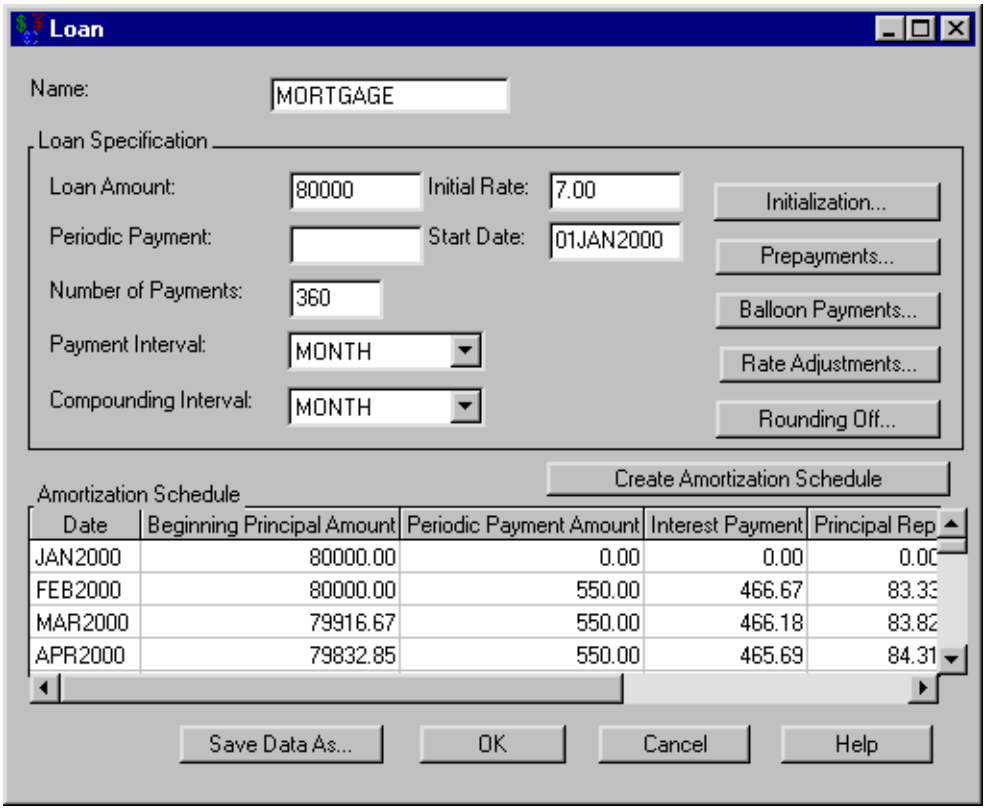


You can specify an arbitrary sequence of prepayments in the **Prepayments** area. If you want a uniform prepayment, clear the **Prepayments** area and enter the uniform payment amount in the **Uniform Prepayment** text box. That amount will be added to each payment until the loan is paid off.

To specify this uniform prepayment, follow these steps:

1. Enter 17.76 for the **Uniform Prepayment**.
2. Click **OK** to return to the Loan dialog box.
3. Click **Create Amortization Schedule**, and the amortization schedule is updated, as displayed in Figure 64.5.

Figure 64.5 The Amortization Schedule with Loan Prepayments



The last payment is on January 2030 without prepayments and February 2027 with prepayment; you would pay the loan off almost three years earlier with the $17.76 prepayments.

To continue this example you must remove the prepayments from the loan specification, following these steps:

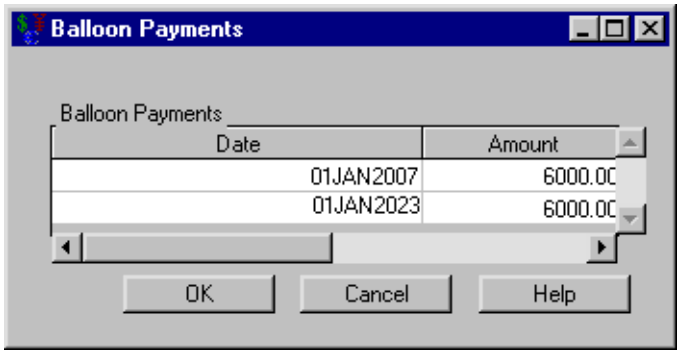
1. Reopen the Loan Prepayments dialog box from the Loan dialog box by clicking **Prepayments**.
2. Enter 0 for **Uniform Prepayment**.
3. Click **OK** to return to the Loan dialog box.

Adding Balloon Payments

Consider the loan described in the section "Loan Tasks" on page 3931. Suppose you cannot afford the payments of $532.24 each month. To lessen your monthly payment, you could pay balloon payments of $6,000 at the end of 2007 and 2023. You wonder how this would affect your monthly payment. (Note that Investment Analysis does not allow both balloon payments and rate adjustments to be specified for a loan.)

In the Loan dialog box, click **Balloon Payments**, which opens the Balloon Payments dialog box shown in Figure 64.6.

Figure 64.6 Defining Loan Balloon Payments



You can specify an arbitrary sequence of balloon payments by adding date-amount pairs to the **Balloon Payments** area.

To specify these balloon payments, follow these steps:

1. Right-click within the **Balloon Payment** area (which pops up a menu) and release on **New**.
2. Set the pair's **Date** to 01JAN2007.
3. Set **Amount** to 6000.
4. Right-click within the **Balloon Payment** area (which pops up a menu) and release on **New**.
5. Set the new pair's **Date** to 01JAN2023.
6. Set its **Amount** to 6000.

Click **OK** to return to the Loan dialog box. Click **Create Amortization Schedule**, and the amortization schedule is updated. Your monthly payment is now $500.30, a difference of approximately $32 each month.

To continue this example you must remove the balloon payments from the loan specification, following these steps:

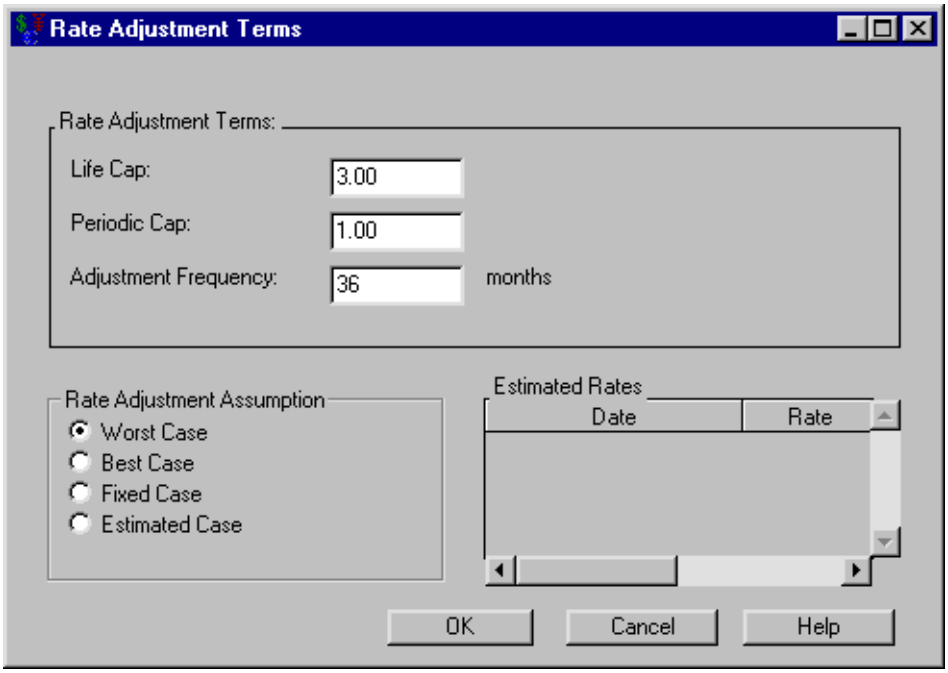
1. Reopen the Balloon Payments dialog box.
2. Right-click within the **Balloon Payment** area (which pops up a menu) and release on **Clear**.
3. Click **OK** to return to the Loan dialog box.

Handling Rate Adjustments

Consider the loan described in the section "Loan Tasks" on page 3931. Another option for lowering your payments is to get a variable rate loan. You can acquire a three-year adjustable rate mortgage (ARM) at 6% with a periodic cap of 1% with a maximum of 9%. (Note that Investment Analysis does not allow both rate adjustments and balloon payments to be specified for a loan.)

In the Loan dialog box, click **Rate Adjustments** to open the Rate Adjustment Terms dialog box shown in Figure 64.7.

Figure 64.7 Setting the Rate Adjustments

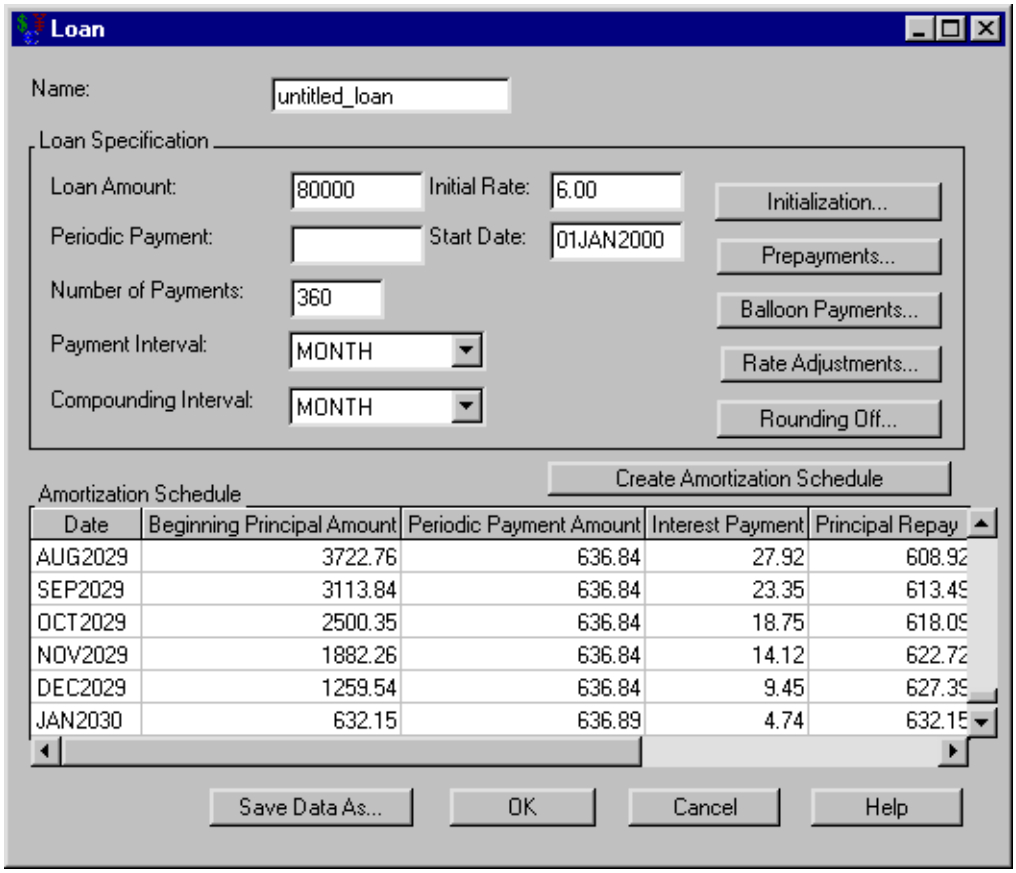


To specify these loan adjustment terms, follow these steps:

1. Enter 3 for the **Life Cap**. The **Life Cap** is the maximum deviation from the Initial Rate.
2. Enter 1 for the **Periodic Cap**.
3. Enter 36 for the **Adjustment Frequency**.
4. Confirm that **Worst Case** is selected from the Rate Adjustment Assumption options.
5. Click **OK** to return to the Loan dialog box.
6. Enter 6 for the **Initial Rate**.
7. Click **Create Amortization Schedule**, and the amortization schedule is updated.

Your monthly payment drops to $479.64 each month. However, if the worst-case scenario plays out, the payments will increase to $636.84 in nine years. Figure 64.8 displays amortization table information for the final few months under this scenario.

Figure 64.8 The Amortization Schedule with Rate Adjustments



Click **OK** to return to the Investment Analysis dialog box.

Specifying Savings Terms to Create an Account Summary

Suppose you put $500 each month into an account that earns 6% interest for 20 years. What is the balance of the account after those 20 years?

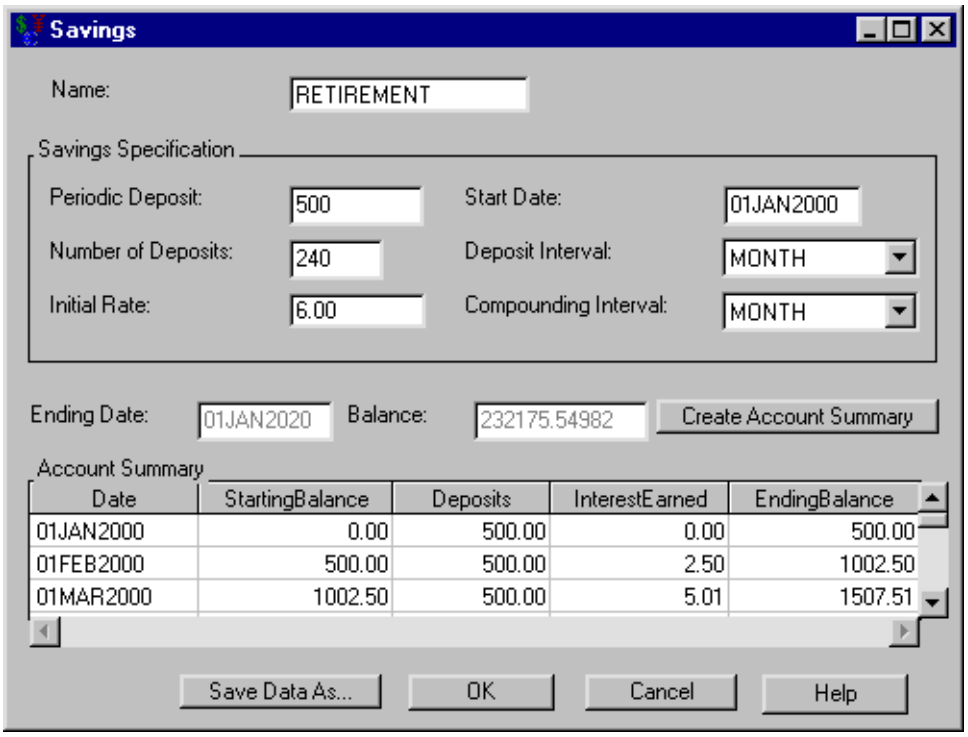
In the Investment Analysis dialog box, select **Investment** → **New** → **Savings** from the menu bar to open the Savings dialog box.

To specify the savings, follow these steps:

1. Enter RETIREMENT for the **Name**.
2. Enter 500 for the **Periodic Deposit**.
3. Enter 240 for the **Number of Deposits**.
4. Enter 6 for the **Initial Rate**.

You must specify the savings before generating the account summary. After you have specified the savings, click **Create Account Summary** to compute the ending date and balance and to generate the account summary displayed in Figure 64.9.

Figure 64.9 Creating an Account Summary



Click **OK** to return to the Investment Analysis dialog box.

Depreciation Tasks

Commercial assets are considered to lose value as time passes. For tax purposes, you want to quantify this loss. This investment structure helps calculate appropriate values.

Suppose you spend $50,000 for a commercial fishing boat that is considered to have a ten-year useful life. How would you depreciate it?

In the Investment Analysis dialog box, select **Investment** → **New** → **Depreciation** from the menu bar to open the Depreciation dialog box.

Specifying Depreciation Terms to Create a Depreciation Table

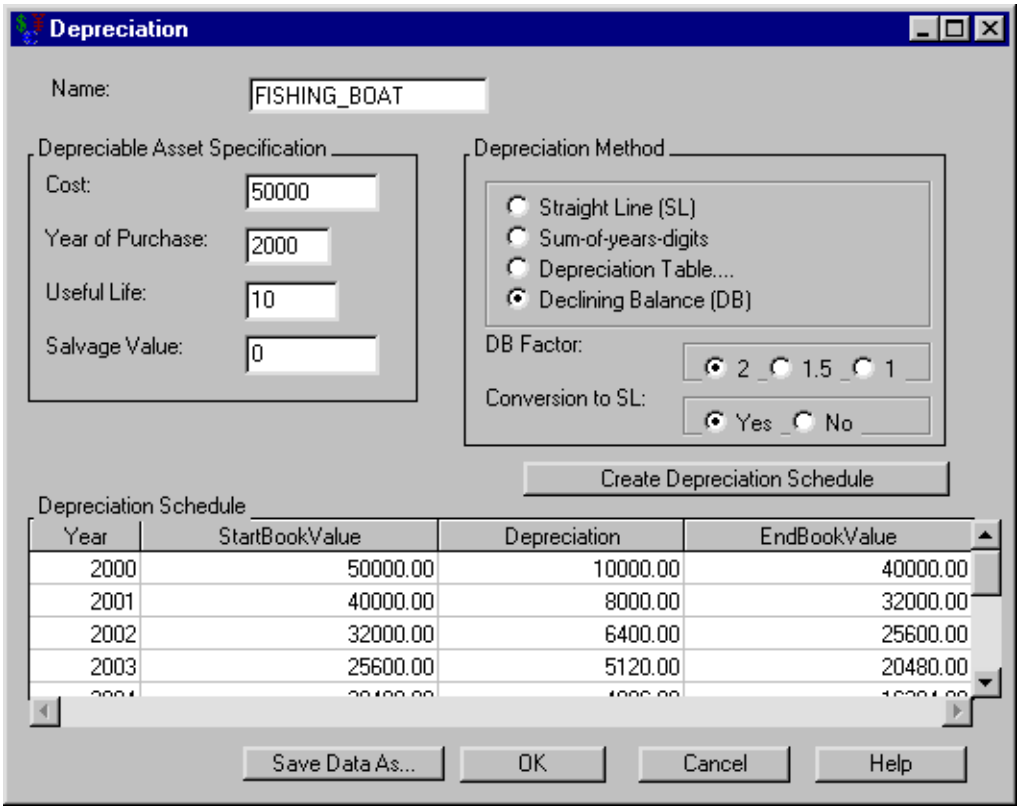
To specify the depreciation, follow these steps:

1. Enter FISHING_BOAT for the **Name**.
2. Enter 50000 for the **Cost**.

3. Enter 2000 for the **Year of Purchase**.
4. Enter 10 for the **Useful Life**.
5. Enter 0 for the **Salvage Value**.

You must specify the depreciation before generating the depreciation schedule. After you have specified the depreciation, click **Create Depreciation Schedule** to generate a depreciation schedule like the one displayed in Figure 64.10.

Figure 64.10 Creating a Depreciation Schedule



The default depreciation method is Declining Balance (with Conversion to Straight Line). Try the following methods to see how they each affect the schedule:

- Straight Line
- Sum-of-years Digits
- Declining Balance (without conversion to Straight Line)

It might be useful to compare the value of the boat at 5 years for each method.

A description of these methods is available in the section "Depreciation Methods" on page 4001.

Using the Depreciation Table

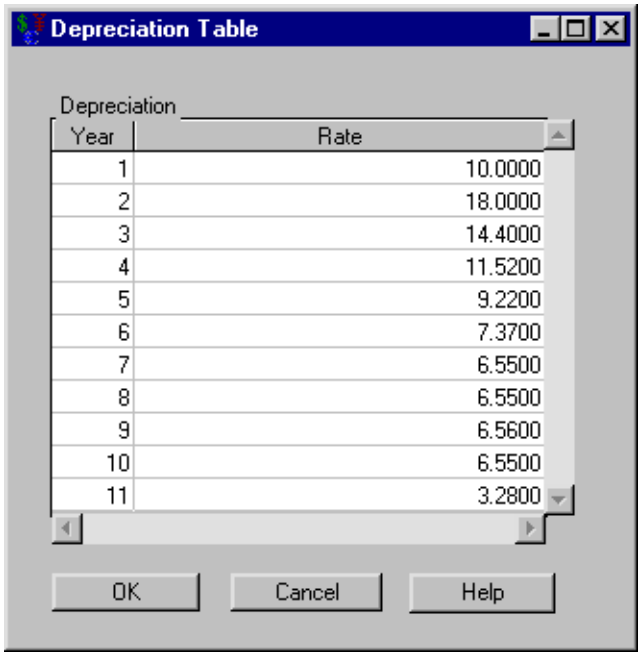
Sometimes you want to force the depreciation rates to be certain percentages each year. This option is particularly useful for calculating modified accelerated cost recovery system (MACRS) depreciations. The United States' Tax Reform Act of 1986 set depreciation rates for an asset based on an assumed lifetime for that asset. Since these lists of rates are important to many people, Investment Analysis provides SAS data sets for situations with yearly rates (using the "half-year convention"). Find them at SASHELP.MACRS* where * refers to the class of the property. For example, use SASHELP.MACRS15 for a fifteen-year property. (When using the MACRS with the Tax Reform Act tables, you must set the **Salvage Value** to zero.)

Suppose you want to compute the depreciation schedule for the commercial fishing boat described in the section "Depreciation Tasks" on page 3938. The boat is a ten-year property according to the Tax Reform Act of 1986.

To employ the MACRS depreciation from the Depreciation dialog box, follow these steps:

1. Select the **Depreciation Table** option within the **Depreciation Method** area and click **OK**. This opens the Depreciation Table dialog box.
2. Right-click within the **Depreciation** area (which pops up a menu) and select **Load**.
3. Enter SASHELP.MACRS10 for the **Dataset Name**. The dialog box should look like Figure 64.11.

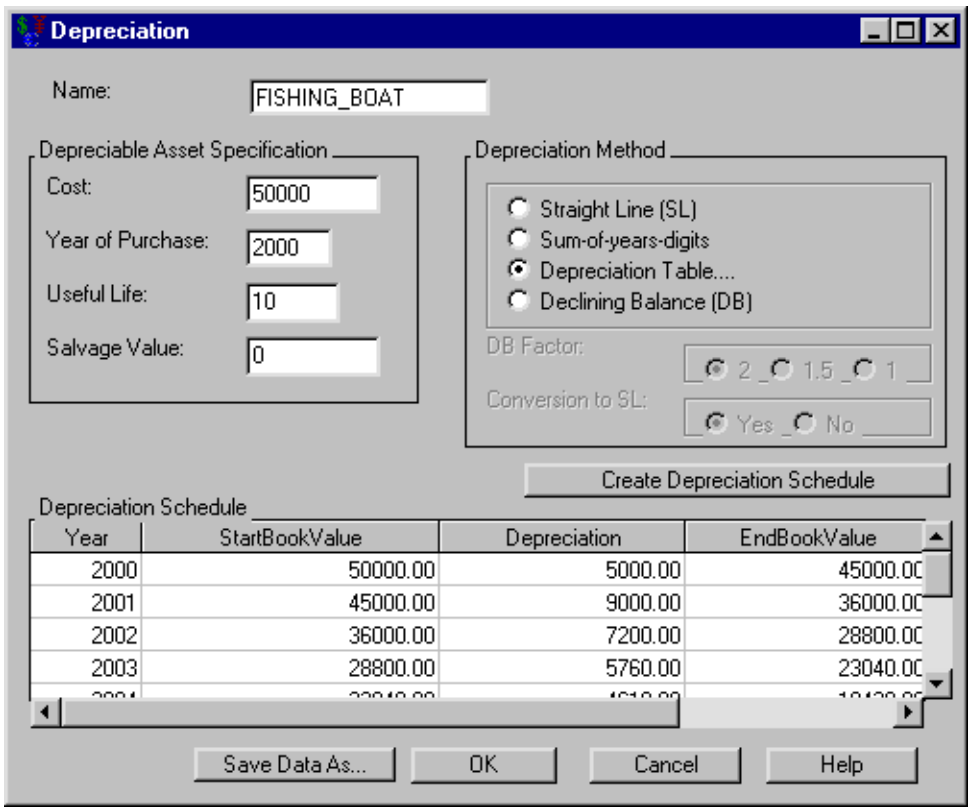
Figure 64.11 MACRS Percentages for a Ten-Year Property



Click **OK** to return to the Depreciation dialog box. Click **Create Depreciation Schedule** and the depreciation schedule fills (see Figure 64.12).

Note there are eleven entries in this depreciation schedule. This is because of the half-year convention that enables you to deduct one half of a year the first year which leaves a half year to deduct after the useful life is over.

Figure 64.12 Depreciation Table with MACRS10



Click **OK** to return to the Investment Analysis dialog box.

Bond Tasks

Suppose someone offers to sell you a 20-year utility bond that was issued six years ago. It has a $1,000 face value and pays semi-year coupons at 2%. You can purchase it for $780. Would you be satisfied with this bond if you expect an 8% minimum attractive rate of return (MARR)?

In the Investment Analysis dialog box, select **Investment** → **New** → **Bond** from the menu bar to open the Bond dialog box.

Specifying Bond Terms

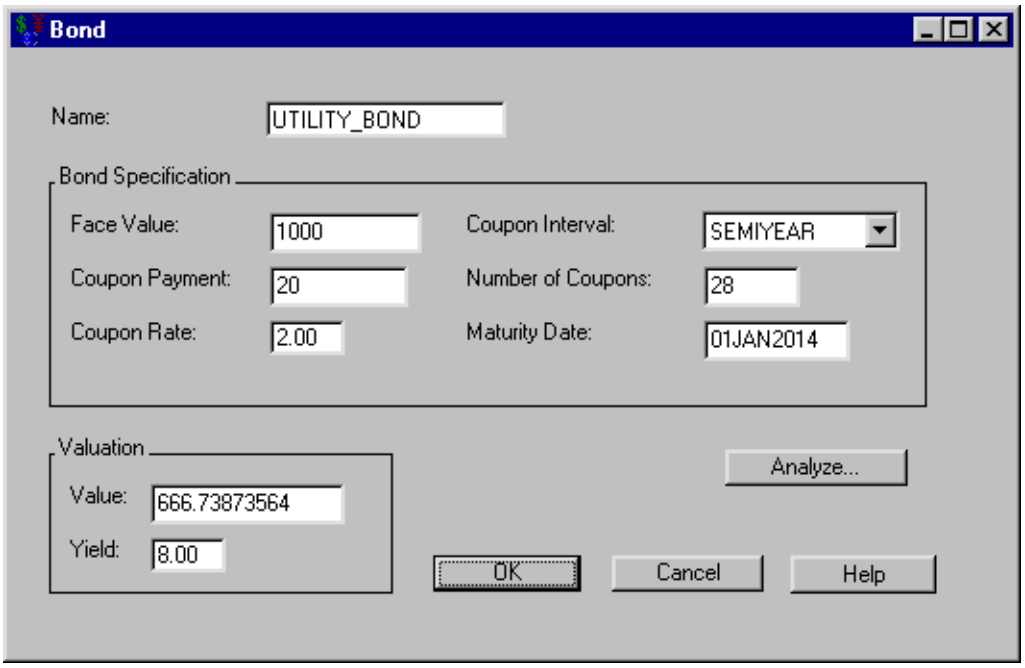
To specify the bond, follow these steps:

1. Enter UTILITY_BOND for the **Name**.
2. Enter 1000 for the **Face Value**.
3. Enter 2 for the **Coupon Rate**. The **Coupon Payment** updates to 20.
4. Select SEMIYEAR for **Coupon Interval**.
5. Enter 28 for the **Number of Coupons**. Because 14 years remain before the bond matures, the bond still has 28 semiyear coupons to pay. The **Maturity Date** updates.

Computing the Price from Yield

Enter 8 for **Yield** within the **Valuation** area. You see the bond's value would be $666.72 as in Figure 64.13.

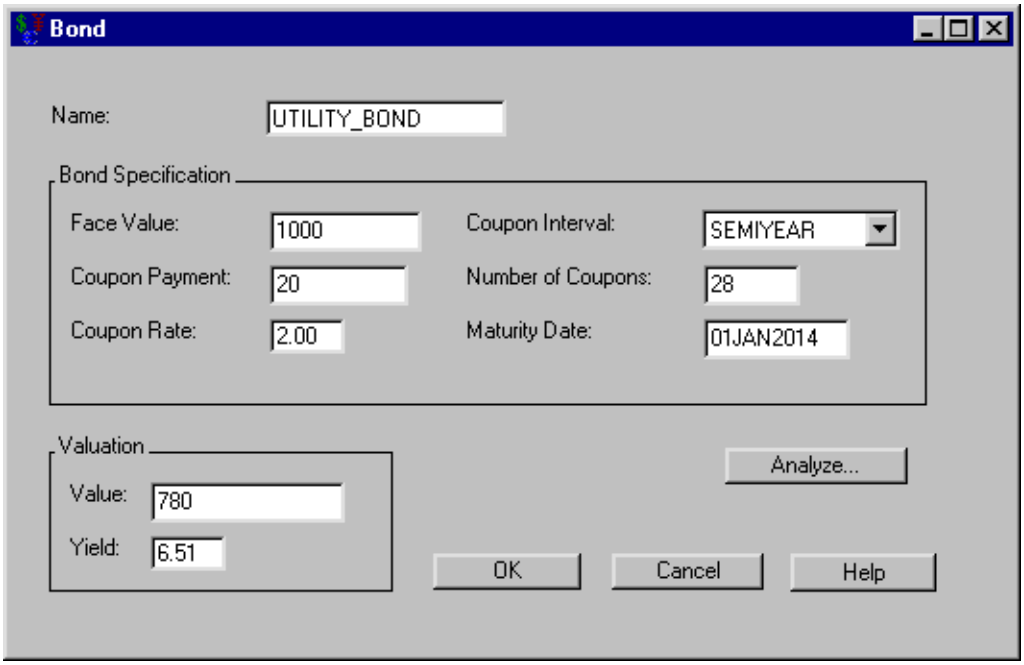
Figure 64.13 Bond Value



Computing the Yield from Price

Now enter 780 for **Value** within the **Valuation** area. You see the yield is only 6.5%, as in Figure 64.14. This is not acceptable if you desire an 8% MARR.

Figure 64.14 Bond Yield



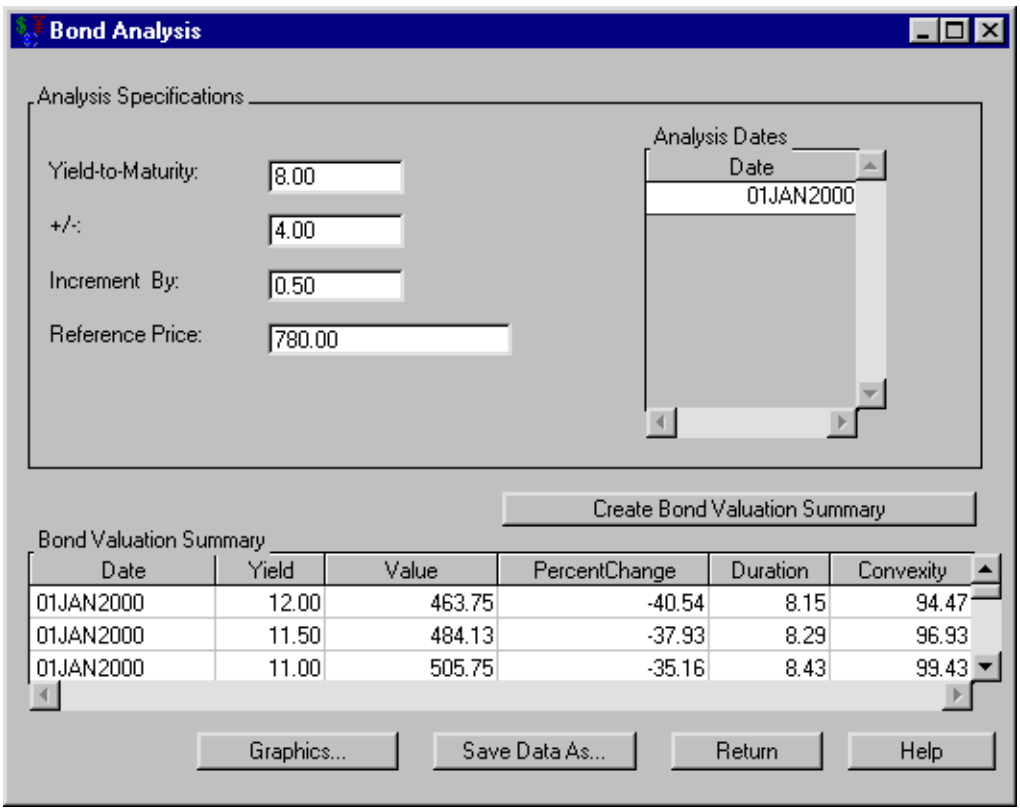
Performing Bond Analysis

To perform bond-pricing analysis, follow these steps:

1. Click **Analyze** to open the Bond Analysis dialog box.
2. Enter 8.0 as the **Yield to Maturity**.
3. Enter 4.0 as the **+/-**.
4. Enter 0.5 as the **Increment by**.
5. Enter 780 as the **Reference Price**.
6. Click **Create Bond Valuation Summary**.

The **Bond Valuation Summary** area fills and shows you the different values for various yields as in Figure 64.15.

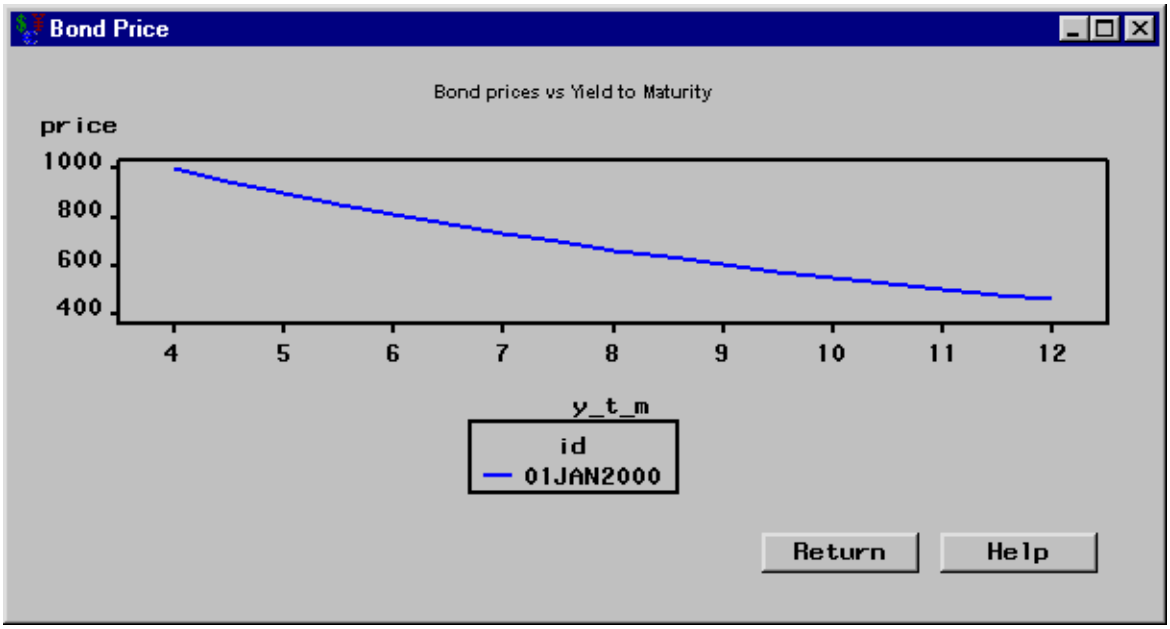
Figure 64.15 Bond Price Analysis



Creating a Price versus Yield-to-Maturity Graph

Click **Graphics** to open the Bond Price dialog box. This contains the price versus yield-to-maturity graph shown in Figure 64.16.

Figure 64.16 Bond Price Graph



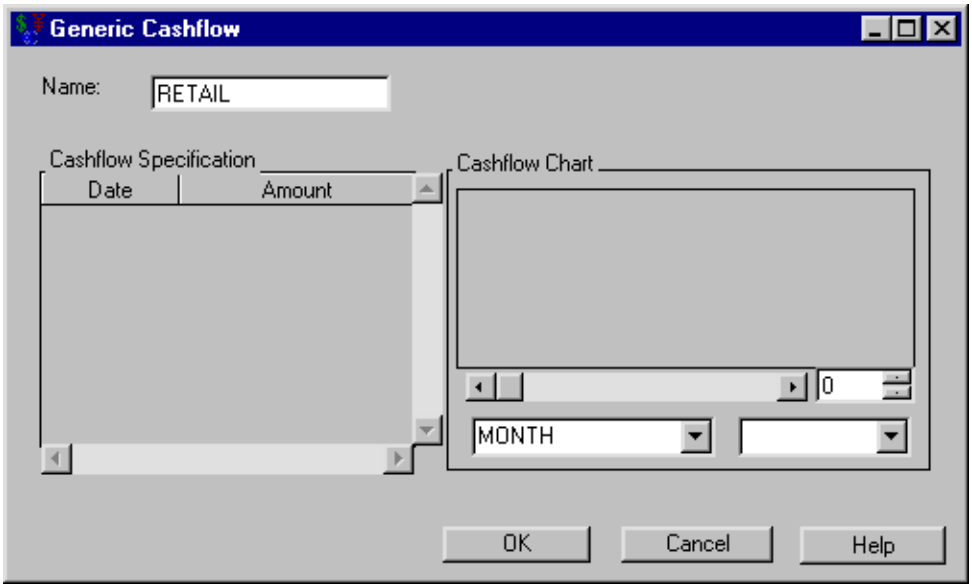
Click **Return** to return to the Bond Analysis dialog box. In the Bond Analysis dialog box, click **OK** to return to the Bond dialog box. In the Bond dialog box, click **OK** to return to the Investment Analysis dialog box.

Generic Cashflow Tasks

To specify a generic cashflow, you merely define any sequence of date-amount pairs. The flexibility of generic cashflows enables the user to represent economic alternatives or investments that do not fit into loan, savings, depreciation, or bond specifications.

In the Investment Analysis dialog box, select **Investment** → **New** → **Generic Cashflow** from the menu bar to open the **Generic Cashflow** dialog box. Enter RETAIL for the **Name** as in Figure 64.17.

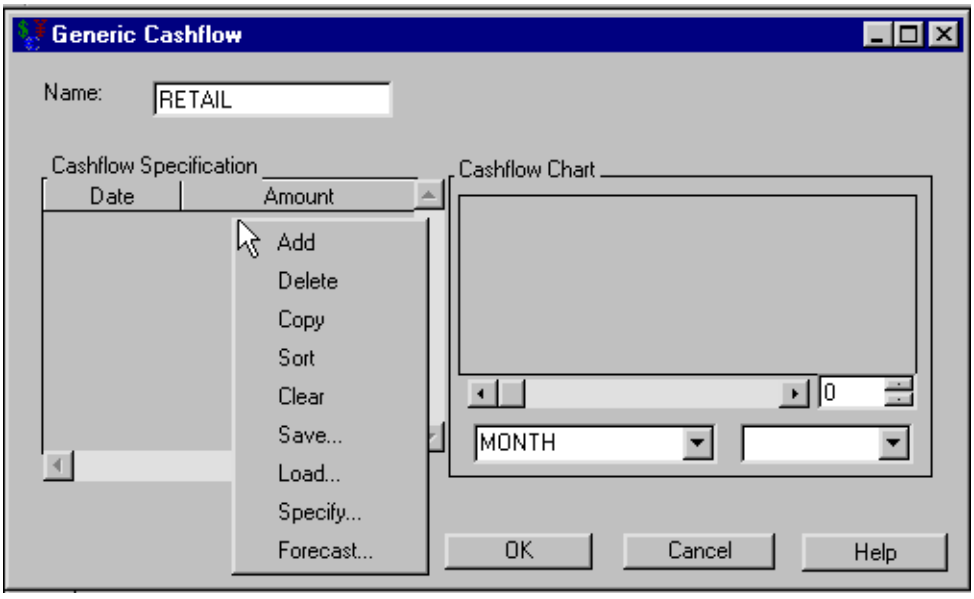
Figure 64.17 Introducing the Generic Cashflow



Right-Clicking within the Cashflow Specification Area

Right-clicking within Generic Cashflow's **Cashflow Specification** area reveals the pop-up menu displayed in Figure 64.18. The menu provides many useful tools to assist you in creating these date-amount pairs.

Figure 64.18 Right-Clicking within the Cashflow Specification Area



The following sections describe how to use most of these right-click options. The **Specify** and **Forecast** menu items are described in the sections "Including a Generated Cashflow" on page 3948 and "Including a Forecasted Cashflow" on page 3950.

Adding a New Date-Amount Pair

To add a new date-amount pair manually, follow these steps:

1. Right-click in the **Cashflow Specification** area as shown in Figure 64.18, and release on **Add**.
2. Enter 01JAN01 for the date.
3. Enter 100 for the amount.

Copying a Date-Amount Pair

To copy a selected date-amount pair, follow these steps:

1. Select the pair you just created.
2. Right-click in the **Cashflow Specification** area as shown in Figure 64.18, but this time release on **Copy**.

Sorting All of the Date-Amount Pairs

Change the second date to 01JAN00. Now the dates are unsorted. Right-click in the **Cashflow Specification** area as shown in Figure 64.18, and release on **Sort**.

Deleting a Date-Amount Pair

To delete a selected date-amount pair, follow these steps:

1. Select a date-amount pair.
2. Right-click in the **Cashflow Specification** area as shown in Figure 64.18, and release on **Delete**.

Clearing All of the Date-Amount Pairs

To clear all date-amount pairs, right-click in the **Cashflow Specification** area as shown in Figure 64.18, and release on **Clear**.

Loading Date-Amount Pairs from a Data Set

To load date-amount pairs from a SAS data set into the **Cashflow Specification** area, follow these steps:

1. Right-click in the **Cashflow Specification** area, and release on **Load**. This opens the Load Dataset dialog box.
2. Enter SASHELP.RETAIL for **Dataset Name**.
3. Click **OK** to return to the Generic Cashflow dialog box.

If there is a **Date** variable in the SAS data set, Investment Analysis loads it into the list. If there is no date-time-formatted variable, it loads the first available date or date-time-formatted variable. Investment Analysis then searches the SAS data set for an **Amount** variable to use. If none exists, it takes the first numeric variable that is not used by the **Date** variable.

Saving Date-Amount Pairs to a Data Set

To save date-amount pairs from the **Cashflow Specification** area to a SAS data set, follow these steps:

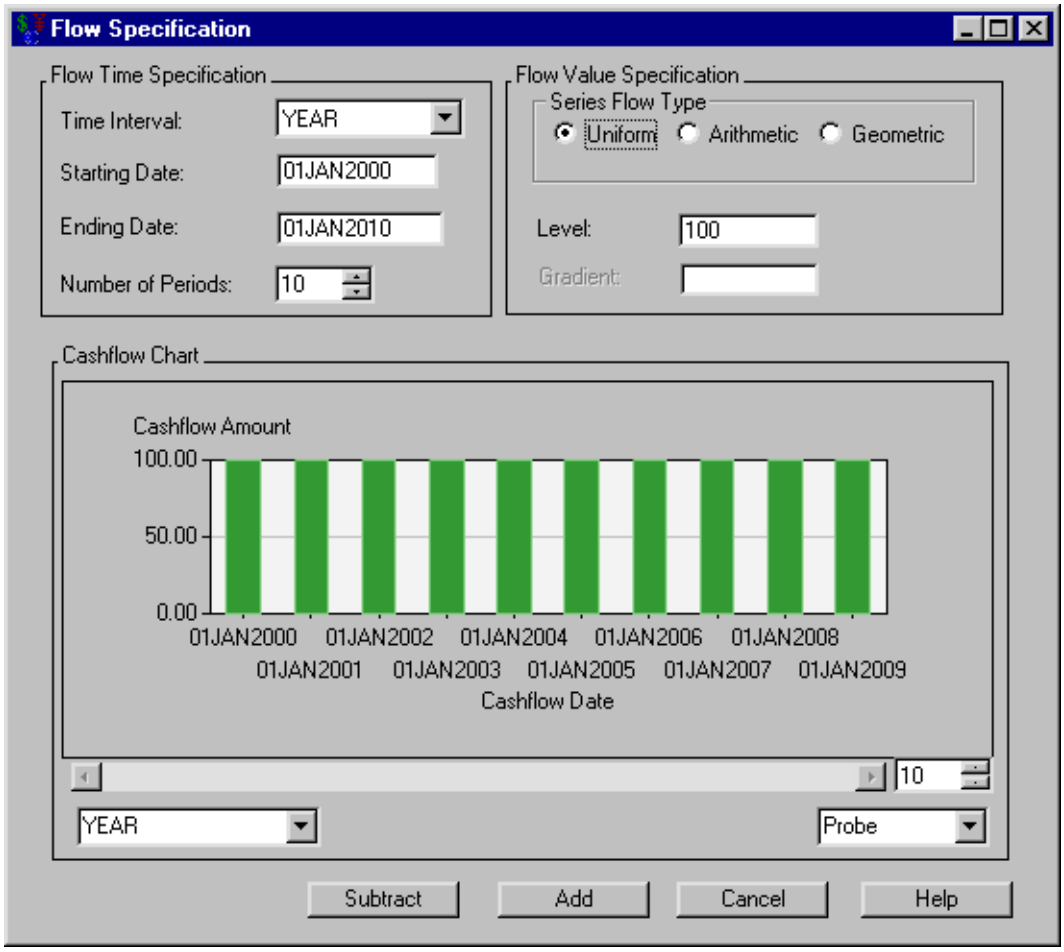
1. Right-click in the **Cashflow Specification** area and release on **Save**. This opens the Save Dataset dialog box.
2. Enter the name of the SAS data set for **Dataset Name**.
3. Click **OK** to return to the Generic Cashflow dialog box.

Including a Generated Cashflow

To generate date-amount pairs for the **Cashflow Specification** area, follow these steps:

1. Right-click in the **Cashflow Specification** area and release on **Specify**. This opens the Flow Specification dialog box.
2. Select YEAR for the **Time Interval**.
3. Enter today's date for the **Starting Date**.
4. Enter 10 for the **Number of Periods**. The **Ending Date** updates.
5. Enter 100 for the level. You can visualize the specification in the Cashflow Chart area (see Figure 64.19).
6. Click **Add** to add the specified cashflow to the list in the Generic Cashflow dialog box. Clicking **Add** also returns you to the Generic Cashflow dialog box.

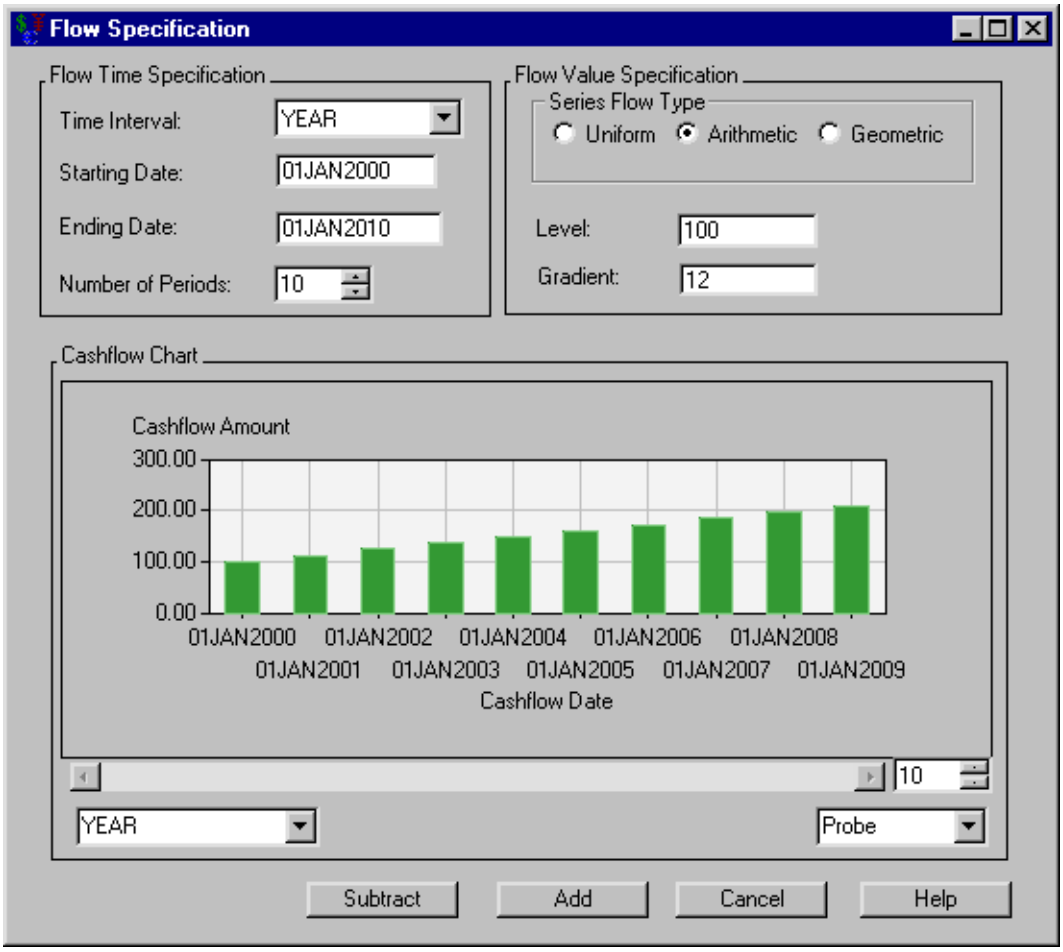
Figure 64.19 Uniform Cashflow Specification



Clicking **Subtract** subtracts the current cashflow from the Generic Cashflow dialog box, and it returns you to the Generic Cashflow dialog box.

You can generate arithmetic and geometric specifications by clicking them within the **Series Flow Type** area. However, you must enter a value for the **Gradient**. In both cases the **Level** value is the value of the list at the **Starting Date**. With an arithmetic flow type, entries increment by the value **Gradient** for each **Time Interval**. With a geometric flow type, entries increase by the factor **Gradient** for each **Time Interval**. Figure 64.20 displays an arithmetic cashflow with a **Level** of 100 and a **Gradient** of 12.

Figure 64.20 Arithmetic Cashflow Specification

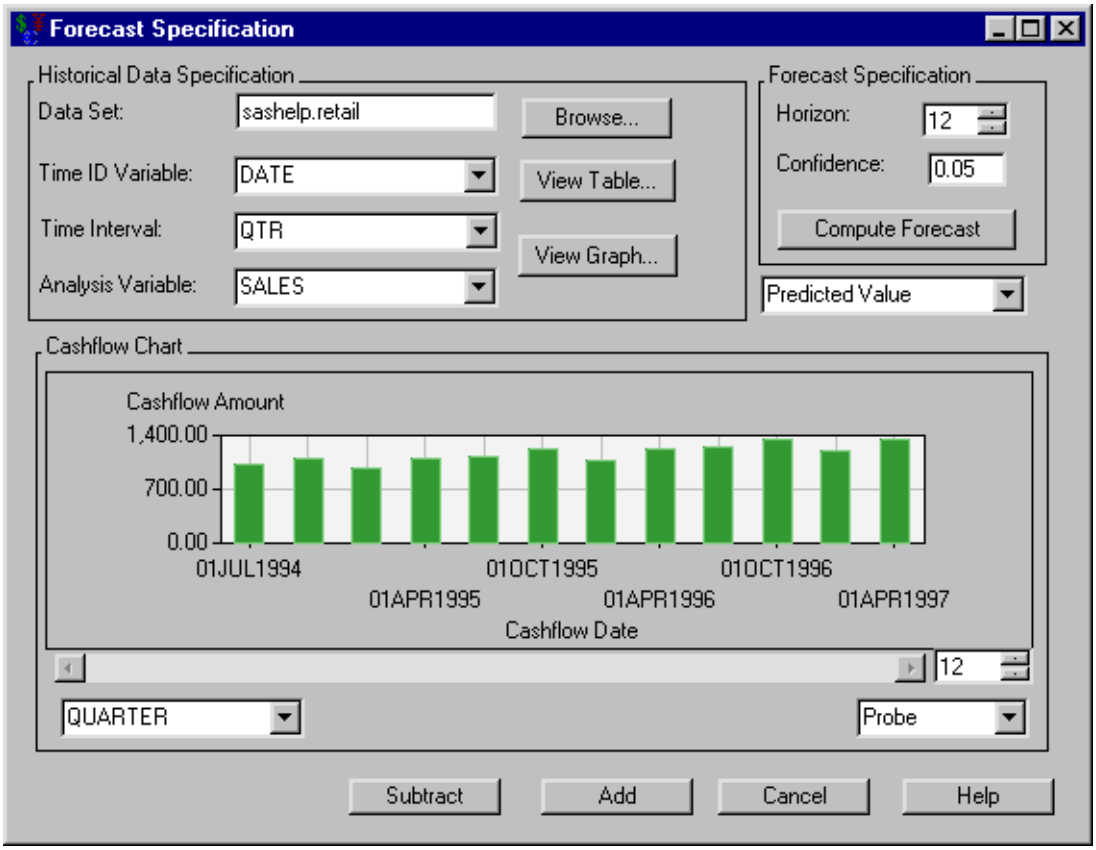


Including a Forecasted Cashflow

To generate date-amount pairs for the **Cashflow Specification** area, follow these steps:

1. Right-click in the **Cashflow Specification** area and release on **Forecast** to open the **Forecast Specification** dialog box.
2. Enter sashelp.retail as the **Data Set**.
3. Select SALES for the **Analysis Variable**.
4. Click **Compute Forecast** to generate the forecast. You can visualize the forecast in the Cashflow Chart area (see Figure 64.21).
5. Click **Add** to add the forecast to the list in the Generic Cashflow dialog box. Clicking **Add** also returns you to the Generic Cashflow dialog box.

Figure 64.21 Cashflow Forecast



Clicking **Subtract** subtracts the current forecast from the Generic Cashflow dialog box, and it returns you to the Generic Cashflow dialog box.

To review the values from the SAS data set you forecast, click **View Table** or **View Graph**.

You can adjust the following values for the SAS data set you forecast: **Time ID Variable**, **Time Interval**, and **Analysis Variable**.

You can adjust the following values for the forecast: the **Horizon**, the **Confidence**, and choice of predicted value, lower confidence limit, and upper confidence limit.

Using the Cashflow Chart

Three dialog boxes contain the Cashflow Chart to aide in your visualization of cashflows: Generic Cashflow, Flow Specification, and Forecast Specification. Within this chart, you possess the following tools:

You can click on a bar in the plot and view its **Cashflow Date** and **Cashflow Amount**.

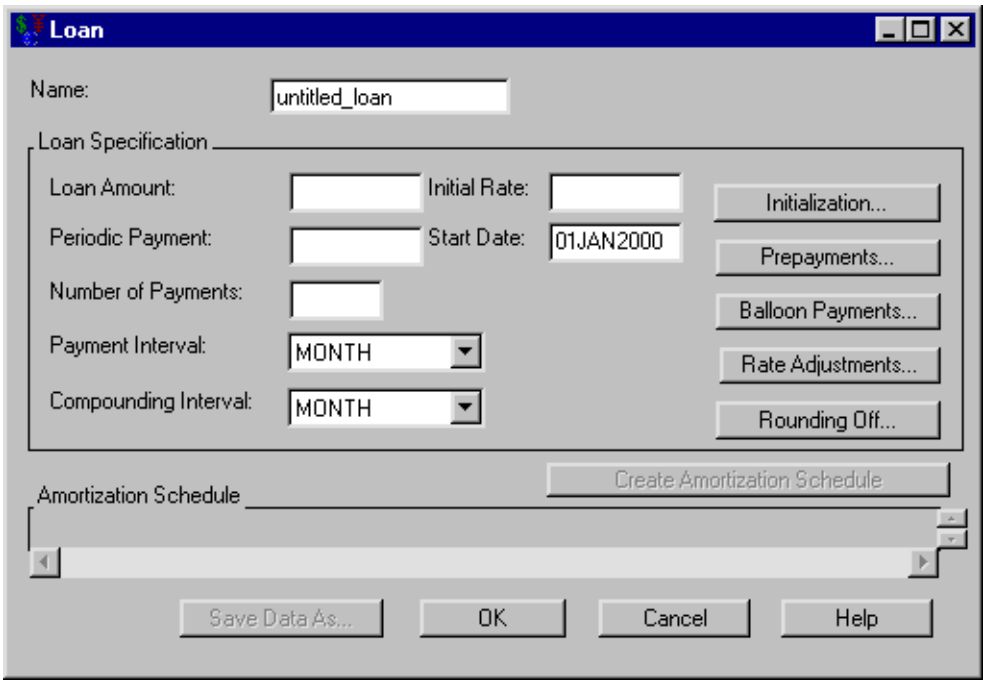
You can change the aggregation period of the view with the box in the lower-left corner of the Cashflow Chart. You can take the quarterly sales figures from the previous example, select YEAR as the value for this box, and view the annual sales figures. You can change the number in the box to the right of the horizontal scroll bar to alter the number of entries you want to view. The number in that box must be no greater than the number of entries in the cashflow list. Lessening this number has the effect of zooming in upon a portion of the cashflow. When the number is less than the number of entries in the cashflow list, you can use the scroll bar at the bottom of the chart to scroll through the chart.

Dialog Box Guide

Loan

Selecting **Investment** → **New** → **Loan** from the Investment Analysis dialog box's menu bar opens the Loan dialog box displayed in Figure 64.22.

Figure 64.22 Loan Dialog Box



The following items are displayed:

Name holds the name you assign to the loan. You can set the name here or within the **Portfolio** area of the Investment Analysis dialog box. This must be a valid SAS name.

The **Loan Specification** area gives access to the values that define the loan.

Loan Amount holds the borrowed amount.

Periodic Payment holds the value of the periodic payments.

Number of Payments holds the number of payments in loan terms.

Payment Interval holds the frequency of the Periodic Payment.

Compounding Interval holds the compounding frequency.

Initial Rate holds the interest rate (a nominal percentage between 0 and 120) you pay on the loan.

Start Date holds the SAS date when the loan is initialized. The first payment is due one Payment Interval after this time.

Initialization opens the Loan Initialization Options dialog box where you can define initialization costs and down-payments relevant to the loan.

Prepayments opens the Loan Prepayments dialog box where you can specify the SAS dates and amounts of any prepayments.

Balloon Payments opens the Balloon Payments dialog box where you can specify the SAS dates and amounts of any balloon payments.

Rate Adjustments opens the Rate Adjustment Terms dialog box where you can specify terms for a variable-rate loan.

Rounding Off opens the Rounding Off dialog box where you can select the number of decimal places for calculations.

Create Amortization Schedule becomes available when you adequately define the loan within the **Loan Specification** area. Clicking it generates the amortization schedule.

Amortization Schedule fills when you click **Create Amortization Schedule**. The schedule contains a row for the loan's start-date and each payment-date with information about the following:

Date is a SAS date, either the loan's start-date or a payment-date.

Beginning Principal Amount is the balance at that date.

Periodic Payment Amount is the expected payment at that date.

Interest Payment is zero for the loan's start-date; otherwise it holds the interest since the previous date.

Principal Repayment is the amount of the payment that went toward the principal.

Ending Principal is the balance at the end of the payment interval.

Print becomes available when you generate the amortization schedule. Clicking it sends the contents of the amortization schedule to the SAS session print device.

Save Data As becomes available when you generate the amortization schedule. Clicking it opens the Save Output Dataset dialog box where you can save the amortization table (or portions thereof) as a SAS Dataset.

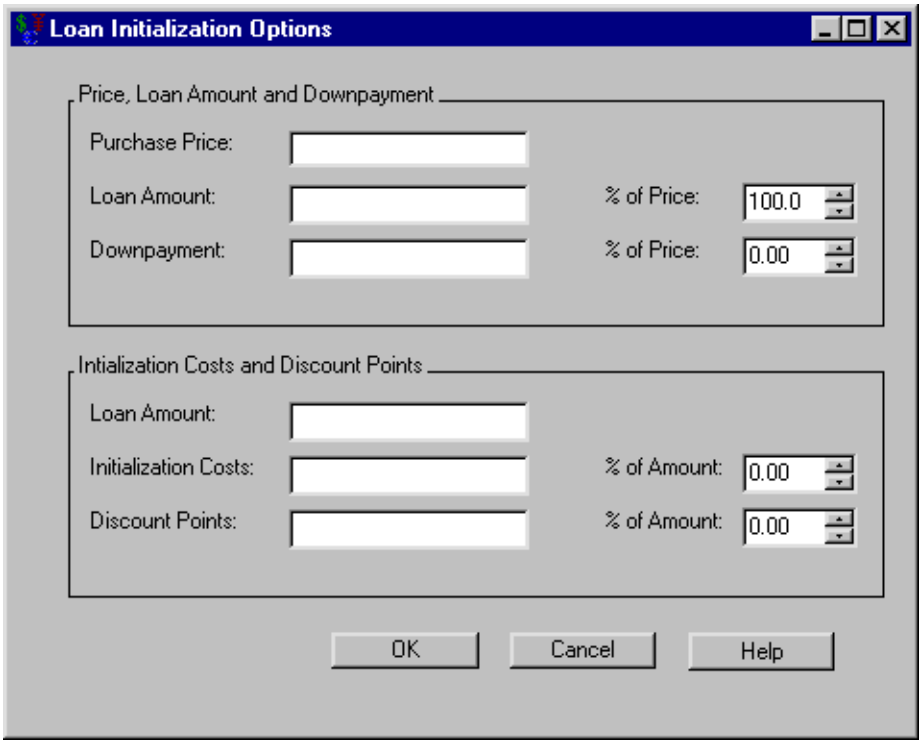
OK returns you to the Investment Analysis dialog box. If this is a new loan specification, clicking **OK** appends the current loan specification to the portfolio. If this is an existing loan specification, clicking **OK** returns the altered loan specification to the portfolio.

Cancel returns you to the Investment Analysis dialog box. If this is a new loan specification, clicking **Cancel** discards the current loan specification. If this is an existing loan specification, clicking **Cancel** discards the current changes.

Loan Initialization Options

Clicking **Initialization** in the Loan dialog box opens the Loan Initialization Options dialog box displayed in Figure 64.23.

Figure 64.23 Loan Initialization Options Dialog Box



The following items are displayed:

The **Price, Loan Amount and Downpayment** area contains the following information:

Purchase Price holds the actual price of the asset. This value equals the loan amount plus the downpayment.

Loan Amount holds the loan amount.

% of Price (to the right of **Loan Amount**) updates when you enter the **Purchase Price** and either the **Loan Amount** or **Downpayment**. This holds the percentage of the **Purchase Price** that comprises the **Loan Amount**. Setting the percentage manually causes the **Loan Amount** and **Downpayment** to update.

Downpayment holds any downpayment paid for the asset.

% of Price (to the right of **Downpayment**) updates when you enter the **Purchase Price** and either the **Loan Amount** or **Downpayment**. This holds the percentage of the **Purchase Price** that comprises the **Downpayment**. Setting the percentage manually causes the **Loan Amount** and **Downpayment** to update.

Initialization Costs and Discount Points area

Loan Amount holds a copy of the **Loan Amount** above.

Initialization Costs holds the value of any initialization costs.

% of Amount (to the right of **Initialization Costs**) updates when you enter the **Purchase Price** and either the **Initialization Costs** or **Discount Points**. This holds the percentage of the **Loan Amount** that comprises the **Initialization Costs**. Setting the percentage manually causes the **Initialization Costs** to update.

Discount Points holds the value of any discount points.

% of Amount (to the right of **Discount Points**) updates when you enter the **Purchase Price** and either the **Initialization Costs** or **Discount Points**. This holds the percentage of the **Loan Amount** that comprises the **Discount Points**. Setting the percentage manually causes the **Discount Points** to update.

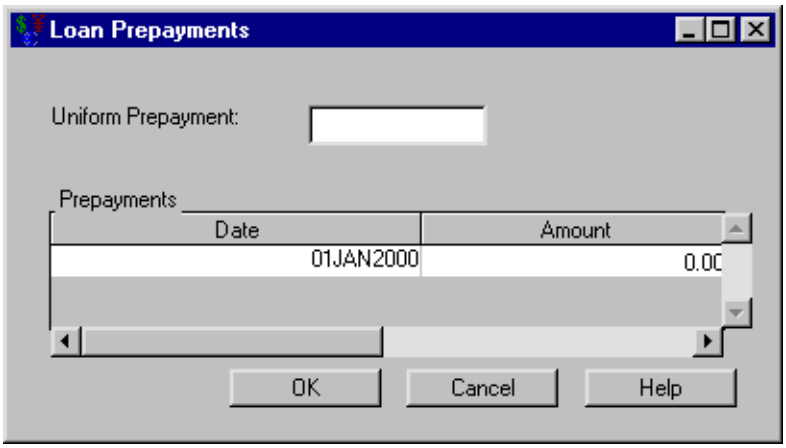
OK returns you to the Loan dialog box, saving the information that is entered.

Cancel returns you to the Loan dialog box, discarding any changes made since you opened the dialog box.

Loan Prepayments

Clicking **Prepayments** in the Loan dialog box opens the Loan Prepayments dialog box displayed in Figure 64.24.

Figure 64.24 Loan Prepayments Dialog Box



The following items are displayed:

Uniform Prepayment holds the value of a regular prepayment concurrent to the usual periodic payment.

Prepayments holds a list of date-amount pairs to accommodate any prepayments. Right-clicking within the **Prepayments** area reveals many helpful tools for managing date-amount pairs.

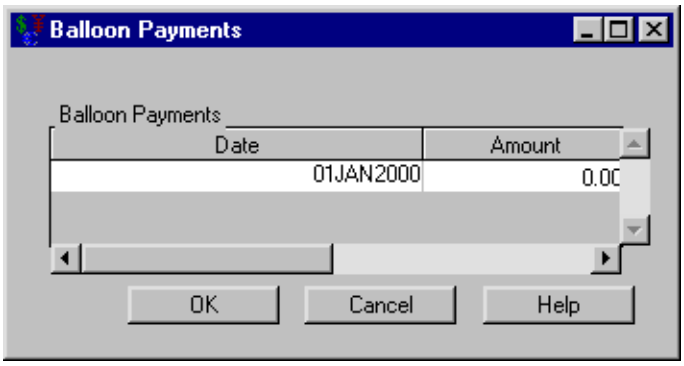
OK returns you to the Loan dialog box, storing the information entered on the prepayments.

Cancel returns you to the Loan dialog box, discarding any prepayments entered since you opened the dialog box.

Balloon Payments

Clicking **Balloon Payments** in the Loan dialog box opens the Balloon Payments dialog box displayed in Figure 64.25.

Figure 64.25 Balloon Payments Dialog Box



The following items are displayed:

Balloon Payments holds a list of date-amount pairs to accommodate any balloon payments. Right-clicking within the **Balloon Payments** area reveals many helpful tools for managing date-amount pairs.

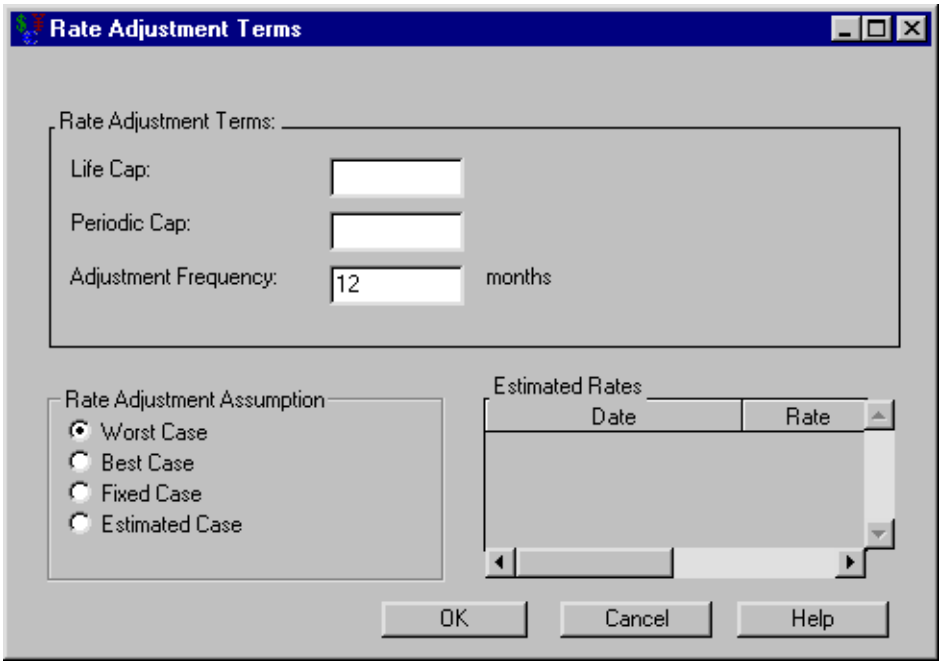
OK returns you to the Loan dialog box, storing the information entered on the balloon payments.

Cancel returns you to the Loan dialog box, discarding any balloon payments entered since you opened the dialog box.

Rate Adjustment Terms

Clicking **Rate Adjustments** in the Loan dialog box opens the Rate Adjustment Terms dialog box displayed in Figure 64.26.

Figure 64.26 Rate Adjustment Terms Dialog Box



The following items are displayed:

The **Rate Adjustment Terms** area

Life Cap holds the maximum deviation from the **Initial Rate** allowed over the life of the loan.

Periodic Cap holds the maximum adjustment allowed per adjustment.

Adjustment Frequency holds how often (in months) the lender can adjust the interest rate.

The **Rate Adjustment Assumption** determines the scenario the adjustments will take.

Worst Case uses information from the **Rate Adjustment Terms** area to forecast a worst-case scenario.

Best Case uses information from the **Rate Adjustment Terms** area to forecast a best-case scenario.

Fixed Case specifies a fixed-rate loan.

Estimated Case uses information from the **Rate Adjustment Terms** and **Estimated Rate** area to forecast a best-case scenario.

Estimated Rates holds a list of date-rate pairs, where each date is a SAS date and the rate is a nominal percentage between 0 and 120. The **Estimated Case** assumption uses these rates for its calculations. Right-clicking within the **Estimated Rates** area reveals many helpful tools for managing date-rate pairs.

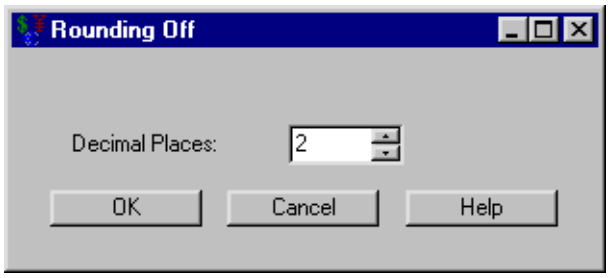
OK returns you to the Loan dialog box, taking rate adjustment information into account.

Cancel returns you to the Loan dialog box, discarding any rate adjustment information provided since opening the dialog box.

Rounding Off

Clicking **Rounding Off** in the Loan dialog box opens the Rounding Off dialog box displayed in Figure 64.27.

Figure 64.27 Rounding Off Dialog Box



The following items are displayed:

Decimal Places fixes the number of decimal places your results will display.

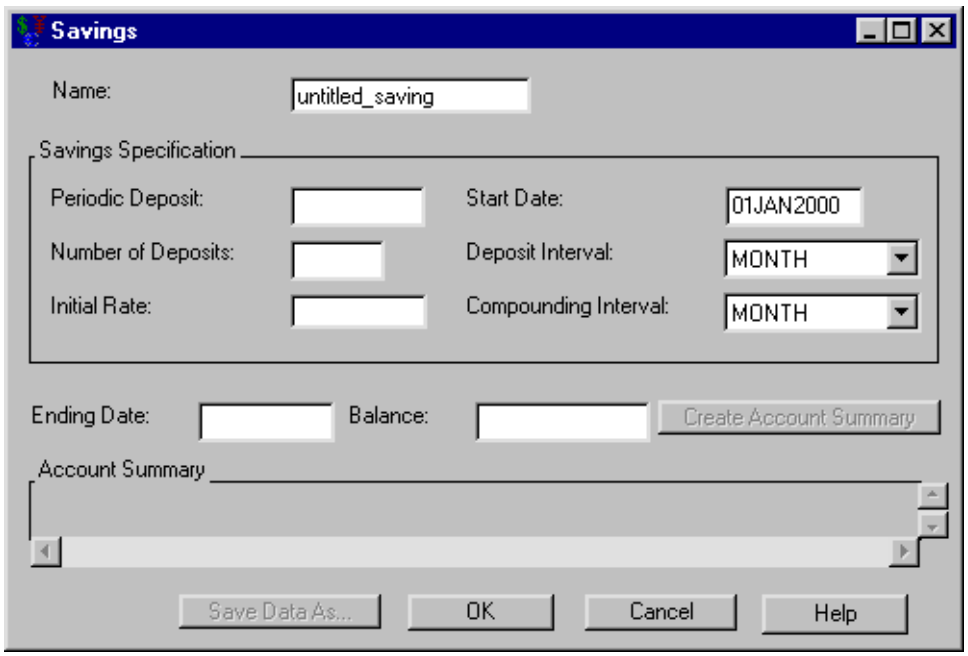
OK returns you to the Loan dialog box. Numeric values will then be represented with the number of decimals specified in **Decimal Places**.

Cancel returns you to the Loan dialog box. Numeric values will be represented with the number of decimals specified prior to opening this dialog box.

Savings

Selecting **Investment** → **New** → **Savings** from the Investment Analysis dialog box's menu bar opens the Savings dialog box displayed in Figure 64.28.

Figure 64.28 Savings Dialog Box



The following items are displayed:

Name holds the name you assign to the savings. You can set the name here or within the **Portfolio** area of the Investment Analysis dialog box. This must be a valid SAS name.

The **Savings Specification** area

> **Periodic Deposit** holds the value of your regular deposits.
>
> **Number of Deposits** holds the number of deposits into the account.
>
> **Initial Rate** holds the interest rate (a nominal percentage between 0 and 120) the savings account earns.
>
> **Start Date** holds the SAS date when deposits begin.
>
> **Deposit Interval** holds the frequency of your **Periodic Deposit**.
>
> **Compounding Interval** holds how often the interest compounds.

Create Account Summary becomes available when you adequately define the savings within the **Savings Specification** area. Clicking it generates the account summary.

Account Summary fills when you click **Create Account Summary**. The schedule contains a row for each deposit-date with information about the following:

> **Date** is the SAS date of a deposit.
>
> **Starting Balance** is the balance at that date.
>
> **Deposits** is the deposit at that date.
>
> **Interest Earned** is the interest earned since the previous date.
>
> **Ending Balance** is the balance after the payment.

Print becomes available when you generate an account summary. Clicking it sends the contents of the account summary to the SAS session print device.

Save Data As becomes available when you generate an account summary. Clicking it opens the Save Output Dataset dialog box where you can save the account summary (or portions thereof) as a SAS Dataset.

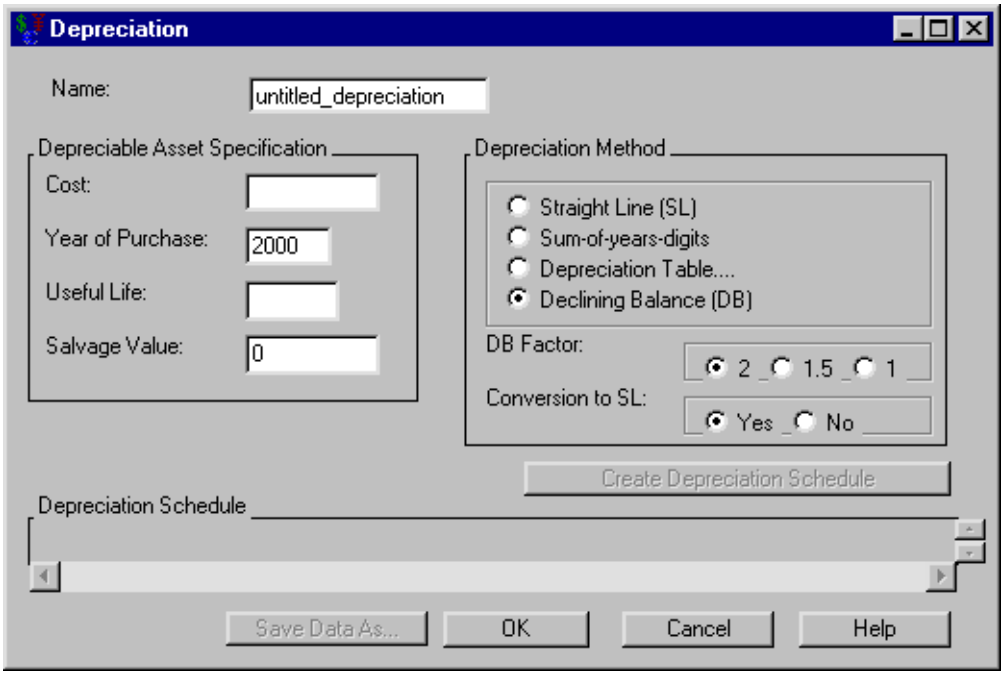
OK returns you to the Investment Analysis dialog box. If this is a new savings, clicking **OK** appends the current savings specification to the portfolio. If this is an existing savings specification, clicking **OK** returns the altered savings to the portfolio.

Cancel returns you to the Investment Analysis dialog box. If this is a new savings, clicking **Cancel** discards the current savings specification. If this is an existing savings, clicking **Cancel** discards the current changes.

Depreciation

Selecting **Investment** → **New** → **Depreciation** from the Investment Analysis dialog box's menu bar opens the Depreciation dialog box displayed in Figure 64.29.

Figure 64.29 Depreciation Dialog Box



The following items are displayed:

Name holds the name you assign to the depreciation. You can set the name here or within the **Portfolio** area of the Investment Analysis dialog box. This must be a valid SAS name.

Depreciable Asset Specification

Cost holds the asset's original cost.

Year of Purchase holds the asset's year of purchase.

Useful Life holds the asset's useful life (in years).

Salvage Value holds the asset's value at the end of its **Useful Life**.

The **Depreciation Method** area holds the depreciation methods available:

- Straight Line
- Sum-of-years Digits
- Depreciation Table
- Declining Balance
  - DB Factor: choice of 2, 1.5, or 1
  - Conversion to SL: choice of Yes or No

Create Depreciation Schedule becomes available when you adequately define the depreciation within the **Depreciation Asset Specification** area. Clicking the **Create Depreciation Schedule** button then fills the **Depreciation Schedule** area.

Depreciation Schedule fills when you click **Create Depreciation Schedule**. The schedule contains a row for each year. Each row holds:

Year is a year.

Start Book Value is the starting book value for that year.

Depreciation is the depreciation value for that year.

End Book Value is the ending book value for that year.

Print becomes available when you generate the depreciation schedule. Clicking it sends the contents of the depreciation schedule to the SAS session print device.

Save Data As becomes available when you generate the depreciation schedule. Clicking it opens the Save Output Dataset dialog box where you can save the depreciation table (or portions thereof) as a SAS Dataset.

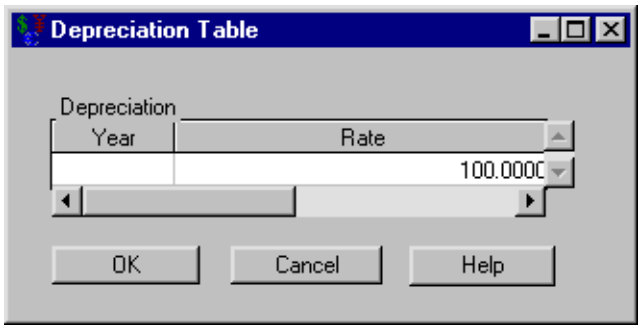
OK returns you to the Investment Analysis dialog box. If this is a new depreciation specification, clicking **OK** appends the current depreciation specification to the portfolio. If this is an existing depreciation specification, clicking **OK** returns the altered depreciation specification to the portfolio.

Cancel returns you to the Investment Analysis dialog box. If this is a new depreciation specification, clicking **Cancel** discards the current depreciation specification. If this is an existing depreciation specification, clicking **Cancel** discards the current changes.

Depreciation Table

Clicking **Depreciation Table** from **Depreciation Method** area of the Depreciation dialog box opens the Depreciation Table dialog box displayed in Figure 64.30.

Figure 64.30 Depreciation Table Dialog Box



The following items are displayed:

The **Depreciation** area holds a list of year-rate pairs where the rate is an annual depreciation rate (a percentage between 0% and 100%). Right-clicking within the **Depreciation** area reveals many helpful tools for managing year-rate pairs.

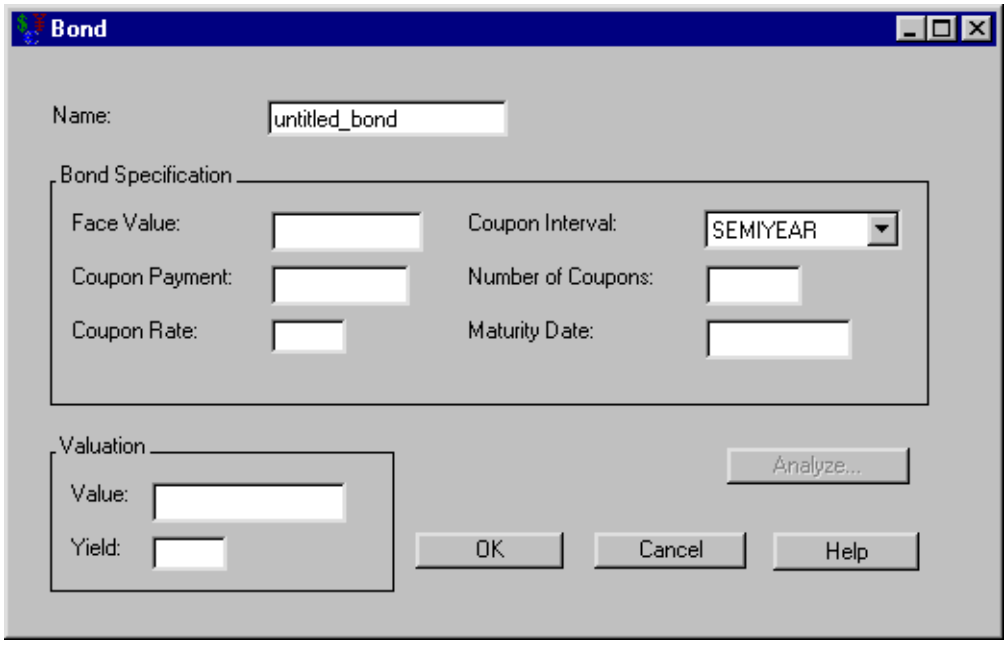
OK returns you to the Depreciation dialog box with the current list of depreciation rates from the **Depreciation** area.

Cancel returns you to the Depreciation dialog box, discarding any editions to the **Depreciation** area since you opened the dialog box.

Bond

Selecting **Investment** → **New** → **Bond** from the Investment Analysis dialog box's menu bar opens the Bond dialog box displayed in Figure 64.31.

Figure 64.31 Bond



The following items are displayed:

Name holds the name you assign to the bond. You can set the name here or within the **Portfolio** area of the Investment Analysis dialog box. This must be a valid SAS name.

Bond Specification

> **Face Value** holds the bond's value at maturity.
>
> **Coupon Payment** holds the amount of money you receive periodically as the bond matures.
>
> **Coupon Rate** holds the rate (a nominal percentage between 0% and 120%) of the **Face Value** that defines the **Coupon Payment**.
>
> **Coupon Interval** holds how often the bond pays its coupons.
>
> **Number of Coupons** holds the number of coupons before maturity.
>
> **Maturity Date** holds the SAS date when you can redeem the bond for its **Face Value**.

The **Valuation** area becomes available when you adequately define the bond within the **Bond Specification** area. Entering either the **Value** or the **Yield** causes the calculation of the other. If you respecify the bond after performing a calculation here, you must reenter the **Value** or **Yield** value to update the calculation.

> **Value** holds the bond's value if expecting the specified **Yield**.
>
> **Yield** holds the bond's yield if the bond is valued at the amount of **Value**.

You must specify the bond before analyzing it. After you have specified the bond, clicking **Analyze** opens the Bond Analysis dialog box where you can compare various values and yields.

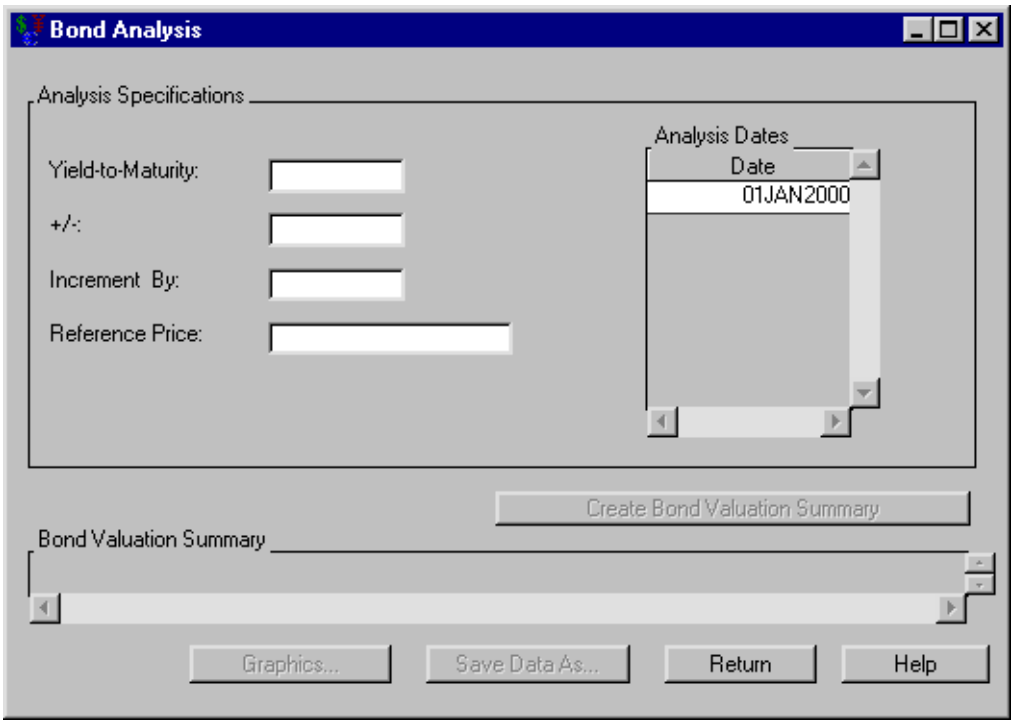
OK returns you to the Investment Analysis dialog box. If this is a new bond specification, clicking **OK** appends the current bond specification to the portfolio. If this is an existing bond specification, clicking **OK** returns the altered bond specification to the portfolio.

Cancel returns you to the Investment Analysis dialog box. If this is a new bond specification, clicking **Cancel** discards the current bond specification. If this is an existing bond specification, clicking **Cancel** discards the current changes.

Bond Analysis

Clicking **Analyze** from the Bond dialog box opens the Bond Analysis dialog box displayed in Figure 64.32.

Figure 64.32 Bond Analysis



The following items are displayed:

Analysis Specifications

Yield-to-maturity holds the percentage yield upon which to center the analysis.

+/- holds the maximum deviation percentage to consider from the **Yield-to-maturity**.

Increment by holds the percentage increment by which the analysis is calculated.

Reference Price holds the reference price.

Analysis Dates holds a list of SAS dates for which you perform the bond analysis.

You must specify the analysis before valuing the bond for the various yields. After you adequately specify the analysis, click **Create Bond Valuation Summary** to generate the bond valuation summary.

Bond Valuation Summary fills when you click **Create Bond Valuation Summary**. The schedule contains a row for each rate with information concerning the following:

Date is the SAS date when the **Value** gives the particular **Yield**.

Yield is the percent yield that corresponds to the **Value** at the given **Date**.

Value is the value of the bond at **Date** for the given **Yield**.

Percent Change is the percent change if the **Reference Price** is specified.

Duration is the duration.

Convexity is the convexity.

Graphics opens the Bond Price graph that represents the price versus yield-to-maturity.

Print becomes available when you generate the **Bond Valuation Summary**. Clicking it sends the contents of the summary to the SAS session print device.

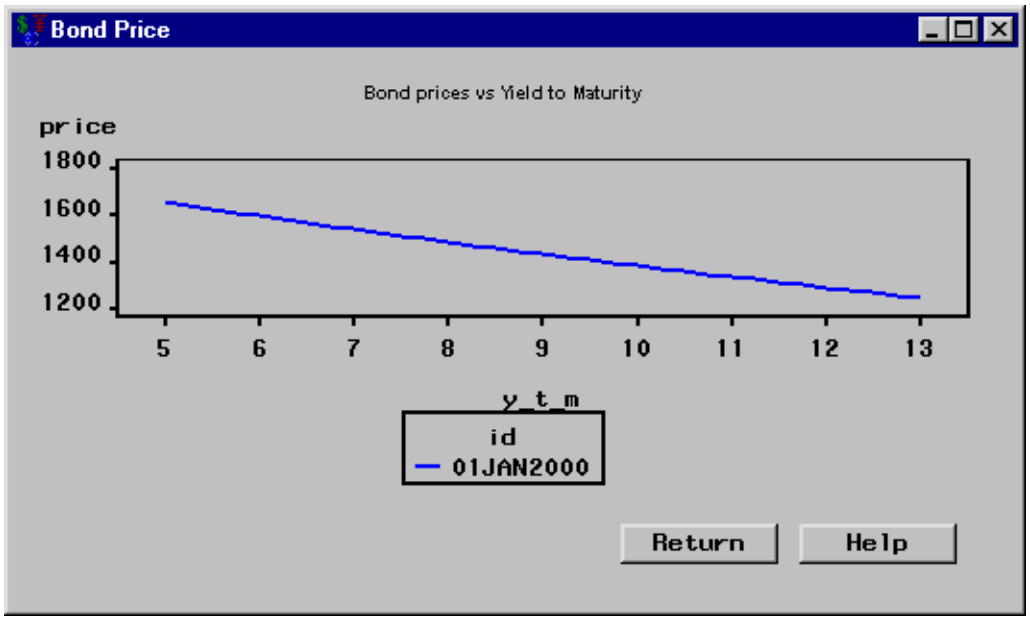
Save Data As becomes available when you fill the **Bond Valuation Summary** area. Clicking it opens the Save Output Dataset dialog box where you can save the valuation summary (or portions thereof) as a SAS Dataset.

Return takes you back to the Bond dialog box.

Bond Price

Clicking **Graphics** from the Bond dialog box opens the Bond Price dialog box displayed in Figure 64.33.

Figure 64.33 Bond Price Graph



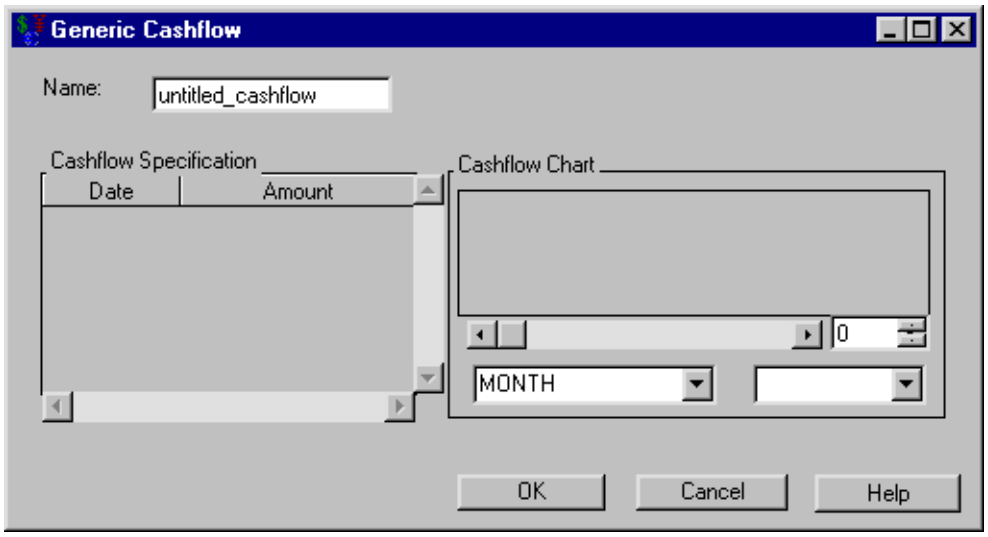
It possesses the following item:

Return takes you back to the Bond Analysis dialog box.

Generic Cashflow

Selecting **Investment** → **New** → **Generic Cashflow** from the Investment Analysis dialog box's menu bar opens the Generic Cashflow dialog box displayed in Figure 64.34.

Figure 64.34 Generic Cashflow



The following items are displayed:

Name holds the name you assign to the generic cashflow. You can set the name here or within the **Portfolio** area of the Investment Analysis dialog box. This must be a valid SAS name.

Cashflow Specification holds date-amount pairs that correspond to deposits and withdrawals (or benefits and costs) for the cashflow. Each date is a SAS date. Right-clicking within the **Cashflow Specification** area reveals many helpful tools for managing date-amount pairs.

The **Cashflow Chart** fills with a graph representing the cashflow when the **Cashflow Specification** area is nonempty. The box to the right of the scroll bar controls the number of entries with which to fill the graph. If the number in this box is less than the total number of entries, you can use the scroll bar to view different segments of the cashflow. The left box below the scroll bar holds the frequency for drilling purposes.

OK returns you to the Investment Analysis dialog box. If this is a new generic cashflow specification, clicking **OK** appends the current cashflow specification to the portfolio. If this is an existing cashflow specification, clicking **OK** returns the altered cashflow specification to the portfolio.

Cancel returns you to the Investment Analysis dialog box. If this is a new cashflow specification, clicking **Cancel** discards the current cashflow specification. If this is an existing cashflow specification, clicking **Cancel** discards the current changes.

Right-Clicking within Generic Cashflow's Cashflow Specification Area

Right-click within the **Cashflow Specification** area of the Generic Cashflow dialog box pops up the menu displayed in Figure 64.35.

Figure 64.35 Right-Clicking



Add creates a blank pair.

Delete removes the currently highlighted pair.

Copy duplicates the currently selected pair.

Sort arranges the entered pairs in chronological order.

Clear empties the area of all pairs.

Save opens the Save Dataset dialog box where you can save the entered pairs as a SAS Dataset for later use.

Load opens the Load Dataset dialog box where you select a SAS Dataset to populate the area.

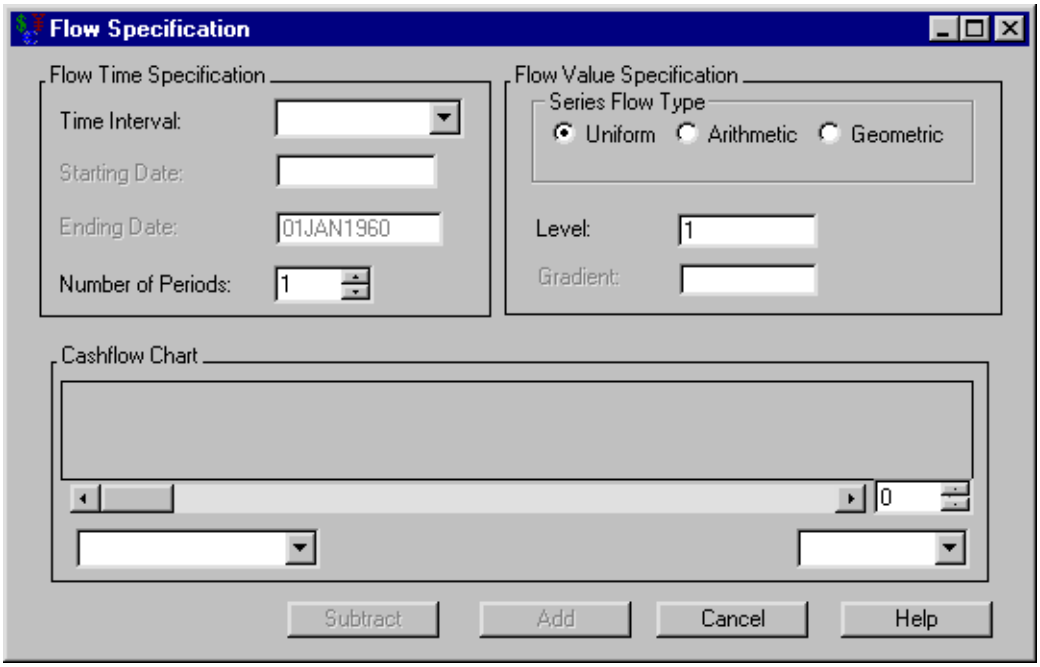
Specify opens the Flow Specification dialog box where you can generate date-rate pairs to include in your cashflow.

Forecast opens the Forecast Specification dialog box where you can generate the forecast of a SAS data set to include in your cashflow.

If you want to perform one of these actions on a collection of pairs, you must select a collection of pairs before right-clicking. To select an adjacent list of pairs, do the following: click the first pair, hold down the SHIFT key, and click the final pair. After the list of pairs is selected, you can release the SHIFT key.

Flow Specification

Figure 64.36 Flow Specification



The following items are displayed:

Flow Time Specification

Time Interval holds the uniform frequency of the entries.

You can set the **Starting Date** when you set the **Time Interval**. It holds the SAS date the entries will start.

You can set the **Ending Date** when you set the **Time Interval**. It holds the SAS date the entries will end.

Number of Periods holds the number of entries.

Flow Value Specification

Series Flow Type describes the movement the entries can assume:

- **Uniform** assumes all entries are equal.
- **Arithmetic** assumes the entries begin at **Level** and increase by the value of **Gradient** per entry.
- **Geometric** assumes the entries begin at **Level** and increase by a factor of **Gradient** per entry.

Level holds the starting amount for all flow types.

You can set the **Gradient** when you select either **Arithmetic** or **Geometric** series flow type. It holds the arithmetic and geometric gradients, respectively, for the **Arithmetic** and **Geometric** flow types.

When the cashflow entries are adequately defined, the **Cashflow Chart** fills with a graph displaying the dates and values of the entries. The box to the right of the scroll bar controls the number of entries with which to fill the graph. If the number in this box is less than the total number of entries, you can use the scroll bar to view different segments of the cashflow. The left box below the scroll bar holds the frequency.

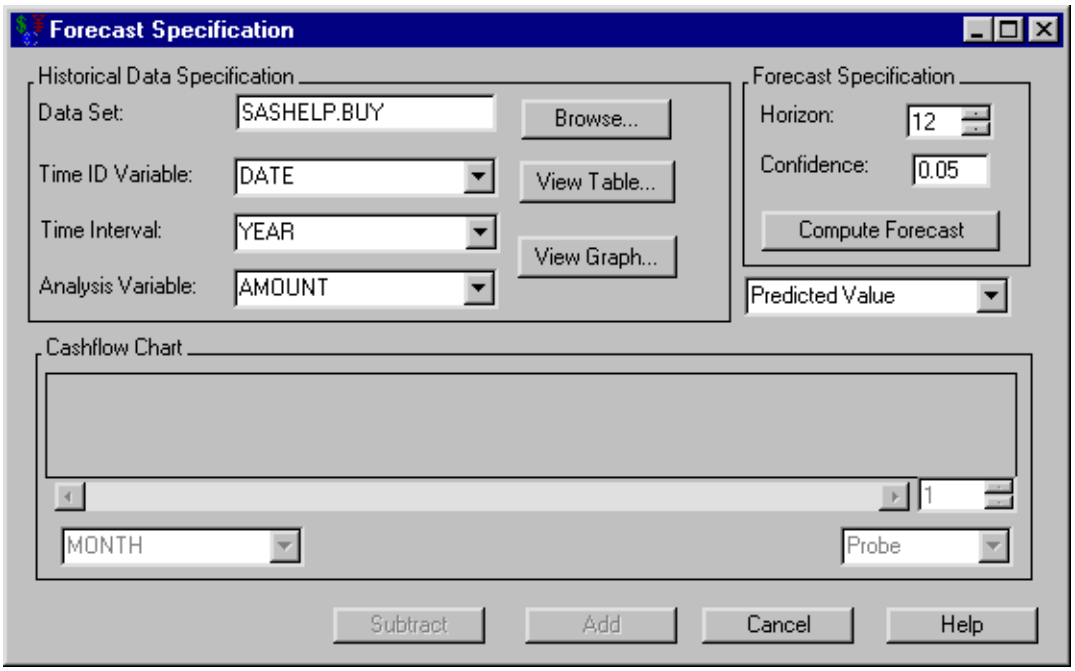
Subtract becomes available when the collection of entries is adequately specified. Clicking **Subtract** then returns you to the Generic Cashflow dialog box and subtracts the entries from the current cashflow.

Add becomes available when the collection of entries is adequately specified. Clicking **Add** then returns you to the Generic Cashflow dialog box and adds the entries to the current cashflow.

Cancel returns you to Generic Cashflow dialog box without changing the cashflow.

Forecast Specification

Figure 64.37 Forecast Specification



The following items are displayed:

Historical Data Specification

Data Set holds the name of the SAS data set to forecast.

Browse opens the standard SAS **Open** dialog box to help select a SAS data set to forecast.

Time ID Variable holds the time ID variable to forecast over.

Time Interval fixes the time interval for the **Time ID Variable**.

Analysis Variable holds the data variable upon which to forecast.

View Table opens a table that displays the contents of the specified SAS data set.

View Graph opens the Time Series Viewer that graphically displays the contents of the specified SAS data set.

Forecast Specification

Horizon holds the number of periods into the future you want to forecast.

Confidence holds the confidence limit for applicable forecasts.

Compute Forecast fills the **Cashflow Chart** with the forecast.

The box below **Forecast Specification** holds the type of forecast you want to generate:

- Predicted Value
- Lower Confidence Limit
- Upper Confidence Limit

The **Cashflow Chart** fills when you click **Compute Forecast**. The box to the right of the scroll bar controls the number of entries with which to fill the graph. If the number in this box is less than the total number of entries, you can use the scroll bar to view different segments of the cashflow. The left box below the scroll bar holds the frequency.

Subtract becomes available when the collection of entries is adequately specified. Clicking **Subtract** then returns you to the Generic Cashflow dialog box subtracting the forecast from the current cashflow.

Add becomes available when the collection of entries is adequately specified. Clicking **Add** then returns you to the Generic Cashflow adding the forecast to the current cashflow.

Cancel returns to Generic Cashflow dialog box without changing the cashflow.

Chapter 65
Computations

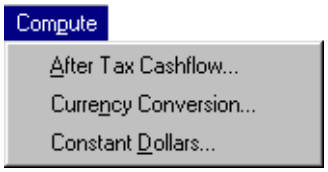
Contents

The Compute Menu **3971**
Tasks **3972**
Taxing a Cashflow 3972
Converting Currency 3974
Deflating Cashflows 3976
Dialog Box Guide **3978**
After Tax Cashflow Calculation 3978
Currency Conversion 3979
Constant Dollar Calculation 3980

The Compute Menu

Figure 65.1 shows the **Compute** menu.

Figure 65.1 The Compute Menu



The **Compute** menu offers the following options that apply to generic cashflows.

After Tax Cashflow opens the After Tax Cashflow Calculation dialog box. Computing an after tax cashflow is useful when taxes affect investment alternatives differently. Comparing after tax cashflows provides a more accurate determination of the cashflows' profitabilities. You can set default values for income tax rates by selecting **Tools** → **Define Rate** → **Income Tax Rate** from the Investment Analysis dialog box. This opens the Income Tax Specification dialog box where you can enter the tax rates.

Currency Conversion opens the Currency Conversion dialog box. Currency conversion is necessary when investments are in different currencies. For data concerning currency conversion rates, see `http://dsbb.imf.org/`, the International Monetary Fund's Dissemination Standards Bulletin Board.

Constant Dollars opens the Constant Dollar Calculation dialog box. A constant dollar (inflation adjusted monetary value) calculation takes cashflow and inflation information and discounts the cashflow to a level where the buying power of the monetary unit is constant over time. Groups quantify inflation (in the

form of price indices and inflation rates) for countries and industries by averaging the growth of prices for various products and sectors of the economy. For data concerning price indices, see the United States Department of Labor at `http://www.dol.gov/` and the International Monetary Fund's Dissemination Standards Bulletin Board at `http://dsbb.imf.org/`. You can set default values for inflation rates by clicking **Tools** → **Define Rate** → **Inflation** from the Investment Analysis dialog box. This opens the Inflation Specification dialog box where you can enter the inflation rates.

Tasks

The next few sections show how to perform computations for the following situation. Suppose you buy a $10,000 certificate of deposit that pays 12% interest a year for five years. Your earnings are taxed at a rate of 30% federally and 7% locally. Also, you want to transfer all the money to an account in England. British pounds convert to American dollars at an exchange rate of $1.00 to £0.60. The inflation rate in England is 3%. The instructions in this example assume familiarity with the following:

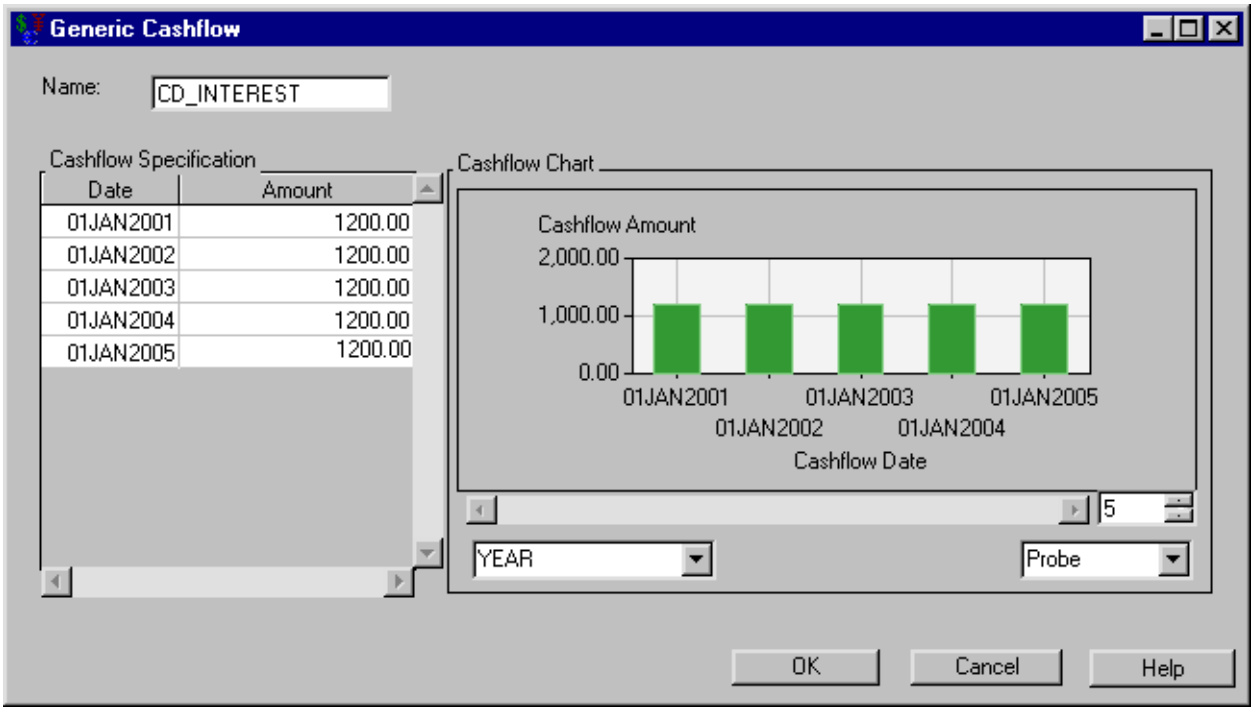
- The right-clicking options of the **Cashflow Specification** area in the Generic Cashflow dialog box (described in the section "Right-Clicking within Generic Cashflow's Cashflow Specification Area" on page 3967.)
- The **Save Data As** button located in many dialog boxes (described in the section "Saving Output to SAS Data Sets" on page 3997.)

Taxing a Cashflow

Consider the example described in the section "The Compute Menu" on page 3971. To create the earnings, follow these steps:

1. Select **Investment** → **New** → **Generic Cashflow** to create a generic cashflow.
2. Enter CD_INTEREST for the **Name**.
3. Enter 1200 for each of the five years starting one year from today as displayed in Figure 65.2.
4. Click **OK** to return to the Investment Analysis dialog box.

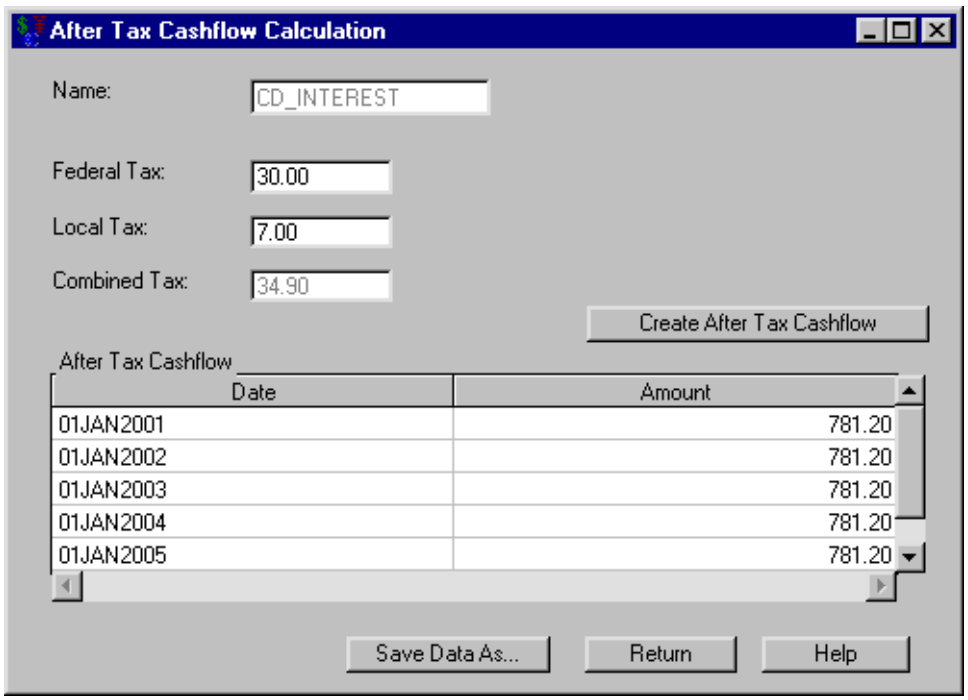
Figure 65.2 Computing the Interest on the CD



To compute the tax on the earnings, follow these steps:

1. Select CD_INTEREST from the **Portfolio** area.
2. Select **Compute** → **After Tax Cashflow** from the pull-down menu.
3. Enter 30 for **Federal Tax**.
4. Enter 7 for **Local Tax**. Note that **Combined Tax** updates.
5. Click **Create After Tax Cashflow**. The **After Tax Cashflow** area fills, as displayed in Figure 65.3.

Figure 65.3 Computing the Interest After Taxes



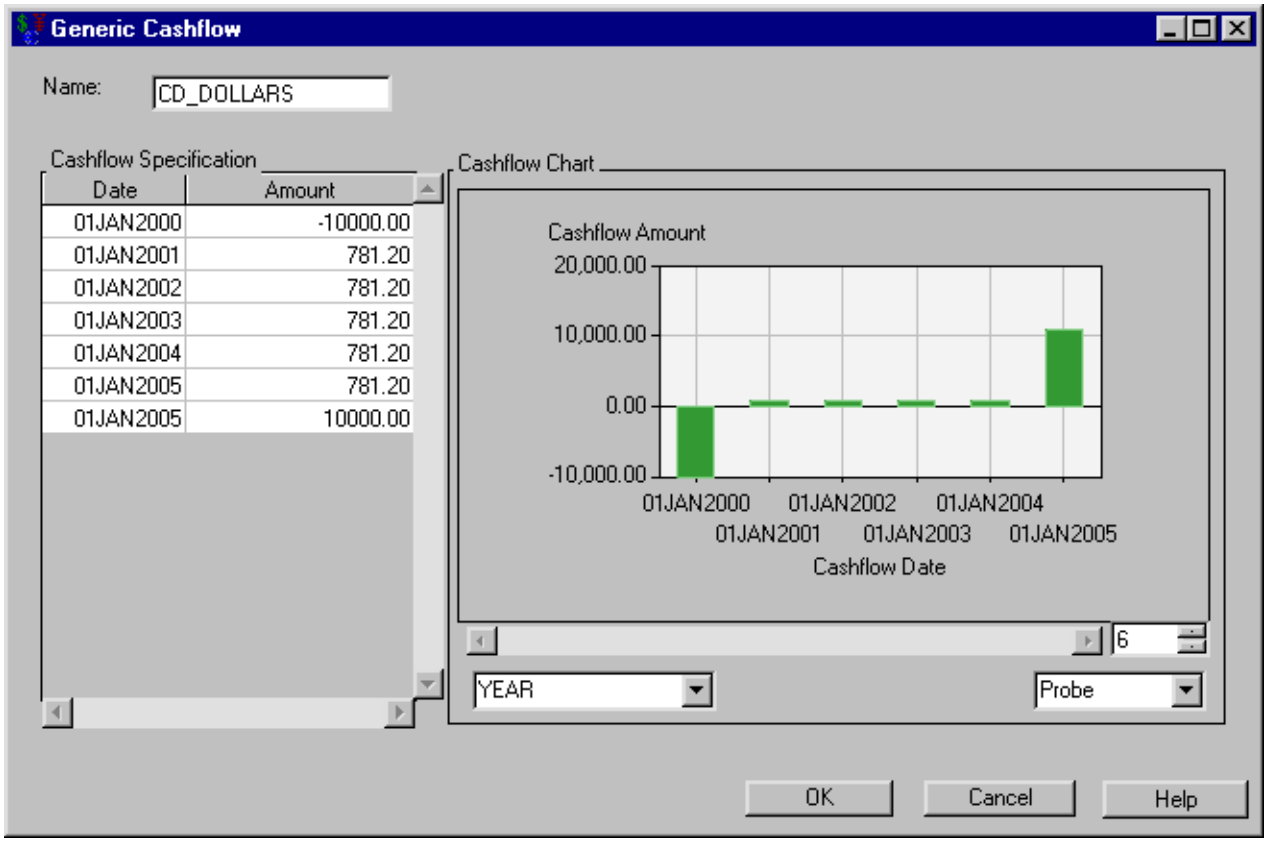
Save the taxed earnings to a SAS data set named WORK.CD_AFTERTAX. Click **Return** to return to the Investment Analysis dialog box.

Converting Currency

Consider the example described in the section "The Compute Menu" on page 3971. To create the cashflow to convert, follow these steps:

1. Select **Investment** → **New** → **Generic Cashflow** to open a new generic cashflow.
2. Enter CD_DOLLARS for the **Name**.
3. Load WORK.CD_AFTERTAX into its **Cashflow Specification**.
4. Add –10,000 for today and +10,000 for five years from today to the cashflow as displayed in Figure 65.4.
5. Sort the transactions by date to aid your reading.
6. Click **OK** to return to the Investment Analysis dialog box.

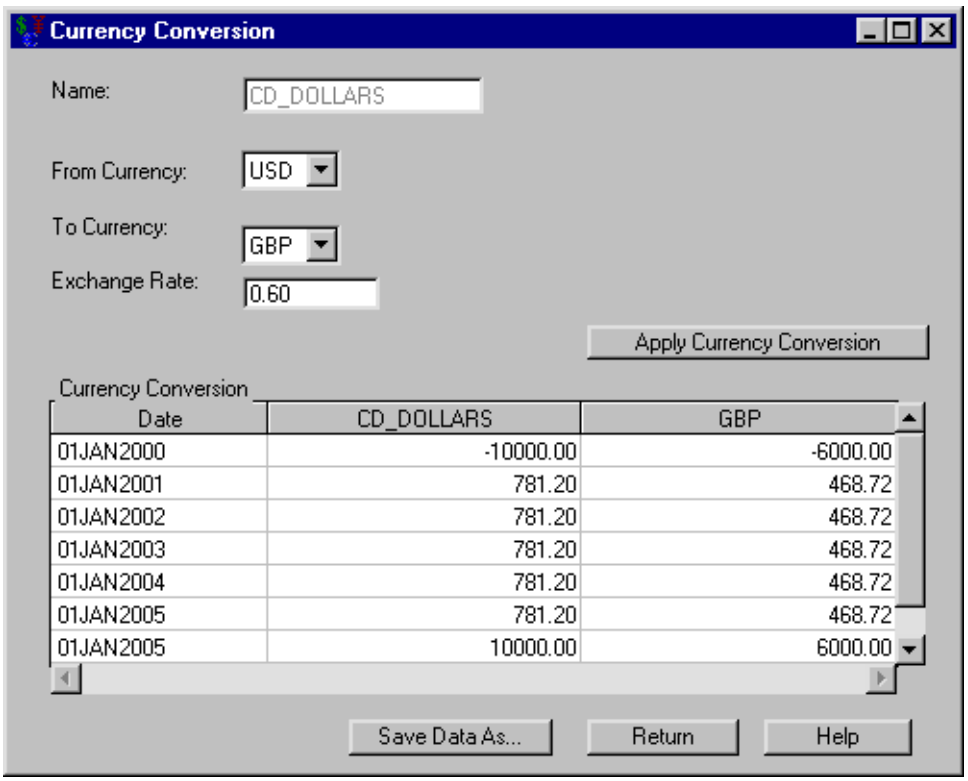
Figure 65.4 The CD in Dollars



To convert from British pounds to American dollars, follow these steps:

1. Select CD_DOLLARS from the portfolio.
2. Select **Compute** → **Currency Conversion** from the pull-down menu. This opens the Currency Conversion dialog box.
3. Select USD for the **From Currency**.
4. Select GBP for the **To Currency**.
5. Enter 0.60 for the **Exchange Rate**.
6. Click **Apply Currency Conversion** to fill the **Currency Conversion** area as displayed in Figure 65.5.

Figure 65.5 Converting the CD to Pounds



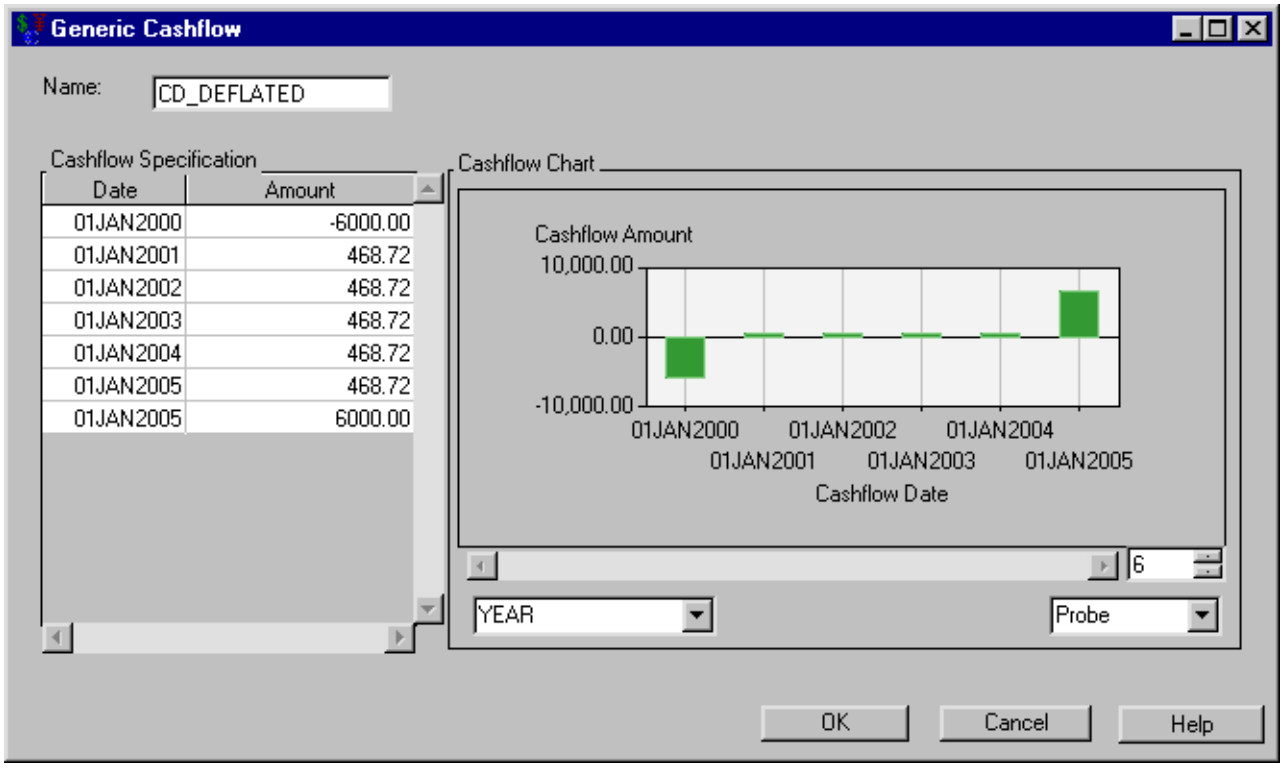
Save the converted values to a SAS data set named WORK.CD_POUNDS. Click **Return** to return to the Investment Analysis dialog box.

Deflating Cashflows

Consider the example described in the section "The Compute Menu" on page 3971. To create the cashflow to deflate, follow these steps:

1. Select **Investment** → **New** → **Generic Cashflow** to open a new generic cashflow.
2. Enter CD_DEFLATED for **Name**.
3. Load WORK.CD_POUNDS into its **Cashflow Specification** (see Figure 65.6).
4. Click **OK** to return to the Investment Analysis dialog box.

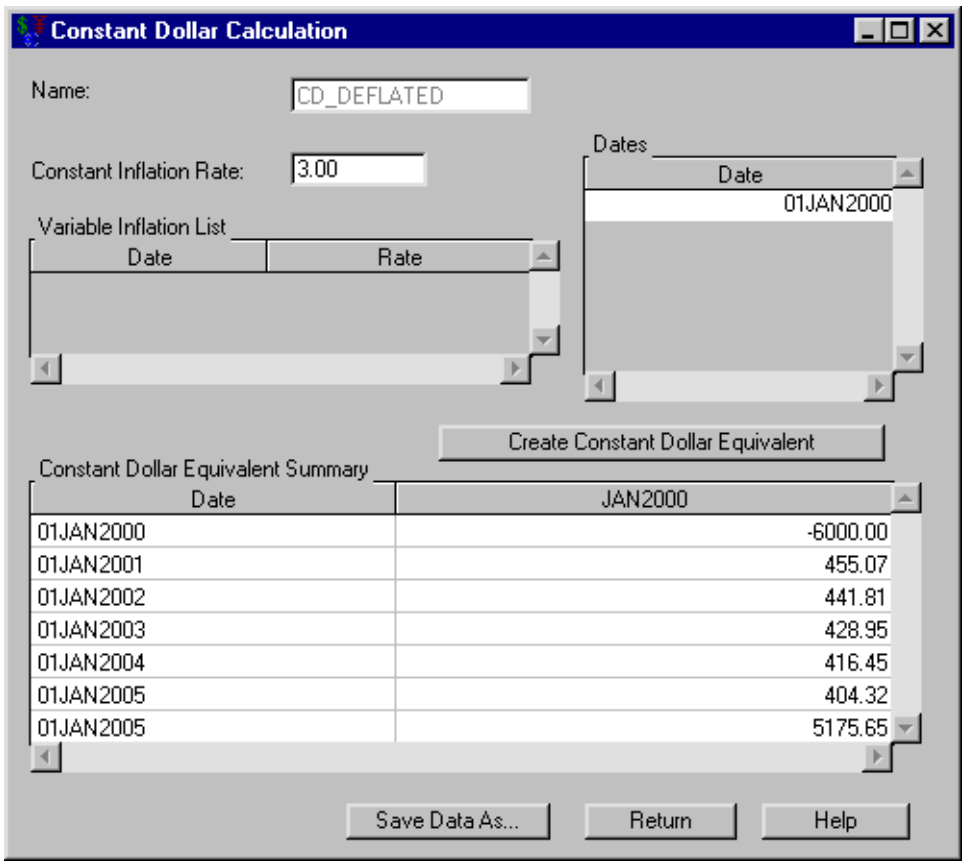
Figure 65.6 The CD before Deflation



To deflate the values, follow these steps:

1. Select CD_DEFLATED from the portfolio.
2. Select **Compute** → **Constant Dollars** from the menu. This opens the Constant Dollar Calculation dialog box.
3. Clear the **Variable Inflation List** area.
4. Enter 3 for the **Constant Inflation Rate**.
5. Click **Create Constant Dollar Equivalent** to generate a constant dollar equivalent summary (see Figure 65.7).

Figure 65.7 CD Values after Deflation



You can save the deflated cashflow to a SAS data set for use in an internal rate of return analysis or breakeven analysis.

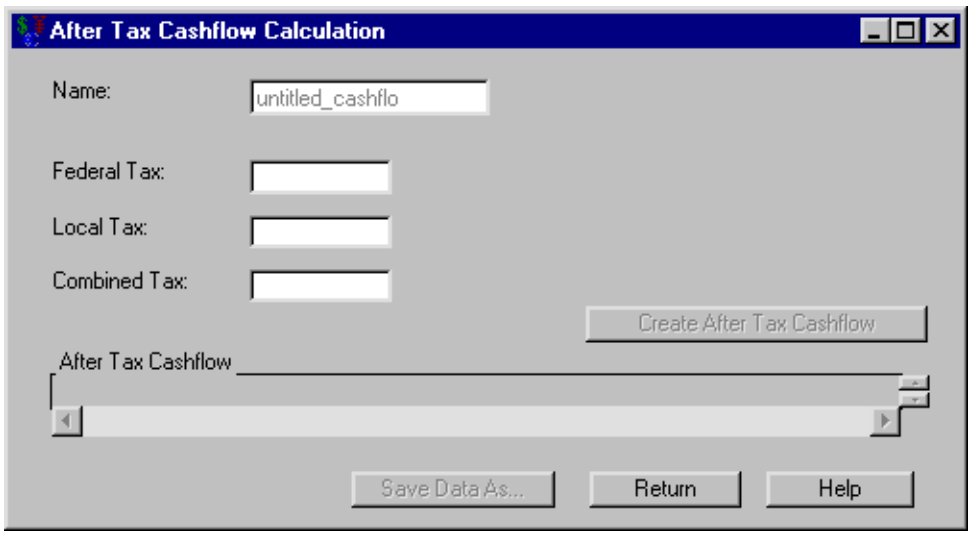
Click **Return** to return to the Investment Analysis dialog box.

Dialog Box Guide

After Tax Cashflow Calculation

Having selected a generic cashflow from the Investment Analysis dialog box, to perform an after tax calculation, select **Compute** → **After Tax** from the Investment Analysis dialog box's menu bar. This opens the After Tax Cashflow Calculation dialog box displayed in Figure 65.8.

Figure 65.8 After Tax Cashflow Calculation Dialog Box



The following items are displayed:

Name holds the name of the investment for which you are computing the after-tax cashflow.

Federal Tax holds the federal tax rate (a percentage between 0% and 100%).

Local Tax holds the local tax rate (a percentage between 0% and 100%).

Combined Tax holds the effective tax rate from federal and local income taxes.

Create After Tax Cashflow becomes available when **Combined Tax** is not empty. Clicking **Create After Tax Cashflow** then fills the **After Tax Cashflow** area.

After Tax Cashflow fills when you click **Create After Tax Cashflow**. It holds a list of date-amount pairs where the amount is the amount retained after taxes for that date.

Print becomes available when you fill the after-tax cashflow. Clicking it sends the contents of the after tax cashflow to the SAS session print device.

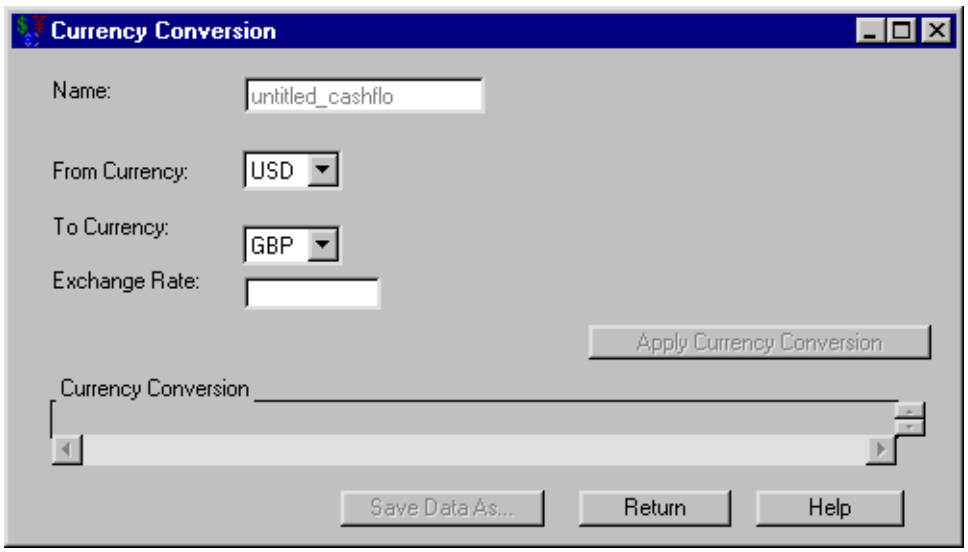
Save Data As becomes available when you fill the after tax cashflow. Clicking it opens the Save Output Dataset dialog box where you can save the resulting cashflow (or portions thereof) as a SAS Dataset.

Return returns you to the Investment Analysis dialog box.

Currency Conversion

Having selected a generic cashflow from the Investment Analysis dialog box, to perform a currency conversion, select **Compute** → **Currency Conversion** from the Investment Analysis dialog box's menu bar. This opens the Currency Conversion dialog box displayed in Figure 65.9.

Figure 65.9 Currency Conversion Dialog Box



The following items are displayed:

Name holds the name of the investment to which you are applying the currency conversion.

From Currency holds the name of the currency the cashflow currently represents.

To Currency holds the name of the currency to which you wish to convert.

Exchange Rate holds the rate of exchange between the **From Currency** and the **To Currency**.

Apply Currency Conversion becomes available when you fill **Exchange Rate**. Clicking **Apply Currency Conversion** fills the **Currency Conversion** area.

Currency Conversion fills when you click **Apply Currency Conversion**. The schedule contains a row for each cashflow item with the following information:

- **Date** is a SAS date within the cashflow.
- The **From Currency** value is the amount in the original currency at that date.
- The **To Currency** value is the amount in the new currency at that date.

Print becomes available when you fill the **Currency Conversion** area. Clicking it sends the contents of the conversion table to the SAS session print device.

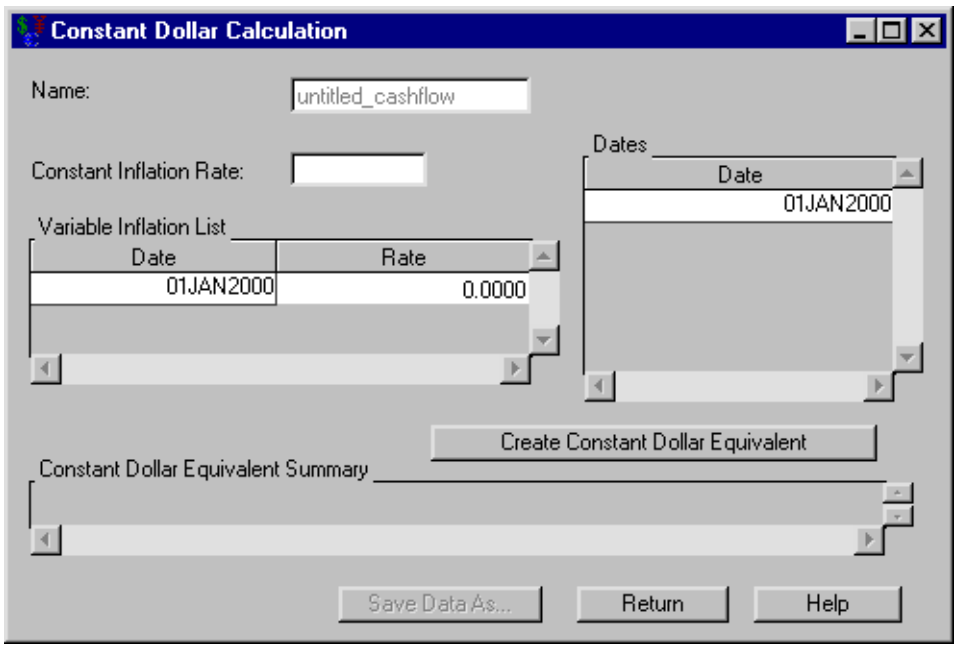
Save Data As becomes available when you fill the **Currency Conversion** area. Clicking it opens the Save Output Dataset dialog box where you can save the conversion table (or portions thereof) as a SAS Dataset.

Return returns you to the Investment Analysis dialog box.

Constant Dollar Calculation

Having selected a generic cashflow from the Investment Analysis dialog box, to perform a constant dollar calculation, select **Compute** → **Constant Dollars** from the Investment Analysis dialog box's menu bar. This opens the Constant Dollar Calculation dialog box displayed in Figure 65.10.

Figure 65.10 Constant Dollar Calculation Dialog Box



The following items are displayed:

Name holds the name of the investment for which you are computing the constant dollars value.

Constant Inflation Rate holds the constant inflation rate (a percentage between 0% and 120%). This value is used if the **Variable Inflation List** area is empty.

Variable Inflation List holds date-rate pairs that describe how inflation varies over time. Each date is a SAS date, and the rate is a percentage between 0% and 120%. Each date refers to when that inflation rate begins. Right-clicking within the **Variable Inflation** area reveals many helpful tools for managing date-rate pairs. If you assume a fixed inflation rate, just insert that rate in **Constant Rate**.

Dates holds the SAS date(s) at which you wish to compute the constant dollar equivalent. Right-clicking within the **Dates** area reveals many helpful tools for managing date lists.

Create Constant Dollar Equivalent becomes available when you enter inflation rate information. Clicking it fills the constant dollar equivalent summary with the computed constant dollar values.

Constant Dollar Equivalent Summary fills with a summary when you click **Create Constant Dollar Equivalent**. The first column lists the dates of the generic cashflow. The second column contains the constant dollar equivalent of the original generic cashflow item of that date.

Print becomes available when you fill the constant dollar equivalent summary. Clicking it sends the contents of the summary to the SAS session print device.

Save Data As becomes available when you fill the constant dollar equivalent summary. Clicking it opens the Save Output Dataset dialog box where you can save the summary (or portions thereof) as a SAS Dataset.

Return returns you to the Investment Analysis dialog box.

Chapter 66
Analyses

Contents

The Analyze Menu **3983**
Tasks **3984**
Performing Time Value Analysis 3984
Computing an Internal Rate of Return 3986
Performing a Benefit-Cost Ratio Analysis 3987
Computing a Uniform Periodic Equivalent 3988
Performing a Breakeven Analysis 3989
Dialog Box Guide **3991**
Time Value Analysis 3991
Uniform Periodic Equivalent 3992
Internal Rate of Return 3993
Benefit-Cost Ratio Analysis 3994
Breakeven Analysis 3995
Breakeven Graph 3996

The Analyze Menu

Figure 66.1 shows the **Analyze** menu.

Figure 66.1 Analyze Menu



The **Analyze** menu offers the following options for use on applicable investments.

Time Value opens the Time Value Analysis dialog box. Time value analysis involves moving money through time across a defined minimum attractive rate of return (MARR) so that you can compare value at a consistent date. The MARR can be constant or variable over time.

Periodic Equivalent opens the Uniform Periodic Equivalent dialog box. Uniform periodic equivalent analysis determines the payment needed to convert a cashflow to uniform amounts over time, given a periodicity, a number of periods, and a MARR. This option helps when making comparisons where one alternative is uniform (such as renting) and another is not (such as buying).

Internal Rate of Return opens the Internal Rate of Return dialog box. The internal rate of return of a cashflow is the interest rate that makes the time value equal to 0. This calculation assumes uniform periodicity of the cashflow. It is particularly applicable where the choice of MARR would be difficult.

Benefit-Cost Ratio opens the Benefit-Cost Ratio Analysis dialog box. The benefit-cost ratio divides the time value of the benefits by the time value of the costs. For example, governments often use this analysis when deciding whether to commit to a public works project.

Breakeven Analysis opens the Breakeven Analysis dialog box. Breakeven analysis computes time values at various MARRs to compare, which can be advantageous when it is difficult to determine a MARR. This analysis can help you determine how the cashflow's profitability varies with your choice of MARR. A graph displaying the relationships between time value and MARR is also available.

Tasks

Performing Time Value Analysis

Suppose a rock quarry needs equipment to use the next five years. It has two alternatives:

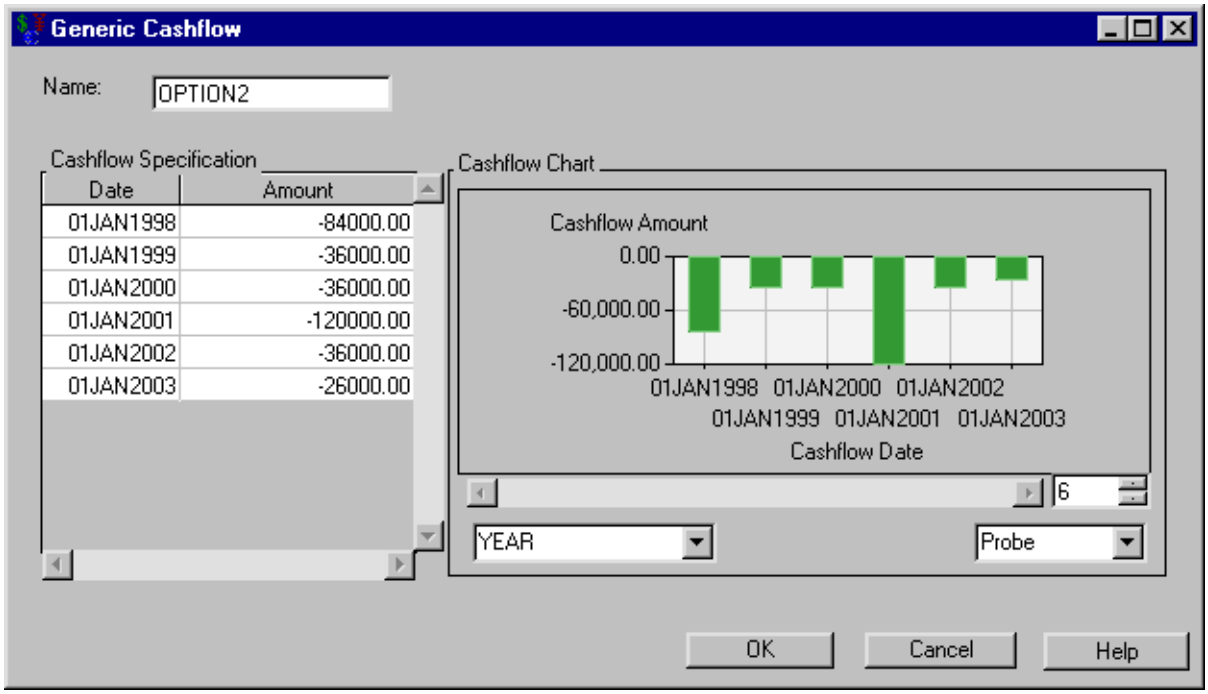
- a box loader and conveyer system that has a one-time cost of $264,000
- a two-shovel loader, which costs $84,000 but has a yearly operating cost of $36,000. This loader has a service life of three years, which necessitates the purchase of a new loader for the final two years of the rock quarry project. Assume the second loader also costs $84,000 and its salvage value after its two-year service is $10,000. A SAS data set that describes this is available at SASHELP.ROCKPIT

You expect a 13% MARR. Which is the better alternative?

To create the cashflows, follow these steps:

1. Create a cashflow with the single amount –264,000. Date the amount 01JAN1998 to be consistent with the SAS data set you load.
2. Load SASHELP.ROCKPIT into a second cashflow, as displayed in Figure 66.2.

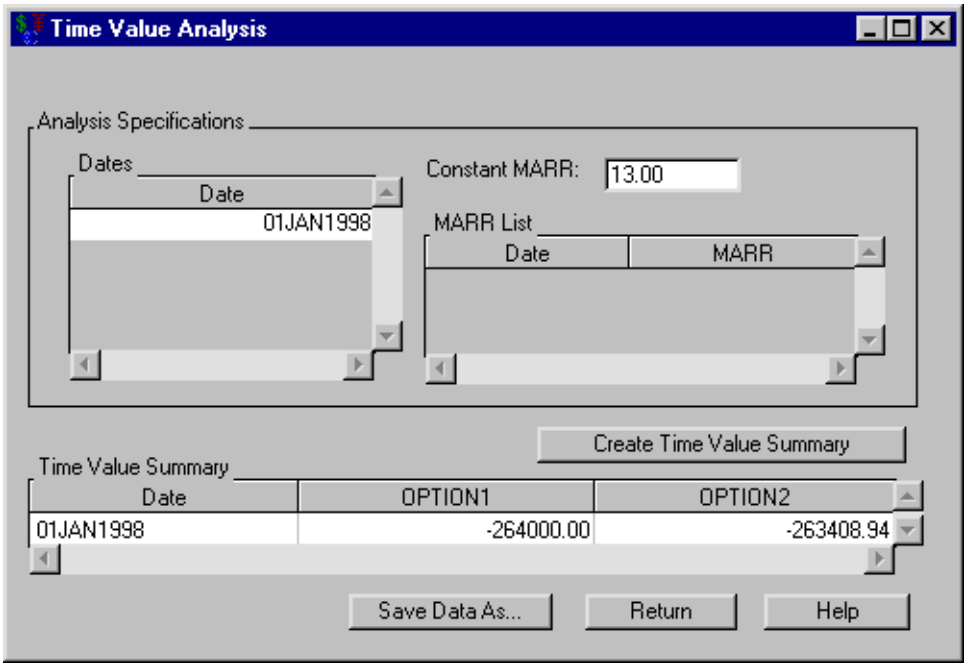
Figure 66.2 The contents of SASHELP.ROCKPIT



To compute the time values of these investments, follow these steps:

1. Select both cashflows.
2. Select **Analyze** → **Time Value**. This opens the Time Value Analysis dialog box.
3. Enter the date 01JAN1998 into the **Dates** area.
4. Enter 13 for the **Constant MARR**.
5. Click **Create Time Value Summary**.

Figure 66.3 Performing the Time Value Analysis



As shown in Figure 66.3, option 1 has a time value of –$264,000.00 naturally on 01JAN1998. However, option 2 has a time value of –$263,408.94, which is slightly less expensive.

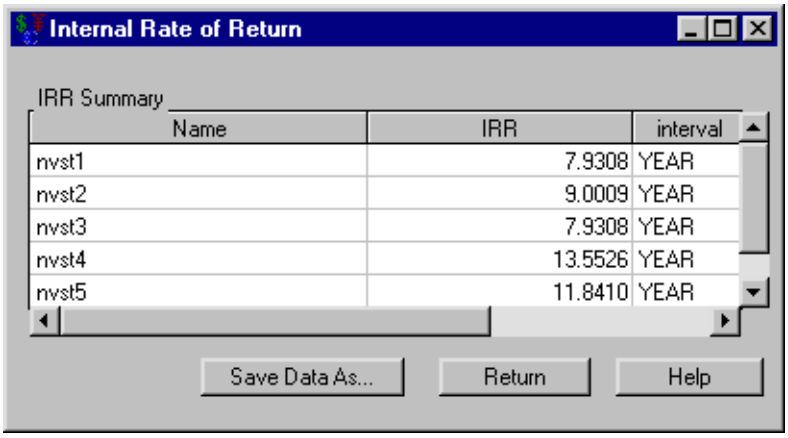
Computing an Internal Rate of Return

You are choosing between five investments. A portfolio containing these investments is available at SASHELP.INVSAMP.NVST. Which investments are acceptable if you expect a MARR of 9%?

Open the portfolio SASHELP.INVSAMP.NVST and compare the investments. Note that Internal Rate of Return computations assume regular periodicity of the cashflow. To compute the internal rates of return, follow these steps:

1. Select all five investments.
2. Select **Analyze** → **Internal Rate of Return**.

Figure 66.4 Computing an Internal Rate of Return



The results displayed in Figure 66.4 indicate that the internal rates of return for investments 2, 4, and 5 are greater than 9%. Hence, each of these is acceptable.

Performing a Benefit-Cost Ratio Analysis

Suppose a municipality has excess funds to invest. It is choosing between the same investments described in the previous example. Government agencies often compute benefit-cost ratios to decide which investment to pursue. Which is best in this case?

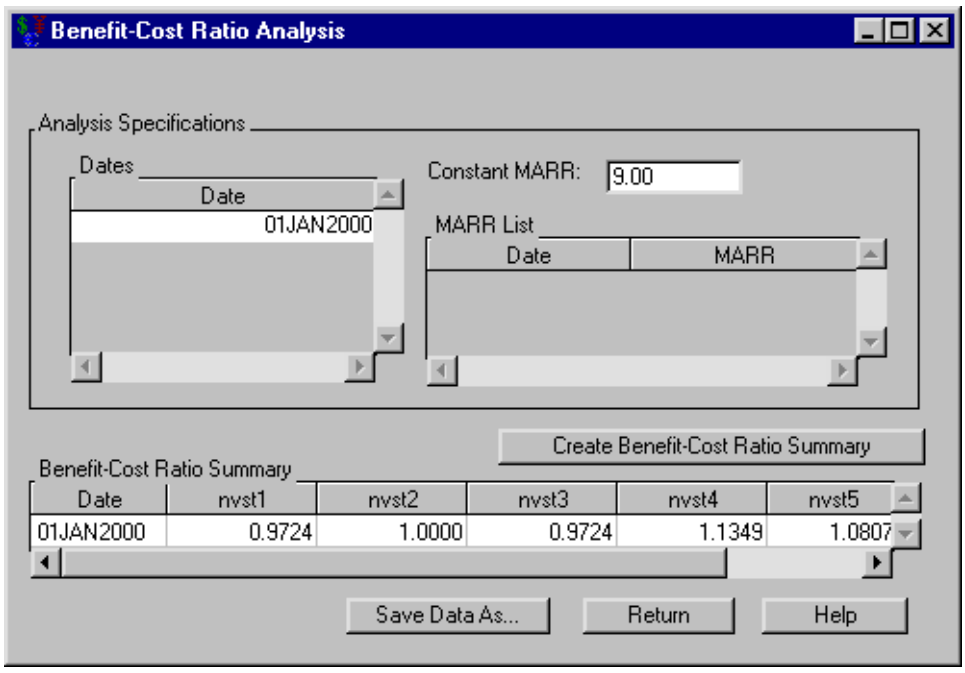
Open the portfolio SASHELP.INVSAMP.NVST and compare the investments.

To compute the benefit-cost ratios, follow these steps:

1. Select all five investments.
2. Select **Analyze** → **Benefit-Cost Ratio**.
3. Enter 01JAN1996 for the **Date**.
4. Enter 9 for **Constant MARR**.
5. Click **Create Benefit-Cost Ratio Summary** to fill the **Benefit-Cost Ratio Summary** area.

The results displayed in Figure 66.5 indicate that investments 2, 4, and 5 have ratios greater than 1. Therefore, each is profitable with a MARR of 9%.

Figure 66.5 Performing a Benefit-Cost Ratio Analysis



Computing a Uniform Periodic Equivalent

Suppose you need a warehouse for ten years. You have two options:

- pay rent for ten years at $23,000 per year
- build a two-stage facility that you will maintain and which you intend to sell at the end of those ten years

Data sets describing these scenarios are available in the portfolio SASHELP.INVSAMP.BUYRENT. Which option is more financially sound if you desire a 12% MARR?

Open the portfolio SASHELP.INVSAMP.BUYRENT and compare the options.

To perform the periodic equivalent, follow these steps:

1. Load the portfolio SASHELP.INVSAMP.BUYRENT.
2. Select both cashflows.
3. Select **Analyze** → **Periodic Equivalent**.
   This opens the Uniform Periodic Equivalent dialog box.
4. Enter 01JAN1996 for the **Start Date**.
5. Enter 10 for the **Number of Periods**.
6. Select YEAR for the **Interval**.

7. Enter 12 for the **Constant MARR**.
8. Click **Create Time Value Summary**.

Figure 66.6 Computing a Uniform Periodic Equivalent

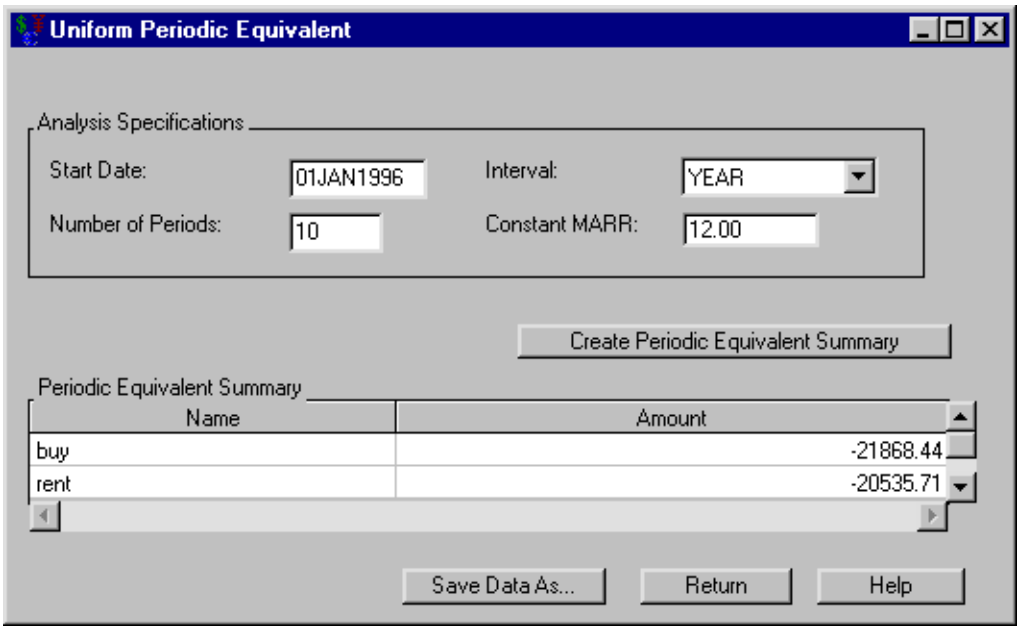


Figure 66.6 indicates that renting costs about $1,300 less each year. Hence, renting is more financially sound. Notice the periodic equivalent for renting is not $23,000. This is because the $23,000 per year does not account for the MARR.

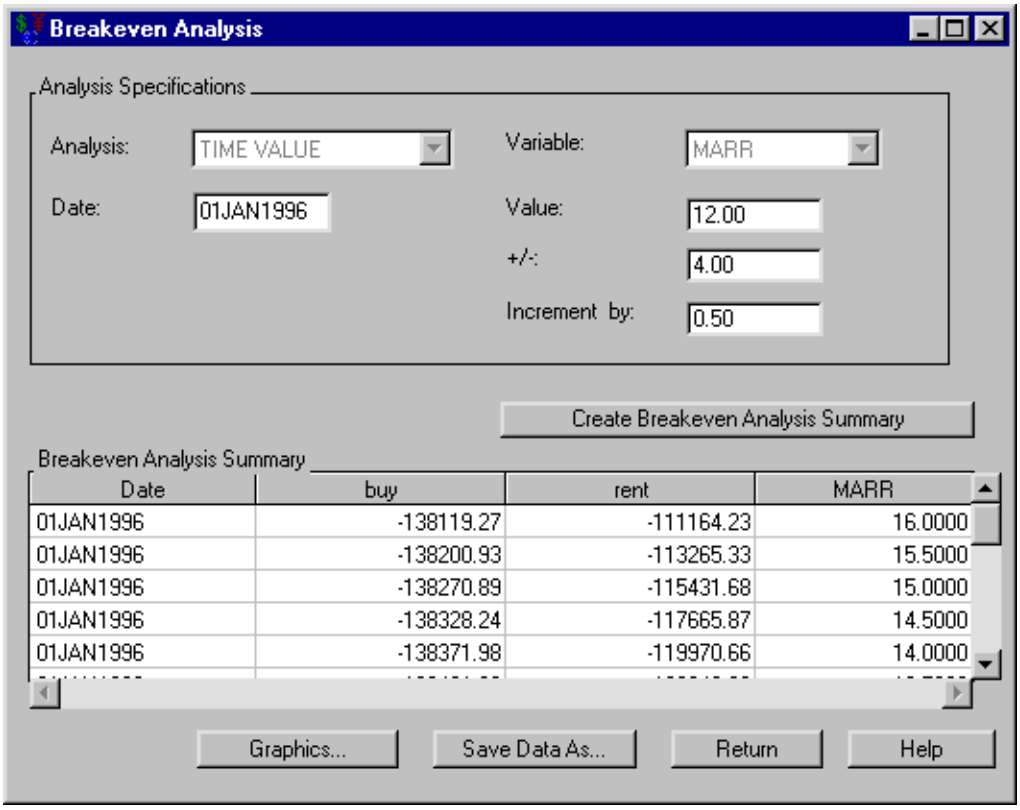
Performing a Breakeven Analysis

In the previous example you computed the uniform periodic equivalent for a rent-buy scenario. Now let's perform a breakeven analysis to see how the MARR affects the time values.

To perform the breakeven analysis, follow these steps:

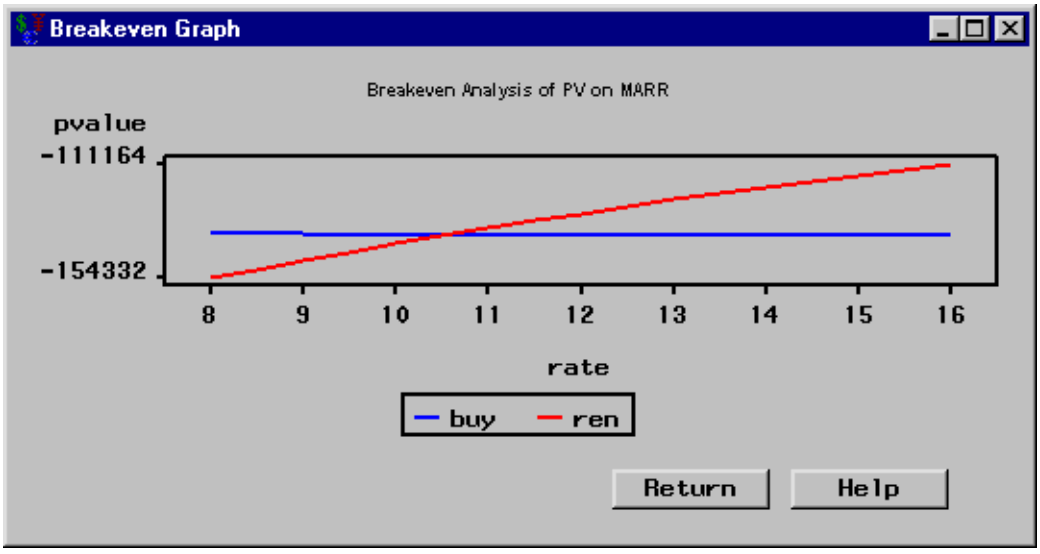
1. Select both options.
2. Select **Analyze** → **Breakeven Analysis**.
3. Enter 01JAN1996 for the **Date**.
4. Enter 12.0 for **Value**.
5. Enter 4.0 for **(+/-)**.
6. Enter 0.5 for **Increment by**.
7. Click **Create Breakeven Analysis Summary** to fill the **Breakeven Analysis Summary** area as displayed in Figure 66.7.

Figure 66.7 Performing a Breakeven Analysis



Click **Graphics** to view a plot displaying the relationship between time value and MARR.

Figure 66.8 Viewing a Breakeven Graph



As shown in Figure 66.8, renting is better if you want a MARR of 12%. However, if your MARR should drop to 10.5%, buying would be better.

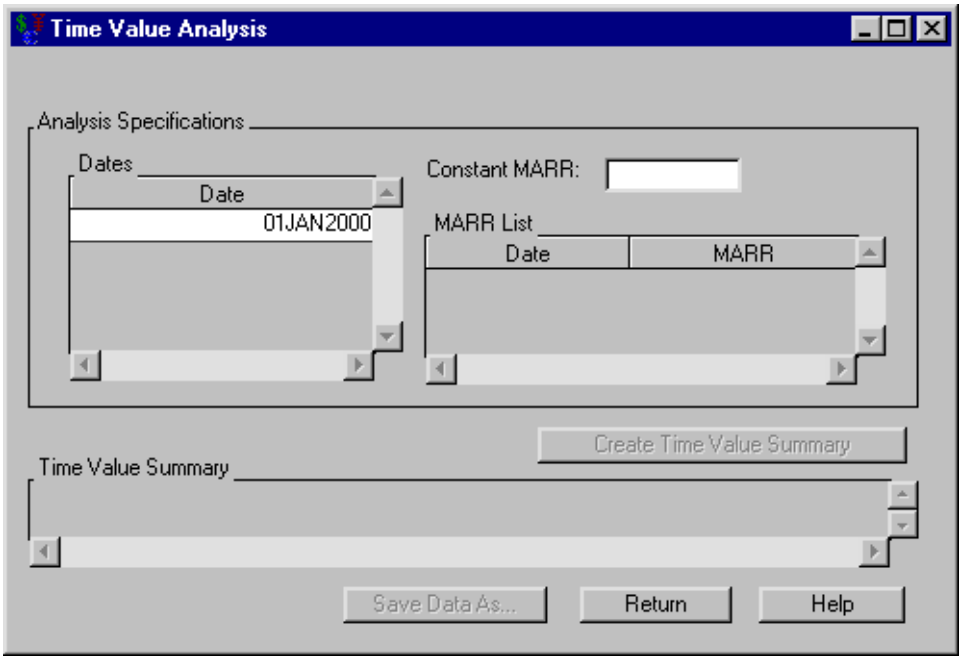
With a single investment, knowing where the graph has a time value of 0 tells the MARR when a venture switches from being profitable to being a loss. With multiple investments, knowing where the graphs for the various investments cross each other tells at what MARR a particular investment becomes more profitable than another.

Dialog Box Guide

Time Value Analysis

Having selected a generic cashflow from the Investment Analysis dialog box, to perform an time value analysis, select **Analyze** → **Time Value** from the Investment Analysis dialog box's menu bar. This opens the Time Value Analysis dialog box displayed in Figure 66.9.

Figure 66.9 Time Value Analysis Dialog Box



The following items are displayed:

Analysis Specifications

Dates holds the list of dates as of which to perform the time value analysis. Right-clicking within the **Dates** area reveals many helpful tools for managing date lists.

Constant MARR holds the desired MARR for the time value analysis. This value is used if the **MARR List** area is empty.

MARR List holds date-rate pairs that express your desired MARR as it changes over time. Each date refers to when that expected MARR begins. Right-clicking within the **MARR List** area reveals many helpful tools for managing date-rate pairs.

Create Time Value Summary becomes available when you adequately specify the analysis within the **Analysis Specifications** area. Clicking **Create Time Value Summary** then fills the **Time Value Summary** area.

Time Value Summary fills when you click **Create Time Value Summary**. The table contains a row for each date in the **Dates** area. The remainder of each row holds the time values at that date, one value for each investment selected.

Print becomes available when you fill the time value summary. Clicking it sends the contents of the summary to the SAS session print device.

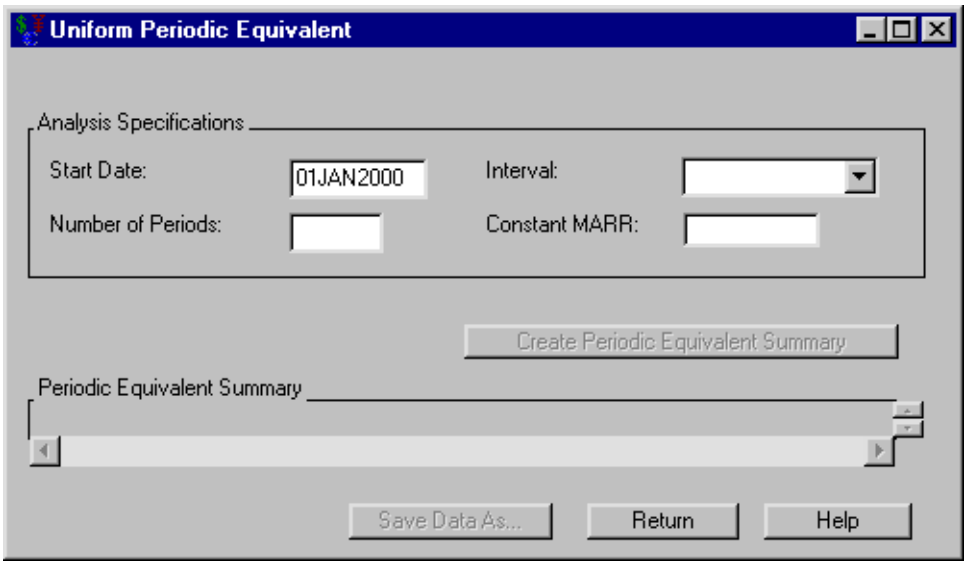
Save Data As becomes available when you fill the time value summary. Clicking it opens the Save Output Dataset dialog box where you can save the summary (or portions thereof) as a SAS Dataset.

Return takes you back to the Investment Analysis dialog box.

Uniform Periodic Equivalent

Having selected a generic cashflow from the Investment Analysis dialog box, to perform a uniform periodic equivalent, select **Analyze** → **Periodic Equivalent** from the Investment Analysis dialog box's menu bar. This opens the Uniform Periodic Equivalent dialog box displayed in Figure 66.10.

Figure 66.10 Uniform Periodic Equivalent Dialog Box



The following items are displayed:

Analysis Specifications

Start Date holds the date the uniform periodic equivalents begin.

Number of Periods holds the number of uniform periodic equivalents.

Interval holds how often the uniform periodic equivalents occur.

Constant MARR holds the Minimum Attractive Rate of Return.

Create Periodic Equivalent Summary becomes available when you adequately fill the **Analysis Specification** area. Clicking **Create Periodic Equivalent Summary** then fills the periodic equivalent summary.

Periodic Equivalent Summary fills with two columns when you click **Create Periodic Equivalent Summary**. The first column lists the investments selected. The second column lists the computed periodic equivalent amount.

Print becomes available when you fill the periodic equivalent summary. Clicking it sends the contents of the summary to the SAS session print device.

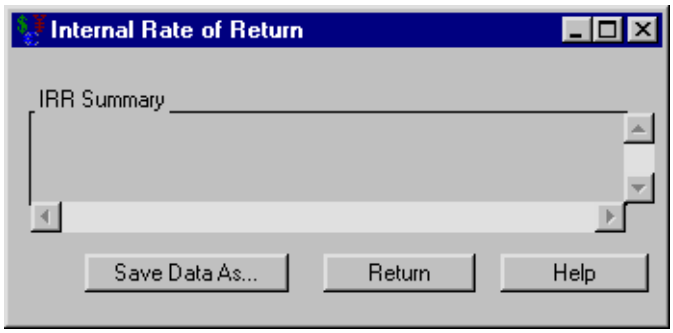
Save Data As becomes available when you generate the periodic equivalent summary. Clicking it opens the Save Output Dataset dialog box where you can save the summary (or portions thereof) as a SAS Dataset.

Return takes you back to the Investment Analysis dialog box.

Internal Rate of Return

Having selected a generic cashflow from the Investment Analysis dialog box, to perform an internal rate of return calculation, select **Analyze** → **Internal Rate of Return** from the Investment Analysis dialog box's menu bar. This opens the Internal Rate of Return dialog box displayed in Figure 66.11.

Figure 66.11 Internal Rate of Return Dialog Box



The following items are displayed:

IRR Summary contains a row for each deposit. Each row holds:

Name holds the name of the investment.

IRR holds the internal rate of return for that investment.

interval holds the interest rate interval for that **IRR**.

Print becomes available when you fill the IRR summary. Clicking it sends the contents of the summary to the SAS session print device.

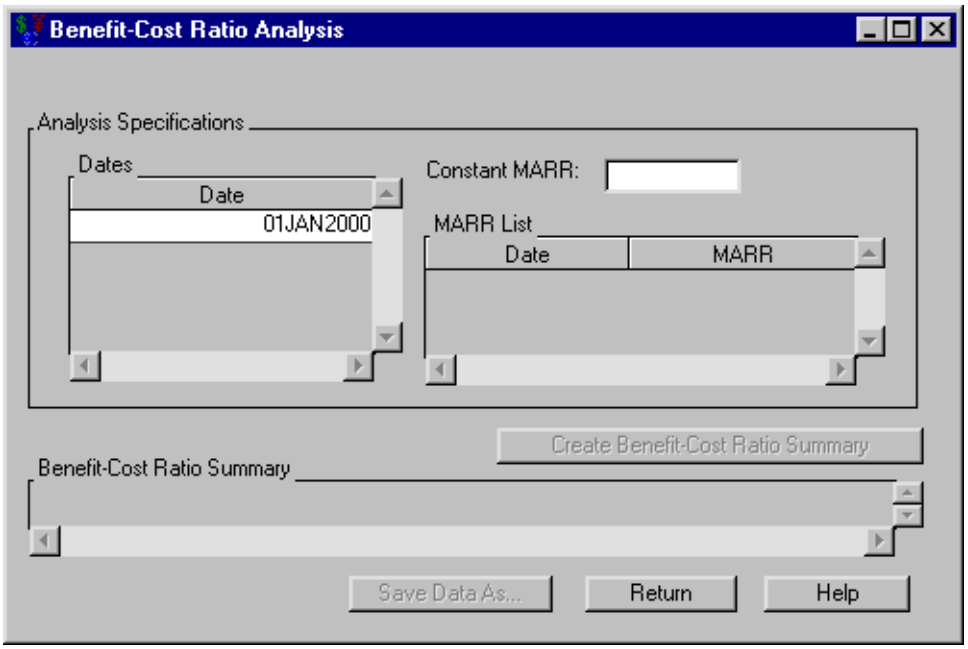
Save Data As opens the Save Output Dataset dialog box where you can save the IRR summary (or portions thereof) as a SAS data set.

Return takes you back to the Investment Analysis dialog box.

Benefit-Cost Ratio Analysis

Having selected a generic cashflow from the Investment Analysis dialog box, to compute a benefit-cost ratio, select **Analyze** → **Benefit-Cost Ratio** from the Investment Analysis dialog box's menu bar. This opens the Benefit-Cost Ratio Analysis dialog box displayed in Figure 66.12.

Figure 66.12 Benefit-Cost Ratio Analysis Dialog Box



The following items are displayed:

Analysis Specifications

Dates holds the dates as of which to compute the Benefit-Cost ratios.

Constant MARR holds the desired MARR. This value is used if the **MARR List** area is empty.

MARR List holds date-rate pairs that express your desired MARR as it changes over time. Each date refers to when that expected MARR begins. Right-clicking within the **MARR List** area reveals many helpful tools for managing date-rate pairs.

Create Benefit-Cost Ratio Summary becomes available when you adequately specify the analysis. Clicking **Create Benefit-Cost Ratio Summary** fills the benefit-cost ratio summary.

Benefit-Cost Ratio Summary fills when you click **Exchange the Rates**. The area contains a row for each date in the **Dates** area. The remainder of each row holds the benefit-cost ratios at that date, one value for each investment selected.

Print becomes available when you fill the benefit-cost ratio summary. Clicking it sends the contents of the summary to the SAS session print device.

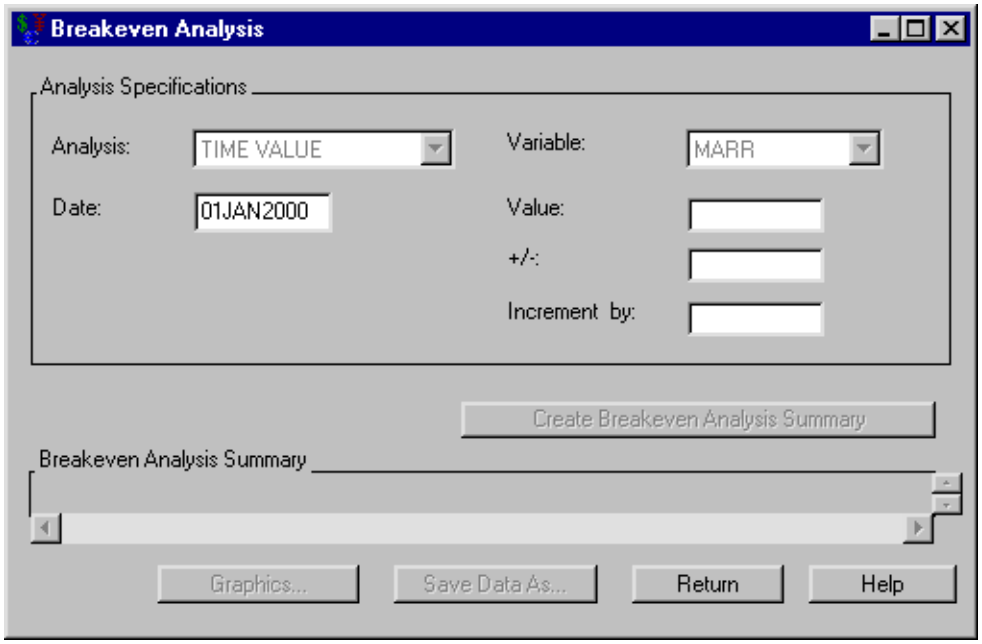
Save Data As becomes available when you generate the benefit-cost ratio summary. Clicking it opens the Save Output Dataset dialog box where you can save the summary (or portions thereof) as a SAS Dataset.

Return takes you back to the Investment Analysis dialog box.

Breakeven Analysis

Having selected a generic cashflow from the Investment Analysis dialog box, to perform a breakeven analysis, select **Analyze** → **Breakeven Analysis** from the Investment Analysis dialog box's menu bar. This opens the Breakeven Analysis dialog box displayed in Figure 66.13.

Figure 66.13 Breakeven Analysis Dialog Box



The following items are displayed:

Analysis Specification

Analysis holds the analysis type. Only Time Value is currently available.

Date holds the date for which you perform this analysis.

Variable holds the variable upon which the breakeven analysis will vary. Only MARR is currently available.

Value holds the desired rate upon which to center the analysis.

+/- holds the maximum deviation from the **Value** to consider.

Increment by holds the increment by which the analysis is calculated.

Create Breakeven Analysis Summary becomes available when you adequately specify the analysis. Clicking **Create Breakeven Analysis Summary** then fills the **Breakeven Analysis Summary** area.

Breakeven Analysis Summary fills when you click **Create Breakeven Analysis Summary**. The schedule contains a row for each MARR and date.

Graphics becomes available when you fill the **Breakeven Analysis Summary** area. Clicking it opens the Breakeven Graph graph representing the time value versus MARR.

Print becomes available when you fill the breakeven analysis summary. Clicking it sends the contents of the summary to the SAS session print device.

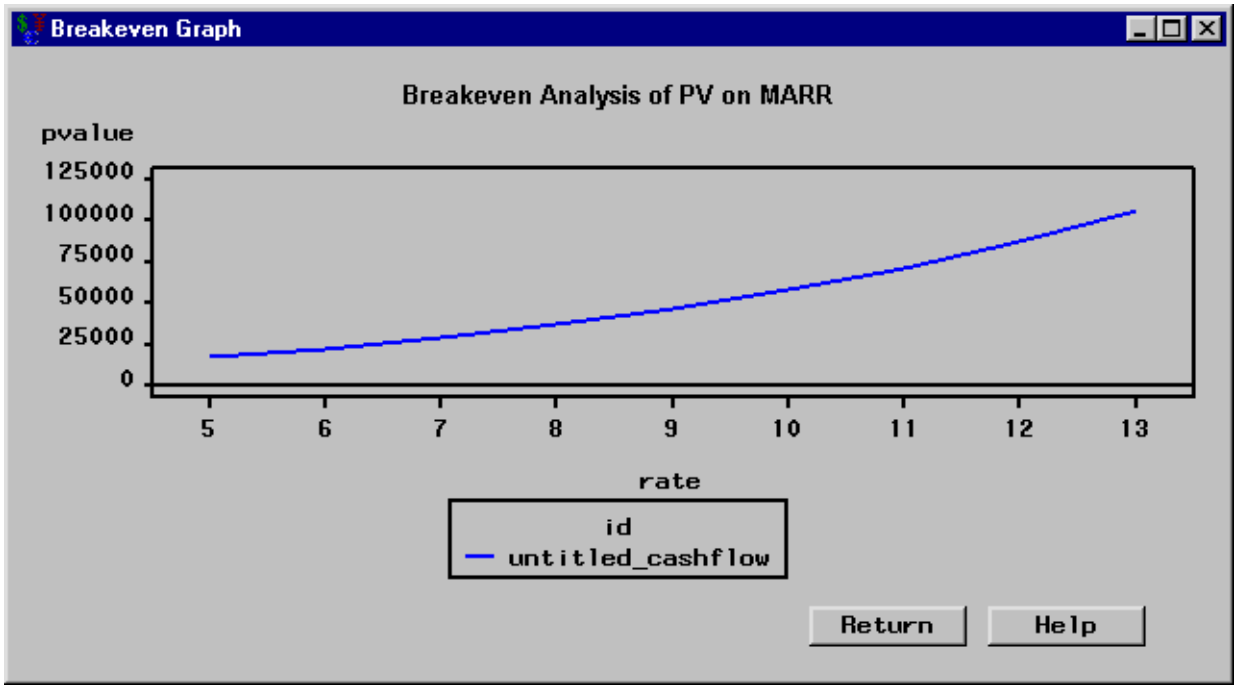
Save Data As becomes available when you generate the breakeven analysis summary. Clicking it opens the Save Output Dataset dialog box where you can save the summary (or portions thereof) as a SAS Dataset.

Return takes you back to the Investment Analysis dialog box.

Breakeven Graph

Suppose you perform a breakeven analysis in the Breakeven Analysis dialog box. Once you create the breakeven analysis summary, you can click the **Graphics** button to open the Breakeven Graph dialog box displayed in Figure 66.14.

Figure 66.14 Breakeven Graph Dialog Box



The following item is displayed:

Return takes you back to the Breakeven Analysis dialog box.

Chapter 67
Details

Contents

Investments and Data Sets **3997**
Saving Output to SAS Data Sets 3997
Loading a SAS Data Set into a List 3999
Saving Data from a List to a SAS Data Set 4000
Right Mouse Button Options **4000**
Depreciation Methods **4001**
Straight Line (SL) 4001
Sum-of-Years Digits 4001
Declining Balance (DB) 4003
Rate Information **4004**
The Tools Menu 4004
Minimum Attractive Rate of Return (MARR) **4005**
Income Tax Specification **4006**
Inflation Specification **4007**
References **4007**

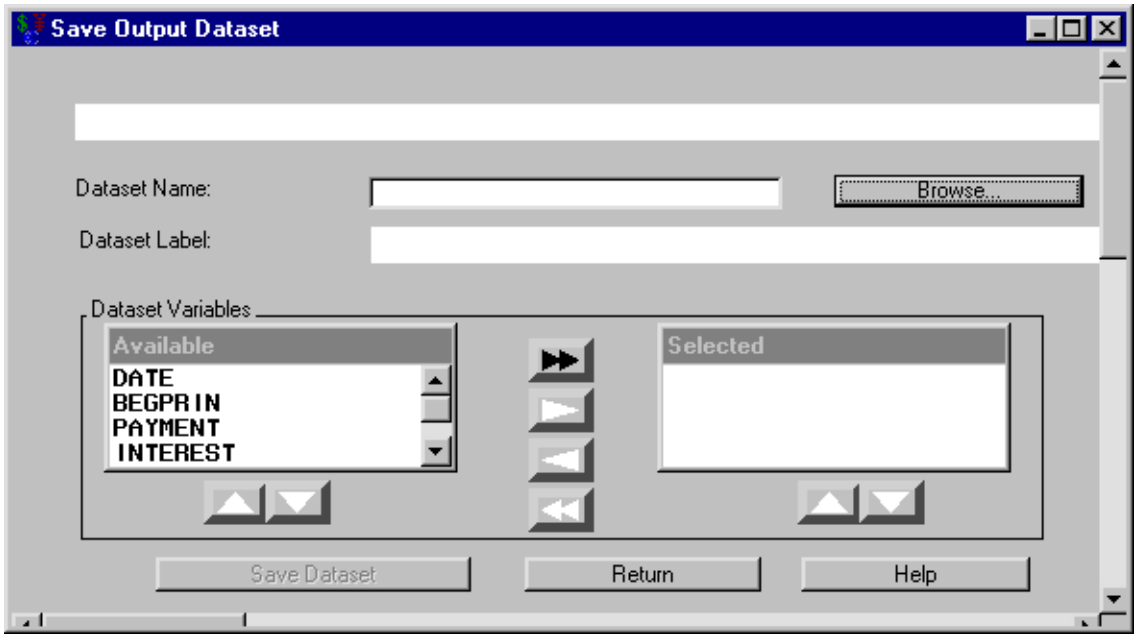
Investments and Data Sets

Investment Analysis provides tools to assist you in moving data between SAS data sets and lists you can use within Investment Analysis.

Saving Output to SAS Data Sets

Many investment specifications have a button that reads **Save Data As**. Clicking that button opens the Save Output Dataset dialog box (see Figure 67.1). This dialog box enables you to save all or part of the area generated by the specification.

Figure 67.1 Saving to a Dataset



The following items are displayed:

Dataset Name holds the SAS data set name to which you want to save.

Browse opens the standard SAS **Open** dialog box, which enables you to select an existing SAS data set to overwrite.

Dataset Label holds the SAS data set's label.

Dataset Variables organizes variables. The variables listed in the **Selected** area will be included in the SAS data set.

- You can select variables one at a time, by clicking the single right-arrow after each selection to move it to the **Selected** area.
- If the desired SAS data set has many variables you want to save, it may be simpler to follow these steps:

  1. Click the double right-arrow to select all available variables.
  2. Remove any unwanted variable by selecting it from the **Selected** area and clicking the single left-arrow.

- The double left-arrow removes all selected variables from the proposed SAS data set.
- The up and down arrows below the **Available** and **Selected** boxes enable you to scroll up and down the list of variables in their respective boxes.

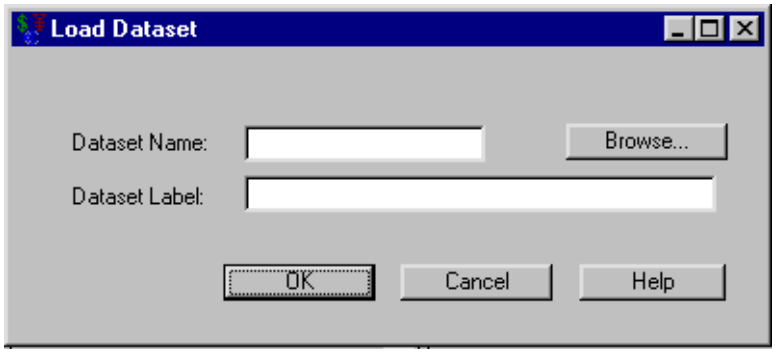
Save Dataset attempts to save the SAS data set. If the SAS data set name exists, you are asked if you want to replace the existing SAS data set, append to the existing SAS data set, or cancel the current save attempt. You then return to this dialog box ready to create another SAS data set to save.

Return takes you back to the specification dialog box.

Loading a SAS Data Set into a List

Right-click in the area that you want to load the list and release on **Load**. This opens the Load Dataset dialog box (see Figure 67.2).

Figure 67.2 Load Dataset Dialog Box



The following items are displayed:

Dataset Name holds the name of the SAS data set that you want to load.

Browse opens the standard SAS **Open** dialog box, which aids in finding a SAS data set to load. If there is a **Date** variable in the SAS data set, Investment Analysis loads it into the list. If there is no **Date** variable, it loads the first available time-formatted variable. If an amount or rate variable is needed, Investment Analysis searches the SAS data set for a **Amount** or **Rate** variable to use. Otherwise it takes the first numeric variable that is not used by the **Date** variable.

Dataset Label holds a SAS data set label.

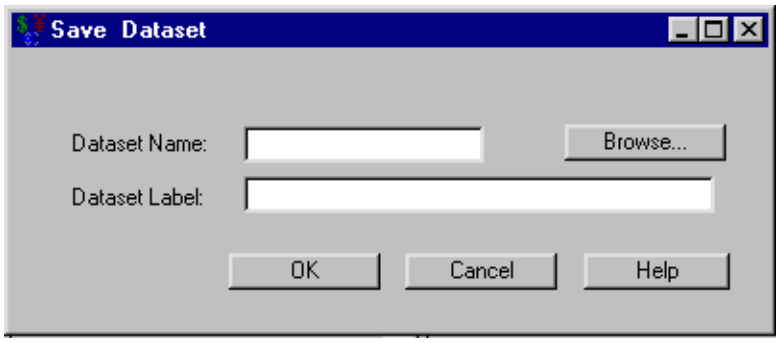
OK attempts to load the SAS data set specified in **Dataset Name**. If the specified SAS data set exists, clicking **OK** returns you to the calling dialog box with the selected SAS data set filling the list. If the specified SAS data set does not exist and you click **OK**, you receive an error message and no SAS data set is loaded.

Cancel returns you to the calling dialog box without loading a SAS data set.

Saving Data from a List to a SAS Data Set

Right-click in the area you want to hold the list, and release on **Save**. This opens the Save Dataset dialog box.

Figure 67.3 Save Dataset Dialog Box



The following items are displayed:

Dataset Name holds the SAS data set name to which you want to save.

Browse opens the standard SAS **Save As** dialog box, which enables you to find an existing SAS data set to overwrite.

Dataset Label holds a user-defined description to be saved as the label of the SAS data set.

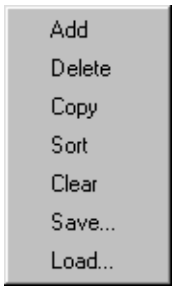
OK saves the current data to the SAS data set specified in **Dataset Name**. If the specified SAS data set does not already exist, clicking **OK** saves the SAS data set and returns you to the calling dialog box. If the specified SAS data set does already exist, clicking **OK** warns you and enables you to replace the old SAS data set with the new SAS data set or cancel the save attempt.

Cancel aborts the save process. Clicking **Cancel** returns you to the calling dialog box without attempting to save.

Right Mouse Button Options

A pop-up menu often appears when you right-click within table editors. The menus offer tools to aid in the management of the table's entries. Most table editors provide the following options.

Figure 67.4 Right-Clicking Options



Add creates a blank row.

Delete removes any currently selected row.

Copy duplicates the currently selected row.

Sort arranges the rows in chronological order according to the date variable.

Clear empties the table of all rows.

Save opens the Save Dataset dialog box where you can save the all rows to a SAS Dataset for later use.

Load opens the Load Dataset dialog box where you select a SAS Dataset to fill the rows.

If you want to perform one of these actions on a collection of rows, you must select a collection of rows before right-clicking. To select an adjacent list of rows, do the following: click the first pair, hold down SHIFT, and click the final pair. After the list of rows is selected, you may release the SHIFT key.

Depreciation Methods

Suppose an asset's price is $20,000 and it has a salvage value of $5,000 in five years. The following sections describe various methods to quantify the depreciation.

Straight Line (SL)

This method assumes a constant depreciation value per year.

Assuming that the price of a depreciating asset is P and its salvage value after N years is S, the annual depreciation is:

$$\frac{P-S}{N}$$

For our example, the annual depreciation would be

$$\frac{\$20,000-\$5,000}{5}=\$3,000$$

Sum-of-Years Digits

An asset often loses more of its value early in its lifetime. A method that exhibits this dynamic is desirable.

Assume an asset depreciates from price P to salvage value N in S years. First compute the sum-of-years as $T=1+2+\cdots+N$. The depreciation for the years after the asset's purchase is:

Table 67.1 Sum-of-Years General Example

Year Number	Annual Depreciation
first	$\frac{N}{T}(P-S)$
second	$\frac{N-1}{T}(P-S)$
third	$\frac{N-2}{T}(P-S)$
⋮	⋮
final	$\frac{1}{T}(P-S)$

For the *i*th year of the asset's use, the annual depreciation is:

$$\frac{N+1-i}{T}(P-S)$$

For our example, $N = 5$ and the sum of years is $T = 1+2+3+4+5 = 15$. The depreciation during the first year is

$$(\$20,000 - \$5,000)\frac{5}{15} = \$5,000$$

Table 67.2 describes how Declining Balance would depreciate the asset.

Table 67.2 Sum-of-Years Example

Year	Depreciation	Year-End Value
1	$(\$20,000 - \$5,000)\frac{5}{15} = \$5,000$	$\$15,000.00$
2	$(\$20,000 - \$5,000)\frac{4}{15} = \$4,000$	$\$11,000.00$
3	$(\$20,000 - \$5,000)\frac{3}{15} = \$3,000$	$\$8,000.00$
4	$(\$20,000 - \$5,000)\frac{2}{15} = \$2,000$	$\$6,000.00$
5	$(\$20,000 - \$5,000)\frac{1}{15} = \$1,000$	$\$5,000.00$

As expected, the value after N years is S.

$$\begin{aligned} S &= P - (5 \text{ years' depreciation}) \\ &= P - \left(\frac{5}{10}(P-S) + \frac{4}{10}(P-S) + \frac{3}{10}(P-S) + \frac{2}{10}(P-S) + \frac{1}{10}(P-S)\right) \\ &= P - (P-S) \end{aligned}$$

Declining Balance (DB)

Recall that the straight line method assumes a constant depreciation value. Conversely, the declining balance method assumes a constant depreciation rate per year. And like the sum-of-years method, more depreciation tends to occur earlier in the asset's life.

Assume the price of a depreciating asset is P and its salvage value after N years is N. You could assume the asset depreciates by a factor of $\frac{1}{N}$ (or a rate of $\frac{100}{N}\%$). This method is known as single declining balance. The annual depreciation is:

$$\frac{1}{N}(\text{previous year's value})$$

So for our example, the depreciation during the first year is $\frac{\$20{,}000}{5} = \$4,000$. Table 67.3 describes how declining balance would depreciate the asset.

Table 67.3 Declining Balance Example

Year	Depreciation	Year-End Value
1	$\frac{\$20{,}000.00}{5} = \$4,000.00$	$\$16,000.00$
2	$\frac{\$16{,}000.00}{5} = \$3,200.00$	$\$12,800.00$
3	$\frac{\$12{,}800.00}{5} = \$2,560.00$	$\$10,240.00$
4	$\frac{\$10{,}240.00}{5} = \$2,048.00$	$\$8,192.00$
5	$\frac{\$12{,}800.00}{5} = \$1,638.40$	$\$6,553.60$

DB Factor

You could also accelerate the depreciation by increasing the factor (and hence the rate) at which depreciation occurs. Other commonly accepted depreciation rates are $\frac{200}{N}\%$ (called double declining balance as the depreciation factor becomes $\frac{2}{N}$) and $\frac{150}{N}\%$. Investment Analysis enables you to choose between these three types for declining balance: 2 (with $\frac{200}{N}\%$ depreciation), 1.5 (with $\frac{150}{N}\%$), and 1 (with $\frac{100}{N}\%$).

Declining Balance and the Salvage Value

The declining balance method assumes that depreciation is faster earlier in an asset's life; this is what you wanted. But notice the final value is greater than the salvage value. Even if the salvage value were greater than \$6,553.60, the final year-end value would not change. The salvage value never enters the calculation, so there is no way for the salvage value to force the depreciation to assume its value. Newnan and Lavelle (1998) describe two ways to adapt the declining balance method to assume the salvage value at the final time. One way is as follows:

Suppose you call the depreciated value after i years $V(i)$. This sets $V(0) = P$ and $V(N) = S$.

- If $V(N) > S$ according to the usual calculation for $V(N)$, redefine $V(N)$ to equal S.
- If $V(i) < S$ according to the usual calculation for $V(i)$ for some i (and hence for all subsequent $V(i)$ values), you can redefine all such $V(i)$ to equal S.

This alteration to declining balance forces the depreciated value of the asset after N years to be S and keeps $V(i)$ no less than S.

Conversion to SL

The second (and preferred) way to force declining balance to assume the salvage value is by conversion to straight line. If $V(N) > S$, the first way redefines $V(N)$ to equal S; you can think of this as converting to the straight line method for the last time step.

If the $V(N)$ value supplied by DB is appreciably larger than S, then the depreciation in the final year would be unrealistically large. An alternate way is to compute the DB and SL step at each timestep and take whichever step gives a larger depreciation (unless DB drops below the salvage value).

After SL assumes a larger depreciation, it continues to be larger over the life of the asset. SL forces the value at the final time to equal the salvage value. As an algorithm, this looks like the following statements:

```
V(0) = P;
for i=1 to N
   if DB step > SL step from (i,V(i))
      take a DB step to make V(i);
   else
      break;
for j = i to N
   take a SL step to make V(j);
```

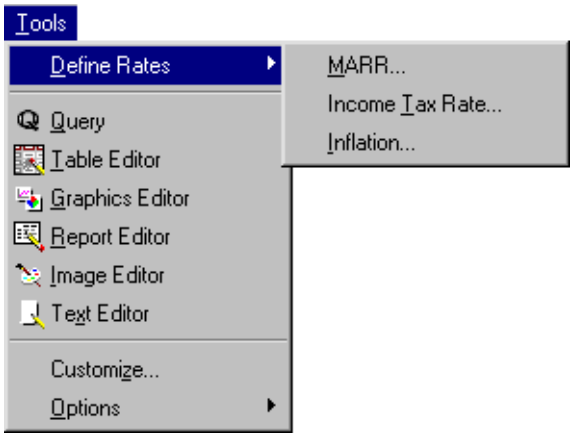
The MACRS, which is discussed in the section that describes the Depreciation Table window, is actually a variation on the declining balance method with conversion to the straight line method.

Rate Information

The Tools Menu

Figure 67.5 shows the **Tools** menu.

Figure 67.5 The Tools Menu



The **Tools** → **Define Rates** menu offers the following options.

MARR opens the Minumum Attractive Rate of Return (MARR) dialog box.

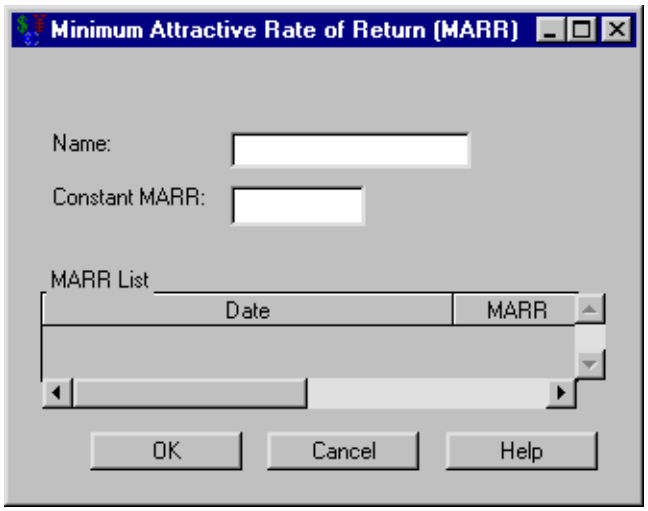
Income Tax Rate opens the Income Tax Specification dialog box.

Inflation opens the Inflation Specification dialog box.

Minimum Attractive Rate of Return (MARR)

Selecting **Tools** → **Define Rates** → **MARR** from the Investment Analysis dialog box menu bar opens the MARR dialog box that is displayed in Figure 67.6.

Figure 67.6 MARR Dialog Box



Name holds the name that you assign to the MARR specification. This must be a valid SAS name.

Constant MARR holds the numeric value that you choose to be the constant MARR. This value is used if the **MARR List** table editor is empty.

MARR List holds date-MARR pairs where the date refers to when the particular MARR value begins. Each date is a SAS date.

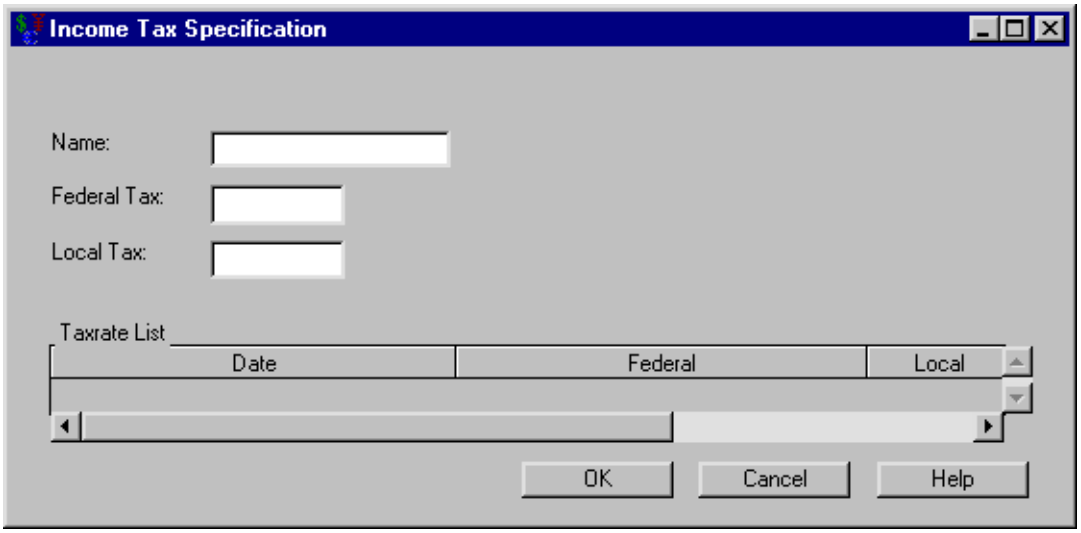
OK returns you to the Investment Analysis dialog box. Clicking it causes the preceding MARR specification to be assumed when you do not specify MARR rates in a dialog box that needs MARR rates.

Cancel returns you to the Investment Analysis dialog box, discarding any work that was done in the MARR dialog box.

Income Tax Specification

Selecting **Tools** → **Define Rates** → **Income Tax Rate** from the Investment Analysis dialog box menu bar opens the Income Tax Specification dialog box displayed in Figure 67.7.

Figure 67.7 Income Tax Specification Dialog Box



Name holds the name you assign to the Income Tax specification. This must be a valid SAS name.

Federal Tax holds the numeric value that you want to be the constant Federal Tax.

Local Tax holds the numeric value that you want to be the constant Local Tax.

Taxrate List holds date-Income Tax triples where the date refers to when the particular Income Tax value begins. Each date is a SAS date, and the value is a percentage between 0% and 100%.

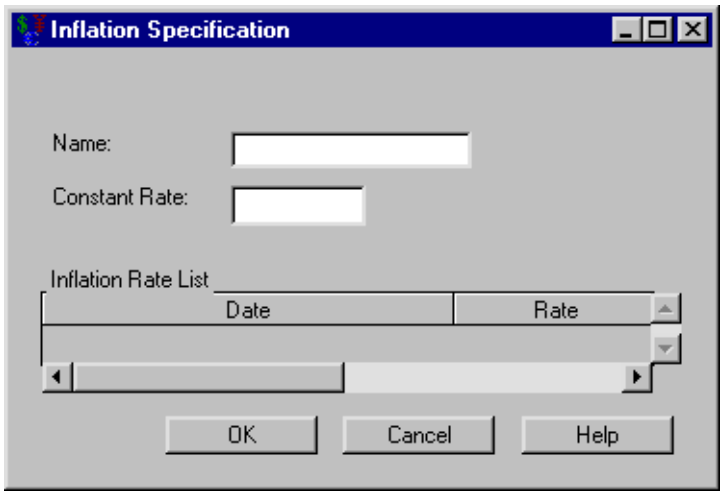
OK returns you to the Investment Analysis dialog box. Clicking it causes the preceding income tax specification to be the default income tax rates when using the After Tax Cashflow Calculation dialog box.

Cancel returns you to the Investment Analysis dialog box, discarding any changes that were made since this dialog box was opened.

Inflation Specification

Selecting **Tools** → **Define Rates** → **Inflation** from the Investment Analysis dialog box menu bar opens the Inflation Specification dialog box displayed in Figure 67.8.

Figure 67.8 Inflation Specification Dialog Box



Name holds the name that you assign to the Inflation specification. This must be a valid SAS name.

Constant Rate holds the numeric value that you want to be the constant inflation rate. This value is used if the **Inflation Rate List** table editor is empty.

Inflation Rate List holds date-rate pairs where the date refers to when the particular inflation rate begins. Each date is a SAS date and the rate is a percentage between 0% and 120%.

OK returns you to the Investment Analysis dialog box. Clicking it causes the preceding inflation specification to be assumed when you use the Constant Dollar Calculation dialog box and do not specify inflation rates.

Cancel returns you to the Investment Analysis dialog box, discarding any changes that were made since this dialog box was opened.

References

Newnan, D. G., and Lavelle, J. P. (1998). *Engineering Economic Analysis*. Austin, TX: Engineering Press.

Subject Index

@CRSPDB date informats
    SASECRSP engine, 3292
@CRSPDR date informats
    SASECRSP engine, 3292
@CRSPDT date informats
    SASECRSP engine, 3292
2SLS estimation method, *see* two-stage least squares
3SLS estimation method, *see* three-stage least squares

add factors, *see* adjustments
additive model
    ARIMA model, 238
additive Winters method
    seasonal forecasting, 965
additive-invertible region
    smoothing weights, 3893
ADDWINTERS method
    FORECAST procedure, 965
ADF test, 373
adjacency graph
    MODEL procedure, 1702
adjust operators, 917
adjustable rate mortgage, *see* LOAN procedure
    LOAN procedure, 1356
adjusted R squared
    MODEL procedure, 1554
adjusted R-square
    statistics of fit, 2820, 3910
adjustments, 3780, 3888
    add factors, 3749
    forecasting models, 3749
    specifying, 3749
After Tax Cashflow Calculation, 3972
AGG= option
    SASEFRED engine, 3386
AGGREGATE method
    EXPAND procedure, 903
aggregation of
    time series data, 885, 888
aggregation of time series
    EXPAND procedure, 885, 888
AIC, *see* Akaike information criterion, *see* Akaike's information criterion
Akaike information criterion
    AIC, 274
    ARIMA procedure, 274
    AUTOREG procedure, 403
    used to select state space models, 2526
Akaike information criterion corrected
    AUTOREG procedure, 403
Akaike's information criterion
    AIC, 3910
    statistics of fit, 3910
    VARMAX procedure, 2981
alignment of
    dates, 3806
    time intervals, 154
alignment of dates, 171, 3806
Almon lag polynomials, *see* polynomial distributed lags
    MODEL procedure, 1625
alternatives to
    DIF function, 131
    LAG function, 131
Amemiya's prediction criterion
    statistics of fit, 3911
Amemiya's R-square
    statistics of fit, 2820, 3910
Amemiya-MaCurdy Estimation
    PANEL procedure, 1850
AMO, 56
amortization schedule
    LOAN procedure, 1382
analyzing models
    MODEL procedure, 1694
and goal seeking
    ordinary differential equations (ODEs), 1598
and state space models
    stationarity, 2507
and tests for autocorrelation
    lagged dependent variables, 344
and the OUTPUT statement
    output data sets, 110
Annuity, *see* Uniform Periodic Equivalent
APIKEY= option
    SASEFRED engine, 3386
    SASEQUAN engine, 3483
AR initial conditions
    conditional least squares, 1613
    Hildreth-Lu, 1613
    maximum likelihood, 1613
    unconditional least squares, 1613
    Yule-Walker, 1613
ARCH model
    AUTOREG procedure, 333
    autoregressive conditional heteroscedasticity, 333

ARIMA model
additive model, 238
ARIMA procedure, 218
autoregressive integrated moving-average model, 218, 3901
Box-Jenkins model, 218
factored model, 238
multiplicative model, 238
notation for, 233
seasonal model, 238
simulating, 3785, 3874
subset model, 238
ARIMA model specification, 3783, 3813
ARIMA models
forecasting models, 3693
specifying, 3693
ARIMA procedure
Akaike information criterion, 274
ARIMA model, 218
ARIMAX model, 218, 239
ARMA model, 218
autocorrelations, 221
autoregressive parameters, 279
BY groups, 252
conditional forecasts, 280
confidence limits, 280
correlation plots, 221
cross-correlation function, 265
data requirements, 246
differencing, 236, 271, 277
factored model, 238
finite memory forecasts, 281
forecasting, 280, 282
Gauss-Marquardt method, 273
ID variables, 282
infinite memory forecasts, 280
input series, 239
interaction effects, 243
intervention model, 239, 242, 244, 318
inverse autocorrelation function, 263
invertibility, 278
Iterative Outlier Detection, 326
log transformations, 281
Marquardt method, 273
Model Identification, 320
moving-average parameters, 279
naming model parameters, 279
ODS graph names, 297
ODS Graphics, 250
Outlier Detection, 325
output data sets, 284–286, 289, 290
output table names, 294
predicted values, 280
prewhitening, 270, 271
printed output, 291
rational transfer functions, 245
regression model with ARMA errors, 239, 240
residuals, 280
Schwarz Bayesian criterion, 274
seasonal model, 238
stationarity, 222
subset model, 238
syntax, 247
time intervals, 282
transfer function model, 239, 243, 274
unconditional forecasts, 281
ARIMA process specification, 3785
ARIMAX model
ARIMA procedure, 218, 239
ARIMAX models and
design matrix, 243
ARMA model
ARIMA procedure, 218
autoregressive moving-average model, 218
MODEL procedure, 1610
notation for, 234
as time ID
observation numbers, 3671
asymptotic distribution of impulse response functions
VARMAX procedure, 2965, 2975
asymptotic distribution of the parameter estimation
VARMAX procedure, 2975
at annual rates
percent change calculations, 133
at sign (@) operator
COUNTREG procedure, 621
attributes
DATASOURCE procedure, 698
attributes of variables
DATASOURCE procedure, 720
audit trail, 3807
Augmented Dickey-Fuller (ADF) test, 373
augmented Dickey-Fuller tests, 256, 270
autocorrelation, 1583
autocorrelation tests, 1583
Durbin-Watson test, 371
Godfrey Lagrange test, 1583
Godfrey's test, 371
autocorrelations
ARIMA procedure, 221
multivariate, 2509
plotting, 221
prediction errors, 3657
series, 3723
AUTOMAP= option
SASEFRED engine, 3386
SASEQUAN engine, 3484
SASEXFSD engine, 3566

automatic forecasting
FORECAST procedure, 937
STATESPACE procedure, 2504
automatic generation
forecasting models, 3623
automatic inclusion of
interventions, 3807
automatic model selection
criterion, 3833
options, 3794
automatic selection
forecasting models, 3688
AUTOREG procedure
Akaike information criterion, 403
Akaike information criterion corrected, 403
ARCH model, 333
autoregressive error correction, 334
BY groups, 360
Cholesky root, 390
Cochrane-Orcutt method, 391
conditional variance, 429
confidence limits, 383
dual quasi-Newton method, 397
Durbin h test, 344
Durbin t test, 344
Durbin-Watson test, 342
EGARCH model, 354
EGLS method, 391
estimation methods, 388
factored model, 346
GARCH model, 333
GARCH-M model, 354
Gauss-Marquardt method, 390
generalized Durbin-Watson tests, 342
Hannan-Quinn information criterion, 403
heteroscedasticity, 347
Hildreth-Lu method, 392
IGARCH model, 354
Kalman filter, 390
lagged dependent variables, 344
maximum likelihood method, 392
nonlinear least-squares, 391
ODS graph names, 437
output data sets, 431
output table names, 434
PGARCH model, 354
Prais-Winsten estimates, 391
predicted values, 383, 427, 428
printed output, 433
QGARCH model, 354
quasi-Newton method, 363
random walk model, 446
residuals, 384
Schwarz Bayesian criterion, 403
serial correlation correction, 334
stepwise autoregression, 345
structural predictions, 427
subset model, 346
TGARCH model, 354
Toeplitz matrix, 388
trust region method, 363
two-step full transform method, 391
Yule-Walker equations, 388
Yule-Walker estimates, 388
autoregressive conditional heteroscedasticity, *see* ARCH model
autoregressive error correction
AUTOREG procedure, 334
autoregressive integrated moving-average model, *see* ARIMA model, *see* ARIMA model
autoregressive models
FORECAST procedure, 959
MODEL procedure, 1610
autoregressive moving-average model, *see* ARMA model
autoregressive parameters
ARIMA procedure, 279
auxiliary data sets
DATASOURCE procedure, 698
auxiliary equations, 1598
MODEL procedure, 1598
available data items
SASEXFSD engine, 3564
available economic data sources
SASEXFSD engine, 3578
available financial data sources
SASEXFSD engine, 3578
available other databases
SASEXFSD engine, 3578

Bai and Perron's multiple structural change tests, 424
balance of payment statistics data files, *see* DATASOURCE procedure
balloon payment mortgage, *see* LOAN procedure
LOAN procedure, 1356
bandwidth functions, 1541
bar (|) operator
COUNTREG procedure, 620
Base SAS software, 46
Basmann test
SYSLIN procedure, 2571, 2584
batch mode, *see* unattended mode
Bayesian vector autoregressive models
VARMAX procedure, 2918, 2970
BDS test, 364, 415
BDS test for
Independence, 364, 415
BEA data files, *see* DATASOURCE procedure

BEA national income and product accounts PC Format
DATASOURCE procedure, 732
BEA S-pages, *see* DATASOURCE procedure
Benefit-Cost Ratio Analysis, 3987
between estimators, 1132, 1840
HPPANEL procedure, 1132
PANEL procedure, 1840
between levels and rates
interpolation, 146
between stocks and flows
interpolation, 146
BIC, *see* Schwarz Bayesian information criterion
block structure
MODEL procedure, 1702
BLS consumer price index surveys
DATASOURCE procedure, 733
BLS data files, *see* DATASOURCE procedure
BLS national employment, hours, and earnings survey
DATASOURCE procedure, 734
BLS producer price index survey
DATASOURCE procedure, 734
BLS state and area employment, hours, and earnings survey
DATASOURCE procedure, 735
Bond, 3941
BOPS data file
DATASOURCE procedure, 750
boundaries
smoothing weights, 3893
bounds on parameter estimates, 813, 1158, 1502, 1970
BOUNDS statement, 813, 1158, 1502, 1970
Box Cox
transformations, 3889
Box Cox transformation, *see* transformations
Box-Cox transformation
BOXCOXAR macro, 180
Box-Jenkins model, *see* ARIMA model
BOXCOXAR macro
Box-Cox transformation, 180
output data sets, 181
SAS macros, 180
BP test, 365
BP test for
Structural Change, 365
break even analysis
LOAN procedure, 1378
Breakeven Analysis, 3989
Breusch-Pagan test, 1576
heteroscedasticity tests, 1576
Brown smoothing model, *see* double exponential smoothing
Bureau of Economic Analysis data files, *see* DATASOURCE procedure
Bureau of Labor Statistics data files, *see* DATASOURCE procedure
buydown rate loans, *see* LOAN procedure
LOAN procedure, 1356
BY groups
ARIMA procedure, 252
AUTOREG procedure, 360
COUNTREG procedure, 601
cross-sectional dimensions and, 106
ESM procedure, 855
EXPAND procedure, 895
FORECAST procedure, 957
HPCDM procedure, 1013
HPCOUNTREG procedure, 1082
HPSEVERITY procedure, 1217
MDC procedure, 1415
PANEL procedure, 1803
PDLREG procedure, 1931
SEVERITY procedure, 2096
SIMILARITY procedure, 2244
SIMLIN procedure, 2306
SPECTRA procedure, 2334
SSM procedure, 2372
STATESPACE procedure, 2520
SYSLIN procedure, 2569
TIMEDATA procedure, 2612
TIMESERIES procedure, 2669
TSCSREG procedure, 2736
UCM procedure, 2762
X11 procedure, 3097
X12 procedure, 3174
X13 procedure, 3174
BY groups and
time series cross-sectional form, 106

CAL, 560
calculation of
leads, 135
calculations
smoothing models, 3890
calendar calculations
functions for, 119, 171
interval functions and, 128
time intervals and, 128
calendar calculations and
INTCK function, 128
INTNX function, 128
time intervals, 128
calendar functions and
date values, 119, 120
calendar variables, 119
computing dates from, 119
computing from dates, 120
computing from datetime values, 121

calibration estimation, 560
canonical correlation analysis
for selection of state space models, 2506, 2527
STATESPACE procedure, 2506, 2527
canonical maximum likelihood estimation, 559
Cashflow, *see* Generic Cashflow
CATALOG procedure, 47
SAS catalogs, 47
Cauchy distribution estimation
example, 1738
examples, 1738
CCM data
SASEXCCM engine, 3520
CCM data item access
SASEXCCM engine, 3527
CDT (COMPUTAB data table)
COMPUTAB procedure, 505
CEIC, Eurostat, FactSet Sourced Economics, Global Insight, IMF, Markit, OECD
SASEXFSD engine, 3578
CEIC_CHINA, CEIC China
SASEXFSD engine, 3568
CEIC_ECON, CEIC Asia
SASEXFSD engine, 3568
ceiling of
time intervals, 125
censored regression models
HPQLIM procedure, 1170
QLIM procedure, 1988
censoring and truncation
HPSEVERITY procedure, 1240
SEVERITY procedure, 2118
Census X-11 method, *see* X11 procedure
Census X-11 methodology
X11 procedure, 3109
Census X-12 method, *see* X12 procedure
X12 procedure, 3149
Census X-13 method, *see* X13 procedure
Center for Research in Security Prices data files, *see* DATASOURCE procedure
centered moving time window operators, 908, 909
change vector, 1555
changes in trend
forecasting models, 3757
changing by interpolation
frequency, 146, 885, 897
periodicity, 146, 885
sampling frequency, 146
changing periodicity
EXPAND procedure, 146
time series data, 146, 885
character functions, 48
character variables
MODEL procedure, 1677
CHART procedure, 47
histograms, 47
checking data periodicity
INTNX function, 127
time intervals, 127
Chirp-Z algorithm
SPECTRA procedure, 2336
choice of
instrumental variables, 1607
Cholesky root
AUTOREG procedure, 390
Chow test, 368, 372, 424
Chow test for
structural change, 368
Chow tests, 1604
MODEL procedure, 1604
CITIBASE format
DATASOURCE procedure, 700
CITIBASE old format
DATASOURCE procedure, 737
CITIBASE PC format
DATASOURCE procedure, 737
CLASS statement
COUNTREG procedure, 601
classical decomposition operators, 912
classification variables
COUNTREG procedure, 619
Clayton copula, 552, 1065
CMLE, 559
CNS_ECON, Consensus Economics
SASEXFSD engine, 3568
Cochrane-Orcutt method
AUTOREG procedure, 391
coherency
cross-spectral analysis, 2341
coherency of cross-spectrum
SPECTRA procedure, 2341
cointegration
VARMAX procedure, 2983
cointegration test, 375, 410
cointegration testing
VARMAX procedure, 2915, 2988
COLLAPSE= option
SASEQUAN engine, 3484
collinearity diagnostics
MODEL procedure, 1559, 1566
column blocks
COMPUTAB procedure, 506
column selection
COMPUTAB procedure, 503, 504
COLxxxxx: label
COMPUTAB procedure, 496
combination models
forecasting models, 3710

specifying, 3710
combined seasonality test, 3125, 3209
combined with cross-sectional dimension
interleaved time series, 108
combined with interleaved time series
cross-sectional dimensions, 108
combining forecasts, 3815, 3888
combining time series data sets, 141
Command Reference, 3771
common trends
VARMAX procedure, 2984
common trends testing
VARMAX procedure, 2917, 2985
COMPARE procedure, 47
comparing SAS data sets, 47
comparing
forecasting models, 3733, 3828
comparing forecasting models, 3733, 3828
comparing loans
LOAN procedure, 1362, 1378, 1382
comparing SAS data sets, *see* COMPARE procedure
compiler listing
MODEL procedure, 1692
compound distribution modeling
HPCDM procedure, 990
COMPUSTAT data files, *see* DATASOURCE procedure
DATASOURCE procedure, 737
COMPUSTAT data items
SASEXFSD engine, 3565
COMPUSTAT IBM 360/370 general format 48 quarter files
DATASOURCE procedure, 739
COMPUSTAT IBM 360/370 general format annual files
DATASOURCE procedure, 738
Compustat Quarterly Point-in-Time data items
SASEXFSD engine, 3584
COMPUSTAT universal character format 48 quarter files
DATASOURCE procedure, 741
COMPUSTAT universal character format annual files
DATASOURCE procedure, 740
Compustat's Xpressfeed data item names
SASEXCCM engine, 3527
COMPUSTAT/Worldscope—global data offerings
SASEXFSD engine, 3558
COMPUTAB procedure
CDT (COMPUTAB data table), 505
column blocks, 506
column selection, 503, 504
COLxxxxx: label, 496
consolidation tables, 497
controlling row and column block execution, 504
input block, 505
missing values, 507
order of calculations, 501
output data sets, 507
program flow, 498
programming statements, 496
reserved words, 507
row blocks, 506
ROWxxxxx: label, 496
table cells, direct access to, 506
computational details
VARMAX procedure, 3032
computing calendar variables from datetime values, 121
computing ceiling of intervals
INTNX function, 125
computing dates from
calendar variables, 119
computing datetime values from
date values, 121
computing ending date of intervals
INTNX function, 124
computing from calendar variables
datetime values, 121
computing from dates
calendar variables, 120
computing from datetime values
calendar variables, 121
date values, 121
time variables, 121
computing from time variables
datetime values, 121
computing lags
RETAIN statement, 131
computing midpoint date of intervals
INTNX function, 124
computing time variables from
datetime values, 121
computing widths of intervals
INTNX function, 125
concatenated
data set, 3637
concentrated likelihood Hessian, 1549
conditional forecasts
ARIMA procedure, 280
conditional least squares
AR initial conditions, 1613
MA Initial Conditions, 1614
conditional logit model
MDC procedure, 1394, 1395, 1428
conditional t distribution
GARCH model, 397
conditional variance
AUTOREG procedure, 429

predicted values, 429
predicting, 429
confidence limits, 3663
ARIMA procedure, 280
AUTOREG procedure, 383
FORECAST procedure, 969
forecasts, 3663
PDLREG procedure, 1935
STATESPACE procedure, 2535
VARMAX procedure, 3018
conjugate-gradient optimization method, 1081
CONNECT= option
SASEXFSD engine, 3567
consolidation tables
COMPUTAB procedure, 497
Constant Dollar Calculation, 3976
constrained estimation
heteroscedasticity models, 381
Consumer Price Index Surveys, *see* DATASOURCE procedure
contemporaneous correlation of
errors across equations, 2580
contents of
SAS data sets, 47
CONTENTS procedure, 47
SASECRSP engine, 3283
SASEFAME engine, 3327
continuous compounding
LOAN procedure, 1376
continuous variables, 619
contrasted with flow variables
stocks, 888
contrasted with flows or rates
levels, 888
contrasted with missing values
omitted observations, 105
contrasted with omitted observations
missing observations, 105
missing values, 105
contrasted with stock variables
flows, 888
contrasted with stocks or levels
rates, 888
control charts, 53
control key
for multiple selections, 3625
control variables
MODEL procedure, 1675
controlling row and column block execution
COMPUTAB procedure, 504
controlling starting values
MODEL procedure, 1561
convergence criteria
MODEL procedure, 1556
convergence problems
VARMAX procedure, 3032
conversion methods
EXPAND procedure, 902
convert option
SASEFAME engine, 3327
converting dates using the CRSP date functions
SASECRSP engine, 3291
converting frequency of
time series data, 885
copula
Clayton, 552, 1065
Frank, 552, 1065
Gumbel, 552, 1065
normal, 548, 1063
Student's *t*, 549
Student's *t*, 1064
COPULA procedure, 533
ODS graph names, 564
output table names, 563
overview, 534
syntax, 538
COPY procedure, 47
copying
SAS data sets, 47
CORR procedure, 47
corrected sum of squares
statistics of fit, 3910
correlation plots
ARIMA procedure, 221
cospectrum estimate
cross-spectral analysis, 2340
SPECTRA procedure, 2340
counting
time intervals, 123, 126
counting time intervals
INTCK function, 126
COUNTREG procedure
bounds on parameter estimates, 601
BY groups, 601
CLASS statement, 601
gamma distribution, 668
Gaussian distribution, 668
initial values, 658
inverse gamma distribution , 668
least squares means, 613
maximum a posteriori, 658
normal distribution, 668
ODS graph names, 672
output table names, 670
restrictions on parameter estimates, 611
standard distributions, 667
syntax, 587
t distribution, 669

uniform distribution, 669
country identifiers for ExtractEconData
SASEXFSD engine, 3569
covariance estimates
GARCH model, 369
Covariance of GMM estimators, 1543
covariance of the parameter estimates, 1535
covariance stationarity
VARMAX procedure, 3013
covariates
heteroscedasticity models, 380, 1161, 1975
CPORT procedure, 47
CPU requirements
VARMAX procedure, 3033
creating
time ID variable, 3667
creating a Fame view, *see* SASEFAME engine
creating a FRED view, *see* SASEFRED engine
creating a Haver view, *see* SASEHAVR engine
creating a Quandl view, *see* SASEQUAN engine
creating from Model Viewer
HTML, 3840
creating from Time Series Viewer
HTML, 3878
criterion
automatic model selection, 3833
cross sectional dimensions
represented by different series, 105
cross sections
DATASOURCE procedure, 704, 705, 709, 719
cross-correlation function
ARIMA procedure, 265
cross-equation covariance matrix
MODEL procedure, 1553
seemingly unrelated regression, 1538
cross-periodogram
cross-spectral analysis, 2330, 2340
SPECTRA procedure, 2340
cross-reference
MODEL procedure, 1691
cross-sectional dimensions, 105
combined with interleaved time series, 108
ID variables for, 106
represented with BY groups, 106
transposing time series, 142
cross-sectional dimensions and
BY groups, 106
cross-spectral analysis
coherency, 2341
cospectrum estimate, 2340
cross-periodogram, 2330, 2340
cross-spectrum, 2341
quadrature spectrum, 2341
SPECTRA procedure, 2329, 2330, 2340, 2341
cross-spectrum
cross-spectral analysis, 2341
SPECTRA procedure, 2341
crossproducts estimator of the covariance matrix, 1548
crossproducts matrix, 1569
crosstabulations, *see* FREQ procedure
CRSP and SAS dates
SASECRSP engine, 3291
CRSP annual data
DATASOURCE procedure, 746
CRSP calendar/indices files
DATASOURCE procedure, 743
CRSP daily binary files
DATASOURCE procedure, 742
CRSP daily character files
DATASOURCE procedure, 742
CRSP daily IBM binary files
DATASOURCE procedure, 742
CRSP daily security files
DATASOURCE procedure, 744
CRSP data files, *see* DATASOURCE procedure
CRSP date formats
SASECRSP engine, 3292
CRSP date functions
SASECRSP engine, 3291
CRSP date informats
SASECRSP engine, 3292
CRSP integer date format
SASECRSP engine, 3291
CRSP monthly binary files
DATASOURCE procedure, 742
CRSP monthly character files
DATASOURCE procedure, 742
CRSP monthly IBM binary files
DATASOURCE procedure, 742
CRSP monthly security files
DATASOURCE procedure, 745
CRSP stock files
DATASOURCE procedure, 742
CRSPAccess Database
DATASOURCE procedure, 742
CRSPDB_SASCAL environment variable
SASECRSP engine, 3283
CRSPDCI date functions
SASECRSP engine, 3293
CRSPDCS date functions
SASECRSP engine, 3293
CRSPDI2S date function
SASECRSP engine, 3293
CRSPDIC date functions
SASECRSP engine, 3293
CRSPDS2I date function
SASECRSP engine, 3293
CRSPDSC date functions

SASECRSP engine, 3293
CRSPDT date formats
SASECRSP engine, 3292
cubic
trend curves, 3906
cubic trend, 3906
cumulative statistics operators, 910
Currency Conversion, 3974
custom model specification, 3795
custom models
forecasting models, 3700
specifying, 3700
CUSUM statistics, 383, 402

Da Silva method
PANEL procedure, 1853
daily STK data item access
SASEXCCM engine, 3531
damped-trend exponential smoothing, 3897
smoothing models, 3897
data frequency, *see* time intervals
data periodicity
FORECAST procedure, 958
data requirements
ARIMA procedure, 246
FORECAST procedure, 968
X11 procedure, 3114
X12 procedure, 3204
X13 procedure, 3204
data set, 3632
concatenated, 3637
forecast data set, 3633
forms of, 3633
interleaved, 3635
simple, 3634
data set selection, 3619, 3798
DATA step, 46
SAS data sets, 46
DATASETS procedure, 47
DATASOURCE procedure
attributes, 698
attributes of variables, 720
auxiliary data sets, 698
balance of payment statistics data files, 698
BEA data files, 698
BEA national income and product accounts PC Format, 732
BEA S-pages, 698
BLS consumer price index surveys, 733
BLS data files, 698
BLS national employment, hours, and earnings survey, 734
BLS producer price index survey, 734
BLS state and area employment, hours, and earnings survey, 735
BOPS data file, 750
Bureau of Economic Analysis data files, 698
Bureau of Labor Statistics data files, 698
Center for Research in Security Prices data files, 698
CITIBASE format, 700
CITIBASE old format, 737
CITIBASE PC format, 737
COMPUSTAT data files, 698, 737
COMPUSTAT IBM 360/370 general format 48 quarter files, 739
COMPUSTAT IBM 360/370 general format annual files, 738
COMPUSTAT universal character format 48 quarter files, 741
COMPUSTAT universal character format annual files, 740
Consumer Price Index Surveys, 698
cross sections, 704, 705, 709, 719
CRSP annual data, 746
CRSP calendar/indices files, 743
CRSP daily binary files, 742
CRSP daily character files, 742
CRSP daily IBM binary files, 742
CRSP daily security files, 744
CRSP data files, 698
CRSP monthly binary files, 742
CRSP monthly character files, 742
CRSP monthly IBM binary files, 742
CRSP monthly security files, 745
CRSP stock files, 742
CRSPAccess Database, 742
direction of trade statistics data files, 698
DOTS data file, 750
DRI Data Delivery Service data files, 698
DRI data files, 698, 736
DRI/McGraw-Hill data files, 698, 736
DRIBASIC data files, 736
DRIBASIC economics format, 700
DRIDDS data files, 736
employment, hours, and earnings survey, 698
event variables, 718, 724
FAME data files, 698
FAME Information Services Databases, 698, 747
formatting variables, 721
frequency of data, 701
frequency of input data, 715
generic variables, 725
GFS data files, 751
Global Insight data files, 698, 736
Global Insight DRI data files, 736
government finance statistics data files, 698

Haver Analytics data files, 749
ID variable, 724
IMF balance of payment statistics, 750
IMF data files, 698
IMF direction of trade statistics, 750
IMF Economic Information System data files, 749
IMF government finance statistics, 751
IMF International Financial Statistics, 704
IMF international financial statistics, 749
indexing the OUT= data set, 714, 778
input file, 714, 715
international financial statistics data files, 698
International Monetary Fund data files, 698, 749
labeling variables, 721
lengths of variables, 711, 721
main economic indicators (OECD) data files, 698
national accounts data files (OECD), 698
national income and product accounts, 698, 732
NIPA Tables, 732
obtaining descriptive information, 702, 705, 706, 709, 725–727
OECD ANA data files, 752
OECD annual national accounts, 752
OECD data files, 698
OECD main economic indicators, 753
OECD MEI data files, 753
OECD QNA data files, 752
OECD quarterly national accounts, 752
Organization for Economic Cooperation and Development data files, 698, 751
OUTALL= data set, 706
OUTBY= data set, 705
OUTCONT= data set, 702, 709
output data sets, 701, 723, 725–727
Producer Price Index Survey, 698
reading data files, 700
renaming variables, 710, 722
SAS YEARCUTOFF= option, 720
state and area employment, hours, and earnings survey, 698
stock data files, 698
subsetting data files, 701, 713
time range, 719
time range of data, 704
time series variables, 701, 724
type of input data file, 714
U.S. Bureau of Economic Analysis data files, 732
U.S. Bureau of Labor Statistics data files, 733
variable list, 722
DATE
ID variables, 99
date values, 3617
calendar functions and, 119, 120
computing datetime values from, 121
computing from datetime values, 121
difference between dates, 125
formats, 97, 167
formats for, 97
functions, 171
incrementing by intervals, 123
informats, 97, 164
informats for, 97
INTNX function and, 123
normalizing to intervals, 124
SAS representation for, 95
syntax for, 96
time intervals, 152
time intervals and, 124
DATE variable, 99
DATE, date ranges, relative dates, absolute dates
SASEXFSD engine, 3579
dates
alignment of, 3806
DATES= option
SASEXFSD engine, 3566
DATETIME
ID variables, 99
datetime values
computing calendar variables from, 121
computing from calendar variables, 121
computing from time variables, 121
computing time variables from, 121
formats, 97, 170
formats for, 97
functions, 171
informats, 97, 164
informats for, 97
SAS representation for, 96
syntax for, 96
time intervals, 152
DATETIME variable, 99
dating variables, 3675
decomposition of prediction error covariance
VARMAX procedure, 2911, 2956
default time ranges, 3800
defined
INTCK function, 123
interleaved time series, 107
INTNX function, 122
omitted observations, 105
time values, 96
defining a custom objective function
HPSEVERITY procedure, 1298
SEVERITY procedure, 2173
defining a custom probability distribution
HPSEVERITY procedure, 1273
SEVERITY procedure, 2148
definition

S matrix, 1536
time series, 3608
degrees of freedom correction, 1553
denominator factors
transfer function model, 244
dependence measures, 547, 1062
dependency list
MODEL procedure, 1699
Depreciation, 3938
derivatives
MODEL procedure, 1679
DERT. variable, 1592
descriptive statistics, *see* UNIVARIATE procedure
HPCDM procedure, 1035
design matrix
ARIMAX models and, 243
details
generalized method of moments, 1539
developing
forecasting models, 3644, 3801
developing forecasting models, 3644, 3801
DFPVALUE macro
Dickey-Fuller test, 182
SAS macros, 182
DFTEST macro
Dickey-Fuller test, 183
output data sets, 184
SAS macros, 183
seasonality, testing for, 183
stationarity, testing for, 183
diagnostic tests, 3681, 3908
time series, 3681
diagnostics and debugging
MODEL procedure, 1689
Dickey-Fuller test, 3909
DFPVALUE macro, 182
DFTEST macro, 183
PROBDF Function, 187
significance probabilities, 187
significance probabilities for, 182
unit root, 187
VARMAX procedure, 2915
Dickey-Fuller tests, 256
DIF function
alternatives to, 131
explained, 129
higher order differences, 132
introduced, 129
MODEL procedure version, 131
multiperiod lags and, 132
percent change calculations and, 133, 134
pitfalls of, 130
second difference, 132
DIF function and
differencing, 129, 130
difference between dates
date values, 125
differences with X11ARIMA/88
X11 procedure, 3108
Differencing, 3808
differencing
ARIMA procedure, 236, 271, 277
DIF function and, 129, 130
higher order, 132
MODEL procedure and, 131
multiperiod differences, 132
percent change calculations and, 133, 134
RETAIN statement and, 131
second difference, 132
STATESPACE procedure, 2522
testing order of, 183
time series data, 129–134
VARMAX procedure, 2907
different forms of
output data sets, 109
differential algebraic equations
ordinary differential equations (ODEs), 1671
differential equations
See ordinary differential equations, 1594
direction of trade statistics data files, *see* DATASOURCE procedure
discrete variables, *see* classification variables
discussed
EXPAND procedure, 145
distributed lag regression models
PDLREG procedure, 1925
distribution
of time series, 888
distribution of
time series data, 888
distribution of time series
EXPAND procedure, 888
DIZ201006 data
SASEXCCM engine, 3531, 3533
DOT as a GLUE character
SASEFAME engine, 3332
DOTS data file
DATASOURCE procedure, 750
double exponential smoothing, *see* exponential smoothing, 3895
Brown smoothing model, 3895
smoothing models, 3895
double-dogleg optimization method, 1081
DRI Data Delivery Service data files, *see* DATASOURCE procedure
DRI data files, *see* DATASOURCE procedure
DATASOURCE procedure, 736
DRI data files in FAME.db, *see* SASEFAME engine

DRI/McGraw-Hill data files, *see* DATASOURCE procedure
DATASOURCE procedure, 736
DRI/McGraw-Hill data files in FAME.db, *see* SASEFAME engine
DRIBASIC data files
DATASOURCE procedure, 736
DRIBASIC economics format
DATASOURCE procedure, 700
DRIDDS data files
DATASOURCE procedure, 736
DROP in the DATA step
SASEFAME engine, 3342
dual quasi-Newton method
AUTOREG procedure, 397
Durbin h test
AUTOREG procedure, 344
Durbin t test
AUTOREG procedure, 344
Durbin-Watson
MODEL procedure, 1553
Durbin-Watson test
autocorrelation tests, 371
AUTOREG procedure, 342
for first-order autocorrelation, 342
for higher-order autocorrelation, 342
p-values for, 342
Durbin-Watson tests, 371
linearized form, 379
dynamic models
SIMLIN procedure, 2302, 2303, 2309, 2323
dynamic multipliers
SIMLIN procedure, 2309, 2310
dynamic regression, 218, 239, 3809, 3810
specifying, 3750
dynamic regressors
forecasting models, 3750
dynamic simulation, 1592
MODEL procedure, 1592, 1639
SIMLIN procedure, 2303
dynamic simultaneous equation models
VARMAX procedure, 2940

EBIT Consolidated formula libraries
SASEXFSD engine, 3584
EBIT data items
SASEXFSD engine, 3584
EBITDA Consolidated formula libraries
SASEXFSD engine, 3584
EBITDA data items
SASEXFSD engine, 3584
econometrics
features in SAS/ETS software, 22
editing selection list
forecasting models, 3707
EGARCH model
AUTOREG procedure, 354
EGLS method
AUTOREG procedure, 391
EIU_ECON, Economist Intelligence Unit (EIU) CountryData database
SASEXFSD engine, 3568
embedded in time series
missing values, 104
embedded missing values, 104
embedded missing values in
time series data, 104
Empirical Distribution Estimation
MODEL procedure, 1551
empirical distribution function
HPSEVERITY procedure, 1260
SEVERITY procedure, 2138
employment, hours, and earnings survey, *see* DATASOURCE procedure
END= option
SASEFRED engine, 3386
SASEQUAN engine, 3486
ending dates of
time intervals, 124
endogenous variables
SYSLIN procedure, 2550
endpoint restrictions for
polynomial distributed lags, 1926, 1931
Engle's Lagrange Multiplier test, 422
Engle's Lagrange Multiplier test for Heteroscedasticity, 422
Enterprise Guide, 55
Enterprise Miner—Time Series nodes, 56
ENTROPY procedure
input data sets, 830
missing values, 829
ODS graph names, 833
output data sets, 831
output table names, 832
environment variable, CRSPDB_SASCAL
SASECRSP engine, 3283
EQ. variables, 1584, 1677
equality restriction
linear models, 817
nonlinear models, 1525, 1600
equation translations
MODEL procedure, 1677
equation variables
MODEL procedure, 1674
Error model options, 3811
error sum of squares
statistics of fit, 3910
ERROR. variables, 1677

errors across equations
contemporaneous correlation of, 2580
ESACF (Extended Sample Autocorrelation Function method), 265
ESM procedure
BY groups, 855
ODS graph names, 870
EST= data set
SIMLIN procedure, 2310
ESTIMATE statement, 1509
estimation convergence problems
MODEL procedure, 1564
estimation methods
AUTOREG procedure, 388
MODEL procedure, 1535
estimation of ordinary differential equations, 1594
MODEL procedure, 1594
EURO_STAT, Statistical Office of the European Commission
SASEXFSD engine, 3568
evaluation range, 3870
event variables
DATASOURCE procedure, 718, 724
exact maximum likelihood estimation, 560
example
Cauchy distribution estimation, 1738
generalized method of moments, 1579, 1628, 1630, 1631, 1633
Goldfeld Quandt Switching Regression Model, 1740
Mixture of Distributions, 1743
Multivariate Mixture of Distributions, 1743
ordinary differential equations (ODEs), 1733
The D-method, 1740
example of Bayesian VAR modeling
VARMAX procedure, 2865
example of Bayesian VECM modeling
VARMAX procedure, 2872
example of causality testing
VARMAX procedure, 2878
example of cointegration testing
VARMAX procedure, 2868
example of multivariate GARCH modeling
VARMAX procedure, 3014
example of restricted parameter estimation and testing
VARMAX procedure, 2876
example of VAR modeling
VARMAX procedure, 2858
example of VARMA modeling
VARMAX procedure, 2976
example of vector autoregressive modeling with exogenous variables
VARMAX procedure, 2873
example of vector error correction modeling
VARMAX procedure, 2866
example, COUNTREG, 673
examples
Cauchy distribution estimation, 1738
Monte Carlo simulation, 1737
Simulating from a Mixture of Distributions, 1743
Switching Regression example, 1740
systems of differential equations, 1733
examples of
time intervals, 158
examples of multivariate GARCH modeling
VARMAX procedure, 2879
exogenous variables
SYSLIN procedure, 2550
EXPAND procedure
AGGREGATE method, 903
aggregation of time series, 885, 888
BY groups, 895
changing periodicity, 146
conversion methods, 902
discussed, 145
distribution of time series, 888
extrapolation, 899
frequency, 885
ID variables, 896, 898
interpolation methods, 902
interpolation of missing values, 146
JOIN method, 903
ODS graph names, 920
output data sets, 918, 919
range of output observations, 899
SPLINE method, 902
STEP method, 903
time intervals, 898
transformation of time series, 890, 904
transformation operations, 904
EXPAND procedure and
interpolation, 145
time intervals, 146
experimental design, 53
explained
DIF function, 129
LAG function, 129
explosive differential equations, 1670
ordinary differential equations (ODEs), 1670
exponential
trend curves, 3906
exponential smoothing, *see* smoothing models
double exponential smoothing, 960
FORECAST procedure, 937, 960
single exponential smoothing, 960
triple exponential smoothing, 960
exponential trend, 3906

Extended Sample Autocorrelation Function (ESACF) method, 265
external
forecasts, 3888
external forecasts, 3888
external sources
forecasting models, 3713, 3812
ExtractBenchmarkDetail
SASEXFSD engine, 3573
ExtractDataSnapshot
SASEXFSD engine, 3572
ExtractEconData
SASEXFSD engine, 3567
ExtractFormulaHistory
SASEXFSD engine, 3571
ExtractOFDBItem
SASEXFSD engine, 3575
ExtractOFDBUniverse
SASEXFSD engine, 3576
ExtractScreenUniverse
SASEXFSD engine, 3576
extrapolation
EXPAND procedure, 899

FACTLET= option
SASEXFSD engine, 3563
Factored ARIMA, 3781, 3808, 3844
Factored ARIMA model specification, 3813
Factored ARIMA models
forecasting models, 3697
specifying, 3697
factored model
ARIMA model, 238
ARIMA procedure, 238
AUTOREG procedure, 346
FactSet data offerings
SASEXFSD engine, 3558
FactSet frequency codes
SASEXFSD engine, 3579
FactSet Fundamentals Data: items(variable names) by formula category
SASEXFSD engine, 3559
FactSet Fundamentals—annual data items
SASEXFSD engine, 3584
FactSet global indices formulas
SASEXFSD engine, 3565
FAME data files, *see* DATASOURCE procedure
Fame data files, *see* SASEFAME engine
Fame GLUE symbol named DOT
SASEFAME engine, 3338
FAME Information Services Databases, *see* DATASOURCE procedure
DATASOURCE procedure, 747
Fame Information Services Databases, *see* SASEFAME engine
fast Fourier transform
SPECTRA procedure, 2336
fatal error when reading from a Fame database
SASEFAME engine, 3327
FCMP procedure, 47
SAS functions, 47
FDS_ECON, FactSet sourced economic data
SASEXFSD engine, 3568
features in SAS/ETS software
econometrics, 22
Federal Reserve Economic Data, *see* SASEFRED engine
FIML estimation method, *see* full information maximum likelihood
Financial Functions
PROBDF Function, 187
financial functions, 48
finishing the Fame CHLI
SASEFAME engine, 3327
finite Fourier transform
SPECTRA procedure, 2330
finite memory forecasts
ARIMA procedure, 281
first-differenced methods, 1839
one-way, 1839
PANEL procedure, 1839
first-stage R squares, 1610
fitting
forecasting models, 3647
fitting custom probability distributions
HPSEVERITY procedure, 1220
SEVERITY procedure, 2098
fitting forecasting models, 3647
fixed rate mortgage, *see* LOAN procedure
LOAN procedure, 1356
fixed-effects model
one-way, 1833
flows
contrasted with stock variables, 888
for first-order autocorrelation
Durbin-Watson test, 342
for higher-order autocorrelation
Durbin-Watson test, 342
for interleaved time series
ID variables, 107
for multiple selections
control key, 3625
for nonlinear models
instrumental variables, 1607
for selection of state space models
canonical correlation analysis, 2506, 2527
for time series data

ID variables, 95
forecast combination, 3815, 3888
FORECAST command, 3772
forecast horizon, 3800, 3870
forecast options, 3818
FORECAST procedure
ADDWINTERS method, 965
automatic forecasting, 937
autoregressive models, 959
BY groups, 957
confidence limits, 969
data periodicity, 958
data requirements, 968
exponential smoothing, 937, 960
forecasting, 937
Holt two-parameter exponential smoothing, 937, 965
ID variables, 957
missing values, 958
output data sets, 968, 969
predicted values, 969
residuals, 969
seasonal forecasting, 962, 965
seasonality, 966
smoothing weights, 965
STEPAR method, 959
stepwise autoregression, 937, 959
time intervals, 958
time series methods, 949
time trend models, 947
Winters method, 937, 962
FORECAST procedure and
interleaved time series, 107, 108
Forecast Studio, 49
forecasting, 3886
ARIMA procedure, 280, 282
FORECAST procedure, 937
MODEL procedure, 1641
STATESPACE procedure, 2504, 2531
VARMAX procedure, 2953
Forecasting menusystem, 43
forecasting models
adjustments, 3749
ARIMA models, 3693
automatic generation, 3623
automatic selection, 3688
changes in trend, 3757
combination models, 3710
comparing, 3733, 3828
custom models, 3700
developing, 3644, 3801
dynamic regressors, 3750
editing selection list, 3707
external sources, 3713, 3812
Factored ARIMA models, 3697
fitting, 3647
interventions, 3755
level shifts, 3760
linear trend, 3741
predictor variables, 3739
reference, 3735
regressors, 3747
seasonal dummy variables, 3766
selecting from a list, 3685
smoothing models, 3690, 3890
sorting, 3732, 3805
specifying, 3681
transfer functions, 3903
trend curves, 3743
forecasting of Bayesian vector autoregressive models
VARMAX procedure, 2971
forecasting process, 3617
forecasting project, 3637
managing, 3822
Project Management window, 3637
saving and restoring, 3639
sharing, 3643
forecasts, 3664
confidence limits, 3663
external, 3888
plotting, 3663
producing, 3631, 3845
form of
state space models, 2505
FORMAT= option
SASEFRED engine, 3387
SASEQUAN engine, 3485
SASEXFSD engine, 3567
formats
date values, 97, 167
datetime values, 97, 170
recommended for time series ID, 98
time values, 170
formats for
date values, 97
datetime values, 97
formatting variables
DATASOURCE procedure, 721
forms of
data set, 3633
Fourier coefficients
SPECTRA procedure, 2340
Fourier transform
SPECTRA procedure, 2330
fractional operators, 914
Frank copula, 552, 1065
FRED data files, *see* SASEHAVR engine
FREQ procedure, 47

crosstabulations, 47
FREQ= option
SASEFRED engine, 3387
SASEQUAN engine, 3486
frequency
changing by interpolation, 146, 885, 897
EXPAND procedure, 885
of time series observations, 110, 146
SPECTRA procedure, 2339
time intervals and, 110, 146
frequency of data, *see* time intervals
DATASOURCE procedure, 701
frequency of input data
DATASOURCE procedure, 715
frequency option
SASEHAVR engine, 3433
from interleaved form
transposing time series, 141
from standard form
transposing time series, 144
full information maximum likelihood
FIML estimation method, 2548
MODEL procedure, 1547
SYSLIN procedure, 2557, 2580
Fuller Battese
variance components, 1841
Fuller's modification to LIML
SYSLIN procedure, 2584
functions, 48
date values, 171
datetime values, 171
lag functions, 1681
mathematical functions, 1680
random-number functions, 1681
time intervals, 171
time values, 171
functions across time
MODEL procedure, 1681
functions for
calendar calculations, 119, 171
time intervals, 122, 171
functions of parameters
nonlinear models, 1509

G4 inverse, 1512
gamma distribution
COUNTREG procedure, 668
definition of (COUNTREG), 668
definition of (HPQLIM), 1176
definition of (QLIM), 2025
HPQLIM procedure, 1176
QLIM procedure, 2025
GARCH in mean model, *see* GARCH-M model
GARCH model
AUTOREG procedure, 333
conditional t distribution, 397
covariance estimates, 369
generalized autoregressive conditional heteroscedasticity, 333
heteroscedasticity models, 380
initial values, 378
starting values, 363
t distribution, 397
GARCH-M model, 396
AUTOREG procedure, 354
GARCH in mean model, 396
Gauss-Marquardt method
ARIMA procedure, 273
AUTOREG procedure, 390
Gauss-Newton method, 1555
Gaussian distribution
COUNTREG procedure, 668
definition of (COUNTREG), 668
definition of (HPQLIM), 1177
definition of (QLIM), 2026
HPQLIM procedure, 1177
MODEL procedure, 1508
QLIM procedure, 2026
General Form Equations
Jacobi method, 1665
Seidel method, 1665
generalized autoregressive conditional heteroscedasticity, *see* GARCH model
generalized Durbin-Watson tests
AUTOREG procedure, 342
generalized least squares
PANEL procedure, 1851
generalized least squares estimator of the covariance matrix, 1549
generalized least-squares
Yule-Walker method as, 391
Generalized Method of Moments
V matrix, 1540, 1545
generalized method of moments
details, 1539
example, 1579, 1628, 1630, 1631, 1633
generating models, 3787
Generic Cashflow, 3945
generic variables
DATASOURCE procedure, 725
GFS data files
DATASOURCE procedure, 751
giving dates to
time series data, 95
global constituents formulas
SASEXFSD engine, 3565
Global Insight data files
DATASOURCE procedure, 736

Global Insight DRI data files, *see* DATASOURCE procedure
DATASOURCE procedure, 736
global statements, 47
GLUE symbol
SASEFAME engine, 3332
GMM
simulated method of moments, 1543
SMM, 1543
GMM in panel: Arellano and Bond's estimator
panel GMM, 1855
goal seeking
MODEL procedure, 1660
goal seeking problems, 1598
Godfrey Lagrange test
autocorrelation tests, 1583
Godfrey's test, 371
autocorrelation tests, 371
Goldfeld Quandt Switching Regression Model
example, 1740
goodness of fit, *see* statistics of fit
goodness-of-fit statistics, *see* statistics of fit, 3864, *see* statistics of fit
government finance statistics data files, *see* DATASOURCE procedure
gradient of the objective function, 1568, 1569
Granger causality test
VARMAX procedure, 2966
graph names
SSM procedure, 2419
graphics
SAS/GRAPH software, 50
graphs, *see* Model Viewer, *see* Time Series Viewer
grid search
MODEL procedure, 1563
Gumbel copula, 552, 1065
GVIIDKEY as composite key for security items
SASEXCCM engine, 3527
GVIIDKEY as Compustat's permanent issue identifier
SASEXCCM engine, 3523
GVKEY access to CCM data
SASEXCCM engine, 3520
GVKEY as Compustat's permanent SPC identifier
SASEXCCM engine, 3523
GVKEY.IID as composite key for accessing CCM Security data
SASEXCCM engine, 3523

HAC
PANEL procedure, 1870
Hannan-Quinn information criterion
AUTOREG procedure, 403
Hausman specification test, 1603
MODEL procedure, 1603
Hausman-Taylor Estimation
PANEL procedure, 1849
Haver Analytics data files
DATASOURCE procedure, 749
Haver data files, *see* SASEHAVR engine
Haver Information Services Databases, *see* SASEHAVR engine
HCCME 2SLS, 1582
HCCME 3SLS, 1582
HCCME =
PANEL procedure, 1866
HCCME OLS, 1580
HCCME SUR, 1582
help system, 21
Henze-Zirkler test, 1574
normality tests, 1574
heteroscedastic errors, 1539
heteroscedastic extreme value model
MDC procedure, 1404, 1429
Heteroscedasticity
Engle's Lagrange Multiplier test for, 422
Lee and King's test for, 423
Portmanteau Q test for, 422
Wong and Li's test for, 423
heteroscedasticity, 1473, 1576
AUTOREG procedure, 347
Lagrange multiplier test, 381
testing for, 347
heteroscedasticity models, *see* GARCH model
constrained estimation, 381
covariates, 380, 1161, 1975
link function, 380
heteroscedasticity tests
Breusch-Pagan test, 1576
Lagrange multiplier test, 381
White's test, 1576
heteroscedasticity- and autocorrelation-consistent covariance matrices, 1870
Heteroscedasticity-Consistent Covariance Matrix Estimation , 1580
heteroscedasticity-corrected covariance matrices, 1866
hierarchical Archimedean copula, 557
higher order
differencing, 132
higher order differences
DIF function, 132
higher order sums
summation, 137
Hildreth-Lu
AR initial conditions, 1613
Hildreth-Lu method
AUTOREG procedure, 392
histograms, *see* CHART procedure
hold-out sample, 3800

hold-out samples, 3735
Holt smoothing model, *see* linear exponential smoothing
Holt two-parameter exponential smoothing
  FORECAST procedure, 937, 965
Holt-Winters Method, *see* Winters Method
Holt-Winters method, *see* Winters method
homoscedastic errors, 1576
HPCDM procedure
  BY groups, 1013
  descriptive statistics, 1035
  ODS graph names, 1042
  ODS table names, 1040
  parameter perturbation analysis, 1035
  scenario analysis, 1019
  simulating aggregate adjusted loss distribution, 1027
  simulating aggregate loss distribution, 1020
HPCOPULA procedure
  multithreading, 1061
  overview, 1057
  syntax, 1059
HPCOUNTREG procedure
  bounds on parameter estimates, 1082
  BY groups, 1082
  multithreading, 1087
  output table names, 1109
  restrictions on parameter estimates, 1088
  syntax, 1076
HPPANEL procedure
  between estimators, 1132
  ID variables, 1121
  linear hypothesis testing, 1132
  multithreading, 1123
  one-way fixed-effects model, 1126
  one-way random-effects model, 1129
  output data sets, 1134
  output table names, 1135
  pooled estimator, 1132
  printed output, 1135
  specification tests, 1133
  two-way fixed-effects model, 1127
  two-way random-effects model, 1130
HPQLIM procedure, 1144
  BY groups, 1159
  censored regression models, 1170
  frontier, 1171
  gamma distribution, 1176
  Gaussian distribution, 1177
  heteroscedasticity, 1172
  inverse gamma distribution, 1177
  limited dependent variable models, 1170
  logit, 1144
  multithreading, 1166
  normal distribution, 1177
  ordinal discrete choice modeling, 1169
  output, 1178
  output ODS Graphics table names, 1184
  output table names, 1183
  probit, 1144
  selection, 1144
  standard distributions, 1175
  syntax, 1147
  t distribution, 1177
  tests on parameters, 1173
  Tobit, 1144
  truncated regression models, 1170
  uniform distribution, 1178
HPSEVERITY procedure
  BY groups, 1217
  censoring and truncation, 1240
  defining a custom objective function, 1298
  defining a custom probability distribution, 1273
  empirical distribution function, 1260
  ODS graph names, 1309
  ODS table names, 1306
  predefined distributions, 1230
  predefined utility functions, 1285
  probability of observability, 1241
  scale regression model, 1245
  scoring functions, 1290
  statistics of fit, 1266
  Tweedie distribution, 1232
HTML
  creating from Model Viewer, 3840
  creating from Time Series Viewer, 3878
hyperbolic
  trend curves, 3906
hyperbolic trend, 3906

I/B/E/S data offerings
  SASEXFSD engine, 3558
IBJ_NIKKO, INDB (I-N Information Systems) Japanese Macroeconomic Database
  SASEXFSD engine, 3568
ID groups
  MDC procedure, 1415
ID values for
  time intervals, 124
ID variable, *see* time ID variable
  DATASOURCE procedure, 724
ID variable for
  time series data, 95
ID variables, 3622
  ARIMA procedure, 282
  DATE, 99
  DATETIME, 99
  EXPAND procedure, 896, 898

for interleaved time series, 107
for time series data, 95
FORECAST procedure, 957
HPPANEL procedure, 1121
PANEL procedure, 1807
SIMLIN procedure, 2307
sorting by, 99
STATESPACE procedure, 2521
TSCSREG procedure, 2737
X11 procedure, 3097, 3099
X12 procedure, 3180
X13 procedure, 3180
ID variables for
cross-sectional dimensions, 106
interleaved time series, 107
time series cross-sectional form, 106
IDLIST= option
SASEFRED engine, 3386
SASEQUAN engine, 3486
IDS= option
SASEXFSD engine, 3564
IGARCH model
AUTOREG procedure, 354
IMF balance of payment statistics
DATASOURCE procedure, 750
IMF data files, *see* DATASOURCE procedure
IMF direction of trade statistics
DATASOURCE procedure, 750
IMF Economic Information System data files
DATASOURCE procedure, 749
IMF government finance statistics
DATASOURCE procedure, 751
IMF International Financial Statistics
DATASOURCE procedure, 704
IMF international financial statistics
DATASOURCE procedure, 749
IMF_IFS, IMF International Financial Statistics
SASEXFSD engine, 3568
IML, *see* SAS/IML software
IML Studio software, 52
impact multipliers
SIMLIN procedure, 2309, 2313
impulse function
intervention model and, 242
impulse response function
VARMAX procedure, 2911, 2943
impulse response matrix
of a state space model, 2534
in SAS data sets
time series, 3608
in standard form
output data sets, 110
incrementing by intervals
date values, 123
incrementing dates
INTNX function, 123
incrementing dates by
time intervals, 122, 123
IND data
SASEXCCM engine, 3521
IND data item access
SASEXCCM engine, 3533
Independence
BDS test for, 364, 415
Rank Version of the von Neumann Ratio test for, 417
Rank version of von Neumann ratio test for, 377
Runs test for, 372, 416
Turning Point test for, 377, 416
independent variables, *see* predictor variables
indexing
OUT= data set, 724
indexing the OUT= data set
DATASOURCE procedure, 714, 778
INDNO access to IND data
SASEXCCM engine, 3521
INDNO as CRSP's primary key for indices access
SASEXCCM engine, 3524
inequality restriction
linear models, 817
nonlinear models, 1502, 1525, 1600
infinite memory forecasts
ARIMA procedure, 280
infinite order AR representation
VARMAX procedure, 2911
infinite order MA representation
VARMAX procedure, 2911, 2943
informats
date values, 97, 164
datetime values, 97, 164
time values, 164
informats for
date values, 97
datetime values, 97
initial values, 378, 1419
COUNTREG procedure, 658
GARCH model, 378
QLIM procedure, 2013
initializations
smoothing models, 3891
initializing lags
MODEL procedure, 1684
SIMLIN procedure, 2311
innovation vector
of a state space model, 2505
input block
COMPUTAB procedure, 505
input data set, 3619, 3798

input data sets
ENTROPY procedure, 830
MODEL procedure, 1627
input file
DATASOURCE procedure, 714, 715
input matrix
of a state space model, 2505
input series
ARIMA procedure, 239
INPUT variables
X12 procedure, 3182
X13 procedure, 3182
inputs, *see* predictor variables
installment loans, *see* LOAN procedure
instrumental regression, 1537
instrumental variables, 1537
choice of, 1607
for nonlinear models, 1607
number to use, 1608
SYSLIN procedure, 2550
instruments, 1536
INTCK function
calendar calculations and, 128
counting time intervals, 126
defined, 123
INTCK function and
time intervals, 123, 126
interaction effects
ARIMA procedure, 243
interest rates
LOAN procedure, 1377
interim multipliers
SIMLIN procedure, 2305, 2309, 2312, 2313
interleaved
data set, 3635
interleaved form
output data sets, 109
interleaved form of
time series data set, 107
interleaved time series
and _TYPE_ variable, 107, 108
combined with cross-sectional dimension, 108
defined, 107
FORECAST procedure and, 107, 108
ID variables for, 107
plots of, 116
Internal Rate of Return, 3986
internal rate of return
LOAN procedure, 1378
internal variables
MODEL procedure, 1675
international financial statistics data files, *see* DATASOURCE procedure
International Monetary Fund data files, *see* DATASOURCE procedure
DATASOURCE procedure, 749
interpolation
between levels and rates, 146
between stocks and flows, 146
EXPAND procedure and, 145
of missing values, 146, 886
time series data, 146
to higher frequency, 146
to lower frequency, 146
interpolation methods
EXPAND procedure, 902
interpolation of
missing values, 146
time series data, 145, 146, 886
interpolation of missing values
EXPAND procedure, 146
interpolation of time series
step function, 903
interrupted time series analysis, *see* intervention model
interrupted time series model, *see* intervention model
interval functions, *see* time intervals, functions
interval functions and
calendar calculations, 128
INTERVAL= option and
time intervals, 110
intervals, *see* time intervals, 3622
intervention analysis, *see* intervention model
intervention model
ARIMA procedure, 239, 242, 244, 318
interrupted time series analysis, 242
interrupted time series model, 239
intervention analysis, 242
intervention model and
impulse function, 242
step function, 243
intervention notation, 3908
intervention specification, 3819, 3820
interventions, 3906
automatic inclusion of, 3807
forecasting models, 3755
point, 3907
predictor variables, 3906
ramp, 3907
specifying, 3755
step, 3907
INTNX function
calendar calculations and, 128
checking data periodicity, 127
computing ceiling of intervals, 125
computing ending date of intervals, 124
computing midpoint date of intervals, 124
computing widths of intervals, 125

defined, 122
incrementing dates, 123
normalizing dates in intervals, 124
INTNX function and
date values, 123
time intervals, 122
introduced
DIF function, 129
LAG function, 129
percent change calculations, 133
time variables, 119
inverse autocorrelation function
ARIMA procedure, 263
inverse gamma distribution
HPQLIM procedure, 1177
inverse gamma distribution
COUNTREG procedure, 668
definition of (COUNTREG), 668
definition of (HPQLIM), 1177
inverse-gamma distribution
definition of (QLIM), 2025
QLIM procedure, 2025
invertibility
ARIMA procedure, 278
VARMAX procedure, 2972
Investment Analysis System, 44
Investment Portfolio, 3925
invoking the system, 3612
IRoR, 3986
irregular component
X11 procedure, 3086, 3091
ISO currency codes
SASEXFSD engine, 3581
ISON= option
SASEXFSD engine, 3559
item-handling access, items, groups, and reporting formats
SASEXCCM engine, 3524
ITEMS= option
SASEXFSD engine, 3564
iterated generalized method of moments, 1543
iterated seemingly unrelated regression
SYSLIN procedure, 2580
iterated three-stage least squares
SYSLIN procedure, 2580
Iterative Outlier Detection
ARIMA procedure, 326

J function
creating a matrix of identical values, 2934
Jacobi method
MODEL procedure, 1664
Jacobi method with General Form Equations
MODEL procedure, 1665
Jacobian, 1537, 1555
Jarque-Bera test, 372
normality tests, 372
JMP, 54
JOIN method
EXPAND procedure, 903
joint generalized least squares, *see* seemingly unrelated regression
jointly dependent variables
SYSLIN procedure, 2550

K-class estimation
SYSLIN procedure, 2579
Kalman filter
AUTOREG procedure, 390
STATESPACE procedure, 2506
used for state space modeling, 2506
Kaplan-Meier's EDF estimator
HPSEVERITY procedure, 1262
SEVERITY procedure, 2139
KEEP in the DATA step
SASEFAME engine, 3342
kernels, 1541, 2336
SPECTRA procedure, 2336
keysets role in classifying and presenting CCM data
SASEXCCM engine, 3527
Kolmogorov-Smirnov test, 1574
normality tests, 1574
KPSS (Kwiatkowski, Phillips, Schmidt, Shin) test, 374
KPSS test, 374, 413
unit roots, 374, 413
Kruskal-Wallis test, 3125

labeling variables
DATASOURCE procedure, 721
LAG function
alternatives to, 131
explained, 129
introduced, 129
MODEL procedure version, 131
multiperiod lags and, 132
percent change calculations and, 133, 134
pitfalls of, 130
LAG function and
Lags, 129
lags, 129, 130
lag functions
functions, 1681
MODEL procedure, 1681
lag lengths
MODEL procedure, 1683
lag logic
MODEL procedure, 1682
lagged dependent variables

and tests for autocorrelation, 344
AUTOREG procedure, 344
lagged endogenous variables
SYSLIN procedure, 2550
lagging
time series data, 129–134
Lagrange multiplier test
heteroscedasticity, 381
heteroscedasticity tests, 381
linear hypotheses, 818
nonlinear hypotheses, 1173, 1440, 1534, 1601, 2000
Lags
LAG function and, 129
lags
LAG function and, 129, 130
MODEL procedure and, 131
multiperiod lagging, 132
percent change calculations and, 133, 134
RETAIN statement and, 131
SIMLIN procedure, 2311
lambda, 1555
language differences
MODEL procedure, 1685
large problems
MODEL procedure, 1570, 1571
leads
calculation of, 135
multiperiod, 135
time series data, 135
Lee and King's test, 423
Lee and King's test for
Heteroscedasticity, 423
left-hand side expressions
nonlinear models, 1674
legacy data access, SASECRSP
SASEXCCM engine, 3520
lengths of variables
DATASOURCE procedure, 711, 721
level shifts
forecasting models, 3760
specifying, 3760
levels
contrasted with flows or rates, 888
levels, of classification variable, 619
LIBNAME *libref* SASEHAVR '*physical name*' on Windows
SASEFAME engine, 3338
LIBNAME *libref* SASEHAVR '*physical name*'on UNIX
SASEFAME engine, 3338
LIBNAME interface engine for Fame database, *see* SASEFAME engine
LIBNAME interface engine for FRED database, *see* SASEFRED engine
LIBNAME interface engine for Haver database, *see* SASEHAVR engine
LIBNAME interface engine for Quandl data sets, *see* SASEQUAN engine
LIBNAME libref SASEFRED statement
SASEFRED engine, 3386
LIBNAME libref SASEQUAN statement
SASEQUAN engine, 3483
LIBNAME libref SASEXFSD statement
SASEXFSD engine, 3563
LIBNAME statement
SASECRSP engine, 3280
SASEFAME engine, 3326
SASEHAVR engine, 3382, 3432, 3480
SASEXCCM engine, 3520
SASEXFSD engine, 3558
likelihood confidence intervals, 1605
MODEL procedure, 1605
Likelihood ratio test
nonlinear hypotheses, 1440
likelihood ratio test
linear hypotheses, 818
nonlinear hypotheses, 1534, 1602
LIMIT= option
SASEFRED engine, 3388
limitations on
ordinary differential equations (ODEs), 1670
limitations on ordinary differential equations
MODEL procedure, 1670
limited dependent variable models
HPQLIM procedure, 1170
QLIM procedure, 1988
limited information maximum likelihood
LIML estimation method, 2548
SYSLIN procedure, 2579
LIML estimation method, *see* limited information maximum likelihood
linear
trend curves, 3906
linear dependencies
MODEL procedure, 1566
linear exponential smoothing, 3896
Holt smoothing model, 3896
smoothing models, 3896
linear hypotheses
Lagrange multiplier test, 818
likelihood ratio test, 818
Wald test, 818
linear hypothesis testing, 1132, 1865
HPPANEL procedure, 1132
linear models
equality restriction, 817

inequality restriction, 817
restricted estimation, 817
linear structural equations
SIMLIN procedure, 2308
linear trend, 3741, 3906
forecasting models, 3741
linearized form
Durbin-Watson tests, 379
link function
heteroscedasticity models, 380
Linux 64-bit
SASEXCCM engine, 3522
Listing the Haver selection keys, OUTSELECT=ON
SASEHAVR engine, 3433
Loan, 3931
LOAN procedure
adjustable rate mortgage, 1355, 1356
amortization schedule, 1382
balloon payment mortgage, 1355, 1356
break even analysis, 1378
buydown rate loans, 1355, 1356
comparing loans, 1362, 1378, 1382
continuous compounding, 1376
fixed rate mortgage, 1355, 1356
installment loans, 1355
interest rates, 1377
internal rate of return, 1378
loan repayment schedule, 1382
loan summary table, 1381
loans analysis, 1355
minimum attractive rate of return, 1378
mortgage loans, 1355
output data sets, 1379, 1380
output table names, 1382
present worth of cost, 1378
rate adjustment cases, 1373
taxes, 1378
true interest rate, 1378
types of loans, 1356
loan repayment schedule
LOAN procedure, 1382
loan summary table
LOAN procedure, 1381
loans analysis, *see* LOAN procedure
log
transformations, 3889
log likelihood value, 372
log test, 3909
log transformation, *see* transformations
log transformations
ARIMA procedure, 281
LOGTEST macro, 185
logarithmic
trend curves, 3906
logarithmic trend, 3906
logistic
transformations, 3889
trend curves, 3906
logistic trend, 3906
logit
HPQLIM procedure, 1144
QLIM Procedure, 1948
LOGTEST macro
log transformations, 185
output data sets, 186
SAS macros, 185
long-run relations testing
VARMAX procedure, 2996
loss distribution modeling
HPSEVERITY procedure, 1194
SEVERITY procedure, 2068

%MA and %AR macros combined, 1621
MA Initial Conditions
conditional least squares, 1614
maximum likelihood, 1614
unconditional least squares, 1614
macro return codes
MODEL procedure, 1689
macros, *see* SAS macros
MAE
AUTOREG procedure, 401
main economic indicators (OECD) data files, *see* DATASOURCE procedure
main economic indicators (OECD) data files in FAME.db, *see* SASEFAME engine
managing
forecasting project, 3822
managing forecasting projects, 3822
MAPE
AUTOREG procedure, 401
MAPREF= option
SASEFRED engine, 3388
SASEQUAN engine, 3484
SASEXFSD engine, 3566
MAPREF= option, SAS XML map
SASEFRED engine, 3395
SASEQUAN engine, 3497
SASEXFSD engine, 3561
Mardia's test, 1574
normality tests, 1574
Marquardt method
ARIMA procedure, 273
Marquardt-Levenberg method, 1555
MARR, *see* minimum attractive rate of return, 4005
mathematical functions, 48
functions, 1680
matrix language

SAS/IML software, 51
maximizing likelihood functions, 54
maximum a posteriori
COUNTREG procedure, 658
QLIM procedure, 2013
maximum likelihood
AR initial conditions, 1613
MA Initial Conditions, 1614
maximum likelihood method
AUTOREG procedure, 392
MDC procedure
binary data modeling example, 1442
binary logit example, 1442, 1446
binary probit example, 1442
bounds on parameter estimates, 1414
BY groups, 1415
conditional logit example, 1446
conditional logit model, 1394, 1395, 1428
goodness-of-fit measures, 1438
Hausman's specification and likelihood ratio tests for nested logit, 1442
heteroscedastic extreme value model, 1404, 1429
ID groups, 1415
introductory examples, 1395
mixed logit model, 1409, 1430
multinomial discrete choice, 1427
multinomial probit example, 1448
multinomial probit model, 1403, 1432
nested logit example, 1455
nested logit model, 1400, 1434
output table names, 1442
restrictions on parameter estimates, 1424
syntax, 1411
Tests on Parameters, 1439
mean absolute error
statistics of fit, 3910
mean absolute percent error
statistics of fit, 2820, 3910
mean percent error
statistics of fit, 3911
mean prediction error
statistics of fit, 3911
mean square error
statistics of fit, 2820
mean squared error
statistics of fit, 3910
MEANS procedure, 47
measure
dependence, 547, 1062
measurement equation
observation equation, 2505
of a state space model, 2505
MELO estimation method, *see* minimum expected loss estimator
memory requirements
MODEL procedure, 1571
VARMAX procedure, 3033
menu interfaces
to SAS/ETS software, 43, 44
merging series
time series data, 141
merging time series data sets, 141
Michaelis-Menten Equations, 1598
midpoint dates of
time intervals, 124
MINIC (Minimum Information Criterion) method, 267
minimization methods
MODEL procedure, 1555
minimization summary
MODEL procedure, 1557
minimum attractive rate of return
LOAN procedure, 1378
MARR, 1378
minimum expected loss estimator
MELO estimation method, 2579
SYSLIN procedure, 2579
minimum information criteria method
VARMAX procedure, 2962
Minimum Information Criterion (MINIC) method, 267
missing observations
contrasted with omitted observations, 105
missing values, 911, 1629
COMPUTAB procedure, 507
contrasted with omitted observations, 105
embedded in time series, 104
ENTROPY procedure, 829
FORECAST procedure, 958
interpolation of, 146
MODEL procedure, 1553, 1666
smoothing models, 3892
time series data, 886
time series data and, 104
VARMAX procedure, 2936
missing values and
time series data, 104
missing values and Compustat's missing codes
SASEXCCM engine, 3526
MISSONLY operator, 911
mixed logit model
MDC procedure, 1409, 1430
Mixture of Distributions
example, 1743
MIZ201006 data
SASEXCCM engine, 3532
MLE, 560
MMAE, 3890
MMSE, 3890
model evaluation, 3884

Model Identification
ARIMA procedure, 320
model list, 3652, 3829
MODEL procedure
adjacency graph, 1702
adjusted R squared, 1554
Almon lag polynomials, 1625
analyzing models, 1694
ARMA model, 1610
autoregressive models, 1610
auxiliary equations, 1598
block structure, 1702
character variables, 1677
Chow tests, 1604
collinearity diagnostics, 1559, 1566
compiler listing, 1692
control variables, 1675
controlling starting values, 1561
convergence criteria, 1556
cross-equation covariance matrix, 1553
cross-reference, 1691
dependency list, 1699
derivatives, 1679
diagnostics and debugging, 1689
Durbin-Watson, 1553
dynamic simulation, 1592, 1639
Empirical Distribution Estimation, 1551
equation translations, 1677
equation variables, 1674
estimation convergence problems, 1564
estimation methods, 1535
estimation of ordinary differential equations, 1594
forecasting, 1641
full information maximum likelihood, 1547
functions across time, 1681
Gaussian distribution, 1508
goal seeking, 1660
grid search, 1563
Hausman specification test, 1603
initializing lags, 1684
input data sets, 1627
internal variables, 1675
Jacobi method, 1664
Jacobi method with General Form Equations, 1665
lag functions, 1681
lag lengths, 1683
lag logic, 1682
language differences, 1685
large problems, 1570, 1571
likelihood confidence intervals, 1605
limitations on ordinary differential equations, 1670
linear dependencies, 1566
macro return codes, 1689
memory requirements, 1571
minimization methods, 1555
minimization summary, 1557
missing values, 1553, 1666
model variables, 1674
Monte Carlo simulation, 1737
Moore-Penrose generalized inverse, 1512
moving average models, 1610
Multivariate t-Distribution Estimation, 1549
n-period-ahead forecasting, 1639
nested iterations, 1554
Newton's Method, 1662
nonadditive errors, 1584
normal distribution, 1508
ODS graph names, 1638
ordinary differential equations and goal seeking, 1598
output data sets, 1632
output table names, 1635
parameters, 1675
polynomial distributed lag models, 1625
program listing, 1690
program variables, 1677
properties of the estimates, 1552
quasi-random number generators, 1651
R squared, 1554, 1560
random-number generating functions, 1681
restrictions on parameters, 1620
S matrix, 1553
S-iterated methods, 1554
Seidel method, 1665
Seidel method with General Form Equations, 1665
SIMNLIN procedure, 1471
simulated nonlinear least squares, 1547
simulation, 1641
simulation dependency analysis, 1694
solution mode output, 1654
solution modes, 1639, 1662
SOLVE Data Sets, 1671
starting values, 1558, 1564
static simulation, 1592
static simulations, 1639
stochastic simulation, 1642
storing programs, 1688
summary statistics, 1657
SYSNLIN procedure, 1471
systems of ordinary differential equations, 1733
tests on parameters, 1601
time variable, 1598
troubleshooting estimation convergence problems, 1558
troubleshooting simulation problems, 1666

using models to forecast, 1641
using solution modes, 1639
variables in model program, 1673
WEIGHT variable, 1577
MODEL procedure and
differencing, 131
lags, 131
MODEL procedure version
DIF function, 131
LAG function, 131
model selection, 3843
model selection criterion, 3730, 3833
model selection for X-11-ARIMA method
X11 procedure, 3117
model selection list, 3834
model variables
MODEL procedure, 1674
Model Viewer, 3654, 3838
graphs, 3646
plots, 3646
saving graphs and tables, 3851, 3852
modified Kaplan-Meier's EDF estimator
HPSEVERITY procedure, 1262
SEVERITY procedure, 2140
Monte Carlo simulation, 1642, 1737
examples, 1737
MODEL procedure, 1737
monthly STK data item access
SASEXCCM engine, 3532
Moore-Penrose generalized inverse, 1512
mortgage loans, *see* LOAN procedure
moving average function, 1681
moving average models, 1611
MODEL procedure, 1610
moving averages
percent change calculations, 134
moving between computer systems
SAS data sets, 47
moving product and geometric mean operators, 915
moving rank operator, 914
moving seasonality test, 3125
moving t-value operators, 918
moving time window operators, 908
moving-average parameters
ARIMA procedure, 279
MSCI Global Index and Constituents data offerings
SASEXFSD engine, 3558
multinomial discrete choice
independence from irrelevant alternatives, 1429
MDC procedure, 1427
multinomial probit model
MDC procedure, 1403, 1432
multiperiod
leads, 135
multiperiod differences
differencing, 132
multiperiod lagging
lags, 132
multiperiod lags and
DIF function, 132
LAG function, 132
summation, 136, 137
multiple selections, 3625
multiplicative model
ARIMA model, 238
multiplicative seasonal smoothing, 3899
smoothing models, 3899
multipliers
SIMLIN procedure, 2305, 2306, 2309, 2312, 2313
multipliers for higher order lags
SIMLIN procedure, 2310, 2323
multithreading
HPCOPULA procedure, 1061
HPCOUNTREG procedure, 1087
HPPANEL procedure, 1123
HPQLIM procedure, 1166
multivariate
autocorrelations, 2509
normality tests, 1574
partial autocorrelations, 2526
multivariate forecasting
STATESPACE procedure, 2504
multivariate GARCH modeling
VARMAX procedure, 2902
Multivariate Mixture of Distributions
example, 1743
multivariate model diagnostic checks
VARMAX procedure, 2981
Multivariate t-Distribution Estimation
MODEL procedure, 1549
multivariate time series
STATESPACE procedure, 2504

n-period-ahead forecasting
MODEL procedure, 1639
naming
time intervals, 111, 152
naming model parameters
ARIMA procedure, 279
national accounts data files (OECD), *see* DATASOURCE procedure
national accounts data files (OECD) in FAME.db, *see* SASEFAME engine
national income and product accounts, *see* DATASOURCE procedure
DATASOURCE procedure, 732
negative log likelihood function, 1548

negative log-likelihood function, 1550
Nerlove
    variance components, 1843
nested iterations
    MODEL procedure, 1554
nested logit model
    MDC procedure, 1400, 1434
Newton's Method
    MODEL procedure, 1662
Newton-Raphson
    optimization methods, 594, 1153, 1420
Newton-Raphson method, 594, 1153, 1420
Newton-Raphson optimization method, 1081
Newton-Raphson Ridge optimization method, 1081
NIPA Tables
    DATASOURCE procedure, 732
NLO Overview
    NLO system, 195
NLO system
    NLO Overview, 195
    Options, 195
    output table names, 213
    remote monitoring, 210
nominal variables, *see also* classification variables
NOMISS operator, 911
nonadditive errors
    MODEL procedure, 1584
nonlinear hypotheses
    Lagrange multiplier test, 1173, 1440, 1534, 1601, 2000
    Likelihood ratio test, 1440
    likelihood ratio test, 1534, 1602
    Wald test, 1440, 1534, 1601, 2000
nonlinear least-squares
    AUTOREG procedure, 391
nonlinear models
    equality restriction, 1525, 1600
    functions of parameters, 1509
    inequality restriction, 1502, 1525, 1600
    left-hand side expressions, 1674
    restricted estimation, 1502, 1525, 1600
    restricted solution, 1525
    test of hypotheses, 1533
nonmissing observations
    statistics of fit, 3909
nonseasonal ARIMA model
    notation, 3902
nonseasonal transfer function
    notation, 3903
nonstationarity, *see* stationarity
normal copula, 548, 1063
normal distribution
    COUNTREG procedure, 668
    definition of (COUNTREG), 668
    definition of (HPQLIM), 1177
    definition of (QLIM), 2026
    HPQLIM procedure, 1177
    MODEL procedure, 1508
    QLIM procedure, 2026
normality tests, 1574
    Henze-Zirkler test, 1574
    Jarque-Bera test, 372
    Kolmogorov-Smirnov test, 1574
    Mardia's test, 1574
    multivariate, 1574
    Shapiro-Wilk test, 1574
normalizing dates in intervals
    INTNX function, 124
normalizing to intervals
    date values, 124
notation
    nonseasonal ARIMA model, 3902
    nonseasonal transfer function, 3903
    seasonal ARIMA model, 3902
    seasonal transfer function, 3904
notation for
    ARIMA model, 233
    ARMA model, 234
NTC_ECON, NTC Economics
    SASEXFSD engine, 3568
number of observations
    statistics of fit, 3910
number to use
    instrumental variables, 1608
numerator factors
    transfer function model, 244

OBJECT convergence measure, 1556
objective function, 1535
observation equation, *see* measurement equation
observation numbers, 3865
    as time ID, 3671
    time ID variable, 3671
obtaining descriptive information
    DATASOURCE procedure, 702, 705, 706, 709, 725–727
ODS graph names
    ARIMA procedure, 297
    AUTOREG procedure, 437
    COPULA procedure, 564
    COUNTREG procedure, 672
    ENTROPY procedure, 833
    ESM procedure, 870
    EXPAND procedure, 920
    HPCDM procedure, 1042
    HPSEVERITY procedure, 1309
    MODEL procedure, 1638
    SEVERITY procedure, 2185

SIMILARITY procedure, 2275
SYSLIN procedure, 2590
TIMEDATA procedure, 2628
TIMESERIES procedure, 2710
UCM procedure, 2814
VARMAX procedure, 3031
X12 procedure, 3218
X13 procedure, 3218
ODS Graphics
ARIMA procedure, 250
SSM procedure, 2370
UCM procedure, 2756
X12 procedure, 3164
X13 procedure, 3164
ODS table names
HPCDM procedure, 1040
HPSEVERITY procedure, 1306
SEVERITY procedure, 2182
OECD ANA data files
DATASOURCE procedure, 752
OECD annual national accounts
DATASOURCE procedure, 752
OECD data files, *see* DATASOURCE procedure
OECD data files in FAME.db, *see* SASEFAME engine
OECD main economic indicators
DATASOURCE procedure, 753
OECD MEI data files
DATASOURCE procedure, 753
OECD QNA data files
DATASOURCE procedure, 752
OECD quarterly national accounts
DATASOURCE procedure, 752
OECD_MEI, OECD Main Economic Indicators
SASEXFSD engine, 3568
OECD_OTLK, OECD Economic Outlook
SASEXFSD engine, 3568
of a state space model
impulse response matrix, 2534
innovation vector, 2505
input matrix, 2505
measurement equation, 2505
state transition equation, 2505
state vector, 2504
transition equation, 2505
transition matrix, 2505
of a time series
unit root, 183
of interleaved time series
overlay plots, 116
of missing values
interpolation, 146, 886
of time series
distribution, 888
overlay plots, 114
sampling frequency, 98, 110, 146
simulation, 3785, 3874
stationarity, 236
summation, 136
time ranges, 104
of time series data set
standard form, 103
time series cross-sectional form, 106
of time series observations
frequency, 110, 146
periodicity, 98, 110, 146
OFFSET= option
SASEFRED engine, 3388
omitted observations
contrasted with missing values, 105
defined, 105
replacing with missing values, 127
omitted observations in
time series data, 105
one-way
first-differenced methods, 1839
fixed-effects model, 1833
random-effects model, 1841
one-way fixed-effects model, 1126, 1833
HPPANEL procedure, 1126
PANEL procedure, 1833
one-way random-effects model, 1129
HPPANEL procedure, 1129
PANEL procedure, 1841
ONS_ECON, ONS - U.K. Office for National Statistics
SASEXFSD engine, 3568
operations research
SAS/OR software, 53
optimization methods
conjugate-gradient, 1081
double-dogleg, 1081
Newton-Raphson, 594, 1081, 1153, 1420
Newton-Raphson Ridge, 1081
none, 1081
quasi-Newton, 380, 594, 1081, 1420
trust region, 380, 594, 1081, 1153, 1420
optimizations
smoothing weights, 3893
Options
NLO system, 195
options
automatic model selection, 3794
options summary
MODEL statement (PANEL), 1810
order of calculations
COMPUTAB procedure, 501
order statistics, *see* RANK procedure
ordinal discrete choice modeling
HPQLIM procedure, 1169

QLIM procedure, 1985
ordinary differential equations (ODEs)
and goal seeking, 1598
differential algebraic equations, 1671
example, 1733
explosive differential equations, 1670
limitations on, 1670
systems of, 1733
ordinary differential equations and goal seeking
MODEL procedure, 1598
Organization for Economic Cooperation and Development data files, *see* DATASOURCE procedure
DATASOURCE procedure, 751
Organization for Economic Cooperation and Development data files in FAME.db, *see* SASEFAME engine
ORIENTATION= option
SASEXFSD engine, 3567
orthogonal polynomials
PDLREG procedure, 1926
OUT= data set
indexing, 724
OUTALL= data set
DATASOURCE procedure, 706
OUTBY= data set
DATASOURCE procedure, 705
OUTCONT= data set
DATASOURCE procedure, 702, 709
Outlier Detection
ARIMA procedure, 325
output data sets
and the OUTPUT statement, 110
ARIMA procedure, 284–286, 289, 290
AUTOREG procedure, 431
BOXCOXAR macro, 181
COMPUTAB procedure, 507
DATASOURCE procedure, 701, 723, 725–727
DFTEST macro, 184
different forms of, 109
ENTROPY procedure, 831
EXPAND procedure, 918, 919
FORECAST procedure, 968, 969
HPPANEL procedure, 1134
in standard form, 110
interleaved form, 109
LOAN procedure, 1379, 1380
LOGTEST macro, 186
MODEL procedure, 1632
PANEL procedure, 1894, 1895
PDLREG procedure, 1938
produced by SAS/ETS procedures, 109
SIMLIN procedure, 2311, 2312
SPECTRA procedure, 2339
STATESPACE procedure, 2534–2536
SYSLIN procedure, 2585, 2586
VARMAX procedure, 3017
X11 procedure, 3119–3121
Output Delivery System (ODS), 3840, 3878
output ODS Graphics table names
HPQLIM procedure, 1184
QLIM procedure, 2035
OUTPUT statement
SAS/ETS procedures using, 110
output table names
ARIMA procedure, 294
AUTOREG procedure, 434
COPULA procedure, 563
COUNTREG procedure, 670
ENTROPY procedure, 832
HPCOUNTREG procedure, 1109
HPPANEL procedure, 1135
HPQLIM procedure, 1183
LOAN procedure, 1382
MDC procedure, 1442
MODEL procedure, 1635
NLO system, 213
PANEL procedure, 1897
PDLREG procedure, 1939
QLIM procedure, 2033
SIMLIN procedure, 2314
SPECTRA procedure, 2342
STATESPACE procedure, 2538
SYSLIN procedure, 2589
TSCSREG procedure, 2740
X11 procedure, 3132
OUTPUT= option
SASEFRED engine, 3388
OUTXML= option
SASEFRED engine, 3388
SASEQUAN engine, 3484
SASEXFSD engine, 3566
OUTXML= option, SAS XML data
SASEFRED engine, 3395
SASEQUAN engine, 3497
SASEXFSD engine, 3561
over identification restrictions
SYSLIN procedure, 2584
overlay plot of
time series data, 114
overlay plots
of interleaved time series, 116
of time series, 114
TYPE variable and, 116

p-values for
Durbin-Watson test, 342
panel data

TSCSREG procedure, 2729
panel GMM, 1855
GMM in panel: Arellano and Bond's estimator, 1855
PANEL procedure
Amemiya-MaCurdy Estimation, 1850
between estimators, 1840
BY groups, 1803
Da Silva method, 1853
first-differenced methods, 1839
generalized least squares, 1851
HAC, 1870
Hausman-Taylor Estimation, 1849
HCCME =, 1866
ID variables, 1807
one-way fixed-effects model, 1833
one-way random-effects model, 1841
output data sets, 1894, 1895
output table names, 1897
Parks method, 1851
pooled estimator, 1840
predicted values, 1827
printed output, 1896
R-square measure, 1872
residuals, 1827
specification tests, 1873
two-way fixed-effects model, 1834
two-way random-effects model, 1843
Zellner's two-stage method, 1852
parameter change vector, 1568
parameter estimates, 3660
parameter estimation, 3884
parameter perturbation analysis
HPCDM procedure, 1035
parameters
MODEL procedure, 1675
SSM procedure, 2373, 2375–2377, 2381, 2388
UCM procedure, 2759–2761, 2763–2766, 2768, 2770–2782
Pareto charts, 53
Parks method
PANEL procedure, 1851
partial autocorrelations
multivariate, 2526
partial autoregression coefficient
VARMAX procedure, 2912, 2959
partial canonical correlation
VARMAX procedure, 2912, 2961
partial correlation
VARMAX procedure, 2960
PASS= option
SASEXFSD engine, 3567
PDL, *see* polynomial distributed lags
PDLREG procedure
BY groups, 1931
confidence limits, 1935
distributed lag regression models, 1925
orthogonal polynomials, 1926
output data sets, 1938
output table names, 1939
polynomial distributed lags, 1926
predicted values, 1934
residuals, 1934
restricted estimation, 1935
percent change calculations
at annual rates, 133
introduced, 133
moving averages, 134
period-to-period, 133
time series data, 133, 134
year-over-year, 133
yearly averages, 134
percent change calculations and
DIF function, 133, 134
differencing, 133, 134
LAG function, 133, 134
lags, 133, 134
percent operators, 918
period of evaluation, 3734
period of fit, 3734, 3800, 3870
period-to-period
percent change calculations, 133
PERIOD= option
SASEXFSD engine, 3566
Periodic Equivalent, *see* Uniform Periodic Equivalent
periodicity
changing by interpolation, 146, 885
of time series observations, 98, 110, 146
periodicity of
time series data, 110, 146
periodicity of time series
time intervals, 110, 146
periodogram
SPECTRA procedure, 2330, 2340
PERMNO access to STK data
SASEXCCM engine, 3521
PERMNO as CRSP's primary key for STK access
SASEXCCM engine, 3523
PGARCH model, 396
AUTOREG procedure, 354
Power GARCH model, 396
Phillips-Ouliaris test, 375, 410
Phillips-Perron test, 375, 408
unit roots, 375, 408
Phillips-Perron tests, 256
physical names on supported hosts
SASEFAME engine, 3338

physical path name syntax for a variety of environments
SASEFAME engine, 3338
pitfalls of
DIF function, 130
LAG function, 130
plot axis and
time intervals, 113
plot axis for time series
SGPLOT procedure, 113
PLOT procedure, 47
plot reference lines and
time intervals, 113
plots, *see* Model Viewer, *see* Time Series Viewer
plots of
interleaved time series, 116
plotting
autocorrelations, 221
forecasts, 3663
prediction errors, 3656
residual, 117
time series data, 112
plotting time series
SGPLOT procedure, 113
Time Series Viewer procedure, 112
point
interventions, 3907
point interventions, 3907
point-in-time values, 885, 888
polynomial distributed lag models
MODEL procedure, 1625
polynomial distributed lags
Almon lag polynomials, 1925
endpoint restrictions for, 1926, 1931
PDL, 1925
PDLREG procedure, 1926
Polynomial specification, 3781, 3808, 3844
pooled estimator, 1132, 1840
HPPANEL procedure, 1132
PANEL procedure, 1840
Portfolio, *see* Investment Portfolio
Portmanteau Q test, 422
Portmanteau Q test for
Heteroscedasticity, 422
power curve
trend curves, 3906
power curve trend, 3906
Power GARCH model, *see* PGARCH model
PPC convergence measure, 1556
Prais-Winsten estimates
AUTOREG procedure, 391
PRED. variables, 1677
predetermined variables
SYSLIN procedure, 2550
predicted values
ARIMA procedure, 280
AUTOREG procedure, 383, 427, 428
conditional variance, 429
FORECAST procedure, 969
PANEL procedure, 1827
PDLREG procedure, 1934
SIMLIN procedure, 2303, 2307
STATESPACE procedure, 2531, 2535
structural, 384, 427, 1934
SYSLIN procedure, 2572
transformed models, 1586
predicting
conditional variance, 429
prediction error covariance
VARMAX procedure, 2910, 2953, 2955
prediction errors
autocorrelations, 3657
plotting, 3656
residuals, 3726
stationarity, 3658
predictions
smoothing models, 3892
predictive Chow test, 372, 424
predictive Chow tests, 1604
predictor variables
forecasting models, 3739
independent variables, 3739
inputs, 3739
interventions, 3906
seasonal dummies, 3908
specifying, 3739
trend curves, 3905
Present Value Analysis, *see* Time Value Analysis
present worth of cost
LOAN procedure, 1378
prewhitening
ARIMA procedure, 270, 271
principal component, 1567
PRINT procedure, 47
printing SAS data sets, 47
printed output
ARIMA procedure, 291
AUTOREG procedure, 433
HPPANEL procedure, 1135
PANEL procedure, 1896
SIMLIN procedure, 2313
STATESPACE procedure, 2537
SYSLIN procedure, 2587
X11 procedure, 3122
printing
SAS data sets, 47
printing SAS data sets, *see* PRINT procedure
prior distribution

distribution specification (COUNTREG), 610
distribution specification (HPQLIM), 1166
distribution specification (QLIM), 1979
probability functions, 48
probability of observability
HPSEVERITY procedure, 1241
SEVERITY procedure, 2119
PROBDF Function
Dickey-Fuller test, 187
Financial Functions, 187
significance probabilities, 187
significance probabilities for Dickey-Fuller tests, 187
PROBDF function
defined, 187
probit
HPQLIM procedure, 1144
QLIM Procedure, 1948
produced by SAS/ETS procedures
output data sets, 109
Producer Price Index Survey, *see* DATASOURCE procedure
producing
forecasts, 3631, 3845
producing forecasts, 3845
program flow
COMPUTAB procedure, 498
program listing
MODEL procedure, 1690
program variables
MODEL procedure, 1677
programming statements
COMPUTAB procedure, 496
Project Management window
forecasting project, 3637
properties of the estimates
MODEL procedure, 1552
properties of time series, 3681
PROTO procedure, 47
printing SAS data sets, 47
PROXY= option
SASEFRED engine, 3388
SASEQUAN engine, 3487
SASEXFSD engine, 3567

QGARCH model, 396
AUTOREG procedure, 354
Quadratic GARCH model, 396
QLIM Procedure, 1948
logit, 1948
probit, 1948
selection, 1948
Tobit model, 1948
QLIM procedure
bivariate limited dependent variable modeling, 1995
Box-Cox modeling, 1994
BY groups, 1971
censored regression models, 1988
frontier, 1991
gamma distribution, 2025
Gaussian distribution, 2026
heteroscedasticity, 1993
initial values, 2013
inverse-gamma distribution , 2025
limited dependent variable models, 1988
maximum a posteriori, 2013
multivariate limited dependent models, 1999
normal distribution, 2026
ordinal discrete choice modeling, 1985
output, 2027
output ODS Graphics table names, 2035
output table names, 2033
selection models, 1996
square root inverse-gamma distribution , 2026
standard distributions, 2024
syntax, 1955
t distribution, 2026
tests on parameters, 2000
truncated regression models, 1990
types of Tobit model, 1989
uniform distribution, 2027
quadratic
trend curves, 3906
Quadratic GARCH model, *see* QGARCH model
quadratic trend, 3906
quadrature spectrum
cross-spectral analysis, 2341
SPECTRA procedure, 2341
qualitative variables, *see* classification variables
Quandl data files, *see* SASEQUAN engine
Quandl economic data, *see* SASEQUAN engine
quasi-Newton
optimization methods, 380, 594, 1420
quasi-Newton method, 380, 594, 1153, 1420
AUTOREG procedure, 363
quasi-Newton optimization method, 1081
quasi-random number generators
MODEL procedure, 1651

R convergence measure, 1556
R square statistic
statistics of fit, 2820
R squared
MODEL procedure, 1554, 1560
R-square measure
PANEL procedure, 1872
R-square statistic

statistics of fit, 3910
SYSLIN procedure, 2581
ramp
interventions, 3907
ramp function, *see* ramp interventions
ramp interventions, 3907
ramp function, 3906
Ramsey's test, *see* RESET test
random number functions, 48
random walk model
AUTOREG procedure, 446
random walk R-square
statistics of fit, 2820, 3910
random-effects model
one-way, 1841
random-number functions
functions, 1681
random-number generating functions
MODEL procedure, 1681
random-walk with drift tests, 256
range of output observations
EXPAND procedure, 899
RANGE= option in the LIBNAME statement
SASEFAME engine, 3345
RANK procedure, 47
order statistics, 47
Rank Version of the von Neumann Ratio test, 417
Rank Version of the von Neumann Ratio test for
Independence, 417
Rank version of von Neumann ratio test, 377
Rank version of von Neumann ratio test for
Independence, 377
rate adjustment cases
LOAN procedure, 1373
rates
contrasted with stocks or levels, 888
ratio operators, 918
rational transfer functions
ARIMA procedure, 245
reading
time series data, 94, 148
reading data files
DATASOURCE procedure, 700
reading from a Fame database
SASEFAME engine, 3327
reading from a Haver DLX database
SASEHAVR engine, 3433
reading from CRSP data files
SASECRSP engine, 3283
reading, with DATA step
time series data, 147, 148
recommended for time series ID
formats, 98
recursive residuals, 384, 402
reduced form coefficients
SIMLIN procedure, 2309, 2313, 2317
SYSLIN procedure, 2583
reference
forecasting models, 3735
SGPLOT procedure, 113
regression model with ARMA errors
ARIMA procedure, 239, 240
regressor
definition, 619
regressor selection, 3850
regressors
forecasting models, 3747
specifying, 3747
relation to ARMA models
state space models, 2533
remote Fame access, using Fame CHLI
SASEFAME engine, 3328
remote monitoring
NLO system, 210
RENAME in the DATA step
SASEFAME engine, 3342
renaming
SAS data sets, 47
renaming variables
DATASOURCE procedure, 710, 722
replacing with missing values
omitted observations, 127
represented by different series
cross sectional dimensions, 105
represented with BY groups
cross-sectional dimensions, 106
reserved words
COMPUTAB procedure, 507
RESET test, 372
Ramsey's test, 372
RESID. variables, 1579, 1584, 1677
residual
plotting, 117
residual analysis, 3726
residuals, *see* prediction errors
ARIMA procedure, 280
AUTOREG procedure, 384
FORECAST procedure, 969
PANEL procedure, 1827
PDLREG procedure, 1934
SIMLIN procedure, 2308
STATESPACE procedure, 2535
structural, 384, 1935
SYSLIN procedure, 2572
response variable, 619
restarting the SASEFAME engine
SASEFAME engine, 3327
RESTRICT statement, 384, 817, 1525

restricted estimates
STATESPACE procedure, 2522
restricted estimation, 384
linear models, 817
nonlinear models, 1502, 1525, 1600
PDLREG procedure, 1935
SYSLIN procedure, 2573, 2574
restricted solution
nonlinear models, 1525
restricted vector autoregression, 1620
restrictions on parameters
MODEL procedure, 1620
RETAIN statement
computing lags, 131
RETAIN statement and
differencing, 131
lags, 131
Reuters data items
SASEXFSD engine, 3565
root mean square error
statistics of fit, 2820, 3910
row blocks
COMPUTAB procedure, 506
ROWS= option
SASEQUAN engine, 3487
ROWxxxxx: label
COMPUTAB procedure, 496
RPC convergence measure, 1556
RTEND= option
SASEFRED engine, 3389
RTSTART= option
SASEFRED engine, 3389
Runs test, 372, 416
Runs test for
Independence, 372, 416

S convergence measure, 1556
S matrix
definition, 1536
MODEL procedure, 1553
S matrix used in estimation, 1554
S-iterated methods
MODEL procedure, 1554
sample cross covariances
VARMAX procedure, 2910, 2958
sample cross-correlations
VARMAX procedure, 2910, 2958
sample data sets, 3608, 3620
sample FactSet data types
SASEXFSD engine, 3577
sampling frequency
changing by interpolation, 146
of time series, 98, 110, 146
time intervals and, 110
sampling frequency of
time series data, 110, 146
sampling frequency of time series
time intervals, 110, 146
SAS and CRSP dates
SASECRSP engine, 3291
SAS catalogs, *see* CATALOG procedure
SAS data sets
contents of, 47
copying, 47
DATA step, 46
moving between computer systems, 47
printing, 47
renaming, 47
sorting, 47
structured query language, 47
summarizing, 47
transposing, 47
SAS data sets and
time series data, 93
SAS DATA step
SASECRSP engine, 3283
SASEFAME engine, 3327
SASEHAVR engine, 3433
SAS date format
SASECRSP engine, 3291
SAS language features for
time series data, 92
SAS macros
BOXCOXAR macro, 180
DFPVALUE macro, 182
DFTEST macro, 183
LOGTEST macro, 185
macros, 179
SAS options statement, using
VALIDVARNAME=ANY
SASEFAME engine, 3338, 3342
SAS output data set
SASECRSP engine, 3290
SASEFAME engine, 3333
SASEFRED engine, 3394
SASEHAVR engine, 3439
SASEQUAN engine, 3491
SASEXCCM engine, 3526
SASEXFSD engine, 3578
SAS OUTXML file, SML file, XML file
SASEXFSD engine, 3579
SAS OUTXML file, XML file
SASEFRED engine, 3394
SASEQUAN engine, 3491
SAS representation for
date values, 95
datetime values, 96
SAS Risk Products, 57

SAS source statements, 3807
SAS XML data, OUTXML= option
  SASEFRED engine, 3395
  SASEQUAN engine, 3497
  SASEXFSD engine, 3561
SAS XML format
  SASEFRED engine, 3395
  SASEQUAN engine, 3497
SAS XML format, SML format
  SASEXFSD engine, 3561
SAS XML map file
  SASEFRED engine, 3394
  SASEXFSD engine, 3492, 3579
SAS XML map, MAPREF= option
  SASEXFSD engine, 3561
SAS XML map, XMLMAP= option
  SASEFRED engine, 3395
  SASEQUAN engine, 3497
  SASEXFSD engine, 3561
SAS YEARCUTOFF= option
  DATASOURCE procedure, 720
SAS/ETS procedures using
  OUTPUT statement, 110
SAS/GRAPH software, 50
  graphics, 50
SAS/HPF, 49
SAS/IML software, 51
  IML, 51
  matrix language, 51
SAS/IML Studio software, 52
SAS/OR software, 53
  operations research, 53
SAS/QC software, 53
  statistical quality control, 53
SAS/STAT software, 50
SASECRSP engine
  @CRSPDB date informats, 3292
  @CRSPDR date informats, 3292
  @CRSPDT date informats, 3292
  CONTENTS procedure, 3283
  converting dates using the CRSP date functions, 3291
  CRSP and SAS dates, 3291
  CRSP date formats, 3292
  CRSP date functions, 3291
  CRSP date informats, 3292
  CRSP integer date format, 3291
  CRSPDB_SASCAL environment variable, 3283
  CRSPDCI date functions, 3293
  CRSPDCS date functions, 3293
  CRSPDI2S date function, 3293
  CRSPDIC date functions, 3293
  CRSPDS2I date function, 3293
  CRSPDSC date functions, 3293
  CRSPDT date formats, 3292
  environment variable, CRSPDB_SASCAL, 3283
  LIBNAME statement, 3280
  reading from CRSP data files, 3283
  SAS and CRSP dates, 3291
  SAS DATA step, 3283
  SAS date format, 3291
  SAS output data set, 3290
  SETID= option, 3283
  SQL procedure, creating a view, 3283
SASEFAME engine
  CONTENTS procedure, 3327
  convert option, 3327
  creating a Fame view, 3326
  DOT as a GLUE character, 3332
  DRI data files in FAME.db , 3326
  DRI/McGraw-Hill data files in FAME.db, 3326
  DROP in the DATA step, 3342
  Fame data files, 3326
  Fame GLUE symbol named DOT, 3338
  Fame Information Services Databases, 3326
  fatal error when reading from a Fame database, 3327
  finishing the Fame CHLI, 3327
  GLUE symbol, 3332
  KEEP in the DATA step, 3342
  LIBNAME *libref* SASEHAVR '*physical name*' on Windows, 3338
  LIBNAME *libref* SASEHAVR '*physical name*' on UNIX, 3338
  LIBNAME interface engine for Fame databases, 3326
  LIBNAME statement, 3326
  main economic indicators (OECD) data files in FAME.db, 3326
  national accounts data files (OECD) in FAME.db, 3326
  OECD data files in FAME.db, 3326
  Organization for Economic Cooperation and Development data files in FAME.db, 3326
  physical names on supported hosts, 3338
  physical path name syntax for a variety of environments, 3338
  RANGE= option in the LIBNAME statement, 3345
  reading from a Fame database, 3327
  remote Fame access, using Fame CHLI, 3328
  RENAME in the DATA step, 3342
  restarting the SASEFAME engine, 3327
  SAS DATA step, 3327
  SAS options statement, using VALIDVARNAME=ANY, 3338, 3342
  SAS output data set, 3333

special characters in SAS variable names, the GLUE symbol DOT, 3338
SQL procedure, creating a view, 3327
SQL procedure, using clause, 3327
using CROSSLIST= option to create a view, 3328
using Fame expressions and Fame functions in an INSET, 3328
using INSET= option with the CROSSLIST= option to create a view, 3328
using INSET= option with the KEEP= clause to create a view, 3328
using KEEPLIST clause to create a view, 3328
using RANGE= option to create a view, 3328
using WHERE clause with INSET= option to create a view, 3328
using WILDCARD= option to create a view, 3328
VALIDVARNAME=ANY, SAS option statement, 3338, 3342
viewing a Fame database, 3326
WHERE in the DATA step, 3345
SASEFRED engine
AGG= option, 3386
APIKEY= option, 3386
AUTOMAP= option, 3386
creating a FRED view, 3381
END= option, 3386
Federal Reserve Economic Data, 3381
FORMAT= option, 3387
FRED data files, 3381
FREQ= option, 3387
IDLIST= option, 3386
LIBNAME interface engine for FRED databases, 3381
LIBNAME libref SASEFRED statement, 3386
LIMIT= option, 3388
MAPREF= option, 3388
MAPREF= option, SAS XML map, 3395
OFFSET= option, 3388
OUTPUT= option, 3388
OUTXML= option, 3388
OUTXML= option, SAS XML data, 3395
PROXY= option, 3388
RTEND= option, 3389
RTSTART= option, 3389
SAS output data set, 3394
SAS OUTXML file, XML file, 3394
SAS XML data, OUTXML= option, 3395
SAS XML format, 3395
SAS XML map file, 3394
SAS XML map, XMLMAP= option, 3395
SORT= option, 3389
START= option, 3389
UNITS= option, 3389
URL= option, 3390
viewing a FRED database, 3381
VINTAGE= option, 3391
XML format, 3395
XML map, MAPREF= option, 3395
XMLMAP= option, 3391
XMLMAP= option, SAS XML map, 3395
SASEHAVR engine
creating a Haver view, 3431
frequency option, 3433
Haver data files, 3431
Haver Information Services Databases, 3431
LIBNAME interface engine for Haver databases, 3431
LIBNAME statement, 3382, 3432, 3480
Listing the Haver selection keys, OUTSELECT=ON, 3433
reading from a Haver DLX database, 3433
SAS DATA step, 3433
SAS output data set, 3439
viewing a Haver database, 3431
SASEQUAN engine
APIKEY= option, 3483
AUTOMAP= option, 3484
COLLAPSE= option, 3484
creating a Quandl view, 3479
END= option, 3486
FORMAT= option, 3485
FREQ= option, 3486
IDLIST= option, 3486
LIBNAME interface engine for Quandl data sets, 3479
LIBNAME libref SASEQUAN statement, 3483
MAPREF= option, 3484
MAPREF= option, SAS XML map, 3497
OUTXML= option, 3484
OUTXML= option, SAS XML data, 3497
PROXY= option, 3487
Quandl data files, 3479
Quandl economic data, 3479
ROWS= option, 3487
SAS output data set, 3491
SAS OUTXML file, XML file, 3491
SAS XML data, OUTXML= option, 3497
SAS XML format, 3497
SAS XML map, XMLMAP= option, 3497
SORT= option, 3487
START= option, 3486
TRANS= option, 3487
viewing a Quandl data set, 3479
XML format, 3497
XML map, MAPREF= option, 3497
XMLMAP= option, 3484
XMLMAP= option, SAS XML map, 3497
SASEXCCM engine

CCM data, 3520
CCM data item access, 3527
Compustat's Xpressfeed data item names, 3527
daily STK data item access, 3531
DIZ201006 data, 3531, 3533
GVIIDKEY as composite key for security items, 3527
GVIIDKEY as Compustat's permanent issue identifier, 3523
GVKEY access to CCM data, 3520
GVKEY as Compustat's permanent SPC identifier, 3523
GVKEY.IID as composite key for accessing CCM Security data, 3523
IND data, 3521
IND data item access, 3533
INDNO access to IND data, 3521
INDNO as CRSP's primary key for indices access, 3524
item-handling access, items, groups, and reporting formats, 3524
keysets role in classifying and presenting CCM data, 3527
legacy data access, SASECRSP, 3520
LIBNAME statement, 3520
Linux 64-bit, 3522
missing values and Compustat's missing codes, 3526
MIZ201006 data, 3532
monthly STK data item access, 3532
PERMNO access to STK data, 3521
PERMNO as CRSP's primary key for STK access, 3523
SAS output data set, 3526
SETIDs, 3520
Solaris SPARC and Solaris X64, 3522
STK data, 3521
supported hosts, 3522
Windows, 3522
SASEXFSD engine
AUTOMAP= option, 3566
available data items, 3564
available economic data sources, 3578
available financial data sources, 3578
available other databases, 3578
CEIC, Eurostat, FactSet Sourced Economics, Global Insight, IMF, Markit, OECD, 3578
CEIC_CHINA, CEIC China, 3568
CEIC_ECON, CEIC Asia, 3568
CNS_ECON, Consensus Economics, 3568
COMPUSTAT data items, 3565
Compustat Quarterly Point-in-Time data items , 3584
COMPUSTAT/Worldscope—global data offerings, 3558
CONNECT= option, 3567
country identifiers for ExtractEconData, 3569
DATE, date ranges, relative dates, absolute dates, 3579
DATES= option, 3566
EBIT Consolidated formula libraries, 3584
EBIT data items, 3584
EBITDA Consolidated formula libraries, 3584
EBITDA data items, 3584
EIU_ECON, Economist Intelligence Unit (EIU) CountryData database, 3568
EURO_STAT, Statistical Office of the European Commission, 3568
ExtractBenchmarkDetail, 3573
ExtractDataSnapshot, 3572
ExtractEconData, 3567
ExtractFormulaHistory, 3571
ExtractOFDBItem, 3575
ExtractOFDBUniverse, 3576
ExtractScreenUniverse, 3576
FACTLET= option, 3563
FactSet data offerings, 3558
FactSet frequency codes, 3579
FactSet Fundamentals Data: items(variable names) by formula category, 3559
FactSet Fundamentals—annual data items, 3584
FactSet global indices formulas, 3565
FDS_ECON, FactSet sourced economic data, 3568
FORMAT= option, 3567
global constituents formulas, 3565
I/B/E/S data offerings, 3558
IBJ_NIKKO, INDB (I-N Information Systems) Japanese Macroeconomic Database, 3568
IDS= option, 3564
IMF_IFS, IMF International Financial Statistics, 3568
ISO currency codes, 3581
ISON= option, 3559
ITEMS= option, 3564
LIBNAME libref SASEXFSD statement, 3563
LIBNAME statement, 3558
MAPREF= option, 3566
MAPREF= option, SAS XML map, 3561
MSCI Global Index and Constituents data offerings, 3558
NTC_ECON, NTC Economics, 3568
OECD_MEI, OECD Main Economic Indicators, 3568
OECD_OTLK, OECD Economic Outlook, 3568
ONS_ECON, ONS - U.K. Office for National Statistics, 3568

ORIENTATION= option, 3567
OUTXML= option, 3566
OUTXML= option, SAS XML data, 3561
PASS= option, 3567
PERIOD= option, 3566
PROXY= option, 3567
Reuters data items, 3565
sample FactSet data types, 3577
SAS output data set, 3578
SAS OUTXML file, SML file, XML file, 3579
SAS XML data, OUTXML= option, 3561
SAS XML format, SML format, 3561
SAS XML map file, 3492, 3579
SAS XML map, MAPREF= option, 3561
SAS XML map, XMLMAP= option, 3561
SIC database, 3578
SML format, SAS XML format, 3561
some FactSet Fundamentals data items, 3564
summary of LIBNAME SASEXFSD options, 3561
supported FactSet OnDemand factlets, 3558
TCB_BCI, The Conference Board's U.S. Business Cycle Indicators, 3568
TCB_CCI, Consumer Confidence Survey, 3568
Thomson Analytics insider trading, 3578
Trucost environmental data, 3578
UNIVERSE= option, 3564
USERNAME= option, 3567
WM/Reuters, 3578
Worldscope data items, 3565
Worldscope—global financial data offerings, 3558
XMLMAP= option, 3566
XMLMAP= option, SAS XML map, 3561
SASHELP library, 3620
saving and restoring
forecasting project, 3639
Savings, 3937
SBC, *see* Schwarz Bayesian criterion, *see* Schwarz Bayesian information criterion
scale operators, 917
scale regression model
HPSEVERITY procedure, 1245
SEVERITY procedure, 2123
SCAN (Smallest Canonical) correlation method, 268
scenario analysis
HPCDM procedure, 1019
Schwarz Bayesian criterion
ARIMA procedure, 274
AUTOREG procedure, 403
SBC, 274
Schwarz Bayesian information criterion
BIC, 3910
SBC, 3910
statistics of fit, 3910
scoring functions
HPSEVERITY procedure, 1290
SEVERITY procedure, 2165
seasonal adjustment
time series data, 3086, 3153
X11 procedure, 3086, 3091
X12 procedure, 3153
X13 procedure, 3153
seasonal ARIMA model
notation, 3902
Seasonal ARIMA model options, 3853
seasonal component
X11 procedure, 3086
X12 procedure, 3153
X13 procedure, 3153
seasonal dummies, 3908
predictor variables, 3908
seasonal dummies and time trends
VARMAX procedure, 2969
seasonal dummy variables
forecasting models, 3766
specifying, 3766
seasonal exponential smoothing, 3898
smoothing models, 3898
seasonal forecasting
additive Winters method, 965
FORECAST procedure, 962, 965
WINTERS method, 962
seasonal model
ARIMA model, 238
ARIMA procedure, 238
seasonal transfer function
notation, 3904
seasonal unit root test, 270
seasonality
FORECAST procedure, 966
testing for, 183
seasonality test, 3909
seasonality tests, 3125
seasonality, testing for
DFTEST macro, 183
second difference
DIF function, 132
differencing, 132
See ordinary differential equations
differential equations, 1594
seemingly unrelated regression, 1538
cross-equation covariance matrix, 1538
joint generalized least squares, 2548
SUR estimation method, 2548
SYSLIN procedure, 2555, 2580
Zellner estimation, 2548
Seidel method

MODEL procedure, 1665
Seidel method with General Form Equations
MODEL procedure, 1665
selecting from a list
forecasting models, 3685
selection
HPQLIM procedure, 1144
QLIM Procedure, 1948
selection criterion, 3833
sequence operators, 915
serial correlation correction
AUTOREG procedure, 334
series
autocorrelations, 3723
series adjustments, 3888
series diagnostics, 3681, 3854, 3908
series selection, 3855
series transformations, 3724
set operators, 916
SETID= option
SASECRSP engine, 3283
SETIDs
SASEXCCM engine, 3520
SETMISS operator, 911
SEVERITY procedure
BY groups, 2096
censoring and truncation, 2118
defining a custom objective function, 2173
defining a custom probability distribution, 2148
empirical distribution function, 2138
ODS graph names, 2185
ODS table names, 2182
predefined distributions, 2108
predefined utility functions, 2160
probability of observability, 2119
scale regression model, 2123
scoring functions, 2165
statistics of fit, 2144
Tweedie distribution, 2110
SGMM simulated generalized method of moments, 1543
SGPLOT procedure
plot axis for time series, 113
plotting time series, 113
reference, 113
time series data, 113
Shapiro-Wilk test, 1574
normality tests, 1574
sharing
forecasting project, 3643
Shewhart control charts, 53
shifted
time intervals, 153
shifted intervals, *see* time intervals, shifted
Shin test, 374, 413
unit roots, 374, 413
SIC database
SASEXFSD engine, 3578
significance probabilities
Dickey-Fuller test, 187
PROBDF Function, 187
unit root, 187
significance probabilities for
Dickey-Fuller test, 182
significance probabilities for Dickey-Fuller tests
PROBDF Function, 187
SIMILARITY procedure
BY groups, 2244
ODS graph names, 2275
SIMLIN procedure
BY groups, 2306
dynamic models, 2302, 2303, 2309, 2323
dynamic multipliers, 2309, 2310
dynamic simulation, 2303
EST= data set, 2310
ID variables, 2307
impact multipliers, 2309, 2313
initializing lags, 2311
interim multipliers, 2305, 2309, 2312, 2313
lags, 2311
linear structural equations, 2308
multipliers, 2305, 2306, 2309, 2312, 2313
multipliers for higher order lags, 2310, 2323
output data sets, 2311, 2312
output table names, 2314
predicted values, 2303, 2307
printed output, 2313
reduced form coefficients, 2309, 2313, 2317
residuals, 2308
simulation, 2303
statistics of fit, 2314
structural equations, 2308
structural form, 2308
total multipliers, 2306, 2309, 2312, 2313
TYPE=EST data set, 2308
SIMNLIN procedure, *see* MODEL procedure
simple
data set, 3634
simple exponential smoothing, 3894
smoothing models, 3894
simulated method of moments
GMM, 1543
simulated nonlinear least squares
MODEL procedure, 1547
simulating
ARIMA model, 3785, 3874
simulating aggregate adjusted loss distribution
HPCDM procedure, 1027

simulating aggregate loss distribution
HPCDM procedure, 1020
Simulating from a Mixture of Distributions
examples, 1743
simulation
MODEL procedure, 1641
of time series, 3785, 3874
SIMLIN procedure, 2303
time series, 3785, 3874
simulation dependency analysis
MODEL procedure, 1694
simultaneous equation bias, 1537
SYSLIN procedure, 2549
single equation estimators
SYSLIN procedure, 2579
single exponential smoothing, *see* exponential smoothing
Sklar's theorem, 547, 1062
sliding spans analysis, 3110
Smallest Canonical (SCAN) correlation method, 268
SML format, SAS XML format
SASEXFSD engine, 3561
SMM, 1543
GMM, 1543
SMM simulated method of moments, 1543
smoothing equations, 3891
smoothing models, 3891
smoothing model specification, 3861, 3863
smoothing models
calculations, 3890
damped-trend exponential smoothing, 3897
double exponential smoothing, 3895
exponential smoothing, 3890
forecasting models, 3690, 3890
initializations, 3891
linear exponential smoothing, 3896
missing values, 3892
multiplicative seasonal smoothing, 3899
predictions, 3892
seasonal exponential smoothing, 3898
simple exponential smoothing, 3894
smoothing equations, 3891
smoothing state, 3891
smoothing weights, 3892
specifying, 3690
standard errors, 3893
underlying model, 3890
Winters Method, 3899, 3901
smoothing state, 3891
smoothing models, 3891
smoothing weights, 3863, 3892
additive-invertible region, 3893
boundaries, 3893
FORECAST procedure, 965
optimizations, 3893
smoothing models, 3892
specifications, 3893
weights, 3892
Solaris SPARC and Solaris X64
SASEXCCM engine, 3522
solution mode output
MODEL procedure, 1654
solution modes
MODEL procedure, 1639, 1662
SOLVE Data Sets
MODEL procedure, 1671
some FactSet Fundamentals data items
SASEXFSD engine, 3564
SORT procedure, 47
sorting, 47
SORT= option
SASEFRED engine, 3389
SASEQUAN engine, 3487
sorting, *see* SORT procedure
forecasting models, 3732, 3805
SAS data sets, 47
time series data, 99
sorting by
ID variables, 99
special characters in SAS variable names, the GLUE symbol DOT
SASEFAME engine, 3338
specification tests
HPPANEL procedure, 1133
PANEL procedure, 1873
specifications
smoothing weights, 3893
specifying
adjustments, 3749
ARIMA models, 3693
combination models, 3710
custom models, 3700
dynamic regression, 3750
Factored ARIMA models, 3697
forecasting models, 3681
interventions, 3755
level shifts, 3760
predictor variables, 3739
regressors, 3747
seasonal dummy variables, 3766
smoothing models, 3690
state space models, 2514
time ID variable, 3869
trend changes, 3757
trend curves, 3743
SPECTRA procedure
BY groups, 2334
Chirp-Z algorithm, 2336

coherency of cross-spectrum, 2341
cospectrum estimate, 2340
cross-periodogram, 2340
cross-spectral analysis, 2329, 2330, 2340, 2341
cross-spectrum, 2341
fast Fourier transform, 2336
finite Fourier transform, 2330
Fourier coefficients, 2340
Fourier transform, 2330
frequency, 2339
kernels, 2336
output data sets, 2339
output table names, 2342
periodogram, 2330, 2340
quadrature spectrum, 2341
spectral analysis, 2329
spectral density estimate, 2329, 2340
spectral window, 2335
white noise test, 2339, 2341
spectral analysis
SPECTRA procedure, 2329
spectral density estimate
SPECTRA procedure, 2329, 2340
spectral window
SPECTRA procedure, 2335
SPLINE method
EXPAND procedure, 902
splitting series
time series data, 140
splitting time series data sets, 140
SQL procedure, 47
structured query language, 47
SQL procedure, creating a view
SASECRSP engine, 3283
SASEFAME engine, 3327
SQL procedure, using clause
SASEFAME engine, 3327
square root
transformations, 3889
square root gamma distribution
definition of (QLIM), 2025
square root inverse-gamma distribution
definition of (QLIM), 2026
QLIM procedure, 2026
square root transformation, *see* transformations
SSM procedure
BY groups, 2372
graph names, 2419
ODS graph names, 2419
ODS Graphics, 2370
ODS table names, 2417
parameters, 2373, 2375–2377, 2381, 2388
syntax, 2367
table names, 2417
time intervals, 2376
stable seasonality test, 3125
standard distributions
COUNTREG procedure, 667
HPQLIM procedure, 1175
QLIM procedure, 2024
standard errors
smoothing models, 3893
standard form
of time series data set, 103
standard form of
time series data, 103
STANDARD procedure, 47
standardized values, 47
standardized values, *see* STANDARD procedure
START= option
SASEFRED engine, 3389
SASEQUAN engine, 3486
starting dates of
time intervals, 124
starting values
GARCH model, 363
MODEL procedure, 1558, 1564
state and area employment, hours, and earnings survey, *see* DATASOURCE procedure
state space model
UCM procedure, 2788
state space models
form of, 2505
relation to ARMA models, 2533
specifying, 2514
state vector of, 2504
STATESPACE procedure, 2504
state space representation
VARMAX procedure, 2937
state transition equation
of a state space model, 2505
state vector
of a state space model, 2504
state vector of
state space models, 2504
STATESPACE procedure
automatic forecasting, 2504
BY groups, 2520
canonical correlation analysis, 2506, 2527
confidence limits, 2535
differencing, 2522
forecasting, 2504, 2531
ID variables, 2521
Kalman filter, 2506
multivariate forecasting, 2504
multivariate time series, 2504
output data sets, 2534–2536
output table names, 2538

predicted values, 2531, 2535
printed output, 2537
residuals, 2535
restricted estimates, 2522
state space models, 2504
time intervals, 2520
Yule-Walker equations, 2525
static simulation, 1592
MODEL procedure, 1592
static simulations
MODEL procedure, 1639
stationarity
and state space models, 2507
ARIMA procedure, 222
nonstationarity, 222
of time series, 236
prediction errors, 3658
testing for, 183
VARMAX procedure, 2964, 2972
stationarity tests, 255, 270, 373
stationarity, testing for
DFTEST macro, 183
statistical quality control
SAS/QC software, 53
statistics of fit, 2820, 3652, 3661, 3864, 3909
adjusted R-square, 2820, 3910
Akaike's information criterion, 3910
Amemiya's prediction criterion, 3911
Amemiya's R-square, 2820, 3910
corrected sum of squares, 3910
error sum of squares, 3910
goodness of fit, 3661
goodness-of-fit statistics, 2820, 3909
HPSEVERITY procedure, 1266
mean absolute error, 3910
mean absolute percent error, 2820, 3910
mean percent error, 3911
mean prediction error, 3911
mean square error, 2820
mean squared error, 3910
nonmissing observations, 3909
number of observations, 3910
R square statistic, 2820
R-square statistic, 3910
random walk R-square, 2820, 3910
root mean square error, 2820, 3910
Schwarz Bayesian information criterion, 3910
SEVERITY procedure, 2144
SIMLIN procedure, 2314
uncorrected sum of squares, 3910
step
interventions, 3907
step function, *see* step interventions
interpolation of time series, 903
intervention model and, 243
step interventions, 3907
step function, 3906
STEP method
EXPAND procedure, 903
STEPAR method
FORECAST procedure, 959
stepwise autoregression
AUTOREG procedure, 345
FORECAST procedure, 937, 959
STK data
SASEXCCM engine, 3521
stochastic simulation
MODEL procedure, 1642
stock data files, *see* DATASOURCE procedure
stocks
contrasted with flow variables, 888
stored in SAS data sets
time series data, 102
storing programs
MODEL procedure, 1688
structural
predicted values, 384, 427, 1934
residuals, 384, 1935
Structural Change
BP test for, 365
structural change
Chow test for, 368
structural equations
SIMLIN procedure, 2308
structural form
SIMLIN procedure, 2308
structural predictions
AUTOREG procedure, 427
structured query language, *see* SQL procedure
SAS data sets, 47
Student's *t* copula, 549
Student's t copula, 1064
subset model
ARIMA model, 238
ARIMA procedure, 238
AUTOREG procedure, 346
subsetting data, *see* WHERE statement
subsetting data files
DATASOURCE procedure, 701, 713
summarizing
SAS data sets, 47
summary of
time intervals, 155
summary of LIBNAME SASEXFSD options
SASEXFSD engine, 3561
summary statistics
MODEL procedure, 1657
summation

higher order sums, 137
multiperiod lags and, 136, 137
of time series, 136
summation of
time series data, 136, 137
supported FactSet OnDemand factlets
SASEXFSD engine, 3558
supported hosts
SASEXCCM engine, 3522
SUR estimation method, *see* seemingly unrelated regression
Switching Regression example
examples, 1740
syntax for
date values, 96
datetime values, 96
time intervals, 111
time values, 96
SYSLIN procedure
Basmann test, 2571, 2584
BY groups, 2569
endogenous variables, 2550
exogenous variables, 2550
full information maximum likelihood, 2557, 2580
Fuller's modification to LIML, 2584
instrumental variables, 2550
iterated seemingly unrelated regression, 2580
iterated three-stage least squares, 2580
jointly dependent variables, 2550
K-class estimation, 2579
lagged endogenous variables, 2550
limited information maximum likelihood, 2579
minimum expected loss estimator, 2579
ODS graph names, 2590
output data sets, 2585, 2586
output table names, 2589
over identification restrictions, 2584
predetermined variables, 2550
predicted values, 2572
printed output, 2587
R-square statistic, 2581
reduced form coefficients, 2583
residuals, 2572
restricted estimation, 2573, 2574
seemingly unrelated regression, 2555, 2580
simultaneous equation bias, 2549
single equation estimators, 2579
system weighted MSE, 2582
system weighted R-square, 2581, 2587
tests of hypothesis, 2575, 2576
three-stage least squares, 2555, 2580
two-stage least squares, 2553, 2579
SYSNLIN procedure, *see* MODEL procedure
system weighted MSE
SYSLIN procedure, 2582
system weighted R-square
SYSLIN procedure, 2581, 2587
systems of
ordinary differential equations (ODEs), 1733
systems of differential equations
examples, 1733
systems of ordinary differential equations
MODEL procedure, 1733

t distribution
GARCH model, 397
t distribution
COUNTREG procedure, 669
definition of (COUNTREG), 669
definition of (HPQLIM), 1177
definition of (QLIM), 2026
HPQLIM procedure, 1177
QLIM procedure, 2026
table cells, direct access to
COMPUTAB procedure, 506
table names
SSM procedure, 2417
UCM procedure, 2811
TABULATE procedure, 47
tabulating data, 47
tabulating data, *see* TABULATE procedure
taxes
LOAN procedure, 1378
TCB_BCI, The Conference Board's U.S. Business Cycle Indicators
SASEXFSD engine, 3568
TCB_CCI, Consumer Confidence Survey
SASEXFSD engine, 3568
tentative order selection
VARMAX procedure, 2958
test of hypotheses
nonlinear models, 1533
TEST statement, 385
testing for
heteroscedasticity, 347
seasonality, 183
stationarity, 183
unit root, 183
testing order of
differencing, 183
testing overidentifying restrictions, 1543
tests of hypothesis
SYSLIN procedure, 2575, 2576
tests of parameters, 385, 817, 1533
tests on parameters
MODEL procedure, 1601
TGARCH model, 396
AUTOREG procedure, 354

Threshold GARCH model, 396
The D-method
example, 1740
Thomson Analytics insider trading
SASEXFSD engine, 3578
three-stage least squares, 1539
3SLS estimation method, 2548
SYSLIN procedure, 2555, 2580
Threshold GARCH model, *see* TGARCH model
time functions, 119
time ID creation, 3865–3868
time ID variable, 3617
creating, 3667
ID variable, 3617
observation numbers, 3671
specifying, 3869
time intervals, 3622
alignment of, 154
ARIMA procedure, 282
calendar calculations and, 128
ceiling of, 125
checking data periodicity, 127
counting, 123, 126
data frequency, 3612
date values, 152
datetime values, 152
ending dates of, 124
examples of, 158
EXPAND procedure, 898
EXPAND procedure and, 146
FORECAST procedure, 958
frequency of data, 3612
functions, 171
functions for, 122, 171
ID values for, 124
incrementing dates by, 122, 123
INTCK function and, 123, 126
INTERVAL= option and, 110
intervals, 111
INTNX function and, 122
midpoint dates of, 124
naming, 111, 152
periodicity of time series, 110, 146
plot axis and, 113
plot reference lines and, 113
sampling frequency of time series, 110, 146
shifted, 153
SSM procedure, 2376
starting dates of, 124
STATESPACE procedure, 2520
summary of, 155
syntax for, 111
UCM procedure, 2769
use with SAS/ETS procedures, 111
VARMAX procedure, 2903
widths of, 125, 898
time intervals and
calendar calculations, 128
date values, 124
frequency, 110, 146
sampling frequency, 110
time intervals, functions
interval functions, 122
time intervals, shifted
shifted intervals, 153
time range
DATASOURCE procedure, 719
time range of data
DATASOURCE procedure, 704
time ranges, 3734, 3800, 3870
of time series, 104
time ranges of
time series data, 104
time series
definition, 3608
diagnostic tests, 3681
in SAS data sets, 3608
simulation, 3785, 3874
time series cross sectional form
TSCSREG procedure and, 106
time series cross-sectional form
BY groups and, 106
ID variables for, 106
of time series data set, 106
TSCSREG procedure and, 2729
time series cross-sectional form of
time series data set, 106
time series data
aggregation of, 885, 888
changing periodicity, 146, 885
converting frequency of, 885
differencing, 129–134
distribution of, 888
embedded missing values in, 104
giving dates to, 95
ID variable for, 95
interpolation, 146
interpolation of, 145, 146, 886
lagging, 129–134
leads, 135
merging series, 141
missing values, 886
missing values and, 104
omitted observations in, 105
overlay plot of, 114
percent change calculations, 133, 134
periodicity of, 110, 146
plotting, 112

reading, 94, 148
reading, with DATA step, 147, 148
sampling frequency of, 110, 146
SAS data sets and, 93
SAS language features for, 92
seasonal adjustment, 3086, 3153
SGPLOT procedure, 113
sorting, 99
splitting series, 140
standard form of, 103
stored in SAS data sets, 102
summation of, 136, 137
time ranges of, 104
Time Series Viewer, 112
transformation of, 890, 904
transposing, 141, 142
time series data and
missing values, 104
time series data set
interleaved form of, 107
time series cross-sectional form of, 106
time series forecasting, 3872
Time Series Forecasting System
invoking, 3772
invoking from SAS/AF and SAS/EIS applications, 3772
running in unattended mode, 3772
time series methods
FORECAST procedure, 949
time series variables
DATASOURCE procedure, 701, 724
Time Series Viewer, 3646, 3719, 3876
graphs, 3646
invoking, 3771
plots, 3646
saving graphs and tables, 3851, 3852
time series data, 112
Time Series Viewer procedure
plotting time series, 112
time trend models
FORECAST procedure, 947
Time Value Analysis, 3984
time values
defined, 96
formats, 170
functions, 171
informats, 164
syntax for, 96
time variable, 1598
MODEL procedure, 1598
time variables
computing from datetime values, 121
introduced, 119
TIMEDATA procedure
BY groups, 2612
ODS graph names, 2628
TIMEPLOT procedure, 47
TIMESERIES procedure
BY groups, 2669
ODS graph names, 2710
to higher frequency
interpolation, 146
to lower frequency
interpolation, 146
to SAS/ETS software
menu interfaces, 43, 44
to standard form
transposing time series, 141, 142
Tobit
HPQLIM procedure, 1144
Tobit model
QLIM Procedure, 1948
Toeplitz matrix
AUTOREG procedure, 388
total multipliers
SIMLIN procedure, 2306, 2309, 2312, 2313
trading-day component
X11 procedure, 3086, 3091
TRANS= option
SASEQUAN engine, 3487
transfer function model
ARIMA procedure, 239, 243, 274
denominator factors, 244
numerator factors, 244
transfer functions, 3903
forecasting models, 3903
transformation and estimation, 1847
transformation and estimation, 1843
transformation of
time series data, 890, 904
transformation of time series
EXPAND procedure, 890, 904
transformations, 3859
Box Cox, 3889
Box Cox transformation, 3889
log, 3889
log transformation, 3889
logistic, 3889
square root, 3889
square root transformation, 3889
transformed models
predicted values, 1586
transition equation
of a state space model, 2505
transition matrix
of a state space model, 2505
TRANSPOSE procedure, 47, 141, 142, 144, 148
transposing SAS data sets, 47

TRANSPOSE procedure and
transposing time series, 141
transposing
SAS data sets, 47
time series data, 141, 142
transposing SAS data sets, *see* TRANSPOSE procedure
transposing time series
cross-sectional dimensions, 142
from interleaved form, 141
from standard form, 144
to standard form, 141, 142
TRANSPOSE procedure and, 141
trend changes
specifying, 3757
trend curves, 3905
cubic, 3906
exponential, 3906
forecasting models, 3743
hyperbolic, 3906
linear, 3906
logarithmic, 3906
logistic, 3906
power curve, 3906
predictor variables, 3905
quadratic, 3906
specifying, 3743
trend cycle component
X11 procedure, 3086, 3091
trend test, 3909
TRIM operator, 911
TRIMLEFT operator, 911
TRIMRIGHT operator, 911
triple exponential smoothing, *see* exponential smoothing
troubleshooting estimation convergence problems
MODEL procedure, 1558
troubleshooting simulation problems
MODEL procedure, 1666
Trucost environmental data
SASEXFSD engine, 3578
true interest rate
LOAN procedure, 1378
truncated regression models
HPQLIM procedure, 1170
QLIM procedure, 1990
trust region
optimization methods, 380, 594, 1153, 1420
trust region method, 380, 594, 1153, 1420
AUTOREG procedure, 363
trust region optimization method, 1081
TSCSREG procedure
BY groups, 2736
estimation techniques, 2732
ID variables, 2737
output table names, 2740
panel data, 2729
TSCSREG procedure and
time series cross sectional form, 106
time series cross-sectional form, 2729
TSVIEW command, 3771
Turnbull's EDF estimator
HPSEVERITY procedure, 1263
SEVERITY procedure, 2140
Turning Point test, 377, 416
Turning Point test for
Independence, 377, 416
Tweedie distribution
HPSEVERITY procedure, 1232
SEVERITY procedure, 2110
two-stage least squares, 1537
2SLS estimation method, 2548
SYSLIN procedure, 2553, 2579
two-step full transform method
AUTOREG procedure, 391
two-way fixed-effects model, 1127, 1834
HPPANEL procedure, 1127
PANEL procedure, 1834
two-way random-effects model, 1130, 1843
HPPANEL procedure, 1130
PANEL procedure, 1843
type of input data file
DATASOURCE procedure, 714
TYPE variable
and interleaved time series, 107, 108
overlay plots, 116
TYPE=EST data set
SIMLIN procedure, 2308
types of loans
LOAN procedure, 1356
types of Tobit model
QLIM procedure, 1989

U.S. Bureau of Economic Analysis data files
DATASOURCE procedure, 732
U.S. Bureau of Labor Statistics data files
DATASOURCE procedure, 733
UCM procedure
BY groups, 2762
ODS graph names, 2814
ODS Graphics, 2756
ODS table names, 2811
parameters, 2759–2761, 2763–2766, 2768, 2770–2782
state space model, 2788
Statistical Graphics, 2801
syntax, 2753
table names, 2811

time intervals, 2769
unattended mode, 3772
unconditional forecasts
ARIMA procedure, 281
unconditional least squares
AR initial conditions, 1613
MA Initial Conditions, 1614
uncorrected sum of squares
statistics of fit, 3910
underlying model
smoothing models, 3890
uniform distribution
COUNTREG procedure, 669
definition of (COUNTREG), 669
definition of (HPQLIM), 1178
definition of (QLIM), 2027
HPQLIM procedure, 1178
QLIM procedure, 2027
Uniform Periodic Equivalent, 3988
unit root
Dickey-Fuller test, 187
of a time series, 183
significance probabilities, 187
testing for, 183
unit roots
KPSS test, 374, 413
Phillips-Perron test, 375, 408
Shin test, 374, 413
UNITS= option
SASEFRED engine, 3389
univariate autoregression, 1615
univariate model diagnostic checks
VARMAX procedure, 2983
univariate moving average models, 1621
UNIVARIATE procedure, 47, 1738
descriptive statistics, 47
UNIVERSE= option
SASEXFSD engine, 3564
unlinking viewer windows, 3723
unrestricted vector autoregression, 1618
URL= option
SASEFRED engine, 3390
use with SAS/ETS procedures
time intervals, 111
used for state space modeling
Kalman filter, 2506
used to select state space models
Akaike information criterion, 2526
vector autoregressive models, 2524
Yule-Walker estimates, 2524
user-defined regression variables, 3208
USERNAME= option
SASEXFSD engine, 3567
using CROSSLIST= option to create a view
SASEFAME engine, 3328
using Fame expressions and Fame functions in an INSET
SASEFAME engine, 3328
using INSET= option with the CROSSLIST= option to create a view
SASEFAME engine, 3328
using INSET= option with the KEEP= clause to create a view
SASEFAME engine, 3328
using KEEPLIST clause to create a view
SASEFAME engine, 3328
using models to forecast
MODEL procedure, 1641
using RANGE= option to create a view
SASEFAME engine, 3328
using solution modes
MODEL procedure, 1639
using WHERE clause with INSET= option to create a view
SASEFAME engine, 3328
using WILDCARD= option to create a view
SASEFAME engine, 3328

V matrix
Generalized Method of Moments, 1540, 1545
VALIDVARNAME=ANY, SAS option statement
SASEFAME engine, 3338, 3342
variable list
DATASOURCE procedure, 722
variables in model program
MODEL procedure, 1673
variance components
Fuller Battese, 1841
Nerlove, 1843
Wallace Hussain, 1842
Wansbeek Kapteyn, 1842
VARMAX procedure
Akaike's information criterion, 2981
asymptotic distribution of impulse response functions, 2965, 2975
asymptotic distribution of the parameter estimation, 2975
Bayesian vector autoregressive models, 2918, 2970
cointegration, 2983
cointegration testing, 2915, 2988
common trends, 2984
common trends testing, 2917, 2985
computational details, 3032
confidence limits, 3018
convergence problems, 3032
covariance stationarity, 3013
CPU requirements, 3033

decomposition of prediction error covariance, 2911, 2956
Dickey-Fuller test, 2915
differencing, 2907
dynamic simultaneous equation models, 2940
example of Bayesian VAR modeling, 2865
example of Bayesian VECM modeling, 2872
example of causality testing, 2878
example of cointegration testing, 2868
example of multivariate GARCH modeling, 3014
example of restricted parameter estimation and testing, 2876
example of VAR modeling, 2858
example of VARMA modeling, 2976
example of vector autoregressive modeling with exogenous variables, 2873
example of vector error correction modeling, 2866
examples of multivariate GARCH modeling, 2879
forecasting, 2953
forecasting of Bayesian vector autoregressive models, 2971
Granger causality test, 2966
impulse response function, 2911, 2943
infinite order AR representation, 2911
infinite order MA representation, 2911, 2943
invertibility, 2972
long-run relations testing, 2996
memory requirements, 3033
minimum information criteria method, 2962
missing values, 2936
multivariate GARCH modeling, 2902
multivariate model diagnostic checks, 2981
ODS graph names, 3031
output data sets, 3017
partial autoregression coefficient, 2912, 2959
partial canonical correlation, 2912, 2961
partial correlation, 2960
prediction error covariance, 2910, 2953, 2955
sample cross covariances, 2910, 2958
sample cross-correlations, 2910, 2958
seasonal dummies and time trends, 2969
state space representation, 2937
stationarity, 2964, 2972
tentative order selection, 2958
time intervals, 2903
univariate model diagnostic checks, 2983
vector autoregressive models, 2963
vector autoregressive models with exogenous variables , 2967
vector autoregressive moving-average models, 2936, 2972
vector error correction models, 2920, 2986
weak exogeneity testing, 2998
Yule-Walker estimates, 2913
vector autoregressive models, 1620
used to select state space models, 2524
VARMAX procedure, 2963
vector autoregressive models with exogenous variables
VARMAX procedure, 2967
vector autoregressive moving-average models
VARMAX procedure, 2936, 2972
vector error correction models
VARMAX procedure, 2920, 2986
vector moving average models, 1623
viewing a Fame database, *see* SASEFAME engine
viewing a FRED database, *see* SASEFRED engine
viewing a Haver database, *see* SASEHAVR engine
viewing a Quandl data set, *see* SASEQUAN engine
viewing time series, 3646
VINTAGE= option
SASEFRED engine, 3391

Wald test
linear hypotheses, 818
nonlinear hypotheses, 1440, 1534, 1601, 2000
Wallace Hussain
variance components, 1842
Wansbeek Kapteyn
variance components, 1842
weak exogeneity testing
VARMAX procedure, 2998
WEIGHT variable
MODEL procedure, 1577
weights, *see* smoothing weights
WHERE in the DATA step
SASEFAME engine, 3345
WHERE statement
subsetting data, 48
white noise test
SPECTRA procedure, 2339, 2341
white noise test of the residuals, 258
white noise test of the series, 256
White's test, 1576
heteroscedasticity tests, 1576
widths of
time intervals, 125, 898
Windows
SASEXCCM engine, 3522
WINTERS method
seasonal forecasting, 962
Winters Method, 3899, 3901
Holt-Winters Method, 3899
smoothing models, 3899, 3901
Winters method
FORECAST procedure, 937, 962
Holt-Winters method, 965
WM/Reuters

SASEXFSD engine, 3578
Wong and Li's test, 423
Wong and Li's test for
Heteroscedasticity, 423
Worldscope data items
SASEXFSD engine, 3565
Worldscope—global financial data offerings
SASEXFSD engine, 3558

X-11 ARIMA methodology
X11 procedure, 3108
X-11 seasonal adjustment method, *see* X11 procedure
X-11-ARIMA seasonal adjustment method, *see* X11 procedure
X-12 seasonal adjustment method, *see* X12 procedure
X-12-ARIMA seasonal adjustment method, *see* X12 procedure
X-13 seasonal adjustment method, *see* X13 procedure
X-13ARIMA-SEATS seasonal adjustment method, *see* X13 procedure
X11 procedure
BY groups, 3097
Census X-11 method, 3086
Census X-11 methodology, 3109
data requirements, 3114
differences with X11ARIMA/88, 3108
ID variables, 3097, 3099
irregular component, 3086, 3091
model selection for X-11-ARIMA method, 3117
output data sets, 3119–3121
output table names, 3132
printed output, 3122
seasonal adjustment, 3086, 3091
seasonal component, 3086
trading-day component, 3086, 3091
trend cycle component, 3086, 3091
X-11 ARIMA methodology, 3108
X-11 seasonal adjustment method, 3086
X-11-ARIMA seasonal adjustment method, 3086
X12 procedure
BY groups, 3174
Census X-12 method, 3149, 3152
data requirements, 3204
ID variables, 3180
INPUT variables, 3182
ODS graph names, 3218
ODS Graphics, 3164
overview, 3149
seasonal adjustment, 3153
seasonal component, 3153
X-12 seasonal adjustment method, 3152
X-12-ARIMA seasonal adjustment method, 3152
X13 procedure
BY groups, 3174
Census X-13 method, 3152
data requirements, 3204
ID variables, 3180
INPUT variables, 3182
ODS graph names, 3218
ODS Graphics, 3164
seasonal adjustment, 3153
seasonal component, 3153
X-13 seasonal adjustment method, 3152
X-13ARIMA-SEATS seasonal adjustment method, 3152
XML format
SASEFRED engine, 3395
SASEQUAN engine, 3497
XML map, MAPREF= option
SASEFRED engine, 3395
SASEQUAN engine, 3497
XMLMAP= option
SASEFRED engine, 3391
SASEQUAN engine, 3484
SASEXFSD engine, 3566
XMLMAP= option, SAS XML map
SASEFRED engine, 3395
SASEQUAN engine, 3497
SASEXFSD engine, 3561

year-over-year
percent change calculations, 133
yearly averages
percent change calculations, 134
Yule-Walker
AR initial conditions, 1613
Yule-Walker equations
AUTOREG procedure, 388
STATESPACE procedure, 2525
Yule-Walker estimates
AUTOREG procedure, 388
used to select state space models, 2524
VARMAX procedure, 2913
Yule-Walker method as
generalized least-squares, 391

Zellner estimation, *see* seemingly unrelated regression
Zellner's two-stage method
PANEL procedure, 1852
zooming graphs, 3721

Syntax Index

2SLS option
FIT statement (MODEL), 1514, 1537
PROC SYSLIN statement, 2568
3SLS option
FIT statement (MODEL), 1514, 1539, 1633
PROC SYSLIN statement, 2568

A option
PROC SPECTRA statement, 2333
A= option
FIXED statement (LOAN), 1368
ABORT, 1687
ABS function, 1680
ACCEPTDEFAULT option
AUTOMDL statement (X12), 3170
AUTOMDL statement (X13), 3170
ACCUMULATE= option
FORECAST statement (ESM), 855
ID statement (ESM), 857
ID statement (SIMILARITY), 2245
ID statement (TIMEDATA), 2613
ID statement (TIMESERIES), 2675
INPUT statement (SIMILARITY), 2248
TARGET statement (SIMILARITY), 2250
VAR statement (TIMEDATA), 2616
VAR statement (TIMESERIES), 2685
ACCURACY= option
BAYES statement (QLIM), 1964
ACCURACY=option
BAYES statement (COUNTREG), 595
ADDITIVE option
MONTHLY statement (X11), 3098
QUARTERLY statement (X11), 3103
ADDMAXIT= option
MODEL statement (MDC), 1418
ADDRANDOM option
MODEL statement (MDC), 1418
ADDVALUE option
MODEL statement (MDC), 1418
ADF= option
ARM statement (LOAN), 1372
ADJMEAN option
PROC SPECTRA statement, 2333
ADJSAMPLEVAR= option
OUTPUT statement (HPCDM), 1015
ADJSMMV option
FIT statement (MODEL), 1512
ADJUST statement
X12 procedure, 3169
X13 procedure, 3169
ADJUSTEDSEVERITY= option
PROC HPCDM statement, 1008
ADJUSTFREQ= option
ARM statement (LOAN), 1372
ADJUSTMEAN
SPECTRA statement (TIMESERIES), 2680
SSA statement (TIMESERIES), 2681
AGG= option
LIBNAME statement (SASEFRED), 3386
AGGMODE=RELAXED option
LIBNAME statement (SASEHAVR), 3437
AGGMODE=STRICT option
LIBNAME statement (SASEHAVR), 3437
AGGREGATION= option
BAYES statement (COUNTREG), 595
BAYES statement (QLIM), 1964
AICTEST= option
REGRESSION statement (X12), 3189
REGRESSION statement (X13), 3189
ALIGN= option
FORECAST statement (ARIMA), 171, 262
ID statement (ENG), 171
ID statement (ESM), 171, 858
ID statement (HPF), 171
ID statement (HPFDIAGNOSE), 171
ID statement (HPFEVENTS), 171
ID statement (SIMILARITY), 171, 2246
ID statement (TIMEDATA), 2614
ID statement (TIMESERIES), 171, 2676
ID statement (UCM), 171, 2769
ID statement (VARMAX), 171, 2903
PROC DATASOURCE statement, 171, 714
PROC EXPAND statement, 171, 892, 898
PROC FORECAST statement, 171, 953
TIMEID procedure, 2640
ALL option
COMPARE statement (LOAN), 1375
MODEL statement (AUTOREG), 363
MODEL statement (MDC), 1419
MODEL statement (PDLREG), 1932
MODEL statement (SYSLIN), 2571
PLOTS option, PROC COUNTREG statement, 593
PROC SYSLIN statement, 2569
SHOW statement (COUNTREG), 614
TEST statement (COUNTREG), 617

TEST statement (ENTROPY), 818
TEST statement (HPCOUNTREG), 1089
TEST statement (HPPANEL), 1124
TEST statement (HPQLIM), 1168
TEST statement (MDC), 1425
TEST statement (MODEL), 1534
TEST statement (PANEL), 1828
TEST statement (QLIM), 1983
ALMOST= option
OUTLIER statement (X12), 3184
OUTLIER statement (X13), 3184
ALPHA option
SPECTRA statement (TIMESERIES), 2680
ALPHA= option
COUNT statement (TIMESERIES), 2671
FORECAST statement (ARIMA), 262
FORECAST statement (ESM), 855
FORECAST statement (UCM), 2767
FORECAST statement (X12), 3179
FORECAST statement (X13), 3179
IDENTIFY statement (ARIMA), 253
MODEL statement (SYSLIN), 2571
OUTLIER statement (ARIMA), 261
OUTLIER statement (X12), 3184
OUTLIER statement (X13), 3184
OUTPUT statement (VARMAX), 2921
PROC FORECAST statement, 953
PROC SYSLIN statement, 2568
ALPHA=option
OUTLIER statement (UCM), 2774
ALPHACLI= option
OUTPUT statement (AUTOREG), 382
OUTPUT statement (PDLREG), 1934
ALPHACLM= option
OUTPUT statement (AUTOREG), 382
OUTPUT statement (PDLREG), 1934
ALPHACSM= option
OUTPUT statement (AUTOREG), 382
ALTPARM option
ESTIMATE statement (ARIMA), 257, 277
ALTW option
PROC SPECTRA statement, 2333
AMACURDY option
MODEL statement (PANEL), 1813
AMOUNT= option
FIXED statement (LOAN), 1368
AMOUNTPCT= option
FIXED statement (LOAN), 1368
ANALYZEDEP option
SOLVE statement, 1499
AOCV= option
OUTLIER statement (X12), 3184
OUTLIER statement (X13), 3184
APCT= option
FIXED statement (LOAN), 1368
APIKEY= option
LIBNAME statement (SASEFRED), 3386
LIBNAME statement (SASEQUAN), 3483
%AR macro, 1619, 1620
AR option
IRREGULAR statement (UCM), 2771
AR= option
BOXCOXAR macro, 181
DFTEST macro, 184
ESTIMATE statement (ARIMA), 259
LOGTEST macro, 186
PROC FORECAST statement, 953
ARCHTEST option
MODEL statement (AUTOREG), 363
ARCOS function, 1680
ARIMA procedure, 247
syntax, 247
ARIMA procedure, PROC ARIMA statement
PLOT option, 250
ARIMA statement
X11 procedure, 3094
X12 procedure, 3169
X13 procedure, 3169
ARM statement
LOAN procedure, 1372
ARMACV= option
AUTOMDL statement (X12), 3170
AUTOMDL statement (X13), 3170
ARMAX= option
PROC STATESPACE statement, 2518
ARSIN function, 1680
ARTEST= option
MODEL statement (PANEL), 1815
ASCII option
PROC DATASOURCE statement, 714
ASTART= option
PROC FORECAST statement, 953
AT= option
COMPARE statement (LOAN), 1375
ATAN function, 1680
ATOL= option
MODEL statement (PANEL), 1815
ATTEMPTS= number
BAYES statement (QLIM), 1964, 1965
ATTEMPTS=number
BAYES statement (COUNTREG), 595, 596
ATTRIBUTE statement
DATASOURCE procedure, 720
AUTOMAP= option
LIBNAME statement (SASEFRED), 3386
LIBNAME statement (SASEQUAN), 3484
LIBNAME statement(SASEXFSD), 3566
AUTOMCMC option

BAYES statement (COUNTREG), 595
BAYES statement (QLIM), 1964
AUTOMDL statement
X12 procedure, 3170
X13 procedure, 3170
AUTOREG procedure, 355
syntax, 355
AUTOREG procedure, AUTOREG statement, 360
AUTOREG statement
UCM procedure, 2759
AUXDATA= option
PROC TIMEDATA statement, 2610
PROC X12 statement, 3162
PROC X13 statement, 3162

B option
ARM statement (LOAN), 1373
BACK= option
ESTIMATE statement (UCM), 2765
FORECAST statement (ARIMA), 262
FORECAST statement (UCM), 2767
OUTPUT statement (VARMAX), 2921
PROC ESM statement, 852
PROC STATESPACE statement, 2520
BACKCAST= option
ARIMA statement (X11), 3094
BACKLIM= option
ESTIMATE statement (ARIMA), 260
BACKSTEP option
MODEL statement (AUTOREG), 378
BALANCED option
AUTOMODL statement (X12), 3171
AUTOMODL statement (X13), 3171
BALLOON statement
LOAN procedure, 1371
BALLOONPAYMENT= option
BALLOON statement (LOAN), 1371
BANDOPT= option
MODEL statement (PANEL), 1816
BASE = option
PROC PANEL statement, 1806
BAYES statement
COUNTREG procedure, 595
HPQLIM procedure, 1154
QLIM procedure, 1964
BCX option
MODEL statement (QLIM), 1977
BDS option
MODEL statement (AUTOREG), 364
BESTCASE option
ARM statement (LOAN), 1373
BETA
PRIOR statement (COUNTREG), 610
PRIOR statement (HPQLIM), 1167
PRIOR statement (QLIM), 1980
BFN option
MODEL statement (PANEL), 1825
BI option
COMPARE statement (LOAN), 1375
BIASCORRECTED option
MODEL statement (PANEL), 1816
BL91 option
MODEL statement (PANEL), 1825
BL95 option
MODEL statement (PANEL), 1826
BLOCK option
PROC MODEL statement, 1499, 1702
BLOCKSEASON statement
UCM procedure, 2760
BLOCKSIZE= option
BLOCKSEASON statement (UCM), 2761
BLUS= option
OUTPUT statement (AUTOREG), 383
BOOTSTRAP= option
FORECAST statement (UCM), 2767
BOUND statement
VARMAX procedure, 2896, 3055
BOUNDARYALIGN= option
ID statement (TIMESERIES), 2676
BOUNDS statement
COUNTREG procedure, 601
ENTROPY procedure, 813
HPCOUNTREG procedure, 1082
HPQLIM procedure, 1158
MDC procedure, 1414
MODEL procedure, 1502
QLIM procedure, 1970
BOXCOXAR
macro, 180
macro variable, 181
BP option
COMPARE statement (LOAN), 1375
MODEL statement (AUTOREG), 365
MODEL statement (PANEL), 1826
BREAKINTEREST option
COMPARE statement (LOAN), 1375
BREAKPAYMENT option
COMPARE statement (LOAN), 1375
BREUSCH= option
FIT statement (MODEL), 1516
BSTART= option
PROC FORECAST statement, 954
BSY option
MODEL statement (PANEL), 1826
BTOL= option
MODEL statement (PANEL), 1816
BTWNG option
MODEL statement (HPPANEL), 1121

MODEL statement (PANEL), 1813
BTWNT option
MODEL statement (HPPANEL), 1121
BUYDOWN statement
LOAN procedure, 1374
BUYDOWNRATES= option
BUYDOWN statement (LOAN), 1374
BW option
MODEL statement (PANEL), 1826
BY statement
ARIMA procedure, 252
AUTOREG procedure, 360
COMPUTAB procedure, 496
COPULA procedure, 541
COUNTREG procedure, 601
ENTROPY procedure, 815
ESM procedure, 855
EXPAND procedure, 895
FORECAST procedure, 957
HPCDM procedure, 1013
HPCOUNTREG procedure, 1082
HPQLIM procedure, 1159
HPSEVERITY procedure, 1217
MDC procedure, 1415
MODEL procedure, 1504
PANEL procedure, 1803
PDLREG procedure, 1931
QLIM procedure, 1971
SEVERITY procedure, 2096
SIMILARITY procedure, 2244
SIMLIN procedure, 2306
SPECTRA procedure, 2334
SSM procedure, 2372
STATESPACE procedure, 2520
SYSLIN procedure, 2569
TIMEDATA procedure, 2612
TIMEID procedure, 2639
TIMESERIES procedure, 2669
TSCSREG procedure, 2736
UCM procedure, 2762
VARMAX procedure, 2897
X11 procedure, 3097
X12 procedure, 3174
X13 procedure, 3174

C
SPECTRA statement (TIMESERIES), 2679
CANCORR option
PROC STATESPACE statement, 2518
CAPS= option
ARM statement (LOAN), 1372
CAUCHY option
ERRORMODEL statement (MODEL), 1508
CAUSAL statement
VARMAX procedure, 2897
CDEC= option
PROC COMPUTAB statement, 490
CDF= option
ERRORMODEL statement (MODEL), 1508
CDTEST option
MODEL statement (PANEL), 1826
CELL statement
COMPUTAB procedure, 494
CENSORED option
ENDOGENOUS statement (HPQLIM), 1160, 1163
ENDOGENOUS statement (QLIM), 1972
MODEL statement (ENTROPY), 816
CENTER
SPECTRA statement (TIMESERIES), 2680
SSA statement (TIMESERIES), 2681
CENTER option
ARIMA statement (X11), 3096
IDENTIFY statement (ARIMA), 253
MODEL statement (AUTOREG), 361
MODEL statement (VARMAX), 2906
PROC SPECTRA statement, 2333
CEV= option
OUTPUT statement (AUTOREG), 383
CHAR option
COLUMNS statement (COMPUTAB), 491
ROWS statement (COMPUTAB), 493
CHARTS= option
MONTHLY statement (X11), 3098
QUARTERLY statement (X11), 3103
CHECK statement
X12 procedure, 3174
X13 procedure, 3174
CHECKBREAK option
LEVEL statement (UCM), 2773
CHICR= option
ARIMA statement (X11), 3095
CHISQUARED option
ERRORMODEL statement (MODEL), 1508
CHOICE= option
MODEL statement (MDC), 1415
CHOW= option
FIT statement (MODEL), 1516, 1605
MODEL statement (AUTOREG), 368
CLASS option
SHOW statement (COUNTREG), 614
CLASS statement
HPSEVERITY procedure, 1218
MDC procedure, 1415
PANEL procedure, 1803
SEVERITY procedure, 2096
CLEAR option
IDENTIFY statement (ARIMA), 253

CLIMIT= option
FORECAST command (TSFS), 3774
CLUSTER option
Model statement (PANEL), 1817
CMPMODEL options, 1498
COEF option
MODEL statement (AUTOREG), 368
MODEL statement (PDLREG), 1932
PROC SPECTRA statement, 2333
COEF= option
HETERO statement (AUTOREG), 381
COINTEG statement
VARMAX procedure, 2898, 2997
COINTTEST= option
MODEL statement (VARMAX), 2915
COINTTEST=(JOHANSEN) option
MODEL statement (VARMAX), 2915
COINTTEST=(JOHANSEN=(IORDER=)) option
MODEL statement (VARMAX), 2916, 3004
COINTTEST=(JOHANSEN=(NORMALIZE=)) option
MODEL statement (VARMAX), 2916, 2991
COINTTEST=(JOHANSEN=(TYPE=)) option
MODEL statement (VARMAX), 2916
COINTTEST=(SIGLEVEL=) option
MODEL statement (VARMAX), 2917
COINTTEST=(SW) option
MODEL statement (VARMAX), 2917, 2985
COINTTEST=(SW=(LAG=)) option
MODEL statement (VARMAX), 2917
COINTTEST=(SW=(TYPE=)) option
MODEL statement (VARMAX), 2917
COLLAPSE= option
LIBNAME statement (SASEQUAN), 3484
COLLIN option
ENTROPY procedure, 810
FIT statement (MODEL), 1517
'column headings' option
COLUMNS statement (COMPUTAB), 491
COLUMNS statement
COMPUTAB procedure, 491
COMMIT= option
PERFORMANCE statement (high-performance analytical procedures), 87
COMMONPACKAGE option
OUTSCORELIB statement (HPSEVERITY), 1226
OUTSCORELIB statement (SEVERITY), 2104
COMPARE statement
LOAN procedure, 1374
PANEL procedure, 1803
COMPONENT statement
SSM procedure, 2372
COMPOUND= option
FIXED statement (LOAN), 1368
COMPRESS= option
TARGET statement (SIMILARITY), 2250
COMPUTAB procedure, 487
syntax, 487
CONDITIONAL
OUTPUT statement (HPQLIM), 1165
OUTPUT statement (QLIM), 1978
CONGRA option
METHOD= option, PROC COUNTREG statement, 594
CONNECT= option
LIBNAME statement (SASEXFSD), 3567
CONST= option
BOXCOXAR macro, 181
LOGTEST macro, 186
CONSTANT= option
OUTPUT statement (AUTOREG), 383
OUTPUT statement (PDLREG), 1934
CONTROL, 1726
CONTROL statement
MODEL procedure, 1507, 1673
CONVERGE= option
ARIMA statement (X11), 3095
ENTROPY procedure, 812
ESTIMATE statement (ARIMA), 260
FIT statement (MODEL), 1519, 1556, 1562, 1564
MODEL statement (AUTOREG), 378
MODEL statement (MDC), 1415
MODEL statement (PDLREG), 1932
PROC SYSLIN statement, 2568
SOLVE statement (MODEL), 1532
CONVERT statement
EXPAND procedure, 895
CONVERT= option
LIBNAME statement (SASEFAME), 3329
COPULA procedure, 533, 538
BY statement, 541
DEFINE statement, 541
FIT statement, 543
PROC COPULA statement, 540
SIMULATE statement, 545
syntax, 538
VAR statement, 546
COPULA= option
SOLVE statement (MODEL), 1531
COPYVAR= option
OUTPUT statement (HPCOUNTREG), 1086
OUTPUT statement (HPPANEL), 1122
OUTPUT statement (HPQLIM), 1165
CORR option
FIT statement (MODEL), 1517
MODEL statement (PANEL), 1826
MODEL statement (TSCSREG), 2738

CORR statement
  TIMESERIES procedure, 2669
CORRB option
  ESTIMATE statement (MODEL), 1510
  FIT statement (MODEL), 1517
  HPQLIM procedure, 1151
  MODEL statement, 608, 1085
  MODEL statement (AUTOREG), 369
  MODEL statement (MDC), 1419
  MODEL statement (PANEL), 1826
  MODEL statement (PDLREG), 1932
  MODEL statement (SYSLIN), 2571
  MODEL statement (TSCSREG), 2738
  PROC COUNTREG statement, 592
  PROC HPCOUNTREG statement, 1079
  PROC HPPANEL statement, 1120
  QLIM procedure, 1960
CORRCONSTANT= option
  GARCH statement, 2902
CORRELATION option
  SHOW statement (COUNTREG), 614
CORROUT option
  PROC HPCOUNTREG statement, 1079
  PROC HPPANEL statement, 1121
  PROC HPQLIM statement, 1150
  PROC PANEL statement, 1802
  PROC QLIM statement, 1959
  PROC TSCSREG statement, 2736
CORRS option
  FIT statement (MODEL), 1517
COS function, 1680
COSH function, 1680
COST option
  ENDOGENOUS statement (HPQLIM), 1161, 1164
  ENDOGENOUS statement (QLIM), 1973
COUNT statement
  TIMESERIES procedure, 2671
COUNT= option
  EXTERNALCOUNTS statement (HPCDM), 1014
COUNTPLOTS= option
  PROC TIMESERIES statement, 2665
COUNTREG procedure, 587
  PRIOR statement, 610
  SCORE statement, 611
  SHOW statement, 613
  syntax, 587
COUNTREG procedure, CLASS statement, 601
  MISSING option, 602
  ORDER= option, 602
  PARAM= option, 602
  REF= option, 603
COUNTREG procedure, SHOW statement, 613
  ALL option, 614
  CLASS option, 614
  CORRELATION option, 614
  COVARIANCE option, 614
  EFFECTS option, 614
  FITSTATS option, 614
  PARAMETERS option, 614
  PROGRAM option, 614
COUNTREG procedure, TEST statement, 616
COUNTS option
  PLOTS option, PROC COUNTREG statement, 593
COUNTSTORE= option
  PROC HPCDM statement, 1008
COV option
  FIT statement (MODEL), 1517
COV3OUT option
  PROC SYSLIN statement, 2567
COVARIANCE option
  SHOW statement (COUNTREG), 614
COVB option
  ESTIMATE statement (MODEL), 1510
  FIT statement (MODEL), 1517
  HPQLIM procedure, 1151
  MODEL statement, 608, 1085
  MODEL statement (AUTOREG), 369
  MODEL statement (MDC), 1419
  MODEL statement (PANEL), 1827
  MODEL statement (PDLREG), 1932
  MODEL statement (SYSLIN), 2571
  MODEL statement (TSCSREG), 2738
  PROC COUNTREG statement, 592
  PROC HPCOUNTREG statement, 1079
  PROC HPPANEL statement, 1120
  PROC STATESPACE statement, 2519
  QLIM procedure, 1960
COVBEST= option
  ENTROPY procedure, 810
  FIT statement (MODEL), 1512, 1548
COVEST= option
  HPQLIM procedure, 1151
  MODEL statement (AUTOREG), 369
  MODEL statement (MDC), 1419
  PROC COUNTREG statement, 592
  PROC HPCOUNTREG statement, 1079
  QLIM procedure, 1960
COVOUT option
  ENTROPY procedure, 811
  FIT statement (MODEL), 1515
  PROC AUTOREG statement, 359
  PROC COUNTREG statement, 592
  PROC HPCOUNTREG statement, 1079
  PROC HPPANEL statement, 1120
  PROC HPQLIM statement, 1150

PROC HPSEVERITY statement, 1209
PROC MDC statement, 1413
PROC PANEL statement, 1801
PROC QLIM statement, 1959
PROC SEVERITY statement, 2087
PROC SYSLIN statement, 2567
PROC TSCSREG statement, 2736
COVS option
FIT statement (MODEL), 1517, 1553
CPEV= option
OUTPUT statement (AUTOREG), 383
CRITERION= option
COUNT statement (TIMESERIES), 2671
PROC HPSEVERITY statement, 1214
PROC SEVERITY statement, 2093
CROSS option
PROC SPECTRA statement, 2333
CROSSCORR statement
TIMESERIES procedure, 2672
CROSSCORR= option
IDENTIFY statement (ARIMA), 253
CROSSLIST= option
LIBNAME statement (SASEFAME), 3333
CROSSPLOTS= option
PROC TIMESERIES statement, 2665
CROSSVAR statement
TIMESERIES procedure, 2685
CSPACE= option
PROC COMPUTAB statement, 490
CSTART= option
PROC FORECAST statement, 954
CUSIP= option
LIBNAME statement (SASECRSP), 3286
CUSUM= option
OUTPUT statement (AUTOREG), 383
CUSUMLB= option
OUTPUT statement (AUTOREG), 383
CUSUMSQ= option
OUTPUT statement (AUTOREG), 383
CUSUMSQLB= option
OUTPUT statement (AUTOREG), 383
CUSUMSQUB= option
OUTPUT statement (AUTOREG), 383
CUSUMUB= option
OUTPUT statement (AUTOREG), 383
CUTOFF= option
SSPAN statement (X11), 3105
CV= option
OUTLIER statement (X12), 3184
OUTLIER statement (X13), 3184
CVMETHOD= option
OUTLIER statement (X12), 3185
OUTLIER statement (X13), 3185
CWIDTH= option
PROC COMPUTAB statement, 490
CYCLE statement
UCM procedure, 2762
CYCLETYPE= option
PROC TIMEDATA statement, 2610

DASILVA option
MODEL statement (PANEL), 1814
MODEL statement (TSCSREG), 2739
DATA Step
IF Statement, 100
WHERE Statement, 100
DATA step
DROP statement, 101
KEEP statement, 101
DATA= option
ENTROPY procedure, 810
FIT statement (MODEL), 1515, 1627
FORECAST command (TSFS), 3772, 3773
IDENTIFY statement (ARIMA), 254
PROC ARIMA statement, 250
PROC AUTOREG statement, 359
PROC COMPUTAB statement, 489
PROC COUNTREG statement, 591
PROC ESM statement, 852
PROC EXPAND statement, 892
PROC FORECAST statement, 954
PROC HPCDM statement, 1009
PROC HPCOUNTREG statement, 1079
PROC HPPANEL statement, 1120
PROC HPQLIM statement, 1150
PROC HPSEVERITY statement, 1209
PROC MDC statement, 1413
PROC MODEL statement, 1496
PROC PANEL statement, 1801
PROC PDLREG statement, 1931
PROC QLIM statement, 1959
PROC SEVERITY statement, 2087
PROC SIMILARITY statement, 2242
PROC SIMLIN statement, 2305, 2311
PROC SPECTRA statement, 2334
PROC SSM statement, 2370
PROC STATESPACE statement, 2517
PROC SYSLIN statement, 2567
PROC TIMEDATA statement, 2610
PROC TIMEID statement, 2638
PROC TIMESERIES statement, 2665
PROC TSCSREG statement, 2736
PROC UCM statement, 2756
PROC VARMAX statement, 2893
PROC X11 statement, 3093
PROC X12 statement, 3162
PROC X13 statement, 3162
SOLVE statement (MODEL), 1528, 1673

TSVIEW command (TSFS), 3772, 3773
DATASOURCE procedure, 712
syntax, 712
DATE
function, 172
DATE= option
MONTHLY statement (X11), 3099
PROC X12 statement, 3162
PROC X13 statement, 3162
QUARTERLY statement (X11), 3103
DATEJUL
function, 120
DATEJUL function, 172
DATEPART function, 121, 172
DATES= option
LIBNAME statement(SASEXFSD), 3566
DATETIME
function, 172
DAY
function, 120, 172
DBLDOG option
METHOD= option, PROC COUNTREG statement, 594
DBNAME= option
PROC DATASOURCE statement, 714
DBTYPE= option
PROC DATASOURCE statement, 714
DBVERSION= option
LIBNAME statement (SASEFAME), 3333
DECOMP statement
TIMESERIES procedure, 2673
DEFINE statement
COPULA procedure, 541
HPCOPULA procedure, 1060
DEGREE= option
SPLINEREG statement (UCM), 2780
SPLINESEASON statement (UCM), 2781
DELETMODEL statement
MODEL procedure, 1507
DELTA= option
ESTIMATE statement (ARIMA), 260
DEPLAG statement
SSM procedure, 2373
UCM procedure, 2763
DESCENDING option
CLASS statement (HPSEVERITY), 1218
CLASS statement (SEVERITY), 2096
DETAILS option
FIT statement (MODEL), 1572
PERFORMANCE statement (COUNTREG), 610
PERFORMANCE statement (high-performance analytical procedures), 87
PERFORMANCE statement (HPCOPULA), 1061
PERFORMANCE statement (HPCOUNTREG), 1087
PERFORMANCE statement (HPPANEL), 1123
PERFORMANCE statement (HPQLIM), 1166
PROC MODEL statement, 1500
DETTOL= option
PROC STATESPACE statement, 2519
DFMIXTURE= option
SCALEMODEL statement (HPSEVERITY), 1228
SCALEMODEL statement (SEVERITY), 2106
DFPVALUE
macro, 183
macro variable, 183, 185
DFTEST
macro, 184
DFTEST option
MODEL statement (VARMAX), 2915, 3033
DFTEST=(DLAG=) option
MODEL statement (VARMAX), 2915
DHMS
function, 121
DHMS function, 172
DIAG= option
FORECAST command (TSFS), 3775
DIAGNOSTICS= option
BAYES statement (COUNTREG), 596
BAYES statement (HPQLIM), 1154
BAYES statement (QLIM), 1966
DIF
function, 129
DIF function
MODEL procedure, 131
DIF= option
BOXCOXAR macro, 181
DFTEST macro, 184
INPUT statement (SIMILARITY), 2248
LOGTEST macro, 186
MODEL statement (VARMAX), 2906
TARGET statement (SIMILARITY), 2252
VAR statement (TIMEDATA), 2616
VAR statement (TIMESERIES), 2685
DIFF= option
IDENTIFY statement (X12), 3181
IDENTIFY statement (X13), 3181
DIFFID= option
AUTOMDL statement (X12), 3171
AUTOMDL statement (X13), 3171
DIFFIDITER= option
AUTOMDL statement (X12), 3171
AUTOMDL statement (X13), 3171
DIFFORDER= option
AUTOMDL statement (X12), 3171
AUTOMDL statement (X13), 3171

DIFX= option
MODEL statement (VARMAX), 2907
DIFY= option
MODEL statement (VARMAX), 2907, 3044
DIMMAX= option
PROC STATESPACE statement, 2518
DISCRETE option
ENDOGENOUS statement (HPQLIM), 1159, 1163
ENDOGENOUS statement (QLIM), 1971
DISPERSION option
PLOTS option, PROC COUNTREG statement, 593
DISPERSION= option
OUTPUT statement (COUNTREG), 608
OUTPUT statement (HPCOUNTREG), 1087
SCORE statement (COUNTREG), 612
DISPMODEL statement
COUNTREG procedure, 603
HPCOUNTREG procedure, 1083
DIST statement
HPSEVERITY procedure, 1220
SEVERITY procedure, 2098
DIST= option
HPCOUNTREG statement (HPCOUNTREG), 1084
MODEL statement (AUTOREG), 361
MODEL statement (HPCOUNTREG), 1084
MODEL statement, COUNTREG procedure, 605
PROC COUNTREG statement, 605
DISTBY statement
HPCDM procedure, 1013
DISTRIBUTION= option
COUNT statement (TIMESERIES), 2672
ENDOGENOUS statement (HPQLIM), 1159, 1163
ENDOGENOUS statement (QLIM), 1972
DLAG= option
DFPVALUE macro, 183
DFTEST macro, 184
DO, 1685
DOL option
ROWS statement (COMPUTAB), 493
DOMAIN option
SPECTRA statement (TIMESERIES), 2680
DOWNPAYMENT= option
FIXED statement (LOAN), 1369
DOWNPAYPCT= option
FIXED statement (LOAN), 1369
DP= option
FIXED statement (LOAN), 1369
DPCT= option
FIXED statement (LOAN), 1369
DROP statement
DATASOURCE procedure, 717
DROP= option
FIT statement (MODEL), 1511
LIBNAME statement (SASEHAVR), 3436
DROPEVENT statement
DATASOURCE procedure, 718
DROPGEOG1= option
LIBNAME statement (SASEHAVR), 3437
DROPGEOG2= option
LIBNAME statement (SASEHAVR), 3437
DROPGROUP= option
LIBNAME statement (SASEHAVR), 3436
DROPH= option
SEASON statement (UCM), 2777
DROPLONG= option
LIBNAME statement (SASEHAVR), 3437
DROPSHORT= option
LIBNAME statement (SASEHAVR), 3437
DROPSOURCE= option
LIBNAME statement (SASEHAVR), 3436
DUAL option
ENTROPY procedure, 812
DUL option
ROWS statement (COMPUTAB), 493
DUPLICATES option
TIMEID procedure, 2640
DW option
FIT statement (MODEL), 1517
MODEL statement (PANEL), 1826
MODEL statement (SYSLIN), 2571
DW= option
MODEL statement (AUTOREG), 371
MODEL statement (PDLREG), 1932
DWPROB option
FIT statement (MODEL), 1517
MODEL statement (AUTOREG), 371
MODEL statement (PDLREG), 1932
DYNAMIC option
FIT statement (MODEL), 1512, 1594, 1596
SOLVE statement (MODEL), 1529, 1592, 1639

E
SPECTRA statement (TIMESERIES), 2679
EASTERMEANS= option
REGRESSION statement (X12), 3189
REGRESSION statement (X13), 3189
EBCDIC option
PROC DATASOURCE statement, 714
ECM= option
MODEL statement (VARMAX), 2920
ECM=(ECTREND) option
MODEL statement (VARMAX), 2920
ECM=(NORMALIZE=) option
MODEL statement (VARMAX), 2920

ECM=(RANK=) option
MODEL statement (VARMAX), 2920
ECTREND option
COINTEG statement (VARMAX), 2899, 2994
EDFALPHA= option
PROC HPSEVERITY statement, 1209
PROC SEVERITY statement, 2088
EFFECTS option
SHOW statement (COUNTREG), 614
Empirical Distribution Estimation
ERRORMODEL statement (MODEL), 1551
EMPIRICAL= option
ERRORMODEL statement (MODEL), 1509
EMPIRICALCDF= option
PROC HPSEVERITY statement, 1215
PROC SEVERITY statement, 2094
END= option
ID statement (ESM), 858
ID statement (SIMILARITY), 2246
ID statement (TIMEDATA), 2614
ID statement (TIMESERIES), 2677
LIBNAME statement (SASEFRED), 3386
LIBNAME statement (SASEHAVR), 3436
LIBNAME statement (SASEQUAN), 3486
MONTHLY statement (X11), 3099
QUARTERLY statement (X11), 3104
ENDOGENOUS statement
MODEL procedure, 1507, 1673
SIMLIN procedure, 2306
SYSLIN procedure, 2570
ENDOTEST option
ENDOGENOUS statement (QLIM), 1974
ENTROPY procedure, 808
syntax, 808
ENTRY= option
FORECAST command (TSFS), 3774
EPSILON = option
FIT statement (MODEL), 1519
%EQAR macro, 1624
EQGROUP statement
MODEL procedure, 1507
%EQMA macro, 1624
ERRORCOMP= option
COUNTREG statement (COUNTREG), 605
HPCOUNTREG statement (HPCOUNTREG), 1084
MODEL statement (COUNTREG), 605
MODEL statement (HPCOUNTREG), 1084
ERRORMODEL statement
MODEL procedure, 1508
ERRSTD
OUTPUT statement (HPQLIM), 1165
OUTPUT statement (QLIM), 1978
ESACF option
IDENTIFY statement (ARIMA), 254
ESM, 847
ESM procedure, 850
syntax, 850
EST= option
PROC SIMLIN statement, 2305, 2310
ESTDATA= option
FIT statement (MODEL), 1515, 1627
SOLVE statement (MODEL), 1528, 1642, 1671
ESTIMATE statement
ARIMA procedure, 257
MODEL procedure, 1509
UCM procedure, 2764
X12 procedure, 3175
X13 procedure, 3175
ESTIMATEDCASE= option
ARM statement (LOAN), 1373
ESTPRINT option
PROC SIMLIN statement, 2305
ESUPPORTS= option
MODEL statement (ENTROPY), 815
EVAL statement
SSM procedure, 2375
EVENT statement
X12 procedure, 3177
X13 procedure, 3177
EXACT= option
ESTIMATE statement (X12), 3175
ESTIMATE statement (X13), 3175
EXCLUDE= option
FIT statement (MODEL), 1609
INSTRUMENTS statement (MODEL), 1521
MONTHLY statement (X11), 3099
EXOGENEITY option
COINTEG statement (VARMAX), 2899, 3000
EXOGENOUS statement
MODEL procedure, 1511, 1673
SIMLIN procedure, 2306
EXP function, 1680
EXPAND procedure, 891
CONVERT statement, 904
syntax, 891
EXPAND= option
TARGET statement (SIMILARITY), 2252
EXPECTED
OUTPUT statement (HPQLIM), 1165
OUTPUT statement (QLIM), 1978
EXTERNALCOUNTS statement
HPCDM procedure, 1014
EXTRADIFFUSE= option
ESTIMATE statement (UCM), 2765
FORECAST statement (UCM), 2768
EXTRAPOLATE option
PROC EXPAND statement, 893

F option
ERRORMODEL statement (MODEL), 1508
FACTLET= option
LIBNAME statement (SASEXFSD), 3563
LIBNAME statement(SASEXFSD), 3563
FACTOR= option
PROC EXPAND statement, 893, 897
FAMEOUT= option
LIBNAME statement (SASEFAME), 3333
FAMEPRINT option
PROC DATASOURCE statement, 714
FCMPOPT statement
SIMILARITY procedure, 2245
TIMEDATA procedure, 2612
FDONE option
MODEL statement (PANEL), 1814
FDONETIME option
MODEL statement (PANEL), 1814
FDTWO option
MODEL statement (PANEL), 1814
FILETYPE= option
PROC DATASOURCE statement, 714
FIML option
FIT statement (MODEL), 1512, 1547, 1628, 1731
PROC SYSLIN statement, 2568
FINAL= option
X11 statement (X12), 3201
X11 statement (X13), 3201
FIRST option
PROC SYSLIN statement, 2569
FIT statement
COPULA procedure, 543
MODEL procedure, 1511
FIT statement, MODEL procedure
GINV= option, 1512
FITSTATS option
SHOW statement (PCOUNTREG), 614
FIXED statement
LOAN procedure, 1367
FIXEDCASE option
ARM statement (LOAN), 1373
FIXONE option
MODEL statement (HPPANEL), 1121
MODEL statement (PANEL), 1814
MODEL statement (TSCSREG), 2738
FIXONETIME option
MODEL statement (HPPANEL), 1121
MODEL statement (PANEL), 1814
FIXTWO option
MODEL statement (HPPANEL), 1121
MODEL statement (PANEL), 1814
MODEL statement (TSCSREG), 2738
FLATDATA statement
PANEL procedure, 1806
FLOW option
PROC MODEL statement, 1500
FORCE= option
X11 statement (X12), 3201
X11 statement (X13), 3201
FORCE=FREQ option
LIBNAME statement (SASEHAVR), 3438
FORECAST
macro, 3773
FORECAST option
SOLVE statement (MODEL), 1530, 1639, 1642
FORECAST procedure, 951
syntax, 951
FORECAST statement
ARIMA procedure, 262
ESM procedure, 855
UCM procedure, 2767
X12 procedure, 3179
X13 procedure, 3179
FORECAST= option
ARIMA statement (X11), 3095
FORM statement
STATESPACE procedure, 2521
FORM= option
GARCH statement, 2902
FORMAT statement
DATASOURCE procedure, 721
FORMAT= option
ATTRIBUTE statement (DATASOURCE), 721
COLUMNS statement (COMPUTAB), 492
ID statement (ESM), 859
ID statement (SIMILARITY), 2246
ID statement (TIMEDATA), 2614
ID statement (TIMESERIES), 2677
LIBNAME statement (SASEFRED), 3387
LIBNAME statement (SASEQUAN), 3485
LIBNAME statement(SASEXFSD), 3567
ROWS statement (COMPUTAB), 494
TIMEID procedure, 2640
FREQ statement
COUNTREG procedure, 604
HPCOUNTREG procedure, 1083
FREQ= option
LIBNAME statement (SASEFRED), 3387
LIBNAME statement (SASEHAVR), 3436
LIBNAME statement (SASEQUAN), 3486
PROC TIMEID statement, 2638
FREQUENCY= option
PROC DATASOURCE statement, 715
FROM= option
PROC EXPAND statement, 893, 897
FRONTIER option
ENDOGENOUS statement (HPQLIM), 1160, 1164

ENDOGENOUS statement (QLIM), 1973
FSRSQ option
FIT statement (MODEL), 1517, 1538, 1610
FULLER option
MODEL statement (TSCSREG), 2738
FULLWEIGHT= option
MONTHLY statement (X11), 3099
QUARTERLY statement (X11), 3104
FUNCTION= option
TRANSFORM statement (X12), 3198
TRANSFORM statement (X13), 3198
FUZZ= option
PROC COMPUTAB statement, 490

GAMMA
PRIOR statement (COUNTREG), 610
PRIOR statement (HPQLIM), 1167
PRIOR statement (QLIM), 1979
GARCH statement
VARMAX procedure, 2902
GARCH= option
MODEL statement (AUTOREG), 362
GCE option
ENTROPY procedure, 810
GCEM option
ENTROPY procedure, 810
GCONV= option
ENTROPY procedure, 812
GDELTA= option
OUTPUT statement (COUNTREG), 609
OUTPUT statement (HPCOUNTREG), 1086
SCORE statement (COUNTREG), 612
GENERAL= option
ERRORMODEL statement (MODEL), 1508
GENGMMV option
FIT statement (MODEL), 1513
GEOG1= option
LIBNAME statement (SASEHAVR), 3437
GEOG2= option
LIBNAME statement (SASEHAVR), 3437
GETDER function, 1680
GHM option
MODEL statement (PANEL), 1826
GINV option
MODEL statement (AUTOREG), 371
MODEL statement (PDLREG), 1932
GINV= option
FIT statement (MODEL), 1512
MODEL statement (PANEL), 1816
GME option
ENTROPY procedure, 810
GMED option
ENTROPY procedure, 810
GMEM option
ENTROPY procedure, 810
GMM option
FIT statement (MODEL), 1513, 1539, 1579, 1628, 1630, 1631, 1633
GMM1 option
MODEL statement (PANEL), 1814
GMM2 option
MODEL statement (PANEL), 1814
GODFREY option
FIT statement (MODEL), 1518
MODEL statement (AUTOREG), 371
GRAPH option
PROC MODEL statement, 1499, 1702
GRID option
ESTIMATE statement (ARIMA), 260
GRIDHOST= option
PERFORMANCE statement (high-performance analytical procedures), 87
GRIDMODE= option
PERFORMANCE statement (high-performance analytical procedures), 88
GRIDTIMEOUT= option
PERFORMANCE statement (high-performance analytical procedures), 88
GRIDVAL= option
ESTIMATE statement (ARIMA), 260
GROUP1 option
CAUSAL statement (VARMAX), 2897
GROUP2 option
CAUSAL statement (VARMAX), 2897
GROUP= option
LIBNAME statement (SASEHAVR), 3436
GROUPID= option
PROC COUNTREG statement, 591
PROC HPCOUNTREG statement, 1079
GROUPS option
SSA statement (TIMESERIES), 2682
GVIIDKEY= option
LIBNAME statement (SASEXCCM), 3523
GVKEY= option
LIBNAME statement (SASEXCCM), 3523

H= option
COINTEG statement (VARMAX), 2899, 2997
HAC = option
MODEL statement (PANEL), 1817
HALT option
RANDOM statement (QLIM), 1981
HALT suboption1
RANDOM statement (QLIM), 1981
HALT suboption2
RANDOM statement (QLIM), 1981
HALTONSTART= option
MODEL statement (MDC), 1416

HAUSMAN option
FIT statement (MODEL), 1518, 1603
HCCME= option
FIT statement (MODEL), 1513
MODEL statement (PANEL), 1818
HCUSIP= option
LIBNAME statement (SASECRSP), 3286
HECKIT option
QLIM procedure, 1960
HERM option
RANDOM statement (QLIM), 1981
HESSIAN= option
FIT statement (MODEL), 1519, 1549
HETERO statement
AUTOREG procedure, 380
HEV option
MODEL statement (MDC), 1416
high-performance analytical procedures, PERFORMANCE statement, 87
COMMIT= option, 87
DETAILS option, 87
GRIDHOST= option, 87
GRIDMODE= option, 88
GRIDTIMEOUT= option, 88
HOST= option, 87
INSTALL= option, 88
INSTALLLOC= option, 88
LASR= option, 88
LASRSERVER= option, 88
MODE= option, 88
NNODES= option, 88
NODES= option, 88
NTHREADS= option, 89
THREADS= option, 89
TIMEOUT= option, 88
HMS
function, 121
HMS function, 172
HOLIDAY function, 172
HONDA option
MODEL statement (PANEL), 1826
Honda option
MODEL statement (PANEL), 1826
HORIZON= option
FORECAST command (TSFS), 3774
HOST= option
PERFORMANCE statement (high-performance analytical procedures), 87
HOUR
function, 173
HPCDM procedure, 1006
DISTBY statement, 1013
EXTERNALCOUNTS statement, 1014
OUTPUT statement, 1014
OUTSUM statement, 1015
PERFORMANCE statement, 1018
SEVERITYMODEL statement, 1018
syntax, 1006
HPCDM procedure, EXTERNALCOUNTS statement
COUNT= option, 1014
ID= option, 1014
HPCDM procedure, OUTPUT statement
ADJSAMPLEVAR= option, 1015
OUT= option, 1014
PERTURBOUT option, 1015
SAMPLEVAR= option, 1015
HPCDM procedure, OUTSUM statement
OUT= option, 1015
PCTLNAME= option, 1017
PCTLNDEC= option, 1018
PCTLPTS= option, 1017
HPCDM procedure, PROC HPCDM statement, 1008
ADJUSTEDSEVERITY= option, 1008
COUNTSTORE= option, 1008
DATA= option, 1009
NOPRINT option, 1009
NPERTURBEDSAMPLES= option, 1009
NREPLICATES= option, 1009
PCTLDEF= option, 1010
PLOTS= option, 1010
PRINT= option, 1011
SEED= option, 1012
SEVERITYEST= option, 1012
VARDEF= option, 1012
HPCOPULA procedure, 1059
DEFINE statement, 1060
PERFORMANCE statement, 1061
PROC HPCOPULA statement, 1059
SIMULATE statement, 1060
syntax, 1059
VAR Statement, 1061
HPCOPULA procedure, PERFORMANCE statement, 1061
HPCOUNTREG procedure, 1076
PERFORMANCE statement, 1087
syntax, 1076
HPCOUNTREG procedure, CLASS statement, 1082
HPCOUNTREG procedure, PERFORMANCE statement, 1087
HPCOUNTREG procedure, TEST statement, 1088
HPCOUNTREG procedure, WEIGHT statement, 1089
HPPANEL procedure, 1119
PERFORMANCE statement, 1123
syntax, 1119
HPPANEL procedure, PERFORMANCE statement, 1123
HPQLIM procedure, 1147
PERFORMANCE statement, 1166

PRIOR statement, 1166
syntax, 1147
HPQLIM procedure, FREQ statement, 1161
HPQLIM procedure, PERFORMANCE statement, 1166
HPQLIM procedure, TEST statement, 1168
HPQLIM procedure, WEIGHT statement, 1169
HPSEVERITY procedure, 1206
CLASS statement, 1218
DIST statement, 1220
LOSS statement, 1222
NLOPTIONS statement, 1224
OUTSCORELIB statement, 1225
PERFORMANCE statement, 1227
SCALEMODEL statement, 1227
syntax, 1206
WEIGHT statement, 1229
HPSEVERITY procedure, CLASS statement
DESCENDING option, 1218
MISSING option, 1219
ORDER= option, 1218
PARAM= option, 1220
REF= option, 1219
TRUNCATE= option, 1220
HPSEVERITY procedure, DIST statement
INIT= option, 1221
LISTONLY option, 1221
VALIDATEONLY option, 1222
HPSEVERITY procedure, LOSS statement
LEFTCENSORED= option, 1222
LEFTTRUNCATED= option, 1223
PROBOBSERVED= option, 1223
RIGHTCENSORED= option, 1223
RIGHTTRUNCATED= option, 1224
HPSEVERITY procedure, OUTSCORELIB statement
COMMONPACKAGE option, 1226
OUTBYID= option, 1226
OUTLIB= option, 1225
HPSEVERITY procedure, PROC HPSEVERITY statement, 1209
COVOUT option, 1209
CRITERION= option, 1214
DATA= option, 1209
EDF=AUTO option, 1215
EDF=KAPLANMEIER option, 1215
EDF=MODIFIEDKM option, 1216
EDF=NOTURNBULL option, 1216
EDF=STANDARD option, 1216
EDF=TURNBULL option, 1216
EMPIRICALCDF= option, 1215
INEST= option, 1209
INITSAMPLE option, 1209
INSTORE= option, 1210
NAMELEN= option, 1210
NOCLPRINT option, 1211
NOPRINT option, 1211
OBJECTIVE= option, 1217
OUTCDF= option, 1211
OUTEST= option, 1211
OUTMODELINFO= option, 1211
OUTSTAT= option, 1211
OUTSTORE= option, 1211
PLOTS= option, 1212
PRINT= option, 1213
VARDEF= option, 1214
HPSEVERITY procedure, SCALEMODEL statement
DFMIXTURE= option, 1228
OFFSET= option, 1229
HRINITIAL option
AUTOMDL statement (X12), 3172
AUTOMDL statement (X13), 3172
HT= option
OUTPUT statement (AUTOREG), 383
HTAYLOR option
MODEL statement (PANEL), 1814

I option
FIT statement (MODEL), 1518, 1568
MODEL statement (PDLREG), 1932
MODEL statement (SYSLIN), 2571
ID statement
ENTROPY procedure, 815
ESM procedure, 857
EXPAND procedure, 896
FORECAST procedure, 957
HPPANEL procedure, 1121
MDC procedure, 1415
MODEL procedure, 1520
PANEL procedure, 1807
SIMILARITY procedure, 2245
SIMLIN procedure, 2307
SSM procedure, 2376
STATESPACE procedure, 2521
TIMEDATA procedure, 2613
TIMEID procedure, 2640
TIMESERIES procedure, 2675
TSCSREG procedure, 2737
UCM procedure, 2769
VARMAX procedure, 2903
X11 procedure, 3097
X12 procedure, 3180
X13 procedure, 3180
ID= option
EXTERNALCOUNTS statement (HPCDM), 1014
FORECAST command (TSFS), 3772, 3773
FORECAST statement (ARIMA), 262
OUTLIER statement (ARIMA), 261

TSVIEW command (TSFS), 3772, 3773
IDENTIFY statement
ARIMA procedure, 253, 261
X12 procedure, 3180
X13 procedure, 3180
IDENTITY statement
SYSLIN procedure, 2570
IDS= option
LIBNAME statement (SASEXFSD), 3564
IF, 1685
IGAMMA
PRIOR statement (COUNTREG), 610
PRIOR statement (HPQLIM), 1167
PRIOR statement (QLIM), 1980
INCLUDE, 1689
INCLUDE statement
MODEL procedure, 1520
INDEX option
PROC DATASOURCE statement, 714
INDID = option
PROC PANEL statement, 1806
INDNO= option
LIBNAME statement (SASECRSP), 3287
LIBNAME statement (SASEXCCM), 3524
INEST= option
PROC HPSEVERITY statement, 1209
PROC SEVERITY statement, 2088
INEVENT= option
PROC X12 statement, 3163
PROC X13 statement, 3163
INFILE= option
PROC DATASOURCE statement, 714
INIT statement
COMPUTAB procedure, 495
COUNTREG procedure, 604
HPCOUNTREG procedure, 1084
HPQLIM procedure, 1162
QLIM procedure, 1976
INIT= option
DIST statement (HPSEVERITY), 1221
DIST statement (SEVERITY), 2099
FIXED statement (LOAN), 1369
INITIAL statement
STATESPACE procedure, 2521
VARMAX procedure, 2904
INITIAL= option
FIT statement (MODEL), 1511, 1596
FIXED statement (LOAN), 1369
MODEL statement (AUTOREG), 378
MODEL statement (MDC), 1419
INITIALPCT= option
FIXED statement (LOAN), 1369
INITMISS option
PROC COMPUTAB statement, 490
INITPCT= option
FIXED statement (LOAN), 1369
INITSAMPLE option
PROC HPSEVERITY statement, 1209
PROC SEVERITY statement, 2088
INITVAL= option
ESTIMATE statement (ARIMA), 259
INPUT statement
SIMILARITY procedure, 2248
X12 procedure, 3182
X13 procedure, 3182
INPUT= option
ESTIMATE statement (ARIMA), 257, 276
INSET= option
LIBNAME statement (SASECRSP), 3288
LIBNAME statement (SASEFAME), 3331, 3332
INSTALL= option
PERFORMANCE statement (high-performance analytical procedures), 88
INSTALLLOC= option
PERFORMANCE statement (high-performance analytical procedures), 88
INSTORE= option
PROC HPSEVERITY statement, 1210
PROC SEVERITY statement, 2089
INSTRUMENTS statement
MODEL procedure, 1520, 1607
SYSLIN procedure, 2570
INTCINDEX function, 173
INTCK
function, 123
INTCK function, 173
INTCYCLE function, 173
INTEGRATE= option
MODEL statement (MDC), 1416
INTERIM= option
PROC SIMLIN statement, 2305
INTERVAL= option
FORECAST command (TSFS), 3772, 3773
FORECAST statement (ARIMA), 263, 282
ID statement (ESM), 859
ID statement (SIMILARITY), 2247
ID statement (SSM), 2376
ID statement (TIMEDATA), 2614
ID statement (TIMESERIES), 2675
ID statement (UCM), 2769
ID statement (VARMAX), 2903
PROC DATASOURCE statement, 715
PROC FORECAST statement, 954
PROC STATESPACE statement, 2520
PROC X12 statement, 3163
PROC X13 statement, 3163
TIMEID procedure, 2640
TSVIEW command (TSFS), 3772, 3773

INTERVAL=option
FIXED statement (LOAN), 1369
INTFIT function, 173
INTFMT function, 174
INTGET function, 174
INTGPRINT option
SOLVE statement (MODEL), 1532
INTINDEX function, 174
INTNX
function, 122
INTNX function, 174
INTONLY option
INSTRUMENTS statement (MODEL), 1522
INTORDER= option
MODEL statement (MDC), 1416
INTPER= option
PROC FORECAST statement, 954
PROC STATESPACE statement, 2520
INTSEAS function, 175
INTSHIFT function, 175
INTTEST function, 175
IRREGULAR statement
SSM procedure, 2376
UCM procedure, 2769
IT2SLS option
FIT statement (MODEL), 1513
IT3SLS option
FIT statement (MODEL), 1513
PROC SYSLIN statement, 2568
ITALL option
FIT statement (MODEL), 1518, 1569
ITDETAILS option
FIT statement (MODEL), 1518, 1568
ITEMLIST= option
LIBNAME statement (SASEXCCM), 3524
ITEMS= option
LIBNAME statement (SASEFRED), 3386
LIBNAME statement (SASEQUAN), 3486
LIBNAME statement(SASEXFSD), 3564
ITGMM option
FIT statement (MODEL), 1513, 1543
MODEL statement (PANEL), 1814
ITOLS option
FIT statement (MODEL), 1513
ITPRINT option
ENTROPY procedure, 811
ESTIMATE statement (X12), 3176
ESTIMATE statement (X13), 3176
FIT statement (MODEL), 1518, 1562, 1568
MODEL statement, 608
MODEL statement (AUTOREG), 371
MODEL statement (MDC), 1419
MODEL statement (PANEL), 1827
MODEL statement (PDLREG), 1932
PROC COPULA statement, 544
PROC STATESPACE statement, 2519
PROC SYSLIN statement, 2569
QLIM procedure, 1960
SOLVE statement (MODEL), 1532, 1668
ITSUR option
FIT statement (MODEL), 1513, 1539, 1772
PROC SYSLIN statement, 2568

J function, 2934
J= option
COINTEG statement (VARMAX), 2901
JACOBI option
SOLVE statement (MODEL), 1530
JULDATE function, 120, 175

K option
PROC SPECTRA statement, 2334
K= option
MODEL statement (SYSLIN), 2571
PROC SYSLIN statement, 2568
KEEP = option
PROC PANEL statement, 1806
KEEP statement
DATASOURCE procedure, 716
KEEP= option
FORECAST command (TSFS), 3775
LIBNAME statement (SASEHAVR), 3436
KEEPEVENT statement
DATASOURCE procedure, 718
KEEPH= option
SEASON statement (UCM), 2777
KERNEL option
FIT statement (MODEL), 1630
Kernel option values
SPECTRA statement (TIMESERIES), 2680
KERNEL= option
FIT statement (MODEL), 1513, 1541
KLAG= option
PROC STATESPACE statement, 2519
KNOTS= option
SPLINEREG statement (UCM), 2780
SPLINESEASON statement (UCM), 2781
KW option
MODEL statement (PANEL), 1826

L= option
FIXED statement (LOAN), 1368
LABEL option
COLUMNS statement (COMPUTAB), 491
ROWS statement (COMPUTAB), 493
LABEL statement
DATASOURCE procedure, 721
MODEL procedure, 1522
LABEL= option

ATTRIBUTE statement (DATASOURCE), 721
FIXED statement (LOAN), 1369
LAG
function, 129
LAG function
MODEL procedure, 131
LAG statement
PANEL procedure, 1809
LAGDEP option
MODEL statement (AUTOREG), 371
MODEL statement (PDLREG), 1932
LAGDEP= option
MODEL statement (AUTOREG), 372
MODEL statement (PDLREG), 1933
LAGDV option
MODEL statement (AUTOREG), 371
MODEL statement (PDLREG), 1932
LAGDV= option
MODEL statement (AUTOREG), 372
MODEL statement (PDLREG), 1933
LAGGED statement
SIMLIN procedure, 2307
LAGMAX= option
MODEL statement (VARMAX), 2909
PROC STATESPACE statement, 2518
LAGRANGE option
TEST statement (ENTROPY), 818
TEST statement (MODEL), 1534
LAGS= option
CORR statement (TIMESERIES), 2670
CROSSCORR statement (TIMESERIES), 2673
DEPLAG statement (UCM), 2764
LAMBDA= option
DECOMP statement (TIMESERIES), 2674
OUTPUT statement (COUNTREG), 609
OUTPUT statement (HPCOUNTREG), 1086
SCORE statement (COUNTREG), 612
LAMBDAHI= option
BOXCOXAR macro, 181
LAMBDALO= option
BOXCOXAR macro, 181
LASR= option
PERFORMANCE statement (high-performance analytical procedures), 88
LASRSERVER= option
PERFORMANCE statement (high-performance analytical procedures), 88
LCL= option
OUTPUT statement (AUTOREG), 383
OUTPUT statement (PDLREG), 1934
LCLM= option
OUTPUT statement (AUTOREG), 383
OUTPUT statement (PDLREG), 1934
LDW option
MODEL statement (AUTOREG), 379
LEAD= option
FORECAST statement (ARIMA), 263
FORECAST statement (UCM), 2768
FORECAST statement (X12), 3179
FORECAST statement (X13), 3179
OUTPUT statement (VARMAX), 2921
PROC ESM statement, 852
PROC FORECAST statement, 954
PROC STATESPACE statement, 2520
PROC TIMEDATA statement, 2610
SEATSDECOMP statement (X12), 3196
SEATSDECOMP statement (X13), 3196
LEFTCENSORED= option
LOSS statement (HPSEVERITY), 1222
LOSS statement (SEVERITY), 2101
LEFTTRUNCATED= option
LOSS statement (HPSEVERITY), 1223
LOSS statement (SEVERITY), 2101
LENGTH
SSA statement (TIMESERIES), 2682
LENGTH option
MONTHLY statement (X11), 3099
LENGTH statement
DATASOURCE procedure, 721
LENGTH= option
ATTRIBUTE statement (DATASOURCE), 721
SEASON statement (UCM), 2778
SPLINESEASON statement (UCM), 2781
LEVEL statement
UCM procedure, 2772
LIBNAME libref SASECRSP statement, 3284
LIBNAME libref SASEFRED statement, 3386
LIBNAME libref SASEQUAN statement, 3483
LIBNAME libref SASEXCCM statement, 3522
LIBNAME libref SASEXFSD statement, 3563
LIFE= option
FIXED statement (LOAN), 1368
LIKE option
TEST statement (ENTROPY), 818
TEST statement (MODEL), 1534
LIMIT1= option
MODEL statement (QLIM), 1976
LIMIT= option
LIBNAME statement (SASEFRED), 3388
LIML option
PROC SYSLIN statement, 2568
LINK= option
HETERO statement (AUTOREG), 380
ZEROMODEL statement, 618
LIST option
FIT statement (MODEL), 1719
PROC MODEL statement, 1499, 1690
LISTALL option

PROC MODEL statement, 1500
LISTCODE option
PROC MODEL statement, 1500, 1692
LISTDEP option
PROC MODEL statement, 1500, 1699
LISTDER option
PROC MODEL statement, 1500
LISTONLY option
DIST statement (HPSEVERITY), 1221
DIST statement (SEVERITY), 2100
LJC option
COLUMNS statement (COMPUTAB), 492
ROWS statement (COMPUTAB), 494
LJUNGBOXLIMIT= option
AUTOMDL statement (X12), 3172
AUTOMDL statement (X13), 3172
LM option
TEST statement (COUNTREG), 617
TEST statement (ENTROPY), 818
TEST statement (HPCOUNTREG), 1089
TEST statement (HPPANEL), 1124
TEST statement (HPQLIM), 1168
TEST statement (MDC), 1425
TEST statement (MODEL), 1534
TEST statement (PANEL), 1828
TEST statement (QLIM), 1983
LOAN procedure, 1365
syntax, 1365
LOG function, 1680
LOG10 function, 1680
LOG2 function, 1681
LOGLIKL option
MODEL statement (AUTOREG), 372
LOGNORMALPARM= option
MODEL statement (MDC), 1416
LOGTEST
macro, 185
macro variable, 186
LONG= option
LIBNAME statement (SASEHAVR), 3437
LOSS statement
HPSEVERITY procedure, 1222
SEVERITY procedure, 2100
LOWERBOUND= option
ENDOGENOUS statement (HPQLIM), 1160, 1164
ENDOGENOUS statement (QLIM), 1972
LR option
TEST statement (COUNTREG), 617
TEST statement (ENTROPY), 818
TEST statement (HPCOUNTREG), 1089
TEST statement (HPPANEL), 1124
TEST statement (HPQLIM), 1168
TEST statement (MDC), 1425
TEST statement (MODEL), 1534
TEST statement (PANEL), 1828
TEST statement (QLIM), 1983
LRECL= option
PROC DATASOURCE statement, 715
LSCV= option
OUTLIER statement (X12), 3185
OUTLIER statement (X13), 3185
LSRUN= option
OUTLIER statement (X12), 3185
OUTLIER statement (X13), 3185
LTEBOUND= option
FIT statement (MODEL), 1519, 1669
MODEL statement (MODEL), 1669
SOLVE statement (MODEL), 1669

M= option
MODEL statement (PANEL), 1815
MODEL statement (TSCSREG), 2739
%MA macro, 1622, 1623
MA= option
ESTIMATE statement (ARIMA), 259
MACURVES statement
X11 procedure, 3097
MAPECR= option
ARIMA statement (X11), 3095
MAPREF= option
LIBNAME statement (SASEFRED), 3388
LIBNAME statement (SASEQUAN), 3484
LIBNAME statement(SASEXFSD), 3566
MARGINAL
OUTPUT statement (HPQLIM), 1165
OUTPUT statement (QLIM), 1979
MARGINALS option
MODEL statement (ENTROPY), 816
MARGINLIKE
BAYES statement (COUNTREG), 598
BAYES statement (QLIM), 1967
MARKOV option
ENTROPY procedure, 810
MARR= option
COMPARE statement (LOAN), 1375
MAXAD= option
ARM statement (LOAN), 1372
MAXADJUST= option
ARM statement (LOAN), 1372
MAXBAND= option
MODEL statement (PANEL), 1816
MAXDIFF= option
AUTOMDL statement (X12), 3172
AUTOMDL statement (X13), 3172
MAXERROR= option
PROC ESM statement, 852
PROC TIMEDATA statement, 2610

PROC TIMEID statement, 2638
PROC TIMESERIES statement, 2666
MAXERRORS= option
PROC FORECAST statement, 954
PROC MODEL statement, 1500
MAXIT=
PROC SYSLIN statement, 2568
MAXIT= option
ESTIMATE statement (ARIMA), 260
PROC STATESPACE statement, 2519
MAXITER= option
ARIMA statement (X11), 3095
ENTROPY procedure, 812
ESTIMATE statement (ARIMA), 260
ESTIMATE statement (X12), 3176
ESTIMATE statement (X13), 3176
FIT statement (MODEL), 1519
MODEL statement (AUTOREG), 379
MODEL statement (MDC), 1420
MODEL statement (PANEL), 1816
MODEL statement (PDLREG), 1933
PROC SYSLIN statement, 2568
SOLVE statement (MODEL), 1532
MAXLAG= option
CHECK statement (X12), 3174
CHECK statement (X13), 3174
IDENTIFY statement (X12), 3181
IDENTIFY statement (X13), 3181
MAXNMCMC=number
BAYES statement (COUNTREG), 596
BAYES statement (QLIM), 1965
MAXNUM= option
OUTLIER statement (ARIMA), 261
OUTLIER statement (UCM), 2774
MAXORDER= option
AUTOMDL statement (X12), 3172
AUTOMDL statement (X13), 3172
MAXPCT= option
OUTLIER statement (ARIMA), 261
OUTLIER statement (UCM), 2775
MAXR= option
ARM statement (LOAN), 1372
MAXRATE= option
ARM statement (LOAN), 1372
MAXSUBITER= option
ENTROPY procedure, 812
FIT statement (MODEL), 1519, 1555
SOLVE statement (MODEL), 1532
MAXTUNE= option
BAYES statement (COUNTREG), 598
BAYES statement (HPQLIM), 1156
BAYES statement (QLIM), 1967
MAXVARLENGTH option
PROC TIMESERIES statement, 2666
MDC procedure, 1411
syntax, 1411
MDC procedure, MODEL statement
ADDMAXIT= option, 1418
ADDRANDOM option, 1418
ADDVALUE option, 1418
ALL option, 1419
CHOICE= option, 1415
CONVERGE= option, 1415
CORRB option, 1419
COVB option, 1419
COVEST= option, 1419
HALTONSTART= option, 1416
HEV= option, 1416
INITIAL= option, 1419
ITPRINT option, 1419
LOGNORMALPARM= option, 1416
MAXITER= option, 1420
MIXED= option, 1416
NCHOICE option, 1417
NOPRINT option, 1419
NORMALEC= option, 1416
NORMALPARM= option, 1416
NSIMUL option, 1417
OPTMETHOD= option, 1420
RANDINIT option, 1417
RANDNUM= option, 1417
RANK option, 1417
RESTART= option, 1417
SAMESCALE option, 1418
SEED= option, 1418
SPSCALE option, 1418
TYPE= option, 1418
UNIFORMEC= option, 1416
UNIFORMPARM= option, 1416
UNITVARIANCE= option, 1418
MDC procedure, OUTPUT statement
OUT= option, 1424
P= option, 1424
XBETA= option, 1424
MDC procedure, PROC MDC statement
COVOUT option, 1413
DATA= option, 1413
OUTEST= option, 1413
MDC procedure, TEST statement, 1425
MDCDATA statement, 1413
MDLINFOIN= option
PROC X12 statement, 3163
PROC X13 statement, 3163
MDLINFOOUT= option
PROC X12 statement, 3163
PROC X13 statement, 3163
MDLVAR= option
PICKMDL statement (X12), 3187

PICKMDL statement (X13), 3187
MDY
function, 119
MDY function, 175
MEAN= option
MODEL statement (AUTOREG), 363
MEASURE= option
TARGET statement (SIMILARITY), 2253
MEDIAN option
FORECAST statement (ESM), 855
MELO option
PROC SYSLIN statement, 2568
MEMORYUSE option
PROC MODEL statement, 1500
METHOD option
RANDOM statement (QLIM), 1981
METHOD= option
ARIMA statement (X11), 3095
CONVERT statement (EXPAND), 896, 902
ENTROPY procedure, 812
ESTIMATE statement (ARIMA), 257
FIT statement (MODEL), 1519, 1555
MODEL statement (AUTOREG), 379
MODEL statement (PDLREG), 1933
MODEL statement (VARMAX), 2907
OUTLIER statement (X12), 3185
OUTLIER statement (X13), 3185
PICKMDL statement (X12), 3187
PICKMDL statement (X13), 3187
PROC COUNTREG statement, 594
PROC EXPAND statement, 893, 902
PROC FORECAST statement, 954
PROC HPCOUNTREG statement, 1081
PROC HPQLIM statement, 1153
QLIM procedure, 1961
MILLS
OUTPUT statement (HPQLIM), 1165
OUTPUT statement (QLIM), 1979
MINIC option
IDENTIFY statement (ARIMA), 254
PROC STATESPACE statement, 2518
MINIC= option
MODEL statement (VARMAX), 2914
MINIC=(P=) option
MODEL statement (VARMAX), 2914, 2963
MINIC=(PERROR=) option
MODEL statement (VARMAX), 2914
MINIC=(Q=) option
MODEL statement (VARMAX), 2914, 2963
MINIC=(TYPE=) option
MODEL statement (VARMAX), 2914
MINR= option
ARM statement (LOAN), 1372
MINRATE= option
ARM statement (LOAN), 1372
MINTIMESTEP= option
FIT statement (MODEL), 1519, 1670
MODEL statement (MODEL), 1670
SOLVE statement (MODEL), 1670
MINTUNE= option
BAYES statement (COUNTREG), 598
BAYES statement (HPQLIM), 1155
BAYES statement (QLIM), 1967
MINUTE
function, 176
MISSING option
CLASS statement (COUNTREG), 602
CLASS statement (HPSEVERITY), 1219
CLASS statement (SEVERITY), 2098
MISSING=option
FIT statement (MODEL), 1515
MIXED option
MODEL statement (MDC), 1416
MODE= option
DECOMP statement (TIMESERIES), 2674
OUTPUT statement (COUNTREG), 609
OUTPUT statement (HPCOUNTREG), 1087
PERFORMANCE statement (high-performance analytical procedures), 88
SCORE statement (COUNTREG), 612
X11 statement (X12), 3201
X11 statement (X13), 3201
MODEL procedure, 1489
syntax, 1489
MODEL statement
AUTOREG procedure, 361
COUNTREG procedure, 605
ENTROPY procedure, 815
HPCOUNTREG procedure, 1084
HPPANEL procedure, 1121
HPQLIM procedure, 1162
MDC procedure, 1415
PANEL procedure, 1810
PDLREG procedure, 1931
QLIM procedure, 1976
SSM procedure, 2376
SYSLIN procedure, 2570
TSCSREG procedure, 2738
UCM procedure, 2774
VARMAX procedure, 2906
MODEL= option
ARIMA statement (X11), 3095
ARIMA statement (X12), 3169
ARIMA statement (X13), 3169
FORECAST statement (ESM), 856
PROC MODEL statement, 1498, 1688
MOMENT statement
MODEL procedure, 1522

MONTH
function, 120, 176
MONTHLY statement
X11 procedure, 3098
MSTAT option
PROC PANEL statement, 1804
MTITLE= option
COLUMNS statement (COMPUTAB), 492
MU= option
ESTIMATE statement (ARIMA), 259
OUTPUT statement (COUNTREG), 609
OUTPUT statement (HPCOUNTREG), 1087
SCORE statement (COUNTREG), 612
MULTIPLIER= value
BAYES statement (QLIM), 1965
MULTIPLIER=value
BAYES statement (COUNTREG), 596

+n option
COLUMNS statement (COMPUTAB), 492
ROWS statement (COMPUTAB), 493
N2SLS option
FIT statement (MODEL), 1514
N3SLS option
FIT statement (MODEL), 1514
NAHEAD= option
SOLVE statement (MODEL), 1529, 1639
NAME option
COLUMNS statement (COMPUTAB), 492
ROWS statement (COMPUTAB), 493
NAMELEN= option
PROC HPSEVERITY statement, 1210
PROC SEVERITY statement, 2089
NBACKCAST= option
FORECAST statement (ESM), 856
FORECAST statement (X12), 3179
FORECAST statement (X13), 3179
SEATSDECOMP statement (X12), 3196
SEATSDECOMP statement (X13), 3196
NBI= option
BAYES statement (COUNTREG), 598
BAYES statement (HPQLIM), 1156
BAYES statement (QLIM), 1967
NBLOCKS= option
BLOCKSEASON statement (UCM), 2761
NBYOBS= option
PROC TIMEID statement, 2638
NCHOICE option
MODEL statement (MDC), 1417
NDEC= option
MONTHLY statement (X11), 3099
PROC MODEL statement, 1500
QUARTERLY statement (X11), 3104
SSPAN statement (X11), 3105
NDRAW option
FIT statement (MODEL), 1514
NDRAW= option
QLIM procedure, 1961
NEST statement
MDC procedure, 1420
NESTIT option
FIT statement (MODEL), 1519, 1554
NEWEYWEST option
MODEL statement (PANEL), 1818
NEWTON option
SOLVE statement (MODEL), 1530
NKNOTS= option
SPLINEREG statement (UCM), 2780
NLAG= option
CORR statement (TIMESERIES), 2671
CROSSCORR statement (TIMESERIES), 2672
IDENTIFY statement (ARIMA), 254
MODEL statement (AUTOREG), 361
MODEL statement (PDLREG), 1933
NLAGS= option
PROC FORECAST statement, 953
NLAMBDA= option
BOXCOXAR macro, 181
NLC option
COINTEG statement (VARMAX), 2901
NLOPTIONS statement
AUTOREG procedure, 382
COUNTREG procedure, 608
HPSEVERITY procedure, 1224
MDC procedure, 1423
QLIM procedure, 1978
SEVERITY procedure, 2103
UCM procedure, 2774
VARMAX procedure, 2921, 2976
NMC= option
BAYES statement (COUNTREG), 598
BAYES statement (HPQLIM), 1156
BAYES statement (QLIM), 1967
NMCPRIOR= option
BAYES statement (QLIM), 1967
NMSIMP option
METHOD= option, PROC COUNTREG statement, 594
NNODES= option
PERFORMANCE statement (high-performance analytical procedures), 88
NO2SLS option
FIT statement (MODEL), 1514
NO3SLS option
FIT statement (MODEL), 1514
NOAPPLY= option
REGRESSION statement (X12), 3190
REGRESSION statement (X13), 3190

NOCENTER option
PROC STATESPACE statement, 2518
NOCLPRINT option
PROC HPSEVERITY statement, 1211
PROC SEVERITY statement, 2089
NOCOMPRINT option
COMPARE statement (LOAN), 1376
NOCONST option
HETERO statement (AUTOREG), 382
NOCONSTANT option
ESTIMATE statement (ARIMA), 257
NOCORR option
PROC COPULA statement, 544
NOCURRENTX option
MODEL statement (VARMAX), 2908
NODES option
PERFOMRANCE statement (HPCOPULA), 1061
PERFOMRANCE statement (HPPANEL), 1123
PERFOMRANCE statement (HPQLIM), 1166
NODES= option
PERFORMANCE statement (high-performance analytical procedures), 88
PERFORMANCE statement (HPCOUNTREG), 1087
NODF option
ESTIMATE statement (ARIMA), 257
NODIFFS option
MODEL statement (PANEL), 1816
NOEST option
AUTOREG statement (UCM), 2759
BLOCKSEASON statement (UCM), 2761
CYCLE statement (UCM), 2763
DEPLAG statement (UCM), 2764
ESTIMATE statement (ARIMA), 259
IRREGULAR statement (UCM), 2770, 2771
LEVEL statement (UCM), 2773
PROC STATESPACE statement, 2519
RANDOMREG statement (UCM), 2776
SEASON statement (UCM), 2778
SLOPE statement (UCM), 2779
SPLINEREG statement (UCM), 2780
SPLINESEASON statement (UCM), 2781
NOESTIM option
MODEL statement (PANEL), 1815
NOGENGMMV option
FIT statement (MODEL), 1514
NOINCLUDE option
PROC SYSLIN statement, 2568
NOINT option
ARIMA statement (X11), 3096
AUTOMODL statement (X12), 3172
AUTOMODL statement (X13), 3172
ESTIMATE statement (ARIMA), 257
INSTRUMENTS statement (MODEL), 1522
MODEL statement (AUTOREG), 361, 363
MODEL statement (COUNTREG), 605
MODEL statement (ENTROPY), 816
MODEL statement (HPCOUNTREG), 1085
MODEL statement (HPPANEL), 1122
MODEL statement (HPQLIM), 1163
MODEL statement (PANEL), 1815
MODEL statement (PDLREG), 1933
MODEL statement (QLIM), 1976
MODEL statement (SYSLIN), 2571
MODEL statement (TSCSREG), 2739
MODEL statement (VARMAX), 2908
NOINTERCEPT option
INSTRUMENTS statement (MODEL), 1522
NOLEVELS option
MODEL statement (PANEL), 1816
NOLS option
ESTIMATE statement (ARIMA), 260
NOMEAN option
MODEL statement (TSCSREG), 2739
NOMISS option
IDENTIFY statement (ARIMA), 255
MODEL statement (AUTOREG), 380
NONE= option
PLOTS option, PROC COUNTREG statement, 594
NONORMALIZE option
PROC COUNTREG statement, 591
WEIGHT statement (COUNTREG), 618
WEIGHT statement (HPCOUNTREG), 1090
WEIGHT statement (HPQLIM), 1169
WEIGHT statement (QLIM), 1984
NOOLS option
FIT statement (MODEL), 1514
NOOUTALL option
FORECAST statement (ARIMA), 263
PROC ESM statement, 852
NOP option
FIXED statement (LOAN), 1371
NOPRINT option
ARIMA statement (X11), 3096
COLUMNS statement (COMPUTAB), 492
ESTIMATE statement (ARIMA), 258
FIXED statement (LOAN), 1371
FORECAST statement (ARIMA), 263
IDENTIFY statement (ARIMA), 255
MODEL statement (AUTOREG), 372
MODEL statement (MDC), 1419
MODEL statement (PANEL), 1827
MODEL statement (PDLREG), 1933
MODEL statement (SYSLIN), 2571
MODEL statement (TSCSREG), 2739
MODEL statement (VARMAX), 2909

OUTPUT statement (VARMAX), 2922
PROC COMPUTAB statement, 490
PROC COPULA statement, 544
PROC COUNTREG statement, 592
PROC ENTROPY statement, 811
PROC HPCDM statement, 1009
PROC HPCOUNTREG statement, 1079, 1085
PROC HPPANEL statement, 1120
PROC HPQLIM statement, 1150
PROC HPSEVERITY statement, 1211
PROC MODEL statement, 1501
PROC QLIM statement, 1959
PROC SEVERITY statement, 2089
PROC SIMLIN statement, 2305
PROC SSM statement, 2370
PROC STATESPACE statement, 2517
PROC SYSLIN statement, 2569
PROC UCM statement, 2756
PROC VARMAX statement (VARMAX), 3018
PROC X11 statement, 3094
PROC X12 statement, 3163
PROC X13 statement, 3163
ROWS statement (COMPUTAB), 494
SSPAN statement (X11), 3105
NOPROFILE
ESTIMATE statement (UCM), 2765
NORED option
PROC SIMLIN statement, 2305
NORMAL
PRIOR statement (COUNTREG), 610
PRIOR statement (HPQLIM), 1166
PRIOR statement (QLIM), 1979
NORMAL option
ERRORMODEL statement (MODEL), 1508
FIT statement (MODEL), 1518
MODEL statement (AUTOREG), 372
NORMALEC= option
MODEL statement (MDC), 1416
NORMALIZE= option
COINTEG statement (VARMAX), 2869, 2901, 3033
INPUT statement (SIMILARITY), 2248
TARGET statement (SIMILARITY), 2253
NORMALPARM= option
MODEL statement (MDC), 1416
NORTR option
PROC COMPUTAB statement, 490
NOSTABLE option
ESTIMATE statement (ARIMA), 260
NOSTORE option
PROC MODEL statement, 1498
NOSUM
TABLES statement (X12), 3197
TABLES statement (X13), 3197
NOSUMMARYPRINT option
FIXED statement (LOAN), 1371
NOSUMPR option
FIXED statement (LOAN), 1371
NOTFSTABLE option
ESTIMATE statement (ARIMA), 260
NOTHREADS option
PERFORMANCE statement (COUNTREG), 610
NOTRANS option
PROC COMPUTAB statement, 490
NOTRANSPOSE option
PROC COMPUTAB statement, 490
NOTRIMMISS option
PROC X12 statement, 3163
PROC X13 statement, 3163
NOTSORTED option
ID statement (ESM), 859
ID statement (SIMILARITY), 2247
ID statement (TIMEDATA), 2614
ID statement (TIMESERIES), 2677
TIMEID procedure, 2640
NOZERO option
COLUMNS statement (COMPUTAB), 492
ROWS statement (COMPUTAB), 494
NPARMS= option
CORR statement (TIMESERIES), 2671
NPERIODS option
SSA statement (TIMESERIES), 2682
NPERIODS= option
DECOMP statement (TIMESERIES), 2674
TREND statement (TIMESERIES), 2684
NPERTURBEDSAMPLES= option
PROC HPCDM statement, 1009
NPREOBS option
FIT statement (MODEL), 1514
NRA option
METHOD= option, PROC COUNTREG statement, 594
NREPLICATES= option
PROC HPCDM statement, 1009
NRRIDG option
METHOD= option, PROC COUNTREG statement, 594
NSEASON= option
MODEL statement (VARMAX), 2908
NSIMUL option
MODEL statement (MDC), 1417
NSSTART= MAX option
PROC FORECAST statement, 955
NSSTART= option
PROC FORECAST statement, 955
NSTART= MAX option
PROC FORECAST statement, 955
NSTART= option

PROC FORECAST statement, 955
NTHREADS option
PERFOMRANCE statement (HPCOPULA), 1061
PERFOMRANCE statement (HPPANEL), 1123
PERFOMRANCE statement (HPQLIM), 1166
NTHREADS= option
PERFORMANCE statement (COUNTREG), 610
PERFORMANCE statement (high-performance analytical procedures), 89
PERFORMANCE statement (HPCOUNTREG), 1087
NTRDS= option
BAYES statement (COUNTREG), 598
BAYES statement (QLIM), 1967
NTU= option
BAYES statement (COUNTREG), 598
BAYES statement (HPQLIM), 1156
BAYES statement (QLIM), 1968
NU= option
OUTPUT statement (COUNTREG), 609
OUTPUT statement (HPCOUNTREG), 1087
SCORE statement (COUNTREG), 612
NVAR= option
PROC COPULA statement, 544, 546
NVDRAW option
FIT statement (MODEL), 1514
NWKDOM
function, 176

OBJECTIVE= option
PROC HPSEVERITY statement, 1217
PROC SEVERITY statement, 2095
OBSERVED= option
CONVERT statement (EXPAND), 896, 900
PROC EXPAND statement, 893
OFFSET= option
BLOCKSEASON statement (UCM), 2761
LIBNAME statement (SASEFRED), 3388
MODEL statement (COUNTREG), 606
MODEL statement (HPCOUNTREG), 1085
SCALEMODEL statement (HPSEVERITY), 1229
SCALEMODEL statement (SEVERITY), 2107
SPLINESEASON statement (UCM), 2781
ZEROMODEL statement, 619
OL option
ROWS statement (COMPUTAB), 494
OLS option
FIT statement (MODEL), 1514, 1704
PROC SYSLIN statement, 2568
ONEPASS option
SOLVE statement (MODEL), 1531
ONLY option
PLOTS option, PROC COUNTREG statement, 593
OPTIMIZE option
SOLVE statement (MODEL), 1530
OPTIONS option
PROC COMPUTAB statement, 490
OPTMETHOD= option
MODEL statement (AUTOREG), 380
MODEL statement (MDC), 1420
ORDER= option
CLASS statement (COUNTREG), 602
CLASS statement (HPSEVERITY), 1218
CLASS statement (SEVERITY), 2096
ENDOGENOUS statement (HPQLIM), 1159, 1163
ENDOGENOUS statement (QLIM), 1971
PROC SIMILARITY statement, 2242
ORIENTATION= option
LIBNAME statement(SASEXFSD), 3567
OTHERWISE, 1687
OUT = option
FlatData statement (PANEL), 1807
OUT1STEP option
FORECAST statement (X12), 3180
FORECAST statement (X13), 3180
PROC FORECAST statement, 956
OUT= option
BOXCOXAR macro, 181
DFTEST macro, 184
ENTROPY procedure, 811
FIT statement (MODEL), 1515, 1632, 1706
FIXED statement (LOAN), 1371, 1379
FORECAST command (TSFS), 3775
FORECAST statement (ARIMA), 263, 284
LOGTEST macro, 186
OUTPUT statement (AUTOREG), 382
OUTPUT statement (COUNTREG), 609
OUTPUT statement (HPCDM), 1014
OUTPUT statement (HPCOUNTREG), 1086
OUTPUT statement (HPPANEL), 1122
OUTPUT statement (HPQLIM), 1165
OUTPUT statement (MDC), 1424
OUTPUT statement (PANEL), 1827
OUTPUT statement (PDLREG), 1934
OUTPUT statement (QLIM), 1979
OUTPUT statement (SIMLIN), 2307
OUTPUT statement (SYSLIN), 2585
OUTPUT statement (VARMAX), 2922, 3017
OUTPUT statement (X11), 3102, 3119
OUTPUT statement (X12), 3186
OUTPUT statement (X13), 3186
OUTSUM statement (HPCDM), 1015
PROC ARIMA statement, 252
PROC COMPUTAB statement, 491

PROC DATASOURCE statement, 715, 723
PROC ESM statement, 852
PROC EXPAND statement, 892, 918
PROC FORECAST statement, 955, 968
PROC SIMILARITY statement, 2242
PROC SIMLIN statement, 2312
PROC SPECTRA statement, 2334, 2339
PROC STATESPACE statement, 2520, 2534
PROC SYSLIN statement, 2567
PROC TIMEDATA statement, 2610
PROC TIMESERIES statement, 2666
SCORE statement (COUNTREG), 612
SEATSDECOMP statement (X12), 3196
SEATSDECOMP statement (X13), 3196
SOLVE statement (MODEL), 1528, 1643, 1672
TEST statement (ENTROPY), 818
TEST statement (MODEL), 1534
OUT= table names
OUTPUT statement (X12), 3186
OUTPUT statement (X13), 3186
OUTACTUAL option
FIT statement (MODEL), 1515
PROC FORECAST statement, 955
SOLVE statement (MODEL), 1528
OUTALL option
FIT statement (MODEL), 1515
PROC FORECAST statement, 955
SOLVE statement (MODEL), 1528
OUTALL= option
PROC DATASOURCE statement, 716, 727
OUTAR= option
PROC STATESPACE statement, 2518, 2535
OUTARRAY= option
PROC TIMEDATA statement, 2611
OUTARRAYS statement
TIMEDATA procedure, 2615
OUTBACKCAST option
FORECAST statement (X12), 3180
FORECAST statement (X13), 3180
OUTBY= option
PROC DATASOURCE statement, 716, 726
OUTBYID= option
OUTSCORELIB statement (HPSEVERITY), 1226
OUTSCORELIB statement (SEVERITY), 2104
OUTCAT= option
PROC MODEL statement, 1498
OUTCDF= option
PROC HPSEVERITY statement, 1211
PROC SEVERITY statement, 2089
OUTCOMP= option
COMPARE statement (LOAN), 1376, 1379
OUTCONT= option
PROC DATASOURCE statement, 716, 725
OUTCORR option
ESTIMATE statement (ARIMA), 258
PROC HPPANEL statement, 1121
PROC PANEL statement, 1802
PROC TSCSREG statement, 2736
OUTCORR= option
PROC TIMESERIES statement, 2666
OUTCOV option
ENTROPY procedure, 811
ESTIMATE statement (ARIMA), 259
ESTIMATE statement (MODEL), 1510
FIT statement (MODEL), 1515
PROC HPPANEL statement, 1120
PROC PANEL statement, 1801
PROC SYSLIN statement, 2567
PROC TSCSREG statement, 2736
PROC VARMAX statement, 2893, 3019
OUTCOV3 option
PROC SYSLIN statement, 2567
OUTCOV= option
IDENTIFY statement (ARIMA), 255, 285
OUTCROSSCORR= option
PROC TIMESERIES statement, 2666
OUTDECOMP= option
PROC TIMESERIES statement, 2666
OUTERRORS option
SOLVE statement (MODEL), 1528
OUTEST= option
ENTROPY procedure, 811
ENTROPY statement, 831
ESTIMATE statement (ARIMA), 258, 286
ESTIMATE statement (MODEL), 1510
ESTIMATE statement (UCM), 2765
FIT statement (MODEL), 1515, 1633
PROC AUTOREG statement, 359
PROC COUNTREG statement, 592
PROC ESM statement, 852
PROC EXPAND statement, 892, 919
PROC FORECAST statement, 955, 969
PROC HPCOUNTREG statement, 1079
PROC HPPANEL statement, 1120, 1134
PROC HPQLIM statement, 1150
PROC HPSEVERITY statement, 1211
PROC MDC statement, 1413
PROC PANEL statement, 1801, 1894
PROC QLIM statement, 1959
PROC SEVERITY statement, 2089
PROC SIMLIN statement, 2305, 2311
PROC SYSLIN statement, 2567, 2585
PROC TSCSREG statement, 2736
PROC VARMAX statement, 2893, 3019
OUTESTALL option
PROC FORECAST statement, 956
OUTESTTHEIL option

PROC FORECAST statement, 956
OUTEVENT= option
PROC DATASOURCE statement, 716, 727
OUTEXTRAP option
PROC X11 statement, 3093
OUTFITSTATS option
PROC FORECAST statement, 956
OUTFOR= option
FORECAST statement (UCM), 2768
PROC ESM statement, 853
OUTFORECAST option
FORECAST statement (X12), 3180
FORECAST statement (X13), 3180
X11 statement (X12), 3201
X11 statement (X13), 3201
OUTFREQ= option
PROC TIMESERIES statement, 2666
OUTFULL option
PROC FORECAST statement, 956
OUTHT= option
GARCH statement, 2902
PROC VARMAX statement, 3021
OUTINTERVAL= option
PROC TIMEID statement, 2638
OUTINTERVALDETAILS= option
PROC TIMEID statement, 2638
OUTL= option
ENTROPY procedure, 811
ENTROPY statement, 832
OUTLAGS option
FIT statement (MODEL), 1515
SOLVE statement (MODEL), 1528
OUTLIB= option
OUTSCORELIB statement (HPSEVERITY), 1225
OUTSCORELIB statement (SEVERITY), 2103
OUTLIER statement
UCM procedure, 2774
X12 procedure, 3183
X13 procedure, 3183
OUTLIMIT option
PROC FORECAST statement, 956
OUTMEASURE= option
PROC SIMILARITY statement, 2242
OUTMODEL= option
ESTIMATE statement (ARIMA), 259, 289
PROC MODEL statement, 1498, 1688
PROC STATESPACE statement, 2519, 2536
OUTMODELINFO= option
PROC HPSEVERITY statement, 1211
PROC SEVERITY statement, 2090
OUTOBJVALS option
SOLVE statement (MODEL), 1528
OUTP= option
ENTROPY procedure, 811
ENTROPY statement, 831
OUTPARM=option
PROC PANEL statement, 1805
OUTPARMS= option
FIT statement (MODEL), 1634
PROC MODEL statement, 1497, 1629
OUTPATH= option
PROC SIMILARITY statement, 2242
OUTPOST= option
BAYES statement (COUNTREG), 598
BAYES statement (HPQLIM), 1156
BAYES statement (QLIM), 1968
OUTPREDICT option
FIT statement (MODEL), 1515
SOLVE statement (MODEL), 1529
OUTPRIOR= option
BAYES statement (QLIM), 1968
OUTPROCINFO= option
PROC ESM statement, 853
PROC TIMEDATA statement, 2611
PROC TIMESERIES statement, 2666
OUTPUT
OUT=, 431
OUTPUT statement
AUTOREG procedure, 382
COUNTREG procedure, 608
HPCDM procedure, 1014
HPCOUNTREG procedure, 1085
HPPANEL procedure, 1122
HPQLIM procedure, 1165
PANEL procedure, 1827
PDLREG procedure, 1934
PROC HPPANEL statement, 1134
PROC PANEL statement, 1894
QLIM procedure, 1978
SIMLIN procedure, 2307
SSM procedure, 2377
SYSLIN procedure, 2572
VARMAX procedure, 2921
X11 procedure, 3101
X12 procedure, 3186
X13 procedure, 3186
OUTPUT= option
LIBNAME statement (SASEFRED), 3388
OUTRESID option
FIT statement (MODEL), 1515, 1706
PROC FORECAST statement, 956
SOLVE statement (MODEL), 1529
OUTS= option
ENTROPY procedure, 811
FIT statement (MODEL), 1516, 1554, 1634
OUTSCALAR= option
PROC TIMEDATA statement, 2611

OUTSCALARS statement
TIMEDATA procedure, 2615
OUTSCORELIB statement
HPSEVERITY procedure, 1225
SEVERITY procedure, 2103
OUTSEASON= option
PROC TIMESERIES statement, 2666
OUTSELECT= option
PROC DATASOURCE statement, 716
OUTSELECT=OFF option
LIBNAME statement (SASEHAVR), 3437
OUTSELECT=ON option
LIBNAME statement (SASEHAVR), 3437
OUTSEQUENCE= option
PROC SIMILARITY statement, 2242
OUTSN= option
FIT statement (MODEL), 1516
OUTSPAN= option
PROC X11 statement, 3094, 3120
VAR statement (X11), 3120
OUTSPECTRA= option
PROC TIMESERIES statement, 2666
OUTSSA= option
PROC TIMESERIES statement, 2666
OUTSSCP= option
PROC SYSLIN statement, 2567, 2586
OUTSTAT= option
DFTEST macro, 184
ESTIMATE statement (ARIMA), 259, 290
PROC ESM statement, 853
PROC HPSEVERITY statement, 1211
PROC SEVERITY statement, 2090
PROC VARMAX statement, 2893, 3023
PROC X12 statement, 3164
PROC X13 statement, 3164
OUTSTAT=option
PROC PANEL statement, 1805
OUTSTB= option
PROC X11 statement, 3094, 3120
OUTSTD option
PROC FORECAST statement, 956
OUTSTORE= option
PROC HPSEVERITY statement, 1211
PROC SEVERITY statement, 2090
OUTSUM statement
HPCDM procedure, 1015
OUTSUM= option
FIXED statement (LOAN), 1371
PROC ESM statement, 853
PROC LOAN statement, 1367, 1380
PROC SIMILARITY statement, 2243
PROC TIMEDATA statement, 2611
PROC TIMESERIES statement, 2667
OUTSUSED= option
ENTROPY procedure, 811
FIT statement (MODEL), 1516, 1554, 1635
OUTTDR= option
PROC X11 statement, 3094, 3121
OUTTRANS= option
PROC PANEL statement, 1895
OUTTRANS=option
PROC PANEL statement, 1801
OUTTREND= option
PROC TIMESERIES statement, 2667
OUTUNWGTRESID option
FIT statement (MODEL), 1516
OUTV= option
FIT statement (MODEL), 1516, 1631, 1635
OUTVARS statement
MODEL procedure, 1524
OUTVIOLATIONS option
SOLVE statement (MODEL), 1529
OUTXML= option
LIBNAME statement (SASEFRED), 3388
LIBNAME statement (SASEQUAN), 3484
LIBNAME statement(SASEXFSD), 3566
OVDIFCR= option
ARIMA statement (X11), 3096
OVERID option
ENDOGENOUS statement (QLIM), 1974
MODEL statement (SYSLIN), 2571
OVERPRINT option
ROWS statement (COMPUTAB), 494

P option
IRREGULAR statement (UCM), 2772
PROC SPECTRA statement, 2334
P= option
ESTIMATE statement (ARIMA), 258
FIXED statement (LOAN), 1368
GARCH statement, 2902
IDENTIFY statement (ARIMA), 255
MODEL statement (AUTOREG), 362
MODEL statement (VARMAX), 2913
OUTPUT statement (AUTOREG), 383
OUTPUT statement (MDC), 1424
OUTPUT statement (PANEL), 1827
OUTPUT statement (PDLREG), 1934
OUTPUT statement (SIMLIN), 2307
PAGE option
COLUMNS statement (COMPUTAB), 492
ROWS statement (COMPUTAB), 494
PANEL procedure, 1798
syntax, 1798
PARAM= option
CLASS statement (COUNTREG), 602
CLASS statement (HPSEVERITY), 1220
CLASS statement (SEVERITY), 2098

PARAMETER= option
COUNTREG statement (COUNTREG), 606, 1085
MODEL statement (COUNTREG), 606, 1085
PARAMETERS option
SHOW statement (COUNTREG), 614
PARAMETERS statement
MODEL procedure, 1524, 1673
PARKS option
MODEL statement (PANEL), 1814
MODEL statement (TSCSREG), 2738
PARMS statement
SSM procedure, 2378
PARMS= option
FIT statement (MODEL), 1512
PARMSDATA= option
PROC MODEL statement, 1497, 1629
SOLVE statement (MODEL), 1529
PARMTOL= option
PROC STATESPACE statement, 2519
PARTIAL option
MODEL statement (AUTOREG), 372
MODEL statement (PDLREG), 1933
PASS= option
LIBNAME statement(SASEXFSD), 3567
PASTMIN= option
PROC STATESPACE statement, 2519
PATH= option
TARGET statement (SIMILARITY), 2254
PAYMENT= option
FIXED statement (LOAN), 1368
PCHOW= option
FIT statement (MODEL), 1518, 1605
MODEL statement (AUTOREG), 372
PCTLDEF= option
PROC HPCDM statement, 1010
PCTLNAME= option
OUTSUM statement (HPCDM), 1017
PCTLNDEC= option
OUTSUM statement (HPCDM), 1018
PCTLPTS= option
OUTSUM statement (HPCDM), 1017
PDATA= option
ENTROPY procedure, 810
ENTROPY statement, 830
%PDL macro, 1626
PDLREG procedure, 1929
syntax, 1929
PDWEIGHTS statement
X11 procedure, 3102
PERFORMANCE statement
COUNTREG procedure, 610
high-performance analytical procedures, 87
HPCDM procedure, 1018
HPCOPULA procedure, 1061
HPCOUNTREG procedure, 1087
HPPANEL procedure, 1123
HPQLIM procedure, 1166
HPSEVERITY procedure, 1227
UCM procedure, 2775
PERIOD= option
CYCLE statement (UCM), 2763
LIBNAME statement(SASEXFSD), 3566
PERIODOGRAM option
PROC X12 statement, 3164
PROC X13 statement, 3164
PERMCO= option
LIBNAME statement (SASECRSP), 3286
PERMNO= option
LIBNAME statement (SASECRSP), 3285
LIBNAME statement (SASEXCCM), 3523
PERROR= option
IDENTIFY statement (ARIMA), 255
PERTURBOUT option
OUTPUT statement (HPCDM), 1015
PH option
PROC SPECTRA statement, 2334
PHI option
MODEL statement (PANEL), 1827
MODEL statement (TSCSREG), 2739
PHI= option
DEPLAG statement (UCM), 2764
PICKMDL statement
X12 procedure, 3187
X13 procedure, 3187
PLOT
HAXIS=, 113
PLOT option
AUTOREG statement (UCM), 2759
BLOCKSEASON statement (UCM), 2761
CYCLE statement (UCM), 2763
ESTIMATE statement (ARIMA), 258
ESTIMATE statement (UCM), 2765
FORECAST statement (UCM), 2768
IRREGULAR statement (UCM), 2770
MODEL statement (SYSLIN), 2571
PROC ARIMA statement, 250
PROC UCM statement, 2756
PROC X12 statement, 3164
PROC X13 statement, 3164
PROCSSM statement, 2370
RANDOMREG statement (UCM), 2776
SEASON statement (UCM), 2778
SLOPE statement (UCM), 2779
SPLINEREG statement (UCM), 2780
SPLINESEASON statement (UCM), 2781
PLOT= option
PROC ESM statement, 853

PROC TIMEID statement, 2638
PLOTS option
HPQLIM statement (HPQLIM), 1153
PROC ARIMA statement, 250
PROC AUTOREG statement, 359
PROC COUNTREG statement, 593
PROC ENTROPY statement, 811
PROC MODEL statement, 1497
PROC PANEL statement, 1802
PROC UCM statement, 2756
PROC X12 statement, 3164
PROC X13 statement, 3164
PROCSSM statement, 2370
QLIM statement (QLIM), 1961
PLOTS= option
PROC EXPAND statement, 894
PROC HPCDM statement, 1010
PROC HPSEVERITY statement, 1212
PROC SEVERITY statement, 2090
PROC SIMILARITY statement, 2243
PROC TIMEDATA statement, 2611
PROC TIMESERIES statement, 2667
PM= option
OUTPUT statement (AUTOREG), 384
OUTPUT statement (PDLREG), 1934
PMFACTOR= option
MONTHLY statement (X11), 3100
PNT= option
FIXED statement (LOAN), 1369
PNTPCT= option
FIXED statement (LOAN), 1370
POINTPCT= option
FIXED statement (LOAN), 1370
POINTS= option
FIXED statement (LOAN), 1369
POISSON option
ERRORMODEL statement (MODEL), 1508
POOLED option
MODEL statement (HPPANEL), 1121
MODEL statement (PANEL), 1814
POOLTEST option
MODEL statement (PANEL), 1826
POWER= option
TRANSFORM statement (X12), 3197
TRANSFORM statement (X13), 3197
PRC= option
FIXED statement (LOAN), 1370
PRED= option
OUTPUT statement (COUNTREG), 609
OUTPUT statement (HPCOUNTREG), 1086
SCORE statement (COUNTREG), 613
PREDEFINED= option
ADJUST statement (X12), 3169
ADJUST statement (X13), 3169
PREDEFINED statement (X12), 3190
PREDEFINED statement (X13), 3190
PREDICTED
OUTPUT statement (HPPANEL), 1123
OUTPUT statement (HPQLIM), 1165
OUTPUT statement (QLIM), 1979
PREDICTED= option
OUTPUT statement (AUTOREG), 383
OUTPUT statement (PANEL), 1827
OUTPUT statement (PDLREG), 1934
OUTPUT statement (SIMLIN), 2307
OUTPUT statement (SYSLIN), 2572
PREDICTEDM= option
OUTPUT statement (AUTOREG), 384
OUTPUT statement (PDLREG), 1934
PREDPROB option
PLOTS option, PROC COUNTREG statement, 594
PREDPROFILE option
PLOTS option, PROC COUNTREG statement, 594
PREPAYMENTS= option
FIXED statement (LOAN), 1370
PRICE= option
FIXED statement (LOAN), 1370
PRIMAL option
ENTROPY procedure, 812
PRINT option
AUTOREG statement (UCM), 2760
BLOCKSEASON statement (UCM), 2761
COMPONENT statement (SSM), 2373
CYCLE statement (UCM), 2763
ESTIMATE statement (UCM), 2766
EVAL statement (SSM), 2375
FORECAST statement (UCM), 2768
IRREGULAR statement (SSM), 2376
IRREGULAR statement (UCM), 2770
LEVEL statement (UCM), 2773
MODEL statement (SSM), 2377
OUTLIER statement (UCM), 2775
PROC STATESPACE statement, 2520
RANDOMREG statement (UCM), 2776
SEASON statement (UCM), 2778
SLOPE statement (UCM), 2779
SPLINESEASON statement (UCM), 2782
SSPAN statement (X11), 3106
STATE statement (SSM), 2381
STEST statement (SYSLIN), 2576
TEST statement (SYSLIN), 2577
TREND statement (SSM), 2388
PRINT= option
AUTOMDL statement (X12), 3172
AUTOMDL statement (X13), 3172
BOXCOXAR macro, 181

CHECK statement (X12), 3174
CHECK statement (X13), 3174
LOGTEST macro, 186
MODEL statement (VARMAX), 2910
PROC HPCDM statement, 1011
PROC HPSEVERITY statement, 1213
PROC SEVERITY statement, 2091
PROC SIMILARITY statement, 2243
PROC TIMEDATA statement, 2611
PROC TIMEID statement, 2639
PROC TIMESERIES statement, 2668
PRINT=(CORRB) option
MODEL statement (VARMAX), 2910
PRINT=(CORRX) option
MODEL statement (VARMAX), 2910
PRINT=(CORRY) option
MODEL statement (VARMAX), 2910, 2958
PRINT=(COVB) option
MODEL statement (VARMAX), 2910
PRINT=(COVPE) option
MODEL statement (VARMAX), 2910, 2954
PRINT=(COVX) option
MODEL statement (VARMAX), 2910
PRINT=(COVY) option
MODEL statement (VARMAX), 2911
PRINT=(DECOMPOSE) option
MODEL statement (VARMAX), 2911, 2956
PRINT=(DIAGNOSE) option
MODEL statement (VARMAX), 2911
PRINT=(DYNAMIC) option
MODEL statement (VARMAX), 2911, 2941
PRINT=(ESTIMATES) option
MODEL statement (VARMAX), 2911
PRINT=(IARR) option
MODEL statement (VARMAX), 2869, 2911
PRINT=(IMPULSE) option
MODEL statement (VARMAX), 2947
PRINT=(IMPULSE=) option
MODEL statement (VARMAX), 2911
PRINT=(IMPULSX) option
MODEL statement (VARMAX), 2944
PRINT=(IMPULSX=) option
MODEL statement (VARMAX), 2912
PRINT=(PARCOEF) option
MODEL statement (VARMAX), 2912, 2959
PRINT=(PCANCORR) option
MODEL statement (VARMAX), 2912, 2962
PRINT=(PCORR) option
MODEL statement (VARMAX), 2912, 2961
PRINT=(ROOTS) option
MODEL statement (VARMAX), 2912, 2964
PRINT=(YW) option
MODEL statement (VARMAX), 2913
PRINT=option
PROC ESM statement, 854
PRINTALL option
ARIMA statement (X11), 3096
ESTIMATE statement (ARIMA), 260
FIT statement (MODEL), 1518
FORECAST statement (ARIMA), 263
MODEL statement, 608, 1085
MODEL statement (VARMAX), 2909
PROC COPULA statement, 544
PROC HPCOUNTREG statement, 1079
PROC HPQLIM statement, 1150
PROC MODEL statement, 1501
PROC QLIM statement, 1959
PROC SSM statement, 2371
PROC UCM statement, 2759
SOLVE statement (MODEL), 1532
SSPAN statement (X11), 3106
PRINTDETAILS option
PROC ESM statement, 854
PROC SIMILARITY statement, 2244
PROC TIMESERIES statement, 2668
PRINTERR option
ESTIMATE statement (X12), 3176
ESTIMATE statement (X13), 3176
PRINTFIXED option
MODEL statement (HPPANEL), 1122
MODEL statement (PANEL), 1827
PRINTFORM= option
MODEL statement (VARMAX), 2910, 2944
PRINTFP option
ARIMA statement (X11), 3096
PRINTOUT= option
MONTHLY statement (X11), 3100
PROC STATESPACE statement, 2518
QUARTERLY statement (X11), 3104
PRINTREG option
IDENTIFY statement (X12), 3181
IDENTIFY statement (X13), 3181
PRIOR option
MODEL statement (VARMAX), 2918
PRIOR statement
COUNTREG procedure, 610
HPQLIM procedure, 1166
QLIM procedure, 1979
PRIOR=(IVAR) option
MODEL statement (VARMAX), 2918
PRIOR=(LAMBDA=) option
MODEL statement (VARMAX), 2918
PRIOR=(MEAN=) option
MODEL statement (VARMAX), 2918
PRIOR=(NREP=) option
MODEL statement (VARMAX), 2919
PRIOR=(THETA=) option
MODEL statement (VARMAX), 2919

PRIORS statement
ENTROPY procedure, 816
PRL= option
FIT statement (MODEL), 1512, 1606
PROB
OUTPUT statement (HPQLIM), 1165
OUTPUT statement (QLIM), 1979
PROB= option
OUTPUT statement (COUNTREG), 609
OUTPUT statement (HPCOUNTREG), 1086
SCORE statement (COUNTREG), 613
PROBALL
OUTPUT statement (HPQLIM), 1165
OUTPUT statement (QLIM), 1979
PROBCOUNT option
OUTPUT statement (COUNTREG), 609
OUTPUT statement (HPCOUNTREG), 1086
SCORE statement (COUNTREG), 613
PROBDF
function, 187
macro, 187
PROBOBSERVED= option
LOSS statement (HPSEVERITY), 1223
LOSS statement (SEVERITY), 2101
PROBZERO= option
OUTPUT statement (COUNTREG), 609
OUTPUT statement (HPCOUNTREG), 1086
SCORE statement (COUNTREG), 613
PROC ARIMA statement, 250
PROC AUTOREG
OUTEST=, 431
PROC AUTOREG statement, 359
PROC COMPUTAB
NOTRANS, 498
OUT=, 507
PROC COMPUTAB statement, 489
PROC COPULA statement
COPULA procedure, 540
PROC DATASOURCE statement, 714
PROC ENTROPY statement, 810
PROC ESM statement, 852
PROC EXPAND statement, 892
PROC FORECAST statement, 953
PROC HPCDM statement, 1008, *see* HPCDM procedure
PROC HPCOPULA statement
HPCOPULA procedure, 1059
PROC HPPANEL statement, 1120
PROC HPSEVERITY statement, 1209
PROC LOAN statement, 1367
PROC MDC statement, 1413
PROC MODEL statement, 1496
PROC PANEL statement, 1801
PROC PDLREG statement, 1931
PROC SEVERITY statement, 2087
PROC SIMILARITY statement, 2241
PROC SIMLIN statement, 2305
PROC SPECTRA statement, 2333
PROC SSM statement, 2370, *see* SSM procedure
PROC STATESPACE statement, 2517
PROC SYSLIN statement, 2567
PROC TIMEDATA statement, 2610
PROC TIMEID statement, 2638
PROC TIMESERIES statement, 2665
PROC TSCSREG statement, 2736
PROC UCM statement, 2756, *see* UCM procedure
PROC VARMAX statement, 2893
PROC X11 statement, 3093
PROC X12 statement, 3162
PROC X13 statement, 3162
PRODUCTION option
ENDOGENOUS statement (HPQLIM), 1161, 1164
ENDOGENOUS statement (QLIM), 1973
PROFILE
ESTIMATE statement (UCM), 2766
PROFILELIKE option
PLOTS option, PROC COUNTREG statement, 594
PROGRAM option
SHOW statement (COUNTREG), 614
Program Statements
TIMEDATA procedure, 2617
PROJECT= option
FORECAST command (TSFS), 3773
PROPCOV= option
BAYES statement (COUNTREG), 599
BAYES statement (HPQLIM), 1156
BAYES statement (QLIM), 1968
PROPORTION= value
BAYES statement (QLIM), 1965
PROPORTION=value
BAYES statement (COUNTREG), 596
PROXY= option
LIBNAME statement (SASEFRED), 3388
LIBNAME statement (SASEQUAN), 3487
LIBNAME statement (SASEXFSD), 3567
PSEUDO= option
SOLVE statement (MODEL), 1531
PSTAT option
PROC PANEL statement, 1805
PURE option
ENTROPY procedure, 810
PURGE option
RESET statement (MODEL), 1525
PUT, 1686
PWC option
COMPARE statement (LOAN), 1375

PWOFCOST option
COMPARE statement (LOAN), 1375

Q option
IRREGULAR statement (UCM), 2772
Q= option
ESTIMATE statement (ARIMA), 258
GARCH statement, 2902
IDENTIFY statement (ARIMA), 255
MODEL statement (AUTOREG), 362
MODEL statement (VARMAX), 2913, 2976
QLIM procedure, 1955
PRIOR statement, 1979
syntax, 1955
QLIM procedure, CLASS statement, 1971
QLIM procedure, FREQ statement, 1975
QLIM procedure, RANDOM statement, 1980
QLIM procedure, TEST statement, 1983
QLIM procedure, WEIGHT statement, 1984
QN option
METHOD= option, PROC COUNTREG statement, 594
QTR
function, 176
QUARTERLY statement
X11 procedure, 3103
QUASI= option
SOLVE statement (MODEL), 1531
QUIET= option
FCMPOPT statement (SIMILARITY), 2245
FCMPOPT statement (TIMEDATA), 2612

R= option
FIXED statement (LOAN), 1368
OUTPUT statement (AUTOREG), 384
OUTPUT statement (PANEL), 1827
OUTPUT statement (PDLREG), 1934
OUTPUT statement (SIMLIN), 2308
RANDINIT option
BAYES statement (COUNTREG), 596
BAYES statement (QLIM), 1965
MODEL statement (MDC), 1417
RANDNUM= option
MODEL statement (MDC), 1417
RANDOM= option
SOLVE statement (MODEL), 1531, 1642, 1658
RANDOMREG
UCM procedure, 2775
RANGE, 1675
RANGE option
MODEL statement (ENTROPY), 816
RANGE statement
DATASOURCE procedure, 719
MODEL procedure, 1524
RANGE= option
LIBNAME statement (SASECRSP), 3287
LIBNAME statement (SASEFAME), 3330
RANK option
MODEL statement (MDC), 1417
RANK= option
COINTEG statement (VARMAX), 2869, 2898, 2997
RANONE option
MODEL statement (HPPANEL), 1122
MODEL statement (PANEL), 1814
MODEL statement (TSCSREG), 2738
RANTWO option
MODEL statement (HPPANEL), 1122
MODEL statement (PANEL), 1815
MODEL statement (TSCSREG), 2738
RAO option
TEST statement (ENTROPY), 818
TEST statement (MODEL), 1534
RATE= option
FIXED statement (LOAN), 1368
RECFM= option
PROC DATASOURCE statement, 715
RECPEV= option
OUTPUT statement (AUTOREG), 384
RECRES= option
OUTPUT statement (AUTOREG), 384
REDUCECV= option
AUTOMDL statement (X12), 3173
AUTOMDL statement (X13), 3173
REDUCED option
PROC SYSLIN statement, 2569
REEVAL option
FORECAST command (TSFS), 3775
REF= option
CLASS statement (COUNTREG), 603
CLASS statement (HPSEVERITY), 1219
CLASS statement (SEVERITY), 2097
REFIT option
FORECAST command (TSFS), 3775
REGRESSION statement
X12 procedure, 3188
X13 procedure, 3188
remote Fame data access
physical name using #port number, 3329
RENAME statement
DATASOURCE procedure, 722
REPLACEBACK option
FORECAST statement (ESM), 856
REPLACEMISSING option
FORECAST statement (ESM), 856
REPORTMISSINGS option
PROC MODEL statement, 1501
RESET option

MODEL statement (AUTOREG), 372
RESET statement
MODEL procedure, 1525
RESIDDATA= option
SOLVE statement (MODEL), 1529
RESIDEST option
PROC STATESPACE statement, 2520
RESIDUAL
OUTPUT statement (HPPANEL), 1123
OUTPUT statement (HPQLIM), 1165
OUTPUT statement (QLIM), 1979
RESIDUAL= option
OUTPUT statement (AUTOREG), 384
OUTPUT statement (PANEL), 1827
OUTPUT statement (PDLREG), 1934
OUTPUT statement (SIMLIN), 2308
OUTPUT statement (SYSLIN), 2572
RESIDUALM= option
OUTPUT statement (AUTOREG), 384
OUTPUT statement (PDLREG), 1935
RESTART option
MODEL statement (MDC), 1417
RESTORE= option
PROC COUNTREG statement, 592
RESTRICT statement
AUTOREG procedure, 384
COUNTREG procedure, 611
ENTROPY procedure, 817
HPCOUNTREG procedure, 1088
HPQLIM procedure, 1167
MDC procedure, 1424
MODEL procedure, 1525
PDLREG procedure, 1935
QLIM procedure, 1982
STATESPACE procedure, 2522
SYSLIN procedure, 2573
VARMAX procedure, 2922, 3055
RETAIN statement
MODEL procedure, 1687
RHO option
MODEL statement (PANEL), 1827
MODEL statement (TSCSREG), 2739
RHO= option
AUTOREG statement (UCM), 2760
CYCLE statement (UCM), 2763
RIGHTCENSORED= option
LOSS statement (HPSEVERITY), 1223
LOSS statement (SEVERITY), 2102
RIGHTTRUNCATED= option
LOSS statement (HPSEVERITY), 1224
LOSS statement (SEVERITY), 2102
RKNOTS option
SPLINESEASON statement (UCM), 2782
RLLIMITS= number
BAYES statement (COUNTREG), 595
BAYES statement (QLIM), 1965
RM= option
OUTPUT statement (AUTOREG), 384
OUTPUT statement (PDLREG), 1935
ROBUST option
MODEL statement (PANEL), 1816
ROUND= NONE option
FIXED statement (LOAN), 1370
ROUND= option
FIXED statement (LOAN), 1370
'row titles' option
ROWS statement (COMPUTAB), 493
ROWS statement
COMPUTAB procedure, 493
ROWS= option
LIBNAME statement (SASEQUAN), 3487
RTEND= option
LIBNAME statement (SASEFRED), 3389
RTS= option
PROC COMPUTAB statement, 490
RTSTART= option
LIBNAME statement (SASEFRED), 3389
RUNS option
MODEL statement (AUTOREG), 372

S option
IRREGULAR statement (UCM), 2772
PROC SPECTRA statement, 2334
SAMESCALE option
MODEL statement (MDC), 1418
SAMPLEVAR= option
OUTPUT statement (HPCDM), 1015
SAMPLING= option
BAYES statement (COUNTREG), 599
BAYES statement (HPQLIM), 1156
BAYES statement (QLIM), 1968
SAR option
IRREGULAR statement (UCM), 2772
SATISFY= option
SOLVE statement (MODEL), 1528
SCALE= option
INPUT statement (SIMILARITY), 2249
SCALEMODEL statement
HPSEVERITY procedure, 1227
SEVERITY procedure, 2105
SCAN option
IDENTIFY statement (ARIMA), 255
SCENTER option
MODEL statement (VARMAX), 2908
SCHEDULE option
FIXED statement (LOAN), 1371
SCHEDULE= option
FIXED statement (LOAN), 1371

SCHEDULE= YEARLY option
FIXED statement (LOAN), 1371
SCORE statement
COUNTREG procedure, 611
SDATA= option
ENTROPY procedure, 811, 830
FIT statement (MODEL), 1516, 1630, 1710
SOLVE statement (MODEL), 1529, 1642, 1671
SDIAG option
PROC SYSLIN statement, 2568
SDIF= option
INPUT statement (SIMILARITY), 2249
TARGET statement (SIMILARITY), 2254
VAR statement (TIMEDATA), 2616
VAR statement (TIMESERIES), 2685
SDIFF= option
IDENTIFY statement (X12), 3181
IDENTIFY statement (X13), 3181
SEASON statement
TIMESERIES procedure, 2678
UCM procedure, 2776
SEASONALITY= option
PROC ESM statement, 854
PROC SIMILARITY statement, 2244
PROC TIMEDATA statement, 2612
PROC TIMESERIES statement, 2668
SEASONALMA= option
X11 statement (X12), 3202
X11 statement (X13), 3202
SEASONS= option
PROC FORECAST statement, 956
PROC X12 statement, 3168
PROC X13 statement, 3168
SEATSDECOMP statement
X12 procedure, 3195
X13 procedure, 3195
SECOND
function, 176
SEED= option
BAYES statement (COUNTREG), 599
BAYES statement (HPQLIM), 1157
BAYES statement (QLIM), 1969
MODEL statement (MDC), 1418
PROC HPCDM statement, 1012
QLIM procedure, 1961
SOLVE statement (MODEL), 1531, 1642
SEIDEL option
SOLVE statement (MODEL), 1530
SELECT, 1687
SELECT option
ENDOGENOUS statement (QLIM), 1973
SELECT= option
DISPMODEL statement, COUNTREG procedure, 604, 615
MODEL statement (COUNTREG), 606
SPATIALDISPEFFECTS statement, COUNTREG procedure, 615
SPATIALZEROEFFECTS statement, COUNTREG procedure, 616
ZEROMODEL statement (COUNTREG), 619
SELECT=PEN option
MODEL statement (COUNTREG), 607
SELECTVAR option
MODEL statement (QLIM), 1977
SETID= option
LIBNAME statement (SASECRSP), 3285
LIBNAME statement (SASEXCCM), 3522
SETMISSING= option
FORECAST statement (ESM), 856
ID statement (ESM), 859
ID statement (SIMILARITY), 2247
ID statement (TIMEDATA), 2614
ID statement (TIMESERIES), 2677
INPUT statement (SIMILARITY), 2249
TARGET statement (SIMILARITY), 2254
VAR statement (TIMEDATA), 2616
VAR statement (TIMESERIES), 2686
SEVERITY procedure, 2085
CLASS statement, 2096
DIST statement, 2098
LOSS statement, 2100
NLOPTIONS statement, 2103
OUTSCORELIB statement, 2103
SCALEMODEL statement, 2105
syntax, 2085
WEIGHT statement, 2107
SEVERITY procedure, CLASS statement
DESCENDING option, 2096
MISSING option, 2098
ORDER= option, 2096
PARAM= option, 2098
REF= option, 2097
TRUNCATE= option, 2098
SEVERITY procedure, DIST statement
INIT= option, 2099
LISTONLY option, 2100
VALIDATEONLY option, 2100
SEVERITY procedure, LOSS statement
LEFTCENSORED= option, 2101
LEFTTRUNCATED= option, 2101
PROBOBSERVED= option, 2101
RIGHTCENSORED= option, 2102
RIGHTTRUNCATED= option, 2102
SEVERITY procedure, OUTSCORELIB statement
COMMONPACKAGE option, 2104
OUTBYID= option, 2104
OUTLIB= option, 2103

SEVERITY procedure, PROC SEVERITY statement, 2087
COVOUT option, 2087
CRITERION= option, 2093
DATA= option, 2087
EDF=AUTO option, 2094
EDF=KAPLANMEIER option, 2094
EDF=MODIFIEDKM option, 2094
EDF=STANDARD option, 2094
EDF=TURNBULL option, 2095
EMPIRICALCDF= option, 2094
INEST= option, 2088
INITSAMPLE option, 2088
INSTORE= option, 2089
NAMELEN= option, 2089
NOCLPRINT option, 2089
NOPRINT option, 2089
OBJECTIVE= option, 2095
OUTCDF= option, 2089
OUTEST= option, 2089
OUTMODELINFO= option, 2090
OUTSTAT= option, 2090
OUTSTORE= option, 2090
PLOTS= option, 2090
PRINT= option, 2091
VARDEF= option, 2093
SEVERITY procedure, SCALEMODEL statement
DFMIXTURE= option, 2106
OFFSET= option, 2107
SEVERITYEST= option
PROC HPCDM statement, 1012
SEVERITYMODEL statement
HPCDM procedure, 1018
SHORT= option
LIBNAME statement (SASEHAVR), 3436
SHOW statement
COUNTREG procedure, 613
SICCD= option
LIBNAME statement (SASECRSP), 3287
SIGCORR= option
PROC STATESPACE statement, 2519
SIGMA= option
OUTLIER statement (ARIMA), 261
SIGMALIM= option
X11 statement (X12), 3203
X11 statement (X13), 3203
SIGSQ= option
FORECAST statement (ARIMA), 263
SIM option
RANDOM statement (QLIM), 1982
SIM suboption1
RANDOM statement (QLIM), 1982
SIM suboption2
RANDOM statement (QLIM), 1982
SIMILARITY procedure, 2240
syntax, 2240
SIMLIN procedure, 2304
syntax, 2304
SIMPLE option
PROC SYSLIN statement, 2569
SIMTIME option
BAYES statement (COUNTREG), 599
BAYES statement (QLIM), 1969
SIMULATE option
SOLVE statement (MODEL), 1530
SIMULATE statement
COPULA procedure, 545
HPCOPULA procedure, 1060
SIN function, 1681
SINGLE option
SOLVE statement (MODEL), 1531, 1662
SINGULAR= option
ESTIMATE statement (ARIMA), 260
FIT statement (MODEL), 1519
MODEL statement (TSCSREG), 2739
PROC FORECAST statement, 956
PROC STATESPACE statement, 2520
PROC SYSLIN statement, 2568
SINGULAR=option
MODEL statement (PANEL), 1815
SINH function, 1681
SINTPER= option
PROC FORECAST statement, 956
SKIP option
ROWS statement (COMPUTAB), 494
SKIPFIRST= option
ESTIMATE statement (UCM), 2766
FORECAST statement (UCM), 2769
SKIPLAST= option
ESTIMATE statement (UCM), 2765
SLENTRY= option
PROC FORECAST statement, 957
SLIDE= option
TARGET statement (SIMILARITY), 2254
SLIST= option
PROC MODEL statement, 1498
SLOPE statement
UCM procedure, 2778
SLSTAY= option
MODEL statement (AUTOREG), 378
PROC FORECAST statement, 957
SMA option
IRREGULAR statement (UCM), 2772
SOLVE statement
MODEL procedure, 1527
SOLVEPRINT option
SOLVE statement (MODEL), 1532
SORT= option

LIBNAME statement (SASEFRED), 3389
LIBNAME statement (SASEQUAN), 3487
SORTNAMES option
PROC ESM statement, 854
PROC SIMILARITY statement, 2244
PROC TIMESERIES statement, 2669
SOURCE= option
LIBNAME statement (SASEHAVR), 3436
SP option
IRREGULAR statement (UCM), 2772
SPAN= option
OUTLIER statement (X12), 3186
OUTLIER statement (X13), 3186
PROC X12 statement, 3168
PROC X13 statement, 3168
SPATIALDISPEFFECTS statement
COUNTREG procedure, 614
SPATIALEFFECTS statement
COUNTREG procedure, 615
SPATIALID statement
COUNTREG procedure, 615
SPATIALZEROEFFECTS statement
COUNTREG procedure, 616
SPECTRA procedure, 2332
syntax, 2332
SPECTRA statement
TIMESERIES procedure, 2679
SPECTRUMSERIES= option
PROC X12 statement, 3168
PROC X13 statement, 3168
SPLINEREG
UCM procedure, 2779
SPLINESEASON
UCM procedure, 2781
SPSCALE option
MODEL statement (MDC), 1418
SQ option
IRREGULAR statement (UCM), 2772
SQGAMMA
PRIOR statement (QLIM), 1980
SQIGAMMA
PRIOR statement (QLIM), 1980
SQRT function, 1681
SRESTRICT statement
SYSLIN procedure, 2574
SSA statement
TIMESERIES procedure, 2681
SSM procedure, 2367
syntax, 2367
SSM procedure, PROC SSM statement
PLOT option, 2370
SSPAN statement
X11 procedure, 3105
START= option
FIT statement (MODEL), 1512, 1561, 1704
FIXED statement (LOAN), 1370
ID statement (ESM), 859
ID statement (SIMILARITY), 2247
ID statement (TIMEDATA), 2615
ID statement (TIMESERIES), 2678
LIBNAME statement (SASEFRED), 3389
LIBNAME statement (SASEHAVR), 3436
LIBNAME statement (SASEQUAN), 3486
MODEL statement (AUTOREG), 378
MONTHLY statement (X11), 3100
PROC FORECAST statement, 957
PROC SIMLIN statement, 2305
PROC X12 statement, 3168
PROC X13 statement, 3168
QUARTERLY statement (X11), 3104
SOLVE statement (MODEL), 1530
STARTITER option
FIT statement (MODEL), 1562
STARTITER= option
FIT statement (MODEL), 1520
STARTSUM= option
PROC ESM statement, 854
STARTUP= option
MODEL statement (AUTOREG), 363
STAT= option
FORECAST command (TSFS), 3774
STATE statement
SSM procedure, 2379
STATESPACE procedure, 2516
syntax, 2516
STATIC option
FIT statement (MODEL), 1594
SOLVE statement (MODEL), 1530, 1592, 1639
STATIONARITY= option
BAYES statement (QLIM), 1965
IDENTIFY statement (ARIMA), 255, 256
MODEL statement (AUTOREG), 373
STATIONARITY=option
BAYES statement (COUNTREG), 596
STATISTICS option
BAYES statement (COUNTREG), 600
BAYES statement (HPQLIM), 1157
BAYES statement (QLIM), 1969
STATS option
SOLVE statement (MODEL), 1533, 1657
STB option
MODEL statement (PDLREG), 1933
MODEL statement (SYSLIN), 2571
STD= option
HETERO statement (AUTOREG), 381
STEST statement
SYSLIN procedure, 2575
STORE statement

COUNTREG procedure, 616
SUBFORM= option
GARCH statement, 2902
SUBJECT option
RANDOM statement (QLIM), 1980
SUMBY statement
COMPUTAB procedure, 497
SUMMARY option
MONTHLY statement (X11), 3100
QUARTERLY statement (X11), 3104
SUMONLY option
PROC COMPUTAB statement, 491
SUR option
ENTROPY procedure, 810
FIT statement (MODEL), 1514, 1538, 1707
PROC SYSLIN statement, 2569
SYSLIN procedure, 2565
syntax, 2565

T
PRIOR statement (COUNTREG), 610
PRIOR statement (HPQLIM), 1167
PRIOR statement (QLIM), 1980
T option
ERRORMODEL statement (MODEL), 1508, 1549
TABLES statement
X11 procedure, 3106
X12 procedure, 3197
X13 procedure, 3197
TABLES table names
TABLES statement (X12), 3197
TABLES statement (X13), 3197
TAIL= option
PROC COPULA statement, 544, 546
TAN function, 1681
TANH function, 1681
TARGET statement
SIMILARITY procedure, 2250
TARGETESS= number
BAYES statement (COUNTREG), 595
BAYES statement (QLIM), 1964
TARGETSTATS<= option>
BAYES statement (COUNTREG), 595
BAYES statement (QLIM), 1964
TAX= option
COMPARE statement (LOAN), 1375
TAXRATE= option
COMPARE statement (LOAN), 1375
TCCV= option
OUTLIER statement (X12), 3186
OUTLIER statement (X13), 3186
TCRATE= option
OUTLIER statement (X12), 3186
OUTLIER statement (X13), 3186
TDCOMPUTE= option
MONTHLY statement (X11), 3100
TDCUTOFF= option
SSPAN statement (X11), 3105
TDREGR= option
MONTHLY statement (X11), 3101
TE1
OUTPUT statement (HPQLIM), 1166
OUTPUT statement (QLIM), 1979
TE2
OUTPUT statement (HPQLIM), 1166
OUTPUT statement (QLIM), 1979
TECH= option
ENTROPY procedure, 812
TECHNIQUE= option
ENTROPY procedure, 812
TEST statement
AUTOREG procedure, 385
ENTROPY procedure, 817
MODEL procedure, 1533
SYSLIN procedure, 2576
VARMAX procedure, 2878, 2934, 3055
TEST= option
HETERO statement (AUTOREG), 381
THEIL option
SOLVE statement (MODEL), 1533, 1657
THIN= option
BAYES statement (COUNTREG), 600
BAYES statement (HPQLIM), 1158
BAYES statement (QLIM), 1970
THREADS= option
PERFORMANCE statement (high-performance analytical procedures), 89
THRESHOLDPCT option
SSA statement (TIMESERIES), 2683
TI option
COMPARE statement (LOAN), 1376
TICKER= option
LIBNAME statement (SASECRSP), 3286
TIME
function, 176
TIME option
MODEL statement (PANEL), 1817
TIME= option
FIT statement (MODEL), 1516, 1598
SOLVE statement (MODEL), 1529, 1598
TIMEDATA procedure, 2608
syntax, 2608
TIMEID procedure, 2636
syntax, 2636
TIMEOUT= option
PERFORMANCE statement (high-performance analytical procedures), 88

TIMEPART function, 121, 176
TIMESERIES procedure, 2662
  syntax, 2662
TIN=, 896
TITLES option
  COLUMNS statement (COMPUTAB), 492
TO= option
  PROC EXPAND statement, 893, 897
TODAY function, 176
TOL= option
  ESTIMATE statement (X12), 3176
  ESTIMATE statement (X13), 3176
TOL= value
  BAYES statement (COUNTREG), 596
  BAYES statement (QLIM), 1965
TOL=value
  BAYES statement (COUNTREG), 596
TOTAL option
  PROC SIMLIN statement, 2306
TOUT=, 896
TP option
  MODEL statement (AUTOREG), 377
TR option
  METHOD= option, PROC COUNTREG statement, 594
  MODEL statement (AUTOREG), 363
TRACE option
  PROC MODEL statement, 1501
TRACE= option
  FCMPOPT statement (SIMILARITY), 2245
  FCMPOPT statement (TIMEDATA), 2612
TRANSFORM statement
  X12 procedure, 3197
  X13 procedure, 3197
TRANSFORM=, 896
TRANSFORM= option
  ARIMA statement (X11), 3096
  FORECAST statement (ESM), 856
  INPUT statement (SIMILARITY), 2249
  OUTPUT statement (AUTOREG), 384
  OUTPUT statement (PDLREG), 1935
  TARGET statement (SIMILARITY), 2254
  VAR statement (TIMEDATA), 2616
  VAR statement (TIMESERIES), 2686
TRANSFORMATION= option
  LIBNAME statement (SASEQUAN), 3487
TRANSFORMIN= option
  CONVERT statement (EXPAND), 896, 904
TRANSFORMOUT= option
  CONVERT statement (EXPAND), 896, 904
TRANSIN=, 896
TRANSOUT=, 896
TRANSPOSE option
  SSA statement (TIMESERIES), 2683
TRANSPOSE procedure, 141
TRANSPOSE= option
  CORR statement (TIMESERIES), 2671
  CROSSCORR statement (TIMESERIES), 2673
  DECOMP statement (TIMESERIES), 2675
  SEASON statement (TIMESERIES), 2678
  TREND statement (TIMESERIES), 2685
TREND statement
  SSM procedure, 2384
  TIMESERIES procedure, 2684
TREND= option
  DFPVALUE macro, 183
  DFTEST macro, 184
  MODEL statement (VARMAX), 2909
  PROC FORECAST statement, 957
TREND=LINEAR option
  MODEL statement (VARMAX), 2994
TRENDADJ option
  MONTHLY statement (X11), 3101
  QUARTERLY statement (X11), 3105
TRENDMA= option
  MONTHLY statement (X11), 3101
  X11 statement (X12), 3203
  X11 statement (X13), 3203
TRIMMISS= option
  INPUT statement (SIMILARITY), 2250
TRIMMISSING= option
  INPUT statement (SIMILARITY), 2255
TRUEINTEREST option
  COMPARE statement (LOAN), 1376
TRUNCATE= option
  CLASS statement (HPSEVERITY), 1220
  CLASS statement (SEVERITY), 2098
TRUNCATED option
  ENDOGENOUS statement (HPQLIM), 1160, 1164
  ENDOGENOUS statement (QLIM), 1972
TSCSREG procedure, 2734
  syntax, 2734
TSFS, 3607
TSFS procedure, 3607
TSNAME = option
  PROC PANEL statement, 1806
TSVIEW
  macro, 3771
TUNECHLI= option
  LIBNAME statement (SASEFAME), 3333
TUNEFAME= option
  LIBNAME statement (SASEFAME), 3333
TYPE= option
  FIT statement (MODEL), 1516
  MODEL statement (AUTOREG), 362
  MODEL statement (MDC), 1418
  OUTLIER statement (ARIMA), 261

OUTLIER statement (X12), 3186
OUTLIER statement (X13), 3186
PROC DATASOURCE statement, 715
PROC SIMLIN statement, 2306
SOLVE statement (MODEL), 1529
TEST statement (AUTOREG), 386
X11 statement (X12), 3203
X11 statement (X13), 3203
TYPE=option
BLOCKSEASON statement (UCM), 2761
SEASON statement (UCM), 2778

U option
ERRORMODEL statement (MODEL), 1508
UCL= option
OUTPUT statement (AUTOREG), 384
OUTPUT statement (PDLREG), 1935
UCLM= option
OUTPUT statement (AUTOREG), 384
OUTPUT statement (PDLREG), 1935
UCM procedure, 2753
syntax, 2753
UCM procedure, PROC UCM statement
PLOT option, 2756
UL option
ROWS statement (COMPUTAB), 494
UNIFORM
PRIOR statement (COUNTREG), 611
PRIOR statement (HPQLIM), 1167
PRIOR statement (QLIM), 1980
UNIFORMEC= option
MODEL statement (MDC), 1416
UNIFORMPARM= option
MODEL statement (MDC), 1416
UNITS= option
LIBNAME statement (SASEFRED), 3389
UNITSCALE= option
MODEL statement (MDC), 1416
UNITVARIANCE= option
MODEL statement (MDC), 1418
UNIVERSE= option
LIBNAME statement (SASEXFSD), 3564
UNPACK option
PLOTS option, PROC COUNTREG statement, 593
UNPACK= option
PROC COPULA statement, 544, 545
UNREST option
MODEL statement (SYSLIN), 2571
UPPERBOUND= option
ENDOGENOUS statement (HPQLIM), 1160, 1164
ENDOGENOUS statement (QLIM), 1972
URL= option
LIBNAME statement (SASEFRED), 3390
URSQ option
MODEL statement (AUTOREG), 377
USE= option
FORECAST statement (ESM), 857
USERDEFINED statement
X12 procedure, 3199
X13 procedure, 3199
USERNAME= option
LIBNAME statement(SASEXFSD), 3567
USSCP option
PROC SYSLIN statement, 2569
USSCP2 option
PROC SYSLIN statement, 2569
UTILITY statement
MDC procedure, 1426

VALIDATEONLY option
DIST statement (HPSEVERITY), 1222
DIST statement (SEVERITY), 2100
VAR option
MODEL statement (PANEL), 1827
MODEL statement (TSCSREG), 2738
VAR Statement
HPCOPULA procedure, 1061
VAR statement
COPULA procedure, 546
FORECAST procedure, 958
MODEL procedure, 1534, 1673
SPECTRA procedure, 2334
STATESPACE procedure, 2522
SYSLIN procedure, 2577
TIMEDATA procedure, 2616
TIMESERIES procedure, 2685
X11 procedure, 3106
X12 procedure, 3199
X13 procedure, 3199
VAR= option
FORECAST command (TSFS), 3772, 3773
IDENTIFY statement (ARIMA), 256
TSVIEW command (TSFS), 3772, 3773
VARDEF= option
FIT statement (MODEL), 1514, 1541, 1553
MODEL statement (VARMAX), 2909
Proc ENTROPY, 810
PROC HPCDM statement, 1012
PROC HPSEVERITY statement, 1214
PROC SEVERITY statement, 2093
PROC SYSLIN statement, 2569
VARGROUP statement
MODEL procedure, 1534
VARIANCE= option
AUTOREG statement (UCM), 2760
CYCLE statement (UCM), 2763

IRREGULAR statement (UCM), 2770
LEVEL statement (UCM), 2773
OUTPUT statement (COUNTREG), 609
OUTPUT statement (HPCOUNTREG), 1087
RANDOMREG statement (UCM), 2776
SCORE statement (COUNTREG), 613
SLOPE statement (UCM), 2779
SPLINEREG statement (UCM), 2781
SPLINESEASON statement (UCM), 2782
VARIANCE=option
BLOCKSEASON statement (UCM), 2761
SEASON statement (UCM), 2778
VARLIST statement, 1413
VARMAX procedure, 2890
syntax, 2890
VCOMP= option
MODEL statement (HPPANEL), 1122
MODEL statement (PANEL), 1815
VDATA= option
FIT statement (MODEL), 1516, 1631
VECTORPLOTS= option
PROC TIMESERIES statement, 2669
VINTAGE= option
LIBNAME statement (SASEFRED), 3391
VNRRANK option
MODEL statement (AUTOREG), 377

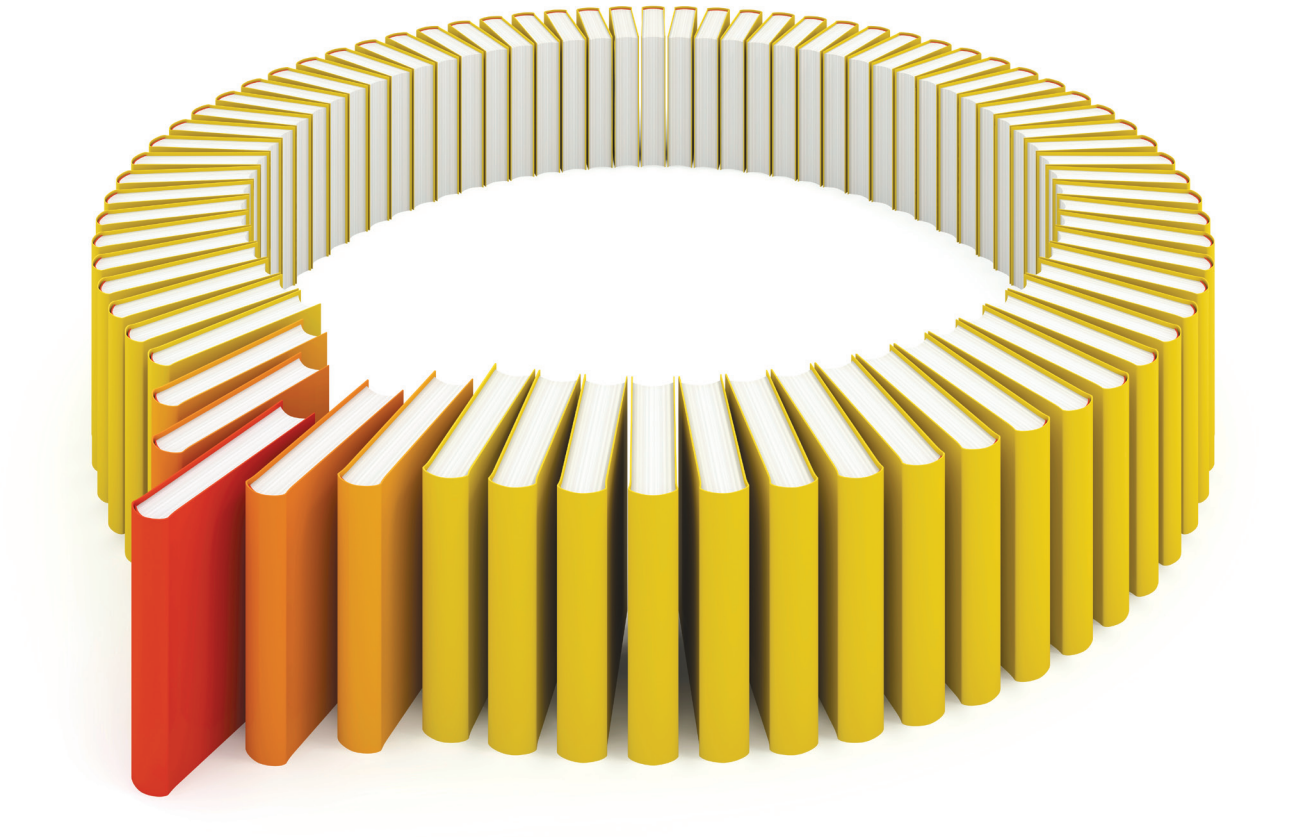
W option
ARM statement (LOAN), 1374
WALD option
TEST statement (COUNTREG), 617
TEST statement (ENTROPY), 818
TEST statement (HPCOUNTREG), 1089
TEST statement (HPPANEL), 1124
TEST statement (HPQLIM), 1168
TEST statement (MDC), 1425
TEST statement (MODEL), 1534
TEST statement (PANEL), 1828
TEST statement (QLIM), 1983
WEEK function, 176
WEEKDAY
function, 120
WEEKDAY function, 176
WEIGHT statement, 1577
COUNTREG procedure, 618
ENTROPY procedure, 819
HPSEVERITY procedure, 1229
MODEL procedure, 1535
SEVERITY procedure, 2107
SYSLIN procedure, 2578
WEIGHT= option
PROC FORECAST statement, 957
WEIGHTS option
SPECTRA statement (TIMESERIES), 2680
WEIGHTS statement
SPECTRA procedure, 2335
WHEN, 1687
WHERE statement
DATASOURCE procedure, 719
WHITE option
FIT statement (MODEL), 1518
WHITENOISE= option
ESTIMATE statement (ARIMA), 258
IDENTIFY statement (ARIMA), 256
WHITETEST option
PROC SPECTRA statement, 2334
WILDCARD= option
LIBNAME statement (SASEFAME), 3330
WISHART= option
SOLVE statement (MODEL), 1532
WMAT=option
PROC COUNTREG statement, 591
WOOLDRIDGE02 option
MODEL statement (PANEL), 1826
WORSTCASE option
ARM statement (LOAN), 1374

X11 procedure, 3091
syntax, 3091
X11 statement
X12 procedure, 3199
X13 procedure, 3199
X12 procedure, 3157
syntax, 3157
X12 procedure, PROC X12 statement
PLOT option, 3164
X13 procedure, 3157
syntax, 3157
X13 procedure, PROC X13 statement
PLOT option, 3164
XBETA
OUTPUT statement (HPQLIM), 1166
OUTPUT statement (QLIM), 1979
XBETA= option
OUTPUT statement (COUNTREG), 609
OUTPUT statement (HPCOUNTREG), 1086
OUTPUT statement (MDC), 1424
SCORE statement (COUNTREG), 613
XLAG= option
MODEL statement (VARMAX), 2914
XMLMAP= option
LIBNAME statement (SASEFRED), 3391
LIBNAME statement (SASEQUAN), 3484
LIBNAME statement(SASEXFSD), 3566
XPX option
FIT statement (MODEL), 1518, 1568
MODEL statement (PDLREG), 1933
MODEL statement (SYSLIN), 2572

XREF option
PROC MODEL statement, 1500, 1691

YEAR
function, 120, 177
YEARSEAS option
OUTPUT statement (X12), 3186
OUTPUT statement (X13), 3186
SEATSDECOMP statement (X12), 3196
SEATSDECOMP statement (X13), 3196
YRAHEADOUT option
PROC X11 statement, 3094
YYQ
function, 119, 177

ZERO= option
COLUMNS statement (COMPUTAB), 492
ROWS statement (COMPUTAB), 494
ZEROMISS option
PROC FORECAST statement, 957
ZEROMISS= option
FORECAST statement (ESM), 857
ID statement (TIMEDATA), 2615
INPUT statement (SIMILARITY), 2250
TARGET statement (SIMILARITY), 2255
VAR statement (TIMEDATA), 2617
ZEROMISSING= option
ID statement (PROC ESM), 860
ID statement (SIMILARITY), 2247
ZEROMODEL statement
COUNTREG procedure, 618
HPCOUNTREG procedure, 1090
ZEROPROFILE option
PLOTS option, PROC COUNTREG statement, 594
ZEROWEIGHT= option
MONTHLY statement (X11), 3101
QUARTERLY statement (X11), 3105
ZGAMMA= option
OUTPUT statement (COUNTREG), 609
OUTPUT statement (HPCOUNTREG), 1086
SCORE statement (COUNTREG), 613



Gain Greater Insight into Your SAS® Software with SAS Books.

Discover all that you need on your journey to knowledge and empowerment.

support.sas.com/bookstore
for additional books and resources.

sas
THE POWER TO KNOW®

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies. © 2013 SAS Institute Inc. All rights reserved. S107969US.0613